Features

- High-performance, Low-power Atmel®AVR®8-bit Microcontroller
- Advanced RISC Architecture
 - 133 Powerful Instructions Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers + Peripheral Control Registers
 - Fully Static Operation
 - Up to 16MIPS Throughput at 16MHz
 - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
 - 128KBytes of In-System Self-programmable Flash program memory
 - 4KBytes EEPROM
 - 4KBytes 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 64KBytes 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 2 to 16 Bits
 - Output Compare Modulator
 - 8-channel, 10-bit ADC
 - 8 Single-ended Channels
 - 7 Differential Channels
 - 2 Differential Channels with Programmable Gain at 1x, 10x, or 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
- Speed Grades
 - 0 16MHz



8-bit **AVR**[®] Microcontroller with 128KBytes In-System Programmable Flash

ATmega128A

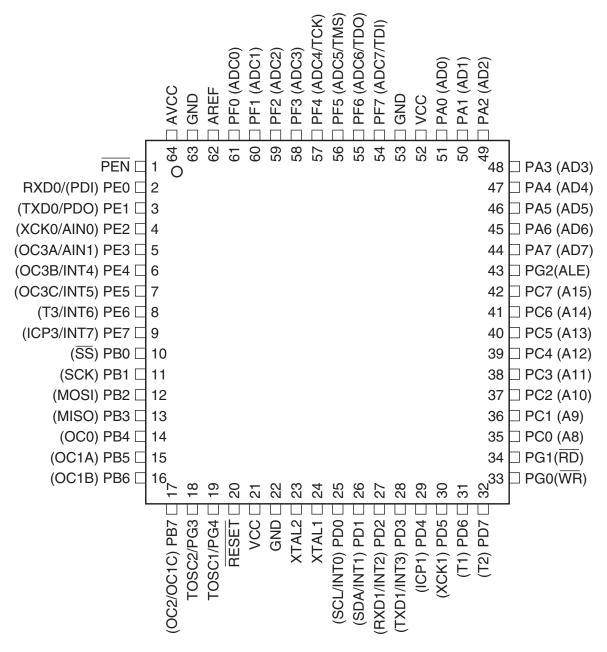
Summary





1. Pin Configurations





Note: The Pinout figure applies to both TQFP and MLF packages. The bottom pad under the QFN/MLF package should be soldered to ground.

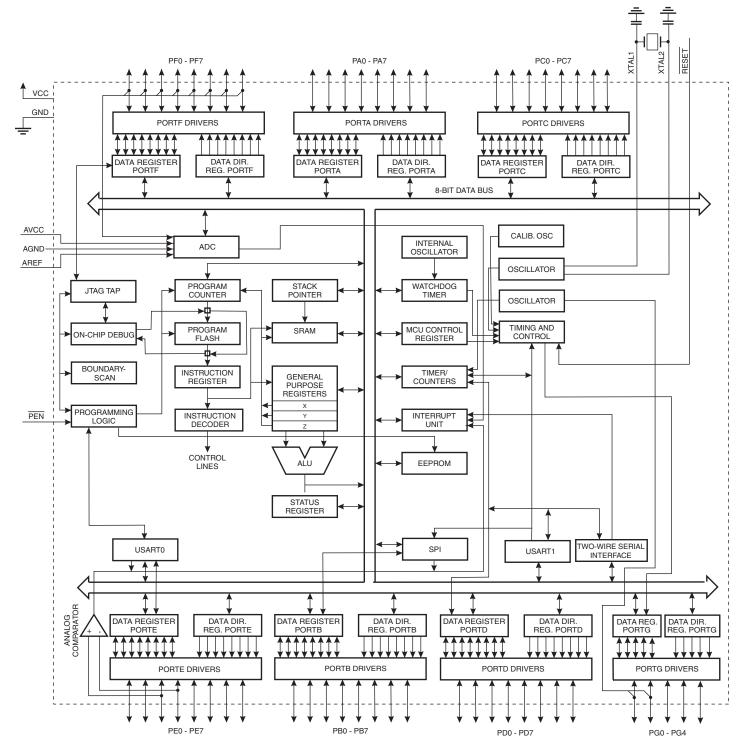
2. Overview

The Atmel[®]AVR[®]ATmega128A 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 ATmega128A achieves throughputs approaching 1MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

² ATmega128A

2.1 Block Diagram

Figure 2-1. Block Diagram







The Atmel[®]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 ATmega128A provides the following features: 128Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 4Kbytes EEPROM, 4Kbytes 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, 2 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 nonvolatile memory technology. The Onchip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile 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 ATmega128A is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega128A 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.

2.2 ATmega103 and ATmega128A Compatibility

The ATmega128A is a highly complex microcontroller where the number of I/O locations supersedes the 64 I/O locations 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 ATmega128A. Most additional I/O locations are added in an Extended I/O space starting from \$60 to \$FF, (i.e., in the ATmega103 internal RAM space). These locations 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.

ATmega128A

The Atmel[®]AVR[®]ATmega128A is 100% pin compatible with ATmega103, and can replace the ATmega103 on current Printed Circuit Boards. The application note "Replacing ATmega103 by ATmega128A" describes what the user should be aware of replacing the ATmega103 by an ATmega128A.

2.2.1 ATmega103 Compatibility Mode

By programming the M103C fuse, the ATmega128 will be compatible with the ATmega103 regards to RAM, I/O pins and interrupt vectors as described above. However, some new features in ATmega128 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-bit Timer/Counters with three compare registers.
- Two-wire serial interface is not supported.
- Port C is output only.
- 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.
- In addition, there are some other minor differences to make it more compatible to ATmega103:
- Only EXTRF and PORF exists in MCUCSR.
- Timed sequence not required for Watchdog Time-out change.
- External Interrupt pins 3 0 serve as level interrupt only.
- USART has no FIFO buffer, so data overrun comes earlier.

Unused I/O bits in ATmega103 should be written to 0 to ensure same operation in ATmega128.

2.3 Pin Descriptions

2.3.1 VCC

Digital supply voltage.

2.3.2 GND

Ground.

2.3.3 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 ATmega128A as listed on page 73.





2.3.4 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 ATmega128A as listed on page 74.

2.3.5 Port C (PC7:PC0)

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 ATmega128A as listed on page 76. In ATmega103 compatibility mode, Port C is output only, and the port C pins are **not** tri-stated when a reset condition becomes active.

Note: The Atmel[®]AVR[®]ATmega128A is by default shipped in ATmega103 compatibility mode. Thus, if the parts are not programmed before they are put on the PCB, PORTC will be output during first power up, and until the ATmega103 compatibility mode is disabled.

2.3.6 Port D (PD7:PD0)

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 ATmega128A as listed on page 78.

2.3.7 Port E (PE7:PE0)

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 ATmega128A as listed on page 81.

2.3.8 Port F (PF7:PF0)

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

6 ATmega128A

ATmega128A

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.

2.3.9 Port G (PG4:PG0)

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.

Port G also serves the functions of various special features.

The port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.

In Atmel[®]AVR[®]ATmega103 compatibility mode, these pins only serves as strobes signals to the external memory as well as input to the 32 kHz Oscillator, and the pins are initialized to PG0 = 1, PG1 = 1, and PG2 = 0 asynchronously when a reset condition becomes active, even if the clock is not running. PG3 and PG4 are oscillator pins.

2.3.10 RESET

XTAL2

AREF

PEN

2.3.12

2.3.14

2.3.15

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 "System and Reset Characteristics" on page 324. Shorter pulses are not guaranteed to generate a reset.

- 2.3.11 XTAL1
 - Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.
 - Output from the inverting Oscillator amplifier.
 - 2.3.13 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 is the analog reference pin for the A/D Converter.

PEN is a programming enable pin for the SPI Serial Programming mode, and is internally pulled high . 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.





3. Resources

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

4. 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.

8 ATmega128A

5. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(\$FF)	Reserved	_	-	_	-	-	-	-	_	
:	Reserved	_	_	_	_	-	_	_	_	
(\$9E)	Reserved	_	_	_	_	_	_	_	_	
(\$9D)	UCSR1C	-	UMSEL1	UPM11	UPM10	USBS1	UCSZ11	UCSZ10	UCPOL1	195
(\$9C)	UDR1				USART1 I/O	Data Register				192
(\$9B)	UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1	193
(\$9A)	UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81	194
(\$99)	UBRR1L		•		USART1 Baud	Rate Register Lov	N			197
(\$98)	UBRR1H	-	-	-	-		USART1 Baud F	Rate Register High	ı	197
(\$97)	Reserved	-	-	-	-	-	-	-	-	
(\$96)	Reserved	-	-	-	-	-	-	-	-	
(\$95)	UCSR0C	_	UMSEL0	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0	195
(\$94)	Reserved	-	-	-	-	-	-	-	-	
(\$93)	Reserved	-	-	-	-	-	-	-	-	
(\$92)	Reserved	-	-	-	-	-	-	-	-	
(\$91)	Reserved	-	-	-	-	-	-	-	-	
(\$90)	UBRR0H	-	-	-	-		USART0 Baud F	Rate Register High	1	197
(\$8F)	Reserved	-	-	-	-	-	-		-	
(\$8E)	Reserved	-	-	-	-	-	-	-	-	
(\$8D)	Reserved	-	-	_	-	_	-	_	-	
(\$8C)	TCCR3C	FOC3A	FOC3B	FOC3C	-	-	-			138
(\$8B)	TCCR3A	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	134
(\$8A)	TCCR3B	ICNC3	ICES3	_	WGM33	WGM32	CS32	CS31	CS30	137
(\$89)	TCNT3H				er/Counter3 – Cou					139
(\$88)	TCNT3L				er/Counter3 – Co					139
(\$87)	OCR3AH				unter3 – Output C					140
(\$86)	OCR3AL				unter3 – Output C					140
(\$85)	OCR3BH				unter3 – Output C		* *			140
(\$84)	OCR3BL				unter3 – Output C					140
(\$83)	OCR3CH				unter3 – Output C					140
(\$82)	OCR3CL				unter3 – Output C	· ·	,			140
(\$81)	ICR3H				Counter3 – Input (141
(\$80)	ICR3L				Counter3 – Input			· · · · · ·		141
(\$7F)	Reserved	-	-	-	-	-	-	-	-	
(\$7E)	Reserved	-	-	-	-	-	-	-	-	
(\$7D)	ETIMSK	-	-	TICIE3	OCIE3A	OCIE3B	TOIE3	OCIE3C	OCIE1C	142
(\$7C)	ETIFR	-	-	ICF3	OCF3A	OCF3B	TOV3	OCF3C	OCF1C	143
(\$7B)	Reserved	-	-	-	-	-	-	-	-	
(\$7A)	TCCR1C	FOC1A	FOC1B	FOC1C	-	-	-	-	-	138
(\$79)	OCR1CH				unter1 – Output C		÷ ,			140
(\$78)	OCR1CL				unter1 – Output C	ompare Register	C Low Byte	,,		140
(\$77)	Reserved	-	-	-	_	-	-	-	-	
(\$76) (\$75)	Reserved	-	-	-		-	-	-	-	
, ,	Reserved		- T\A/E A	-	-	-		-	-	000
(\$74)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	226
(\$73)	TWDR	TMAG	T\A/ A =		Two-wire Serial In	÷		TMAA	TWOOL	228
(\$72) (\$71)	TWAR TWSR	TWA6 TWS7	TWA5 TWS6	TWA4 TWS5	TWA3 TWS4	TWA2 TWS3	TWA1	TWA0 TWPS1	TWGCE TWPS0	229 228
(\$71)	TWSR	10057	14420		o-wire Serial Inte		_	100731	100730	228
(\$70) (\$6F)	OSCCAL			IW		bration Register	913101			44
(\$6F) (\$6E)	Reserved	_	_	_	–	–	-	-	-	44
(\$6E) (\$6D)	XMCRA	-	- SRL2	- SRL1	- SRL0	- SRW01	- SRW00	- SRW11	-	34
(\$6D) (\$6C)	XMCRA	 XMBK	- 5RL2	- SHLT	- SRLU	- SRW01	XMM2	XMM1	XMM0	34 35
(\$6B)	Reserved		_	_	_	_		_	_	
(\$6A)	EICRA	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	90
(\$69)	Reserved	-	-	-	-	-	-	-	-	30
(\$68)	SPMCSR	SPMIE	RWWSB	_	RWWSRE	BLBSET	– PGWRT	– PGERS	_ SPMEN	289
(\$67)	Reserved			_	-	-	-	-		203
(\$66)	Reserved	_	_	_	_	_	_	_		
		_	_	_	PORTG4	PORTG3	PORTG2	PORTG1	PORTG0	89
	PORTG				1 1 0 11 04	1 011100	1 011102	i ontai	101100	03
(\$65) (\$64)	PORTG DDRG	_	_	-	DDG4	DDG3	DDG2	DDG1	DDG0	89





5. Register Summary (Continued)

Address	Neme	Bit 7	Dit C	Dit E	Dit 4	Dia 0	Dit 0	Dia 1	Bit 0	Derre
	Name	-	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1		Page
(\$62)	PORTF	PORTF7	PORTF6	PORTF5	PORTF4	PORTF3	PORTF2	PORTF1	PORTF0	89
(\$61)	DDRF	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0	89
(\$60) \$3F (\$5F)	Reserved SREG	-	_ Т	-	S	V	N	Z	C	10
\$3F (\$5F) \$3E (\$5E)	SPH	SP15	SP14	SP13	SP12	V SP11	SP10	SP9	SP8	13
\$3D (\$5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP9 SP1	SP0	13
\$3C (\$5C)	XDIV	XDIVEN	XDIV6	XDIV5	XDIV4	XDIV3	XDIV2	XDIV1	XDIVO	38
\$3B (\$5B)	RAMPZ	-	-	-	-	-	-	-	RAMPZO	14
\$3A (\$5A)	EICRB	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	91
\$39 (\$59)	EIMSK	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INTO	92
\$38 (\$58)	EIFR	INTF7	INTF6	INTF5	INTF4	INTF3	INTF	INTF1	INTF0	92
\$37 (\$57)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	110, 141, 162
\$36 (\$56)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	110, 143, 162
\$35 (\$55)	MCUCR	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE	33, 50, 63
\$34 (\$54)	MCUCSR	JTD	-	-	JTRF	WDRF	BORF	EXTRF	PORF	56, 257
\$33 (\$53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	106
\$32 (\$52)	TCNT0					unter0 (8 Bit)				109
\$31 (\$51)	OCR0				mer/Counter0 Ou	tput Compare Re	Ť.			109
\$30 (\$50)	ASSR	-	-	-	-	AS0	TCN0UB	OCROUB	TCR0UB	109
\$2F (\$4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	134
\$2E (\$4E)	TCCR1B	ICNC1	ICES1		WGM13	WGM12	CS12	CS11	CS10	137
\$2D (\$4D)	TCNT1H TCNT1L					unter Register Hig				139 139
\$2C (\$4C) \$2B (\$4B)	OCR1AH					unter Register Lo				139
\$2B (\$4B) \$2A (\$4A)	OCR1AL					compare Register	* ·			139
\$29 (\$49)	OCR1BH					compare Register	,			139
\$28 (\$48)	OCR1BL					Compare Register	0 /			139
\$27 (\$47)	ICR1H					Capture Register				140
\$26 (\$46)	ICR1L				•	Capture Register	8 ,			140
\$25 (\$45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	159
\$24 (\$44)	TCNT2		•	•	Timer/Co	unter2 (8 Bit)	•			161
\$23 (\$43)	OCR2			Tir	mer/Counter2 Ou	put Compare Re	gister			162
\$22 (\$42)	OCDR	IDRD/OCDR7	OCDR6	OCDR5	OCDR4	OCDR3	OCDR2	OCDR1	OCDR0	276
\$21 (\$41)	WDTCR	-	-	-	WDCE	WDE	WDP2	WDP1	WDP0	57
\$20 (\$40)	SFIOR	TSM	-	-	-	ACME	PUD	PSR0	PSR321	86, 111, 146, 231
\$1F (\$3F)	EEARH	-	-	-	-			ess Register High		30
\$1E (\$3E)	EEARL					s Register Low B	yte			30
\$1D (\$3D)	EEDR				EEPROM	Data Register				30
\$1C (\$3C)	EECR	-	-	-	-	EERIE	EEMWE	EEWE	EERE	30
\$1B (\$3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	87
\$1A (\$3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	87
\$19 (\$39) \$18 (\$38)	PINA PORTB	PINA7 PORTB7	PINA6 PORTB6	PINA5 PORTB5	PINA4 PORTB4	PINA3 PORTB3	PINA2 PORTB2	PINA1 PORTB1	PINA0 PORTB0	87 87
\$18 (\$38) \$17 (\$37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	87
\$16 (\$36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	87
\$15 (\$35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTCO	87
\$14 (\$34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	87
\$13 (\$33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	88
\$12 (\$32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	88
\$11 (\$31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	88
\$10 (\$30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	88
\$0F (\$2F)	SPDR				SPI Da	ta Register				173
\$0E (\$2E)	SPSR	SPIF	WCOL	-	-	-	-	-	SPI2X	173
\$0D (\$2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	171
\$0C (\$2C)	UDR0		1	1		Data Register	1		1	192
\$0B (\$2B)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	193
\$0A (\$2A)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	194
\$09 (\$29)	UBRROL	1.6-				Rate Register Lo				197
\$08 (\$28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACISO	231
\$07 (\$27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	245
\$06 (\$26)	ADCSRA	ADEN	ADSC	ADFR	ADC Data Ba	ADIE	ADPS2	ADPS1	ADPS0	247
\$05 (\$25) \$04 (\$24)	ADCH ADCL					gister High Byte				248 248
\$03 (\$23)	PORTE	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0	88
ψυυ (ψευ)	TORIL		TOTILO	1 OITL3	I UNIL4	TONIES	TOTTLE	TONILI	TOTTLU	00

5. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
\$02 (\$22)	DDRE	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0	88
\$01 (\$21)	PINE	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0	88
\$00 (\$20)	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 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 \$00 to \$1F only.





6. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND	LOGIC INSTRUCTION	8	l	1	
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 ← 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	$Rd \gets Rd \bullet K$	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	$Rd \leftarrow Rd \lor K$	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 ← \$FF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← \$00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \lor K$	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \bullet (\$FF - K)$	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	$Rd \leftarrow FF$	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 \leftarrow Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	$R1:R0 \leftarrow Rd x Rr$	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 \leftarrow Rd x 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	TIONS			•	·
RJMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2
IJMP		Indirect Jump to (Z)	$PC \leftarrow Z$	None	2
JMP	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	$PC \leftarrow PC + k + 1$	None	3
ICALL		Indirect Call to (Z)	$PC \leftarrow Z$	None	3
CALL	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	$PC \leftarrow STACK$	None	4
RETI		Interrupt Return	$PC \leftarrow STACK$	1	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC \leftarrow PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if $(SREG(s) = 0)$ then $PC \leftarrow PC+k + 1$	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC \leftarrow PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC \leftarrow PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC \leftarrow PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC \leftarrow PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC \leftarrow PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC \leftarrow PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC \leftarrow PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC \leftarrow PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N \oplus V= 0) then PC \leftarrow PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N \oplus V= 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	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	None	1/2

6. Instruction Set Summary (Continued)

Memonical Biolitic Operation Operation Flags #0000 No Seech Flammary Durabid F11-10 more C - FC + k+1 Nore 11/1 DMD k Seech Flammary Durabid F11-10 more C - FC + k+1 Nore 11/1 DMV MA, R More Researce Registres Nore Nore 1 DMV MA, R Coor Projects WOrd Bis List. Nore 1 DM MA, K Load Index and Decision R4 - K Nore 2 DL R4, K Load Index and Decision Nore 2 Nore 2 DL R4, K Load Index and Decision Nore + C Nore 2 2 DL R4, K Load Index and Decision Nore + C + 1 Nore 2 2 DL R4, K Load Index and Decision Nore + C + 1 Nore 2 2 DL R4, Z Load Index and Decision Nore + C + 1 Nore 2 2 DL R4, Z Load Index and Decis	BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC \leftarrow PC + k + 1	None	1/2
BIRE Number of Interrupt Detailed If 11 If Inter 02 + PC + k + 1 Nume 11/2 DATA TABASEE INSTRUCTIONS If 1 - 0 prom PC + PC + k + 1 Nome 11/2 DATA TABASEE INSTRUCTIONS If 1 - 0 prom PC + PC + k + 1 Nome 11/2 DATA TABASEE INSTRUCTIONS If 1 - 0 prom PC + PC + k + 1 Nome 11 DATA TABASEE INSTRUCTIONS If 4 - PC Nome 1 DATA TABASEE INSTRUCTIONS If 4 - PC Nome 1 DATA TABASEE INSTRUCTIONS If 4 - PC Nome 1 DATA TABASEE INSTRUCTIONS If 4 - PC Nome 1 DATA TABASEE INSTRUCTIONS If 4 - PC Nome 2 DATA TABASEE INSTRUCTIONS If 4 - PC Nome 2 DATA TABASEE INSTRUCTIONS If 4 - PC Nome 2 DATA TABASEE INSTRUCTIONS If 4 - PC Nome 2 DATA TABASEE INSTRUCTIONS If 4 - PC Nome 2 DATA TABASEE INSTRUCTIONS If 4 - PC Nome 2 DATA TABASEE INSTRUCTIONSTRUCTIONSTRUCTIONSTRUCTIONSTRUCTIONSTRUCTIONSTRUCTIONSTRUCTION	BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC \leftarrow PC + k + 1		1/2
BRD Is anoth Insung Disables II.1 = 0 (BR - 0 C - V - V + 1 Nove II.7 DAY TANSEGERS-INSUEDIONS Nove Selesen Registere R4 - RC Nove II.1 DAY RAP, RC Copy Register World R1 - Life - (N - Life) Nove II.1 LDI R4, KL Load Index on Post-Inc. R4 - KL Nove II.1 LDI R4, XL Load Index on Post-Inc. R4 - KL Nove II.2 LDI R4, XL Load Index on Post-Inc. R4 - KL Nove II.2 LDI R4, XL Load Index on Post-Inc. R4 - (N - V - (N - N -	Mnemonics	Operands	Description	Operation	Flags	#Clocks
DATA TABASEEN INSTRUCTION Norm Bal. Print Owner Bady and State S	BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC \leftarrow PC + k + 1	None	1/2
MOVRL /rMore Bosen RegisterN - rN - rNome1LDRL /rLode IntroductRd - KLode IntroductN - KNome1LDRL /rLode IntroductRd + ONomeNome2LDRL /rLode Introduct and Paylon.Rd + ONome2LDRL /rLode Introduct and Paylon.Nome7Nome2LDRL /rLode Introduct and Paylon.Nome2Nome2LDRL /rLode Introduct and Paylon.Nome2Nome2LDRL /rLode Introduct and Paylon.Rd + ONome2Nome2LDRL /rLode Introduct and Paylon.Rd + ONome2Nome2LDRL /rLode Introduct and Paylon.Rd + ONome2Nome2LDRL /rLode Introduct and Paylon.Note +ONome2Nome2LDRL /rLode Introduct and Paylon.Note +ONome2Nome2LDRL /rLode Introduct and Paylon.Note +ONome2Nome2LDRL /rLode Introduct and Paylon.Note +NNome2Nome2<	BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC \leftarrow PC + k + 1	None	1 / 2
MOWR4, R4Cogy Bagion WordR6 + R4None1LDR5 KLoad InfrademR4 - (X)None	DATA TRANSFER	INSTRUCTIONS				n
D.D. P6 K.X Load Indicat P6 ± - K. None 1 D.D. P6 X.X Load Indicet and Poelsine. Pd ± - (K), X + X + 1 None 2 D.D. P6 X.X Load Indicet and Poelsine. Pd ± - (K), X + X + 1 None 2 D.D. P6 X.Y Load Indicet and Poelsine. Pd ± - (Y), Y + 1 None 2 D.D. P6 X.Y Load Indicet and Poelsine. Pd ± - (Y), Y + 1 None 2 D.D. P6 X.Y Load Indicet and Poelsine. Pd = (Y), Y + 1 None 2 D.D. P6 X.Y Load Indicet and Poelsine. Pd = (Y) None 2 D.D. P6 X.Z Load Indicet and Poelsine. Pd = (Z) None 2 D.D. P6 X.Z Load Indicet and Poelsine. Pd = (Z) None 2 D.D. P6 X.Z Load Indicet and Poelsine. Pd = (Z) None 2 D.D. P6 X.Z Load Indicet and Poelsine. Pd = (Z) None 2 D.T. N.F. Size	MOV	Rd, Rr	Move Between Registers		None	1
D.D. Pd ₁ X, D. Load Index and Peel-ho. Pd - (D), X + X = 1. None 2.2 LD Rd, Y. Load Index and Peel-ho. X + X + 1. None 2.2 LD Rd, Y. Load Index and Peel-ho. Rd - (T), Y + Y + 1. None 2.2 LD Rd, Y. Load Index and Peel-ho. Y + Y + 1, Rd - (T) None 2.2 LD Rd, Y. Load Index and Peel-ho. Y + Y + 1, Rd - (T) None 2.2 LD Rd, Z. Load Index and Peel-ho. Pd - (D, Y + Y + 1, Rd - (T) None 2.2 LD Rd, Z. Load Index and Peel-ho. Pd - (D, Z + Z + 1, Rd + (D) None 2.2 LD Rd, Z. Load Index and Peel-ho. Pd - (D, Z + Z + 1, Rd + (D) None 2.2 LDS Rd, X. Load Index and Peel-ho. Pd - (D, C + X + 1 + 1) None 2.2 LDS Rd, X. Load Index and Peel-ho. Pd - (D, -R' + X + 1) None 2.2 LDS Rd, X. Store Index and Peel-ho. X + X + 1, Q, -R' None 2.2	MOVW	Rd, Rr	Copy Register Word	$Rd+1:Rd \leftarrow Rr+1:Rr$	None	1
DD P61 x X Load Indices and Peebe. Y + X + 1 + (P) None I > 2 DL0 P61 y X Load Indices and Peebe. P4 + (P) + Y + 1 None I > 2 DL0 P61 y X Load Indices and Peebe. P4 - (P) + Y + 1 None I > 2 DL0 R61 y X Load Indices and Peebe. Y = Y + 1, R1 - (P) None I > 2 DL0 R61 y X Load Indices and Peebe. Y = Y + 1, R1 - (P) None I > 2 DL0 R61 y X Load Indices and Peebe. R1 - (P) None I > 2 DL0 R61 y X Load Indices and Peebe. Z + Z + 1, R1 - (D) None I > 2 DL0 R61 y X Load Indices and Peebe. Z + Z + 1, R1 - (D) None I > 2 DS R1 x R Load Indices and Peebe. Q + x + X + (A) None I > 2 ST X, R Store Indice and Preebe. Q + x + X + (A) None I > 2 ST Y, R Store Indice and Preebe. Q + x + X + (A) None I > 2	LDI					
D.D. Rd. × Load Indirect and ProDec. X × 1, 164 - (b), None 2 D.D. Rd, Y Load Indirect and Posi-fra. Rd - (Y), Y - Y - 1 None 2 D.D. Rd, Y Load Indirect and Posi-fra. Rd - (Y) - (Y - Y - 1) None 2 D.D. Rd, Y Load Indirect and Posi-fra. Rd - (Y - q) None 2 D.D. Rd Z Load Indirect and Posi-fra. Rd - (Z) - Z-11 None 2 D.D. Rd Z Load Indirect and Posi-fra. Rd - (Z) - Z-11 None 2 D.D. Rd Z Load Indirect and Posi-fra. Rd - (Z) - Z-11 None 2 D.D. Rd Z Load Indirect and Posi-fra. Rd - (Z) - Z-11 None 2 D.D. Rd Z Load Indirect and Posi-fra. Q) - Fra None 2 D.D. Rd Z Load Indirect and Posi-fra. Q) - Fra None 2 S.T. X, Fr Store Indirect and Posi-fra. (Y) - Fr None 2 S.T. X, Fr<						
D.D. Rd. Y. Load indicat Description Pair (Y) Y. Y. S1 None P.2 LD Rd. Y. Load indicet and Pre-Dec. Y. Y. Y. S1, Bi-4: (Y) None P.2 LD Rd. Y. Load indicet Mich Dipacement Pd - (2) None P.2 LD Rd. Y. Load indicet Mich Dipacement Pd - (2) None P.2 LD Rd. Y. Load indicet And Pre-Dec. Z - X. 1, Rd - (2) None P.2 LD Rd. Y. Load indicet And Pre-Dec. Z - X. 1, Rd - (2) None P.2 LD Rd. Y. R Load Indicet And Pre-Dec. Z - X. 1, Rd - (2) None P.2 LD Rd. Y. R Soon Indicet and Pre-Dec. None None P.2 ST X. Fr. Soon Indicet and Pre-Dec. Y - Y. Y. Y. 1 None P.2 ST Y. Fr. Soon Indicet and Pre-Dec. Y - Y. Y. Y. 1 None P.2 ST Y. Fr. Soon Indicet and Pre-Dec. Y - Y. 1, (Y) - Fr. None P.2 <t< td=""><td></td><td></td><td></td><td></td><td></td><td></td></t<>						
D.D. Rd. Y Load indicat and Prop. Dr. Pict. P. 1184 (?) None 2 LDD Rd. Y Load indicat with Daplacement Rd. Y (Y + Q) None 2 LDD Rd. Z Load indicat with Daplacement Rd. Y (Y + Q) None 2 LDD Rd. Z Load indicat and Prop. Dr. Rd. Y (Z - Z-1) None 2 LD Rd. Z Load indicat and Prop. Dr. Z - Z-1. Rd. Pd. (Z) None 2 LD Rd. Z Load indicat and Prop. Dr. Z - Z-1. Rd. Pd. (Z) None 2 LD Rd. X Load indicat and Prop. Dr. Rd. Y - X. X. None None 2 LD Rd. X Load indicat and Prop. Dr. X. Y. Y. 1 None 2 ST X. Rr Store indicat and Prop. Dr. Y - Y. Y. 1 None 2 ST Y. Rr Store indicat and Prop. Dr. Y - Y. Y. 1 None 2 ST Y. Rr Store indicat and Prop. Dr. Y - Y. Y. 1 None 2 ST Y. Rr						
DDRd. + YLoad indicat and Pre-Dec.Y + Y. 1, Bd - (Y)None2LDRd Y-qLoad indicat MD ExplorementBd + (Y)None2LDRd ZLoad indicat and Pre-Dec.Log (Z) + Z-11None2LDRd ZLoad indicat and Pre-Dec.Z + Z-1, Bd - (Z)None2LDRd Z-41Load indicat and Pre-Dec.Z + Z-1, Bd - (Z)None2LDRd Z-41Load indicat and Pre-Dec.Z + Z-1, Bd - (Z)None2LDRd X-41Load indicat and Pre-Dec.D + R-2None2STX, RrStore indicat and Pre-Dec.(X + R/X, C) + IPNone2STY, RrStore indicat and Pre-Dec.(X + N, Y + X + 1)None2STY, RrStore indicat and Pre-Dec.(Y + R/Y, V + Y + 1)None2STY, RrStore indicat and Pre-Dec.(Y + Q + RNone2STY, RrStore indicat and Pre-Dec.(Y + Q + RNone2STY, RrStore indicat and Pre-Dec.(Y + Q + RNone2STY, RrStore indicat and Pre-Dec.(Z + R/Y + Y + 1)None2STY, RrStore indicat and Pre-Dec.(Z + R/Y + Y + 1)None2STZ, RrStore indicat and Pre-Dec.(Z + R/Y + Z + 1)None2STZ, RrStore indicat and Pre-Dec.(Z + R/Y + Z + 1)None2STZ, RrStore indicat and Pre-Dec.		,				
DDD FAX+n Load Indired Pat-(Y - q) None 2 DD Fd,Z Load Indired and Pra-ben. Fd - (Z), Z - 2.1 None 2 DD Fd,Z Load Indired and Pra-ben. Z - 2.1. Rbt - (Z) None 2 DD Fd,Z Load Indired and Pra-ben. Z - 2.1. Rbt - (Z) None 2 LDD Fd,Z Load Direct from SFAM Fd - Fd,Z - q) None 2 ST X, Fr Store Indirect and Post-fnc. 100 - FR, X - X + 1 None 2 ST X, Fr Store Indirect and Post-fnc. 100 - FR, X - X + 1 None 2 ST Y, FR Store Indirect and Post-fnc. 100 - FR None 2 ST Y, FR Store Indirect and Post-fnc. 100 - FR None 2 ST Y, FR Store Indirect and Post-fnc. 100 - FR None 2 ST Z, FR Store Indirect and Post-fnc. 100 - FR None 2 ST Z, FR Store Indirect and Post-fnc.<		-				
D.D. Pid. 2 Load Indiced and Post-Inc. Pid(Z). None P D.D. Pid.2. Load Indiced and Post-Inc. ZZ.1.Bd.+(Z) None P D.D. Pid.2. Load Indiced and Post-Inc. ZZ.1.Bd.+(Z) None P D.D. Pid.2. Load Indiced and Post-Inc. Pid(X) None P D.D. Pid.2. Load Indiced and Post-Inc. Pid(X) None P ST X. fit Store Indicet and Post-Inc. (X) + Fit.X + X + 1 None P ST X. fit Store Indicet and Post-Inc. (Y) + Fit.X + X + 1 None P ST Y. fit Store Indicet and Post-Inc. (Y) + Fit.X + X + 1 None P ST Y. fit Store Indicet and Post-Inc. (Y) + Fit.X + Y + 1 None P ST Y. fit Store Indicet and Post-Inc. (Z) - fit.Y + Y + 1 None P ST Z. fit Store Indicet and Post-Inc. (Z) - fit.Y + X + 2 + 1 None P ST						
LD Fd.2.* Load Indiced and Pre-Dec. P.2. = 2.1. Rd+ (-2) None P.2. LDD Fd.3.*Q Load Indiced and Pre-Dec. Z. = 2.1. Rd+ (-2) None P.2. LDD Fd.4.*Q Load Under twin Digbacement Rd+ (A) None P.2. LDS Fd.4.* Load Direct from SRAM Rd+ (A) None P.2. ST X, Rr Store Indirect and Post-Inc. (X) + fr, X += X + 1 None P.2. ST X, Rr Store Indirect and Post-Inc. (X) + fr, X += X + 1 None P.2. ST Y, Rr Store Indirect and Post-Dec. Y + Y + Y + 1 None P.2. ST Y, Rr Store Indirect and Post-Dec. Y + Y + Y + 1 None P.2. ST Y, Rr Store Indirect and Post-Dec. Z - 2. + (Y) + Fr None P.2. ST Z, Rr Store Indirect and Post-Dec. Z - 2. + 1. None P.2. ST Z, Rr Store Indirect and Post-Dec. Z - 2. + 1.0. None P.2. <t< td=""><td></td><td></td><td></td><td></td><td></td><td></td></t<>						
DD Bd.2-q Load Indiced and Pro-Dec. 2 + 2 + 1, dt - (2) None 9.2 DD0 Bd.2-q Load Indiced with Dipplacement Bd + (2) None 9.2 DS R.4 Load Direct two BRAM Bd + (9) None 9.2 ST X, Rr Store Indiced and Pa-Dec. (0) + Rr None 9.2 ST X, Rr Store Indiced and Pa-Dec. (V) + Rr None 2.2 ST Y, Rr Store Indiced and Pa-Dec. (V) + Rr None 2.2 ST Y, Rr Store Indiced and Pa-Dec. (V) + Rr None 2.2 ST Y, Rr Store Indiced and Pa-Dec. (V + V + 1) None 2.2 ST Z, Rr Store Indiced and Pa-Dec. (Z) + Rr, Z + Z + 1 None 2.2 ST Z, Rr Store Indiced and Pa-Dec. (Z + 1, Q) + Rr None 2.2 ST Z, Rr Store Indiced and Pa-Dec. (Z + 1, Q) + Rr None 2.2 ST Z, Rr Store Indiced and Pa-Dec.<		-				
LDD Rd + Q + Q + Q + Q + Q + Q + Q + Q + Q +						
LDS Fld k Load Direct from SRAM Rd - (a) None 2 ST X, Hr Store Indirect and Post-Inc. (b) (- Rr None 2 ST X, R Store Indirect and Pro-Dec. X - X + 1, (b) - Nr None 2 ST -X, Rr Store Indirect and Pro-Dec. X - X + 1, (b) - Nr None 2 ST -Y, Rr Store Indirect and Pro-Dec. Y - K + 1 None 2 ST -Y, Rr Store Indirect and Pro-Dec. Y - K + 1 None 2 ST -Y, Rr Store Indirect and Pro-Dec. Y - K + 1 None 2 ST -Y, Rr Store Indirect and Pro-Dec. Y - K + 1 None 2 ST Z, Rr Store Indirect and Pro-Dec. Z - 2 + 1 None 2 ST Z, Rr Store Indirect and Pro-Dec. Z - 1, (2) - Fr None 2 ST Z, Rr Store Indirect and Pro-Dec. Z - 1, (2) - Fr None 2 ST Z, Rr Store Indirect and Pro-D	LDD					
ST X, Rr Store Indirect and Post-Inc. (D) + Rr None 2 ST X, Rr Store Indirect and Post-Inc. (D) + Rr None 2 ST -X, Rr Store Indirect and Pre-Dec. X - K X + 1, (D) + Rr None 2 ST Y, Rr Store Indirect and Post-Inc. (M + Rr Y + Y + 1 None 2 ST Y, Rr Store Indirect and Post-Inc. (Y + N' + 1) None 2 ST Y, Rr Store Indirect and Post-Inc. (Y + N' + 1) None 2 ST Z, Rr Store Indirect with Doptacement (Z + Rr None 2 ST Z, Rr Store Indirect with Doptacement (Z + C + 1, 1) None 2 STD Z, Rr Store Indirect and Post-Inc. (Z + C + 1, 1) None 2 STD Z, Rr Store Indirect and Post-Dec. Z + 2 + 1, (D) + Rr None 2 STD Z, Rr Store Direct to SPAM (D + C) None 3 STD Z, Rr S	LDS					
ST X, βr Stree Indirect and Peachen. (μ) = Pt, X × 1, 1 None 2 ST ×, Rr Stree Indirect and Peachen. X + X + 1, (λ) ← Pt None 2 ST Y, Rr Store Indirect and Peachen. (Y) + Pt, Y + Y None 2 ST Y, Rr Store Indirect and Peachen. (Y) + Pt, Y + Y None 2 ST Y, Rr Store Indirect and Peachen. (Y + q) + Pt None 2 ST Z, Rr Store Indirect and Peachen. (Y + q) + Pt None 2 ST Z, Rr Store Indirect and Peachen. (Z) + Pt None 2 ST Z, Rr Store Indirect and Peachen. Z(+ Pt = Z + 1. None 2 ST Z, Rr Store Indirect and Peachen. Z(+ Pt = T None 2 ST Z, Rr Store Indirect and Peachen. Z(+ Z + 1, Z) + Pt None 2 ST Z, Rr Store Indirect and Peachen. Z(+ Z + 1, Z) + Pt None 2 ST LadP Peachen Mem	ST					
ST ····································	ST					
STV, RrStore Indirect and Post-Inc.(Y) $+$ Rr, Y $+$ Y + 1None2STY+, RrStore Indirect and Post-Dc.(Y) $+$ Rr, Y $+$ Y + 1None2ST-Y, RrStore Indirect and Post-Dc.Y $+$ Y + 1, (Y) $-$ RrNone2STZ, RrStore Indirect and Post-Dc.(Y $+$ Q) $+$ RrNone2STZ, RrStore Indirect and Post-Dc.(Q) $+$ RrNone2STZ, RrStore Indirect and Post-Dc.(Q) $+$ RrNone2STZ-, RrStore Indirect and Post-Dc.(Z $+ Z + 1, (Z) - RrNone2STZ-, RrStore Indirect and Post-Dc.(Z + Z + 1, (Z) - RrNone2STZ-, RrStore Indirect with Displacement(Z + Q + RrNone2STK, RrStore Direct Do SPAM(h) - RrNone3LPMLoad Porgram MemoryRd - (Z)None3LPMRd, ZLoad Porgram MemoryRd - (Z)None3LPMRd, ZLoad Porgram MemoryRd - (Z)None3Store Program Memory and Post-IncRd - (Z) - RT.None3LPMRd, Z-Extended Load Porgram MemoryRd - (RAMPZ2)None3Store Program Memory and Post-IncRd - (RAMPZ2)None3LPMRd, Z-Extended Load Porgram MemoryRd - (RAMPZ2)None3LPMRd, Z-Extended Load Porgram MemoryRd - (RAMPZ2)None$	ST		Store Indirect and Pre-Dec.		None	2
ST·· \ / R'Store Indirect and Pre-Dec. $Y \leftarrow Y \cdot (1) \leftarrow Rr$ None2STDY \leftarrow q, RrStore Indirect with Displacement $(Y + q) \leftarrow Rr$ None2STZ, RrStore Indirect and Pre-Dec. $(Z) \leftarrow Rr$ None2STZ, RrStore Indirect and Pre-Dec. $(Z + Z + 1, Z) \leftarrow Rr$ None2STDZ-q, RrStore Indirect and Pre-Dec. $(Z + Z + 1, Z) \leftarrow Rr$ None2STDZ-q, RrStore Indirect with Displacement $(Z + Q) \leftarrow Rr$ None2LPMI.oad Program MemoryR0 $\leftarrow (Z)$ None3LPMRd, ZLoad Program MemoryR0 $\leftarrow (Z)$ None3LPMRd, ZLoad Program MemoryRd $\leftarrow (Z)$ None3LPMRd, ZLoad Program MemoryRd $\leftarrow (Z)$ None3LPMRd, ZLoad Program MemoryRd $\leftarrow (Z)$ None3LPMRd, ZExtended Load Program Memory and Post-IncRd $\leftarrow (Z)$ None3LPMRd, ZExtended Load Program MemoryRd $\leftarrow (RAMPZZ)$ None3SMStore Program MemoryRd $\leftarrow P$ None3LPMRd, PIn PortRd $\leftarrow P$ None3LPMRd, PIn PortNone2NNRdPop Register forn StackStack $\leftarrow Rr$ None2NNRdPop Register forn StackRd $\leftarrow P$ None2LPMRdLogical Stift LiftRd(p) $\leftarrow 1$ No	ST	Y, Rr	Store Indirect	$(Y) \leftarrow Rr$	None	2
STD Y+qRr Store Indirect with Digplacement (Y+q) ← Rr None 2 ST Z, Rr Store Indirect and Post-Inc. (Z) ← Rr. (Z) ← Rr. None 2 ST Z, Rr Store Indirect and Post-Inc. (Z) ← Rr. None 2 ST Z, Rr Store Indirect with Displacement (Z) ← Rr. None 2 STS K, Rr Store Direct to STAM (b) ← Rr None 2 STM Load Program Memory Rd ← (Z) None 3 LPM Load Program Memory Rd ← (Z) None 3 LPM Rd, Z Load Program Memory Rd ← (Z) None 3 LPM Rd, Z Extended Load Program Memory Rd ← (RAMPZZ) None 3 Store Store Program Memory Rd ← (RAMPZZ), RAMPZ ← RAMPZZ-1 None 1 NDT Rd, P In Port Rd Poptam Memory (Z) ← R1R0 None 1 NDT Rd, Z Extended Load Program Memory Rd ← RAMPZZ) <	ST	Y+, Rr	Store Indirect and Post-Inc.	$(Y) \leftarrow Rr, Y \leftarrow Y + 1$	None	2
STZ. frStore indirect(Z) - RrNone2STZ., RrStore indirect and Post-Inc.(Z) - Rr, Z + Z + 1None2STZ., RrStore indirect and Pre-Dec.Z + Z - 1, (Z) - RrNone2STDZtq, RrStore indirect tand Pre-Dec.Z + Z - 1, (Z) - RrNone2STDZtq, RrStore indirect tand Pre-Dec.Z + Z - 1, (Z) - RrNone2STSK. RrStore indirect tand Pre-Dec.Z + Z - 1, (Z) - RrNone2LPMLoad Program MemoryR0 - (Z)None3LPMRd, ZLoad Program Memory and Post-IncRd - (Z), Z - Z + 1None3LPMRd, ZLoad Program Memory and Post-IncRd - (RAMPZZ)None3ELPMRd, Z +Extended Load Program MemoryRd - (RAMPZZ)None3SPMStore Program Memory and Post-IncRd - (RAMPZZ), RAMPZZ + RAMPZZ-1None3SPMStore Program Memory and Post-IncRd - PNone1OUTP, RrOur PortRd - PNone1OUTP, RrOur PortSTACKNone2POPRdIn PortRdSTACK + RrNone2POPRdLogical Shift Int/D Register on StackRd - STACKNone2SIP,bSet Bit In I/D RegisterI/O(P,b) - 0None2LSLRdLogical Shift RightRd(n) - Rd(n), Rd(0) - 0Z,C,N,V1RD	ST	- Y, Rr	Store Indirect and Pre-Dec.	$Y \leftarrow Y - 1$, $(Y) \leftarrow Rr$	None	2
STZ, RrStore Indirect and Post-Inc. $(Z) \leftarrow Rr, Z \leftarrow Z + 1$ NonePSTZ, RrStore Indirect and Pre-Dec. $Z \leftarrow 2 + 1, (Z) \leftarrow Rr$ NonePSTDZ <q, rr<="" td="">Store Indirect and Pre-Dec.$(Z + q) \leftarrow Rr$NonePSTSK, RrStore Direct to SRAM$(q) \leftarrow Rr$NonePSTSK, RrStore Direct to SRAM$(q) \leftarrow Rr$NonePLPMLoad Program MemoryR0 $\leftarrow (Z)$None3LPMRd, ZLoad Program MemoryRd $\leftarrow (Z), Z \leftarrow Z + 1$None3LPMRd, ZLoad Program MemoryRd $\leftarrow (Z), Z \leftarrow Z + 1$None3LPMRd, ZExtended Load Program MemoryRd $\leftarrow (RAMPZ-Z)$None3SPMRd, ZExtended Load Program MemoryRd $\leftarrow (RAMPZ-Z)$None3SPMRd, ZExtended Load Program MemoryRd $\leftarrow (RAMPZ-Z)$None3SPMRd, PIn PortRd $\leftarrow P$None3DUTP, RrOut PortRd $\leftarrow P$None1DUTP, RrPush Register on StackSTACKNone2PORdPop Register from StackSTACKNone2SIP.JSet Bit in UO RegisterUO(P.b) $\leftarrow 1$None2LSLRdLogical Shift LightRd(n) $\leftarrow Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 2$Z,C,NV1RORdRotate Left Through CarryRd(n) $\leftarrow Rd(n+1), Rd(0)$Z,C,NV1RORd<td< td=""><td>STD</td><td>Y+q,Rr</td><td>Store Indirect with Displacement</td><td>(Y + q) ← Rr</td><td>None</td><td>2</td></td<></q,>	STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
STZ, RrStore Indirect and Pre-Dec.Z - Z - 1, (Z) \leftarrow RrNonePSTDZ + q, RrStore Indirect and Pre-Dec.(Z + Q + RrNone2STSK, RrStore Direct to SRAM(k) \leftarrow RrNone2LPMLead Program MemoryR0 \leftarrow (Z)None3LPMRd, ZLoad Program MemoryRd \leftarrow (Z), Z \leftarrow Z+1None3LPMRd, Z+Load Program MemoryRd \leftarrow (C, RAMPZ-Z)None3LPMRd, ZExtended Load Program MemoryRd \leftarrow (RAMPZ-Z)None3ELPMRd, ZExtended Load Program MemoryRd \leftarrow (RAMPZ-Z)None3SPMStore Program MemoryRd \leftarrow (RAMPZ-Z)None3SPMStore Program MemoryRd \leftarrow (RAMPZ-Z), RAMPZ-Z \leftarrow RAMPZ-Z+1None3DUTP, RrOut PortPd \leftarrow RAMPZ-Z)None1OUTP, RrOut PortPd \leftarrow RNone2POPRdPop Register from StackSTACK \leftarrow RrNone2PDRdLogical Shift RightRd(n+1) \leftarrow Rd(n), Rd(n) \leftarrow Rd(n), Rd(n) \leftarrow 2, C, NV1LSRdLogical Shift RightRd(n+1) \leftarrow Rd(n), Rd(n) \leftarrow Rd(n), Rd(n) \leftarrow 2, C, NV1LSRdLogical Shift RightRd(n+1) \leftarrow Rd(n), Rd(n) \leftarrow Rd(n), Rd(n) \leftarrow 2, C, NV1LSRdLogical Shift RightRd(n+1) \leftarrow Rd(n), Rd(n) \leftarrow Rd(n+1), Rd(n)11LSRdLogical Shift Right	ST	Z, Rr	Store Indirect	$(Z) \leftarrow Rr$	None	2
STDZ+q,RrStore Indirect with Displacement $(Z + q) \leftarrow Rr$ None2STSk, RrStore Direct to SRAM(k) $\leftarrow Rr$ None2LPMLoad Program MemoryR0 \leftarrow (2)None3LPMRd, ZLoad Program MemoryRd \leftarrow (2), $Z - Z + 1$ None3LPMRd, Z+Load Program Memory and Post-IncRd \leftarrow (2), $Z - Z + 1$ None3LPMRd, Z+Extended Load Program MemoryRd \leftarrow (RAMPZ.2)None3ELPMRd, ZExtended Load Program Memory and Post-IncRd \leftarrow (RAMPZ.2)None3SPMStore Program Memory and Post-IncRd \leftarrow (RAMPZ.2, RAMPZ.2 \leftarrow RAMPZ.2+1None3SPMStore Program MemoryRd \leftarrow PNone1UTP, RrOut PortRd \leftarrow PNone1PUSHRrPush Rogister no StackStACK \leftarrow RrNone2POPRdPop Register from StackStACK \leftarrow RrNone2SB1P,bClear Bit in UO RegisterUO(P,b) \leftarrow 1None2SL1RdLogical Shift RightRd(n) \leftarrow Rd(n+1), Rd(n) \leftarrow 0Z,C,N,V11LSLRdLogical Shift RightRd(n) \leftarrow Rd(n+1), Rd(n) \leftarrow Rd(n)Z,C,N,V11RGRRdLogical Shift RightRd(n) \leftarrow Rd(n+1), Rd(n) \leftarrow Rd(n)Z,C,N,V11RGRRdArithmetic Shift RightRd(n) \leftarrow Rd(n+1), Rd(n) \leftarrow Rd(n)Z,C,N,V11RGRRdStore from Register for T<	ST		Store Indirect and Post-Inc.	$(Z) \leftarrow Rr, Z \leftarrow Z + 1$	None	2
STSk, Rr Store Direct to SRAM(k) $\leftarrow Rr$ None2LPMLoad Program MemoryR0 \leftarrow (2)None3LPMRd, ZLoad Program Memory and Post-IncR0 \leftarrow (2) Z - Z+1None3LPMRd, Z+Load Program Memory and Post-IncR0 \leftarrow (2) Z - Z+1None3ELPMRd, ZExtended Load Program MemoryR0 \leftarrow (RAMPZ2)None3ELPMRd, ZExtended Load Program MemoryR0 \leftarrow (RAMPZ2, RAMPZ2, TAMPZ2+None3SIDPStore Program Memory and Post-IncRd \leftarrow (RAMPZ2, RAMPZ2, TAMPZ2, TAMPZ2+None3SPMStore Program Memory(2) \leftarrow R1:R0None1OUTP, RrOut PortRd \leftarrow (RAMPZ2, RAMPZ2, TAMPZ2+None3PUSHRrPush Register on StackSTACK \leftarrow RrNone1OUTP, RrOut PortRd \leftarrow STACK \leftarrow RrNone2PDPRdPop Register from StackSTACK \leftarrow RrNone2SIIP,bSet Bit In I/O Register $UO(P,b) \leftarrow 0$ None2CB1P,bClear Bit In I/O Register $UO(P,b) \leftarrow 0$ None2SISRdLogical Shift LeftRd(n) \leftarrow Rd(n+1), \leftarrow Rd(n)Z,CN,V11ROLRdRotate Left Through CarryRd(n) \leftarrow Rd(n+1), \leftarrow Rd(n)Z,CN,V11ROLRdRdtate Rght Through CarryRd(n) \leftarrow Rd(n+1), \leftarrow Rd(n)Z,CN,V11RORRdSuber from Register to TT	ST					
LPMNoneNone3LPMRd, ZLoad Program MemoryRd $-(Z)$ None3LPMRd, Z-Load Program MemoryRd $-(Z)$ None3ELPMExtended Load Program MemoryRd $-(Z)$ None3ELPMRd, ZExtended Load Program MemoryRd $-(RAMP2Z)$ None3ELPMRd, ZExtended Load Program MemoryRd $-(RAMP2Z)$ None3SPMRd, Z-Extended Load Program Memory and Post-IncRd $-(RAMP2Z)$, RAMP2Z \leftarrow RAMP2Z \leftarrow None3SPMStore Program Memory(Z) \leftarrow R1:R0None3SPMTot ProfRd $-Q$ None1OUTP, RrOut PortRd \leftarrow RAMP2Z, RAMP2Z \leftarrow RAMP2Z, NAMP2Z \leftarrow RAMP2Z, RAMP2Z, NAMP2Z \leftarrow RAMP2Z, RAMP2Z, NAMP2Z \leftarrow RAMP2Z, NAMP2Z, NAM						
LPMRd, ZLoad Program MemoryRd + (Z)None3LPMRd, Z+Load Program Memory and Post-IncRd - (Z), Z + (Z+1)None3ELPMExtended Load Program MemoryR0 - (RAMPZZ)None3BLPMRd, ZExtended Load Program MemoryRd + (RAMPZZ), RAMPZZ + RAMPZZ+1None3BLPMRd, Z+Extended Load Program Memory and Post-IncRd + (RAMPZZ), RAMPZZ + RAMPZZ+1None3Store Program Memory(2) - R1:R0None1001INRd, PIn PortRd - PNone11OUTP, RrOut PortP - RrNone11OUTP, RrOut PortP - RrNone22POPRdPop Register from StackSTACK + RrNone22POPRdPop Register from StackRd - STACKNone22StatRdPop Register from StackRd - Aff(A) - Aff(A) - Aff(A) - Aff(A)221StatRdLogical Shift LeftRd (P) - 0None221ROLRdRotate Left Through CarryRd (P) - C-Rd(n+1) - Rd(n), Rd(0) - 0Z, C, N, V1ROLRdRotate Left Through CarryRd (P) - C-Rd(n+1) - Rd(n), C-Rd(n)Z, C, N, V1RORRdArithmetic Shift RightRd(n - Rd(n+1), R-Rd(n))Z, C, N, V1RORRdArithmetic Shift RightRd(n - Rd(n+1), Rd(0) - 0Z, C, N,		k, Rr				
LPMRd, Z+Load Program Memory and Post-IncRd $\leftarrow (2), Z \leftarrow Z+1$ None3ELPMExtended Load Program MemoryRd $\leftarrow (RAMPZ.Z)$ None3BLPMRd, ZExtended Load Program MemoryRd $\leftarrow (RAMPZ.Z)$ None3BLPMRd, Z+Extended Load Program Memory and Post-IncRd $\leftarrow (RAMPZ.Z)$, RAMPZ.Z + RAMPZ.Z+1None3SPMStore Program MemoryRd $\leftarrow (RAMPZ.Z)$, RAMPZ.Z + RAMPZ.Z+1None3SPMStore Program MemoryRd $\leftarrow P$ None1QUTP, RrOut PortRd $\leftarrow P$ None1QUTP, RrOut PortRd $\leftarrow P$ None2POPRdPopegister from StackRd $\leftarrow STACK + Rr$ None2BIP,bSet Bit in I/O RegisterI/O(P,b) $\leftarrow 1$ None2CBIP,bClear Bit in I/O RegisterI/O(P,b) $\leftarrow 0$ None2SILSRdLogical Shift RightRd(n+1) $\leftarrow Rd(n), Rd(0) \leftarrow 0$ Z,C,N,V1ROLRdRotate Left Triough CarryRd(0) $\leftarrow CRd(n+1) \leftarrow Rd(n), C-Rd(0)$ Z,C,N,V1RORRdRotate Right Through CarryRd(0) $\leftarrow CRd(n+1) \leftarrow Rd(n), C-Rd(0)$ Z,C,N,V1SWAPRdState from StackRd(3:0) $\leftarrow Rd(n+1) \leftarrow Rd(n), Rd(n+1) \leftarrow Rd(n), C-Rd(0)$ Z,C,N,V1RORRdAnthemics Shift RightRd(0) $\leftarrow CRd(n+1) \leftarrow Rd(n+1) \leftarrow Rd(n)$ X,C,N,V1RORRdState Left Triough CarryRd(3:0) $\leftarrow Rd(n+1) \leftarrow Rd(n+1) \leftarrow Rd(n)$ X,C,N,V						
ELPMNoneSELPMRd, ZExtended Load Program MemoryRd \leftarrow (RAMP2.Z)None3ELPMRd, ZExtended Load Program MemoryRd \leftarrow (RAMP2.Z)None3SPMRd, Z+Extended Load Program Memory and Post-IncRd \leftarrow (RAMP2.Z), RAMP2.Z \leftarrow RAMP2.Z+1None3SPMStore Program Memory(Z) \leftarrow R1:R0None1INRd, PIn PortRd \leftarrow PNone1OUTP, RrOut PortP \leftarrow RrNone2POPRdPop Register on StackSTACK \leftarrow RrNone2POPRdPop Register from StackSTACK \leftarrow RrNone2POPRdLogical Shift LeftV(D(P,b) \leftarrow 1None2SBIP,bSet Bit in I/O RegisterI/O(P,b) \leftarrow 0None2CBIP,bClear Bit in I/O RegisterI/O(P,b) \leftarrow 0None2SBIRdLogical Shift LeftRd(n \leftarrow Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0Z,C,NV1LSLRdLogical Shift RightRd(n) \leftarrow Rd(n+1), C-Rd(n), C-Rd(n)Z,C,NV1SNPRdRdRotate Left Through CarryRd(0) \leftarrow Rd(n+1), C-Rd(n), C-Rd(n)Z,C,NV1SNPRdStarg SetSREG(s) \leftarrow 1SREG(s)1SWPRdSwap NibblesSREG(s) \leftarrow 1SREG(s)1SWPRdStarg Set Register 1Rd(0) \leftarrow Rd(n+1), C-Rd(n), C-Rd(n)Z,C,NV1SWAPRdStarg						
ELPMRd, ZExtended Load Program MemoryRd \leftarrow (RAMPZ:Z)None3ELPMRd, Z+Extended Load Program Memory and Post-IncRd \leftarrow (RAMPZ:Z), RAMPZ:Z+None3SPMStore Program Memory(Z) \leftarrow R1:R0None1NRd, PIn PortRd \leftarrow PNone1OUTP, RrOut PortP \leftarrow RrNone1PUSHRrPop Register on StackSTACK \leftarrow RrNone2POPRdPop Register from StackRd \leftarrow STACK \leftarrow RrNone2BT AND BIT-TEST INSTRUCTIONSSet Bit in I/O RegisterI/O(P,b) \leftarrow 1None2CBIP,bSet Bit in I/O RegisterI/O(P,b) \leftarrow 1None2LSLRdLogical Shift LeftRd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0Z,C,N,V1LSRRdLogical Shift RightRd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0Z,C,N,V1ROLRdRotate Right Through CarryRd(7), \leftarrow Rd(n+1), C=Rd(7)Z,C,N,V1RORRdArithmetic Shift RightRd(n) \leftarrow Rd(n+1), n=0.6Z,C,N,V1SWAPRdSwap NibblesRd(3:0) \leftarrow Rd(7:4), Rd(7:4) \leftarrow Rd(0:0)None1SWAPRdShita eff Through CarryREG(s) \leftarrow 1SREG(s)1SWAPRdSwap NibblesRd(3:0) \leftarrow Rd(7:4), Rd(7:4) \leftarrow Rd(0:0)None1SWAPRdSwap NibblesRd(3:0) \leftarrow Rd(7:4), Rd(7:4) \leftarrow Rd(0:0)None1BET\$Flag SetSREG(s) \leftarrow 1 </td <td></td> <td>Rd, Z+</td> <td></td> <td></td> <td></td> <td></td>		Rd, Z+				
ELPMRd, Z+Extended Load Program Memory and Post-IncRd \leftarrow (RAMPZ.Z), RAMPZ.Z \leftarrow RAMPZ.Z+1None3SPMStore Program Memory(Z) \leftarrow R1:R0None-QUTP, RIn PortRd \leftarrow PNone1QUTP, RrOut PortP \leftarrow RrNone1PUSHRrPush Register on StackSTACK \leftarrow RrNone2POPRdPop Register from StackSTACK \leftarrow RrNone2BIT AND BIT-TEST INSTRUCTIONSSet Bit In /O Register $VO(P,b) \leftarrow 1$ None2CBIP,bSet Bit In /O Register $VO(P,b) \leftarrow 0$ None2CBIP,bGlaar Bit In I/O Register $VO(P,b) \leftarrow 0$ None2CBIP,bClaar Bit In I/O Register $VO(P,b) \leftarrow 0$ None2CBIRdLogical Shift RightRd(n+1), Rd(r) $\leftarrow 0$ Z,C,N,V1ROCRdRotate Left Through CarryRd(O) \leftarrow C,Rd(n+1), C,Rd(r)Z,C,N,V1RORRdAntimetic Shift RightRd(n) \leftarrow Rd(n+1), C,Rd(r)Z,C,N,V1SWAPRdAntimetic Shift RightRd(n) \leftarrow Rd(r), P, A(r), P, C,Rd(r)Z,C,N,V1SWAPRdSwap NibblesRd(S) \leftarrow Rd(r), Rd(r), A(r), A(r), C, Rd(r)Z,C,N,V1SWAPRdSwap NibblesRd(S) \leftarrow Rd(r), Rd(r), A(r), A(Dd 7	· · · ·			
SPMStore Program Memory(Z) \leftarrow R1:R0None.INRd, PIn PotRd \leftarrow PNone1OUTP, RrOut PortP \leftarrow RrNone1OUTP, RrOut PortP \leftarrow RrNone2POPRdPop Register on StackSTACK \leftarrow RrNone2POPRdPop Register from StackRd \leftarrow STACKNone2BIT AND BIT-TEST INSTRUCTIONSSBIP,bSet Bit in I/O RegisterI/O(P,b) \leftarrow 1None2CBIP,bClear Bit in I/O RegisterI/O(P,b) \leftarrow 0None2LSLRdLogical Shift LeftRd(n) \leftarrow Rd(n), Rd(0) \leftarrow 0Z,C,N,V1ROLRdRotate Left Through CarryRd(n) \leftarrow Rd(n+1), C-Rd(n), C-C,Rd(n)Z,C,N,V1RORRdRdate Left Through CarryRd(n) \leftarrow Rd(n+1), L-C-Rd(0)Z,C,N,V1SWAPRdSwap NibblesRd(30), ERd(n+1), L-C-Rd(30)None1BSTsFlag SetSREG(s) \leftarrow 1SREG(s)1BCLRsFlag SetSREG(s) \leftarrow 0SREG(s)1BCLRsFlag ClearSREG(s) \leftarrow 0SREG(s)1BLDRd, bBit Store from Register to TT \leftarrow Rt(b)T1LSClear CarryC \leftarrow 0C \leftarrow 011SECSet RegrayRd(b) \leftarrow TNone11SENSet Regative FlagN \leftarrow 0N11						
$\begin{array}{c c c c c c c c c c c c c c c c c c c $		nu, 2+				
$\begin{array}{c c c c c c c c c c c c c c c c c c c $		Bd P				
PUSH Rr Push Register on StackSTACK \leftarrow RrNone2POPRdPop Register from StackRd \leftarrow STACKNone2BIT AND BIT-TEST INSTRUCTIONSSBIP,bSet Bit In/O Register $VO(P,b) \leftarrow 1$ None2CBIP,bClear Bit in I/O Register $VO(P,b) \leftarrow 0$ None2LSLRdLogical Shift LeftRd(n+1) \leftarrow Rd(n), Rd(0) $\leftarrow 0$ Z,C,N,V1LSRRdLogical Shift RightRd(n+1) \leftarrow Rd(n), Rd(0) $\leftarrow 0$ Z,C,N,V1ROLRdRotate Left Through CarryRd(0) \leftarrow C,Rd(n+1) \leftarrow Rd(n), C,C+Rd(7)Z,C,N,V1ASRRdArithmetic Shift RightRd(n) \leftarrow Rd(n+1), n=0:6Z,C,N,V1SWAPRdSwap NibblesRd(3:0) \leftarrow Rd(7:4), Rd(7:4) \leftarrow Rd(3:0)None1BSETsFlag SetSREG(s) $\leftarrow 1$ SREG(s)1BCLRsFlag SetSREG(s) $\leftarrow 0$ SREG(s)1BLDRd, bBit Store from Register to TT \leftarrow Rr(b)T1BLDRd, bBit Store from Register to TT \leftarrow Rd(b) \leftarrow TNone1SECSet CarryC $\leftarrow 0$ C11SELSet Register FlagN $\leftarrow 1$ N1SELClear Negative FlagN $\leftarrow 0$ N1SEZSet Negative FlagZ $\leftarrow 0$ Z1SEIGlobal Interrupt EnableZ $\leftarrow 0$ Z1LSet Negative FlagZ $\leftarrow 0$ Z<		-				
POPRdPop Register from StackRd \leftarrow STACKNone2BIT AND BIT-TEST INSTRUCTIONSSBIP.bSet Bit in I/O RegisterI/O(P.b) $\leftarrow 1$ None2CBIP.bClear Bit in I/O RegisterI/O(P.b) $\leftarrow 0$ None2CBIP.bClear Bit in I/O RegisterI/O(P.b) $\leftarrow 0$ None2CBIRdLogical Shift LeftRd(n+1) \leftarrow Rd(n), Rd(0) $\leftarrow 0$ Z,C,N,V1LSRRdLogical Shift RightRd(n) \leftarrow Rd(n) \leftarrow Rd(n), C,C-Rd(7)Z,C,N,V1ROLRdRotate Left Through CarryRd(7) \leftarrow C,Rd(n+1), C,C-Rd(0)Z,C,N,V1RORRdRdArithmetic Shift RightRd(n) \leftarrow Rd(n+1), n=0:6Z,C,N,V1SWAPRdSwap NibolesRd(3.0) \leftarrow Rd(7.4),Rd(7.4),C-Rd(3.0)None1BSETsFlag SetSREG(s) $\leftarrow 1$ SREG(s)1BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)1BLDRd, bBit Store from Register to TT \leftarrow Rt(b) \leftarrow T1BLDRd, bBit load from T to RegisterRd(b) \leftarrow TC $\leftarrow 0$ C1CLCClear CarryC $\leftarrow 0$ C $\leftarrow 1$ C1CLNClear CarryC $\leftarrow 0$ C11SEIGlobal Interrupt EnableN $\leftarrow 0$ N11SEIGlobal Interrupt EnableI $\leftarrow 1$ I11						
BIT AND BIT-TEST INSTRUCTIONSSBIP,bSet Bit in I/O RegisterI/O(P,b) $\leftarrow 1$ None2CBIP,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), C,Rd(7)Z,C,N,V1RORRdRotate Left Through CarryRd(7) \leftarrow C,Rd(n+1), C,C,Rd(0)Z,C,N,V1SWAPRdSwap NibblesRd(30) \leftarrow Rd(n+1), C,C,Rd(30)None1BSETsFlag SetSREG(s) $\leftarrow 1$ SREG(s)1BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)1BLDRd, bBit Store from Register to TT \leftarrow Rt(b) \leftarrow TNone1SECSet CarryC $\leftarrow 0$ C $\leftarrow 0$ 11CLCClear CarryC $\leftarrow 0$ C $\leftarrow 0$ 1SENSet Negative FlagN $\leftarrow 0$ N1CLASet Zero FlagZ $\leftarrow 1$ Z1SEIGlobal Interrupt EnableI $\leftarrow 0$ I1SEIGlobal Interrupt EnableI $\leftarrow 1$ 11	POP					
SBIP,bSet Bit in I/O RegisterI/O(P,b) $\leftarrow 1$ None2CBIP,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), C,C,Rd(7)Z,C,N,V1RORRdRotate Left Through CarryRd(7) \leftarrow C,Rd(n+1), C,C-Rd(7)Z,C,N,V1ASRRdAnithmetic Shift RightRd(n) \leftarrow Rd(n+1), n=0:6Z,C,N,V1SWAPRdSwap NibblesRd(3:0) \leftarrow Rd(7:4),Rd(7:4) \leftarrow Rd(3:0)None1BSETsFlag SetSREG(s) $\leftarrow 1$ SREG(s)1BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)1BCLRsStor from Register to TT \leftarrow Rt(b)T1BLDRd, bBit load from T to RegisterRd(b) \leftarrow TNone1SECSet CarryC $\leftarrow 0$ C11CL2Clear Arguive FlagN $\leftarrow 0$ N11SEZSet Negative FlagN $\leftarrow 0$ N11SEZClear Zero FlagZ $\leftarrow 0$ Z11SEZGlobal Interrupt EnableI $\leftarrow 0$ I $\leftarrow 0$ I1SEIGlobal Interrupt EnableI $\leftarrow 0$ I $\leftarrow 0$ I1SEISet Zero FlagZ $\leftarrow 0$ Z <	BIT AND BIT-TEST	INSTRUCTIONS				
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 $Rd(0) \leftarrow C, Rd(n+1), Rd(7) \leftarrow C$ Z, C, N, V 1RORRdRotate Right Through Carry $Rd(7) \leftarrow C, Rd(n+1), C \leftarrow Rd(0)$ Z, C, N, V 1ASRRdArithmetic Shift Right $Rd(7) \leftarrow C, Rd(n+1), C \leftarrow Rd(0)$ Z, C, N, V 1ASRRdArithmetic Shift Right $Rd(3, 0) \leftarrow Rd(n+1), n=0:6$ Z, C, N, V 1SWAPRdSwap Nibbles $Rd(3, 0) \leftarrow Rd(n+1), R=0:6$ Z, C, N, V 1BSETsFlag SetSREG(s) $\leftarrow 1$ SREG(s)1BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)1BLDRd, bBit Store from Register to T $T \leftarrow Rr(b)$ T1BLDRd, bBit load from T to RegisterRd(b) $\leftarrow T$ None1SECSet Carry $C \leftarrow 1$ C1CLClear Carry $C \leftarrow 0$ C1SENSet Negative Flag $N \leftarrow 0$ N1CLNClear Regative Flag $Z \leftarrow 1$ Z1CL2Clear Zero Flag $Z \leftarrow 1$ Z1SEIGlobal Interrupt Enable $I \leftarrow 1$ 11	SBI		Set Bit in I/O Register	I/O(P,b) ← 1	None	2
LSRRdLogical Shift RightRd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0Z,C,N,V1ROLRdRotate Left Through CarryRd(0) \leftarrow C,Rd(n+1) \leftarrow Rd(n), C \leftarrow Rd(7)Z,C,N,V1RORRdRotate Right Through CarryRd(7) \leftarrow C,Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)Z,C,N,V1ASRRdArithmetic Shift RightRd(n) \leftarrow Rd(n+1), n=0:6Z,C,N,V1SWAPRdSwap NibblesRd(3:0) \leftarrow Rd(7:4), Rd(7:4) \leftarrow Rd(3:0)None1BSETsFlag SetSREG(s) \leftarrow 1SREG(s)1BCLRsFlag ClearSREG(s) \leftarrow 0SREG(s)1BSTRr, bBit Store from Register to TT \leftarrow Rr(b)T1BLDRd, bBit load from T to RegisterRd(b) \leftarrow TNone1SECSet CarryC \leftarrow 0C1CLCClear CarryC \leftarrow 0C1SENSet Negative FlagN \leftarrow 1N1CLNClear Xegative FlagZ \leftarrow 1Z1SEZSet Zero FlagZ \leftarrow 1I1SEIGlobal Interrupt EnableI \leftarrow 1I1	CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSRRdLogical Shift RightRd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0Z, C, N,V1ROLRdRotate Left Through CarryRd(0) \leftarrow C,Rd(n+1) \leftarrow Rd(n), C \leftarrow Rd(7)Z, C, N,V1RORRdRotate Right Through CarryRd(7) \leftarrow C,Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)Z, C, N,V1ASRRdArithmetic Shift RightRd(n) \leftarrow Rd(n+1), n=0:6Z, C, N,V1SWAPRdSwap NibblesRd(3:0) \leftarrow Rd(7:4), Rd(7:4) \leftarrow Rd(3:0)None1BSETsFlag SetSREG(s) \leftarrow 1SREG(s)1BCLRsFlag ClearSREG(s) \leftarrow 0SREG(s)1BLDRd, bBit load from T to Register to TT \leftarrow Rr(b)T1BLDRd, bBit load from T to RegisterC \leftarrow 1C1CLCClear CarryC \leftarrow 0C1SENSet Negative FlagN \leftarrow 0N1CLNClear Negative FlagN \leftarrow 0N1SEZGlobal Interrupt EnableI \leftarrow 11 \leftarrow 11	LSL	Rd			Z,C,N,V	1
RORRdRotate Right Through Carry $Rd(7) \leftarrow C, Rd(n) \leftarrow Rd(n+1), C\leftarrow Rd(0)$ Z, C, N, 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 1BSETsFlag Set $Rd(3:0) \leftarrow Rd(n+1), Rd(3:0)$ None1BSETsFlag Clear $SREG(s) \leftarrow 1$ $SREG(s)$ 1BCLRsFlag Clear $SREG(s) \leftarrow 0$ $SREG(s)$ 1BSTRr, bBit Store from Register to T $T \leftarrow Rr(b)$ T1BLDRd, bBit load from T to Register $Rd(b) \leftarrow T$ None1SECSet Carry $C \leftarrow 1$ C1CLCClear Carry $C \leftarrow 0$ C1SENSet Negative Flag $N \leftarrow 1$ N1CLNClear Negative Flag $X \leftarrow 0$ N1SEZSet Zero Flag $Z \leftarrow 0$ Z1SEIGlobal Interrupt Enable $I \leftarrow 1$ 11	LSR	Rd	Logical Shift Right		Z,C,N,V	1
ASRRdArithmetic Shift Right $Rd(n) \leftarrow Rd(n+1), n=0:6$ Z,C,N,V 1SWAPRdSwap Nibbles $Rd(3:0) \leftarrow Rd(7:4),Rd(7:4) \leftarrow Rd(3:0)$ None1BSETsFlag SetSREG(s) $\leftarrow 1$ SREG(s)1BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)1BSTRr, bBit Store from Register to T $T \leftarrow Rr(b)$ T1BLDRd, bBit load from T to RegisterRd(b) $\leftarrow T$ None1SECSet CarryC $\leftarrow 1$ C $\leftarrow 0$ 1CLCClear CarryC $\leftarrow 0$ C1SENSet Negative FlagN $\leftarrow 1$ N1SEZSet Zero FlagZ $\leftarrow 1$ Z $\leftarrow 0$ N1SEZClear Zero FlagZ $\leftarrow 0$ Z1SEIGlobal Interrupt EnableI $\leftarrow 1$ I $\leftarrow 1$ 1	ROL	Rd	Rotate Left Through Carry		Z,C,N,V	1
SWAPRdSwap Nibbles $Rd(3:0)\leftarrow Rd(7:4), Rd(7:4), Rd(3:0)$ None1BSETsFlag SetSREG(s)SREG(s)1BCLRsFlag ClearSREG(s)SREG(s)0BSTRr, bBit Store from Register to TT \leftarrow Rr(b)T1BLDRd, bBit load from T to RegisterRd(b) \leftarrow TNone1SECSet CarryC \leftarrow 1C1CLCSet Negative FlagN \leftarrow 1N1SENSet Negative FlagN \leftarrow 0N1SEZISet Zero FlagZ \leftarrow 1Z1CLZGlobal Interrupt EnableI \leftarrow 1I \leftarrow 111	ROR	Rd			Z,C,N,V	1
BSETsFlag SetSREG(s) $\leftarrow 1$ SREG(s)1BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)1BSTRr, bBit Store from Register to T $T \leftarrow Rr(b)$ T1BLDRd, bBit load from T to RegisterRd(b) $\leftarrow T$ None1SECSet Carry $C \leftarrow 1$ C1CLCClear 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 $\leftarrow 1$ 11	ASR					
BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)1BSTRr, bBit Store from Register to T $T \leftarrow Rr(b)$ T1BLDRd, bBit load from T to RegisterRd(b) $\leftarrow T$ None1SECSet Carry $C \leftarrow 1$ C1CLCClear Carry $C \leftarrow 0$ C1SENSet Negative FlagN $\leftarrow 1$ N1CLNClear Negative FlagN $\leftarrow 0$ N1SEZSet Zero Flag $Z \leftarrow 1$ Z1CLZClear Zero Flag $Z \leftarrow 0$ Z1SEIGlobal Interrupt Enable $I \leftarrow 1$ 11	SWAP	Rd				
BSTRr, bBit Store from Register to T $T \leftarrow Rr(b)$ T1BLDRd, bBit load from T to RegisterRd(b) \leftarrow TNone1SECSet CarryC \leftarrow 1C \leftarrow 1C1CLCClear CarryC \leftarrow 0C1SENSet Negative FlagN \leftarrow 1N1CLNClear Negative FlagN \leftarrow 0N1SEZSet Zero FlagZ \leftarrow 1Z1CLZClear Zero FlagZ \leftarrow 0I1SEIGlobal Interrupt EnableI \leftarrow 1I1	BSET					
BLDRd, bBit load from T to RegisterRd(b) \leftarrow TNone1SECSet CarryC \leftarrow 1C \leftarrow 1C1CLCClear CarryC \leftarrow 0C1SENSet Negative FlagN \leftarrow 1N1CLNClear Negative FlagN \leftarrow 0N1SEZSet Zero FlagZ \leftarrow 1Z1CLZClear Zero FlagZ \leftarrow 0Z1SEIGlobal Interrupt EnableI \leftarrow 1I1	BCLR					
$\begin{array}{c c c c c c c c c c c c c c c c c c c $						
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	BLD	Rd, b				
SEN Set Negative Flag N ← 1 N 1 CLN Clear Negative Flag N ← 0 N 1 SEZ Set Zero Flag Z ← 1 Z 1 CLZ Clear Zero Flag Z ← 0 Z 1 SEI Global Interrupt Enable I ← 1 I 1						
CLN Clear Negative Flag N ← 0 N 1 SEZ Set Zero Flag Z ← 1 Z 1 CLZ Clear Zero Flag Z ← 0 Z 1 SEI Global Interrupt Enable I ← 1 I 1		-				
SEZ Set Zero Flag Z ← 1 Z 1 CLZ Clear Zero Flag Z ← 0 Z 1 SEI Global Interrupt Enable I ← 1 I 1						
CLZ Clear Zero Flag Z ← 0 Z 1 SEI Global Interrupt Enable I ← 1 I 1						
SEI Global Interrupt Enable I ← 1 I 1						
					2	
	CLI		Global Interrupt Enable Global Interrupt Disable	I ← 1 I ← 0		1





6. Instruction Set Summary (Continued)

SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
Mnemonics	Operands	Description	Operation	Flags	#Clocks
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	$V \leftarrow 0$	V	1
SET		Set T in SREG	T ← 1	Т	1
CLT		Clear T in SREG	$T \leftarrow 0$	Т	1
SEH		Set Half Carry Flag in SREG	H ← 1	н	1
CLH		Clear Half Carry Flag in SREG	$H \leftarrow 0$	н	1
MCU CONTROL IN	ISTRUCTIONS				
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

7. Ordering Information

7.1 ATmega128A

Speed (MHz)	Power Supply	Ordering Code ⁽²⁾	Package ⁽¹⁾	Operation Range
16	2.7 - 5.5V	ATmega128A-AU ATmega128A-AUR ⁽³⁾ ATmega128A-MU ATmega128A-MUR ⁽³⁾	64A 64A 64M1 64M1	Industrial (-40°C to 85°C)

Notes: 1. The 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.

3. Tape & Reel

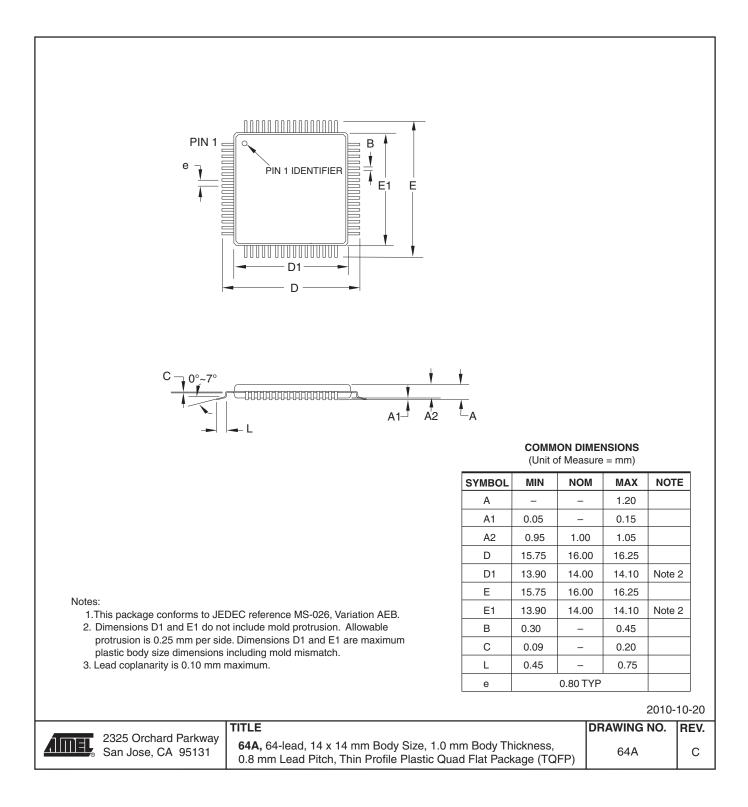
Package Type				
64 A	64-lead, 14 x 14 x 1.0 mm, Thin Profile Plastic Quad Flat Package (TQFP)			
64M1	64-pad, 9 x 9 x 1.0 mm, Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)			





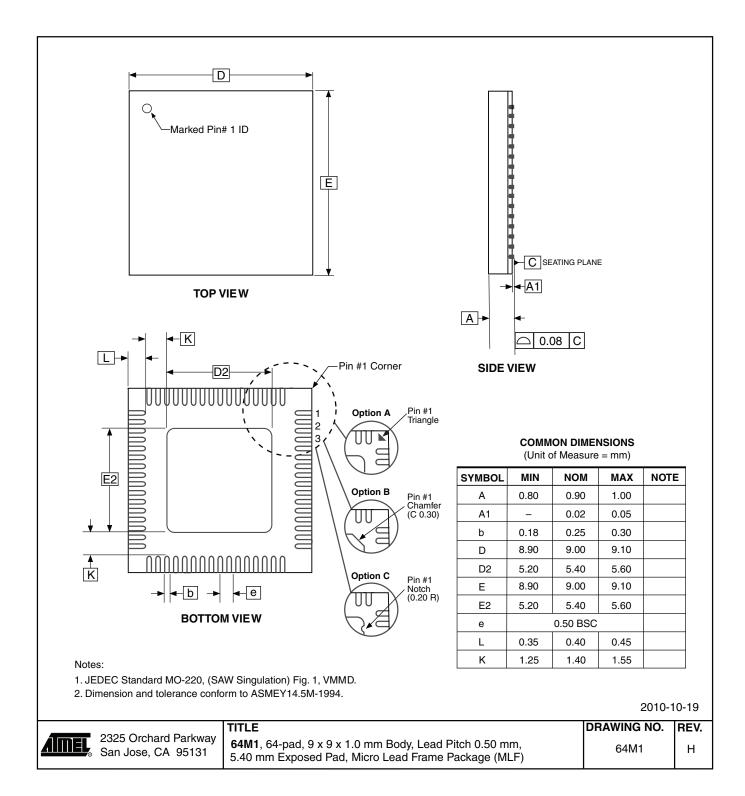
8. Packaging Information

8.1 64A



16 ATmega128A

8.2 64M1







9. Errata

The revision letter in this section refers to the revision of the ATmega128A device.

9.1 ATmega128A Rev. U

- Wrong value for Version in the JTAG Device Identification Register
- 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
NOP		; no operation
NOP		; no operation

NOP	; no operation
NOP	; no operation
SEI	; set 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 ATmega128A is the only device in the scan chain, the problem is not visible.
- Select the Device ID Register of the ATmega128A 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 ATmega128A 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 ATmega128A must be the fist 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.





10. 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.

10.1 Rev. 8151H – 02/11

- 1. Editing update according to the Atmel new style guide. No more space betweeen the numbers and their units.
- 2. Updated the last page.

10.2 Rev. 8151G - 07/10

1. Updated the table note of Table 27-3 on page 324. The test is performed using BODLEVEL=0 and BODLEVEL=1

10.3 Rev. 8151F - 06/10

- 1. Inserted cross reference in "Minimizing Power Consumption" on page 48
- 2. Updated Technical Terminology according to Atmel standard
- 3. Note 6 and Note 7 below "Two-wire Serial Bus Requirements" on page 325 have been removed
- 4. The text in "Bit 6 TXCIEn: TX Complete Interrupt Enable" on page 194 has been corrected by adding an "n"

10.4 Rev. 8151E - 02/10

- 1. Updated "Receiving Frames with 9 Data Bits" on page 185. The C code updated.
- 2. Updated "Packaging Information" on page 373.
- 3. Updated "Performing Page Erase by SPM" on page 283.

10.5 Rev. 8151D - 07/09

- 1. Updated "Errata" on page 376.
- 2. Updated the last page with Atmel's new addresses.

10.6 Rev. 8151C - 05/09

1. Updated "Errata" on page 375. ATmega128A Rev. U.

10.7 Rev. 8151B - 03/09

- 1. Updated view of "Typical Characteristics" on page 337 view.
- 2. Editorial updates.

10.8 Rev. 8151A-08/08

1. Initial revision. (Based on the ATmega128/L datasheet 2467R-AVR-06/08)

Changes done compared to the ATmega128/L datasheet 2467R-AVR-06/08:

- Updated "Stack Pointer" on page 13 description.
- "Power Management and Sleep Modes" on page 46 is reorganized.
- All Electrical characteristics is moved to "Electrical Characteristics" on page 321.

- Output Low Voltage (V_{OL}) and Reset Pull-up Resistor (R_{RST}) limits updated in "DC Characteristics" on page 321.

- Register descriptions are moved to sub sections at the end of each chapter.
- New graphs in "Typical Characteristics" on page 338.
- New "Ordering Information" on page 373.



Atmel Corporation

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

Atmel Asia Limited 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 Munich GmbH Business Campus Parkring 4 D-85748 Garching b. Munich GERMANY Tel: (+49) 89-31970-0 Fax: (+49) 89-3194621

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

© 2011 Atmel Corporation. All rights reserved. / Rev. CORP0XXXX

Atmel[®], Atmel logo and combinations thereof, AVR[®], AVR Studio[®], and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Windows[®] and others are registered trademarks of Microsoft Corporation in U.S. and or other countries. Other terms and product names may be trademarks of others.

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 THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, 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 DIFCT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROF-ITS, 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 products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.