Features

- High-performance, Low-power AVR[®] 8-bit Microcontroller
- Advanced RISC Architecture
 - 130 Powerful Instructions Most Single-clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Fully Static Operation
 - Up to 16 MIPS Throughput at 16 MHz
 - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
 - 8K Bytes of In-System Self-programmable Flash program memory
 - 512 Bytes EEPROM
 - 1K Byte 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
 - Programming Lock for Software Security
- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescaler, one Compare Mode
 - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
 - Real Time Counter with Separate Oscillator
 - Three PWM Channels
 - 8-channel ADC in TQFP and QFN/MLF package Eight Channels 10-bit Accuracy
 - 6-channel ADC in PDIP package Six Channels 10-bit Accuracy
 - Byte-oriented Two-wire Serial Interface
 - Programmable Serial USART
 - Master/Slave SPI Serial Interface
 - Programmable Watchdog Timer with Separate On-chip Oscillator
 - On-chip Analog Comparator
- Special Microcontroller Features
 - Power-on Reset and Programmable Brown-out Detection
 - Internal Calibrated RC Oscillator
 - External and Internal Interrupt Sources
 - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and
 - Standby
- I/O and Packages
 - 23 Programmable I/O Lines
 - 28-lead PDIP, 32-lead TQFP, and 32-pad QFN/MLF
- Operating Voltages
 - 2.7 5.5V (ATmega8L)
 - 4.5 5.5V (ATmega8)
- Speed Grades
 - 0 8 MHz (ATmega8L)
 - 0 16 MHz (ATmega8)
- Power Consumption at 4 Mhz, 3V, 25°C
 - Active: 3.6 mA
 - Idle Mode: 1.0 mA
 - Power-down Mode: 0.5 μA



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

ATmega8 ATmega8L

Summary

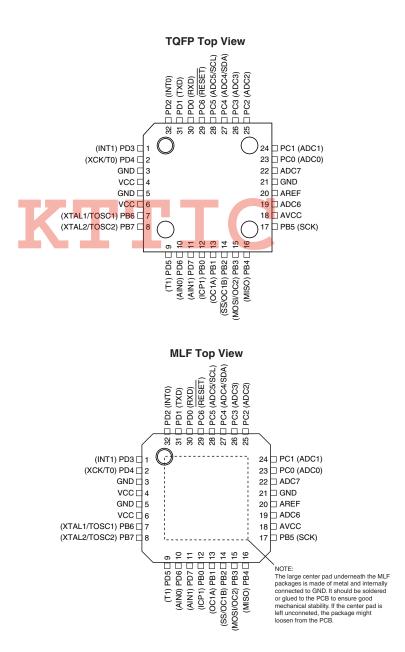


http://www.kttic.com

Pin Configurations

	FDIF		
(RESET) PC6	1	28	PC5 (ADC5/SCL)
(RXD) PD0 🗆	2	27	□ PC4 (ADC4/SDA)
(TXD) PD1 🗆	3	26	PC3 (ADC3)
(INT0) PD2 🗆	4	25	PC2 (ADC2)
(INT1) PD3 🗆	5	24	PC1 (ADC1)
(XCK/T0) PD4 🗆	6	23	PC0 (ADC0)
VCC 🗆	7	22	🗆 GND
GND 🗆	8	21	□ AREF
(XTAL1/TOSC1) PB6	9	20	AVCC
(XTAL2/TOSC2) PB7	10	19	PB5 (SCK)
(T1) PD5 🗆	11	18	□ PB4 (MISO)
(AIN0) PD6 🗆	12	17	PB3 (MOSI/OC2)
(AIN1) PD7 🗆	13	16	PB2 (SS/OC1B)
(ICP1) PB0 🗆	14	15	PB1 (OC1A)

PDIP



² ATmega8(L)

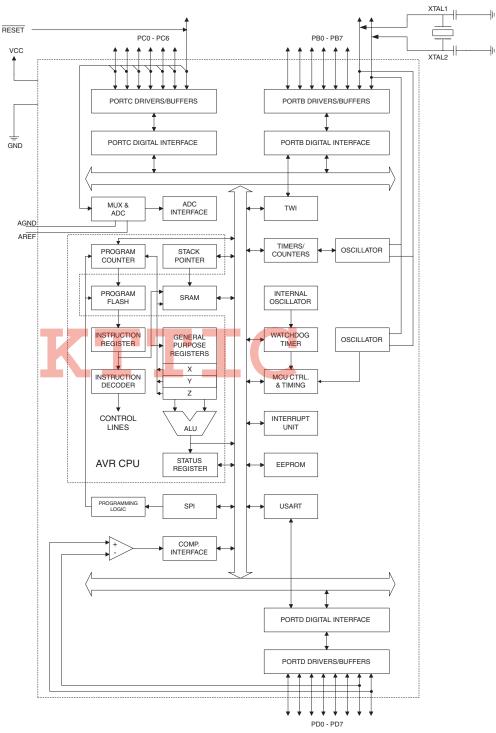
http://www.kttirs.wronon

ATmega8(L)

Overview

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

Block Diagram Figure 1. Block Diagram







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

The ATmega8 provides the following features: 8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes of EEPROM, 1K byte of SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, a 6-channel ADC (eight channels in TQFP and QFN/MLF packages) with 10-bit accuracy, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next Interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption.

The device is manufactured using Atmel's high density non-volatile memory technology. The Flash Program memory can be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash Section will continue to run while the Application Flash Section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega8 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega8 AVR is supported with a full suite of program and system development tools, including C compilers, macro assemblers, program debugger/simulators, In-Circuit Emulators, and evaluation kits.

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

http://www.kttic.com

Pin Descriptions

VCC	Digital supply voltage.
GND	Ground.
Port B (PB7PB0) XTAL1/XTAL2/TOSC1/ TOSC2	Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.
	Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscil- lator amplifier and input to the internal clock operating circuit.
	Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.
	If the Internal Calibrated RC Oscillator is used as chip clock source, PB76 is used as TOSC21 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.
	The various special features of Port B are elaborated in "Alternate Functions of Port B" on page 58 and "System Clock and Clock Options" on page 25.
Port C (PC5PC0)	Port C is an 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.
PC6/RESET	If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical char- acteristics of PC6 differ from those of the other pins of Port C.
	If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on 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 15 on page 38. Shorter pulses are not guaranteed to generate a Reset.
	The various special features of Port C are elaborated on page 61.
Port D (PD7PD0)	Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.
	Port D also serves the functions of various special features of the ATmega8 as listed on page 63.
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 15 on page 38. Shorter pulses are not guaranteed to generate a reset.



AV _{cc}	AV_{CC} is the supply voltage pin for the A/D Converter, Port C (30), and ADC (76). It should be externally connected to V_{CC} , even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter. Note that Port C (54) use digital supply voltage, V_{CC} .
AREF	AREF is the analog reference pin for the A/D Converter.
ADC76 (TQFP and QFN/MLF Package Only)	In the TQFP and QFN/MLF package, ADC76 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

Resources	A comprehensive set of development tools, application notes and datasheets are 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

	1	-	1			1	1	1	1	
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x3F (0x5F)	SREG	I	Т	Н	S	V	Ν	Z	С	11
0x3E (0x5E)	SPH	-	-	-	-	-	SP10	SP9	SP8	13
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	13
0x3C (0x5C)	Reserved									
0x3B (0x5B)	GICR	INT1	INT0	-	-	-	-	IVSEL	IVCE	49, 67
0x3A (0x5A)	GIFR	INTF1	INTF0	-	-	-	-	-	-	68
0x39 (0x59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	-	TOIE0	72, 102, 122
0x38 (0x58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	-	TOV0	73, 102, 122
0x37 (0x57)	SPMCR	SPMIE	RWWSB	-	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	213
0x36 (0x56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	171
0x35 (0x55)	MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	33, 66
0x34 (0x54)	MCUCSR	-	-	-	-	WDRF	BORF	EXTRF	PORF	41
0x33 (0x53)	TCCR0	-	-	-	-	-	CS02	CS01	CS00	72
0x32 (0x52)	TCNT0				Timer/Cou	inter0 (8 Bits)				72
0x31 (0x51)	OSCCAL				Oscillator Cal	ibration Register				31
0x30 (0x50)	SFIOR	-	-	-	-	ACME	PUD	PSR2	PSR10	58, 75, 123, 193
0x2F (0x4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	96
0x2E (0x4E)	TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	100
0x2D (0x4D)	TCNT1H			Tim	er/Counter1 – Co	unter Register Hi	gh byte			101
0x2C (0x4C)	TCNT1L			Tim	er/Counter1 – Co	unter Register Lo	ow byte			101
0x2B (0x4B)	OCR1AH			Timer/Co	unter1 – Output C	Compare Register	r A High byte			101
0x2A (0x4A)	OCR1AL			Timer/Co	ounter1 – Output 0	Compare Registe	r A Low byte			101
0x29 (0x49)	OCR1BH			Timer/Co	unter1 – Output C	Compare Register	B High byte			101
0x28 (0x48)	OCR1BL			Timer/Co	ounter1 – Output 0	Compare Registe	r B Low byte			101
0x27 (0x47)	ICR1H			Timer/	Counter1 – Input	Capture Register	High byte			102
0x26 (0x46)	ICR1L			Timer/	Counter1 – Input	Capture Register	Low byte			102
0x25 (0x45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	117
0x24 (0x44)	TCNT2				Timer/Cou	inter2 (8 Bits)				119
0x23 (0x43)	OCR2			Ti	mer/Counter2 Ou	tput Compare Re	gister			119
0x22 (0x42)	ASSR	-	-	_	_	AS2	TCN2UB	OCR2UB	TCR2UB	119
0x21 (0x41)	WDTCR	-	-		WDCE	WDE	WDP2	WDP1	WDP0	43
0x20 ⁽¹⁾ (0x40) ⁽¹⁾	UBRRH	URSEL	-	_	-		UBR	R[11:8]		158
0x20(** (0x40)(**	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	156
0x1F (0x3F)	EEARH	-	_		-	-		-	EEAR8	20
0x1E (0x3E)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	20
0x1D (0x3D)	EEDR				EEPROM	Data Register				20
0x1C (0x3C)	EECR	-	-	-	-	EERIE	EEMWE	EEWE	EERE	20
0x1B (0x3B)	Reserved									
0x1A (0x3A)	Reserved									
0x19 (0x39)	Reserved									
0x18 (0x38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	65
0x17 (0x37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	65
0x16 (0x36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	65
0x15 (0x35)	PORTC	-	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	65
0x14 (0x34)	DDRC	-	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	65
0x13 (0x33)	PINC	-	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	65
0x12 (0x32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	65
0x11 (0x31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	65
0x10 (0x30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	65
0x0F (0x2F)	SPDR				SPI Da	ta Register				131
0x0E (0x2E)	SPSR	SPIF	WCOL	-	-	-	-	-	SPI2X	131
0x0D (0x2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	129
0x0C (0x2C)	UDR				USART I/O	Data Register	. <u> </u>			153
0x0B (0x2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	154
0x0A (0x2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	155
0x09 (0x29)	UBRRL				USART Baud Ra	te Register Low b	oyte			158
0x08 (0x28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	194
0x07 (0x27)	ADMUX	REFS1	REFS0	ADLAR	_	MUX3	MUX2	MUX1	MUX0	205
0x06 (0x26)	ADCSRA	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	207
	ADCH				ADC Data Re	egister High byte				208
0x05 (0x25)	ADOIT									
0x05 (0x25) 0x04 (0x24)	ADCL				ADC Data Re	egister Low byte				208
				1	ADC Data Re Two-wire Serial In		ister			208 173

Register Summary (Continued)

Address Name Bit 7 Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 Bit 0 Page TWSR TWS6 TWS5 TWS4 TWS3 TWPS1 TWPS0 0x01 (0x21) TWS7 173 0x00 (0x20) TWBR Two-wire Serial Interface Bit Rate Register 171

Notes:

1. Refer to the USART description for details on how to access UBRRH and UCSRC. 2. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses

should never be written. 3. Some of the Status Flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers 0x00 to 0x1F only.



Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND L	OGIC INSTRUCTION	Ş	-		-
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	$Rdh:RdI \leftarrow Rdh:RdI + K$	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← 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.	$Rd \gets Rd - K - C$	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	$Rd \leftarrow Rd v Rr$	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	$Rd \leftarrow Rd \vee K$	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 ← Rd v 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 Z,N,V	1
TST	Rd	Test for Zero or Minus	$Ra \leftarrow Ra - 1$ $Rd \leftarrow Rd \bullet Rd$	Z,N,V Z,N,V	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULS	Rd, Rr	Multiply Signed	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	$R1:R0 \leftarrow (Rd \times Rr) << 1$	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	$R1:R0 \leftarrow (Rd \times Rr) << 1$	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \le 1$	Z,C	2
BRANCH INSTRUC	TIONS			- 1	1
RJMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
RCALL	k	Relative Subroutine Call	$PC \leftarrow PC + k + 1$	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
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 $(Rr(b)=0) PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC \leftarrow PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if $(P(b)=0) PC \leftarrow PC + 2 \text{ or } 3$	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 if (Z = 0) then PC \leftarrow PC + k + 1	None	1/2
BRCS	k k	Branch if Carry Set	if (C = 1) then PC \leftarrow PC + k + 1 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
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC \leftarrow PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC \leftarrow PC + k + 1	None	1/2
DIVUS					
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC \leftarrow PC + k + 1	None	1/2

Instruction Set Summary (Continued)

BND * Branch Interrupt Disability # (1 = 0 pm R) = - P = + + 1 None DATA TANASSER TRAVEOROS Ref. FX Move Benzee Registers. Ref. FX Nore Nore MOV Ref. FX Load Internative Move Benzee Registers. Ref. FX Nore Nore L01 Ref. X Load Internative Move Benzee Ref. FX Nore L01 Ref. X Load Internative Move Benzee Ref. FX Nore L0 Ref. X Load Internative Move Benzee FX Nore L0 Ref. Y Load Internative Move Benzee FX Nore L0 Ref. Y Load Internative Move Benzee FX Nore L0 Ref. Y Load Internative Move Benzee FX Nore L0 Ref. Y Load Internative Move Benzee FX Nore L0 Ref. Y Load Internative Move Benzee FX Nore L0 Ref. Y Sover Internative Move Benzee FX Nore L0 Ref. Y Sover Internative Move Benzee F	BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC \leftarrow PC + k + 1	None	1/2
MOV R.J. R. Nove Source Registers R0 + R1 Nove MOVM R.J. R. Copy Register Word Ret + K1 Nove LDI R.J. K. Last Indires Ret + K1 Nove LDI R.J. K. Last Indires Ret + C0 Nove LD R.J. X. Last Indires Ret + C0 Nove LD R.J. X. Last Indires and Position. Ret - (Y) + V + 1 Nove LD R.J. V. Last Indires and Position. Ret - (Y) = (Y + V + 1) Nove LD R.J. V. Last Indires and Position. Ret - (Y) = (Y + V + 1) Nove LD R.J. V. Last Indires and Position. Ret - (Y) = (Y + V + 1) Nove LD Last Indires and Position. Ret - (Z) = (Z + 1) Nove Nove LD Last Indires and Position. Ret - (Z) = (Z + 1) Nove LD Last Indires and Position. Ret - (Z) = (Z + 1) Nove LD Last Indires and Position. Ret - (Z) = (Z + 1) Nove LD Last I	BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC \leftarrow PC + k + 1	None	1/2
MOW R.A. K. Cog/ Regiser Wood Red + K. None LDI Rd, K. Load Indication Rd + CQ None LD Rd, X. Load Indication Rd + CQ None LD Rd, X. Load Indicat and ProCes. Rd + CQ None LD Rd, Y. Load Indication ProCes. X + X + 1. Rd + CQ None LD Rd Y. Load Indication ProCes. Y + Y + 7. Rd + CY None LD Rd Y. Load Indication ProCes. Y + Y + 7. Rd + CY None LD Rd Y. Load Indication ProCes. Y + Y + 7. Rd + CY None LD Rd Z. Load Indication ProCes. Y + Y + 7. Rd + CY None LD Rd Z. Load Indication ProCes. Y + Y + 7. Rd + CZ None LD Rd Z. Load Indication ProCes. Y + Y + 7. Rd + CZ None LD Rd Z. Load Indication ProCes. Y + Y + 7. Rd + CZ None LD Rd Z. Load Indication ProCes. Y + Y + Y + Y + Y + Y + Y + Y + Y + Y +	DATA TRANSFER	NSTRUCTIONS				
Lb1 R±, K Land Indian R± + C Nome LD R±, X Land Indian and Post-Inc. R± + C(x), X + X + 1 Nome LD R±, X Land Indian and Post-Inc. R± + C(x), X + X + 1 Nome LD R±, X Land Indian and Post-Inc. R± + (Y), X + Y + 1 Nome LD R±, Y Land Indian and Post-Inc. R± + (Y) + (Y + Y + 1) Nome LD R±, Y Land Indian and Post-Inc. R± + (Y) Nome LD R±, Y Land Indian and Post-Inc. R± + (Z) Nome LD R±, Z Land Indian and Post-Inc. R± + (Z) Nome LD R±, Z Land Indian and Post-Inc. R± + (Z) Nome LD R±, Z Land Indian and Post-Inc. R± + (Z) Nome LD R±, Z Land Indian and Post-Inc. R± + (Z) Nome LD R± = (Z) Nome Nome Nome LD Land Indian and Post-Inc. X + X + 1, R± N Nome LD Land Indigent and	MOV	Rd, Rr	Move Between Registers	$Rd \leftarrow Rr$	None	1
LDRd. ×Land Indicar of Polin.Rd + (D)NumeLDRd. ×Land Indicar of Polin.Rd + (D)NumeLDRd. ×Land Indicar of Polin.X + × 1, Rd + (D)NumeLDRd. ×Land Indicar of Polin.Rd + (T)NumeLDRd. ×Land Indicar of Polin.Rd + (T)NumeLDRd. ×Land Indicar of Polin.Rd + (T) + 1NumeLDRd. ×Land Indicar of Polin.Rd + (T) + 0NumeLDRd. ×Land Indicar of Polin.Rd + (T) + 0NumeLDRd. ×Land Indicar of Polin.Rd + (Z) 2 - 2.11NumeLDRd. ×Land Indicar of Polin.Rd + (Z) 2 - 2.11NumeLDRd. ×Land Indicar of Polin.Rd + (Z) 2 - 2.11NumeLDRd. ×Land Indicar of Polin.Rd + (Z) 2 - 2.11NumeLDRd. ×Land Indicar of Polin.Rd + (Z) 2 - 2.11NumeSTX, RTStam Indicar of Polin.(D) + R, × + 1.1NumeSTX, RTStam Indicar of Polin.(D) + R, × + 1.1NumeSTX, RTStam Indicar of Polin.(D) + R, × + 1.1NumeSTX, RTStam Indicar of Polin.(D) + R, × + 1.1NumeSTX, RTStam Indicar of Polin.(D) + R, × + 1.1NumeSTX, RTStam Indicar of Polin.(D) + R, × + 1.1NumeSTX, RTStam Indicar of Polin.(D) + R, × + 1.1NumeST<	MOVW	Rd, Rr	Copy Register Word	$Rd+1:Rd \leftarrow Rr+1:Rr$	None	1
LD RdX Load Indiget and Pre-Boc. Rd - (X), K - X + 1 None LD Rd, Y Load Indiget and Pre-Boc. X + X + 1, Rd + (D) None LD Rd, Y Load Indiget and Pre-Boc. Y + Y + 1, Rd + (P) None LD Rd, Y Load Indiget and Pre-Boc. Y + Y + 1, Rd + (P) None LD Rd, Y Load Indiget and Pre-Boc. Y + Y + 1, Rd + (P) None LD Rd, Z Load Indiget and Pre-Boc. Z + Z + 1, Rd + (P) None LD Rd, Z Load Indiget and Pre-Boc. Z + Z + 1, Rd + (P) None LD Rd, Z Load Indiget and Pre-Boc. Z + Z + 1, Rd + (P) None LD Rd, Z Load Indiget and Pre-Boc. Z + Z + 1, Rd + (P) None ST X, Rr Store Indiget and Pre-Boc. X + X + (X + 1) None ST X, Rr Store Indiget and Pre-Boc. X + X + (Y + 1) None ST X, Rr Store Indiget and Pre-Boc. X + X + (Y + 1) None ST Y, Rr Store Indinget and Pr	LDI	Rd, K	Load Immediate	$Rd \leftarrow K$	None	1
LD Rd, Y Load Indired are Pro-Dec. X ← X + 1, Rd ← (Y), Nome LD Rd, Y Load Indired are Pro-Dec. Rd ← (Y), Y ← Y + 1, Nome LD Rd, Y Load Indired are Pro-Dec. Y ← Y + 1, Rd ← (Y), Nome LDD Rd, Y Load Indired are Pro-Dec. Y ← Y + 1, Rd ← (Y), Nome LD Rd, Z Load Indired are Pro-Dec. Z + ∠ + 1, Rd ← (Z), Nome LD Rd, Z Load Indired are Pro-Dec. Z + ∠ + 1, Rd ← (Z), Nome LD Rd, Z Load Indired are Pro-Dec. Z + ∠ + 1, Rd ← (Z), Nome LDS Rd, X Load Dreed from SMAM Rd + (Z + q) Nome ST X, Rr Store Indired and Pro-Dec. Z + Z + 1, Rd + (Z, Z + 1) Nome ST X, Rr Store Indired and Pro-Dec. (Y + K + X + 1, R) Nome ST X, Rr Store Indired and Pro-Dec. (Y + K + 1, (Y) + Fr. Nome ST X, Rr Store Indired and Pro-Dec. (Y + K + 1, (Y) + Fr. Nome ST X, Rr					None	2
LD Rd, Y Load Indiget and Peeche. Rd + (Y) None LD Rd, Y Load Indiget and Peeche. Y + Y, 1, Rd + (Y) None LD Rd, Y Load Indiget and Peeche. Rd + (Q) None LD Rd, Z Load Indiget and Peeche. Rd + (Q) None LD Rd, Z Load Indiget and Peeche. Rd + (Q) None LD Rd, Z Load Indiget and Peeche. Rd + (Q) None LD Rd, Z Load Indiget and Peeche. Rd + (Q) None LDS Rd, R Load Indiget and Peeche. Rd + (Q) None ST X, Rr Store Indiget and Peoche. Y+ (X) None ST X, Rr Store Indiget and Peoche. Y+ Y+ 1, (Y) + Rr None ST Y, Rr Store Indiget and Peoche. Y+ Y+ 1, (Y) + Rr None ST Y, Rr Store Indiget and Peoche. Y+ Y+ 1, (Y) + Rr None ST Y, Rr Store Indiget and Peoche. Y+ Y+ 1, (Y) + Rr None <			Load Indirect and Post-Inc.		None	2
LD Rd. Y Load Indiana and PeoDec. Rd. + (Y, Y - Y + 1) None LDD Rd. Y Load Indiana with Displacement Rd + (Y, q) None LDD Rd. Z Load Indiana with Displacement Rd + (Y, q) None LD Rd. Z Load Indiana and PeoDec. Rd + (Z, Z + Z+1) None LD Rd. Z Load Indiana and PeoDec. Z + Z + 1, Rd + (Q) None LDS Rd. X Load Indiana and PeoDec. Z + Z + 1, Rd + (Q) None LDS Rd. X Load Orient from SAM Rd + (X + Q) None ST X, Rr Store Indiana and PeoDec. X + X + 1, (X) + Rr None ST X, Rr Store Indiana and PeoDec. (Y + X + X + 1) None ST X, Rr Store Indiana and PeoDec. (Y + K + Y + 1) None ST X, Rr Store Indiana and PeoDec. (Y + X + Y + Y + 1) None ST X, Rr Store Indiana and PeoDec. (Y + X + Y + Y + 1) None ST X, Rr Store Indiana and PeoDec.						2
LD Rs/Y-q Load Indust and Pa-bac. Y - Y - 1, Rd - (Y) - (Y) None LD Rs/Y-q Load Indust with Deplayment Rd + (Z) None LD Rs/Z Load Indust and Po-sho. Rd + (Z) None LD Rs/Z Load Indust and Po-sho. Rd + (Z) None LD Rs/Z Load Indust and Po-sho. Rd + (Z) None LD Rs/Z Load Indust and Po-sho. Rd + (Z) None LDS Rs/K Load Indust and Po-sho. Rd + (Z) None LDS Rs/K Stree Induct and Po-sho. (Y) + R' None ST X, R' Stree Induct and Po-sho. (Y) + R' None ST X, R' Stree Induct and Po-sho. (Y) + R' None ST X, R' Stree Induct and Po-sho. (Y + R') None ST X, R' Stree Induct and Po-sho. (Y + R') None ST X, R' Stree Induct and Po-sho. (Y + R') None ST Z, R'						2
LDD Bd_r / r_q Load Indiance with Desponsement $Bd_r - (2, r)$ NomeLD $Bd_r / 2$ Load Indiance and Post-bac. $Bd_r - (2)_r / - r / r / q)$ NomeLD $Bd_r / 2$ Load Indiance and Post-bac. $Z - r - 1, Bd_r - (2)_r / q)$ NomeLD $Bd_r / 2$ Load Indiance and Post-bac. $Z - r - 1, Bd_r - (2)_r / q)$ NomeLDD $Bd_r / 2 + q$ Load Indiance into Post-bac. $Z + r - 1, Bd_r - (2)_r / q)$ NomeLDS $Bd_r / r / q$ Load Diset from SBAM $Bd_r + (q)$ NomeST X_r / Rr Store Indiance and Post-bac. $X + X + 1 / r / R$ NomeST X_r / Rr Store Indiance and Post-bac. $Y + V + 1 / r / Rr$ NomeST Y_r / Rr Store Indiance and Post-bac. $Y' - V - 1, (y) - Rr$ NomeST Y_r / Rr Store Indiance and Post-bac. $Y' - V - 1, (y) - Rr$ NomeST Y_r / Rr Store Indiance and Post-bac. $Y' - V - 1, (y) - Rr$ NomeST Z_r / Rr Store Indiance and Post-bac. $Z' - Z + 1, (Z) - Rr$ NomeST Z_r / Rr Store Indiance and Post-bac. $Z' - Z + 1, (Z) - Rr$ NomeST Z_r / Rr Store Indiance and Post-bac. $Z' - Z + 1, (Z) - Rr$ NomeST Z_r / Rr Store Indiance and Post-bac. $Z' - Z + 1, (Z) - Rr$ NomeST Z_r / Rr Store Indiance and Post-bac. $Z' - Z + 1, (Z) - Rr$ NomeST Z_r / Rr Store Indiance and Post-bac. $Z' - Z + 2 + 1$ Nom						2
LDRd 2 Load Indirect and Post-Inc.Rd $-(2)$ NoneLDRd, ZLoad Indirect and Post-Inc.Rd $-(2)$ NoneLDRd, ZLoad Indirect and Post-Inc. $2 - 2 - 1$, Rd $-(2)$ NoneLDSRd, LLoad Indirect with DoplacementRd $-(2, q)$ NoneSTX, RrStore IndirectD(2 - Rr, X + X + 1)NoneSTX, RrStore Indirect and Post-Inc.D(2 - Rr, X + X + 1)NoneST-X, RrStore Indirect and Post-Inc.D(2 - Rr, X + X + 1)NoneST-Y, RrStore Indirect and Post-Inc.D(2 - Rr, X + X + 1)NoneST-Y, RrStore Indirect and Post-Inc.D(Y - Rr, Y + Y + 1)NoneST-Y, RrStore Indirect and Post-Inc.D(Y - Rr, Y + Y + 1)NoneST-Y, RrStore Indirect and Post-Inc.D(Y - Rr, Y + Y + 1)NoneST-Y, RrStore Indirect and Post-Inc.D(Y - Rr, Y + Y + 1)NoneST-Z, RrStore Indirect and Post-Inc.D(Y - Rr, Y + Y + 1)NoneST-Z, RrStore Indirect and Post-Inc.D(Y - Rr, Y + Y + 1)NoneST-Z, RrStore Indirect and Post-Inc.D(Y - Rr, Y + Y + 1)NoneST-Z, RrStore Indirect and Post-Inc.D(Y - Rr, Y + Y + 1)NoneST-Z, RrStore Indirect and Post-Inc.D(Y - Rr, Y + Y + 1)NoneST-Z, RrStore Indirect and Post-Inc.D(Y - Rr, Y + Y + 1)NoneST-Z, Rr <t< td=""><td></td><td></td><td></td><td></td><td></td><td>2</td></t<>						2
LDBd $2 +$ Load Indirect and Pro-Boc. $Bd + (2), 2 - 2.1$ NoneLDDBd $2 + q_1$ Load Indirect with Displacement $Bd + (2), 2 - 2.1$ NoneLDSBd k Load Direct with Displacement $Bd + - h)$ NoneSTX, RrStree Indirect and Pro-Boc. $(X) + Rr$ NoneSTX, RrStree Indirect and Pro-Boc. $(X) + Rr, X - X + 1 + 1$ NoneST-X, RrStree Indirect and Pro-Boc. $(Y) - Rr, Y - X + 1 + 1$ NoneST-Y, RrStree Indirect and Pro-Boc. $(Y) - Rr, Y - Y + 1$ NoneST-Y, RrStree Indirect and Pro-Boc. $(Y - Rr, Y - Y + 1)$ NoneST-Y, RrStree Indirect and Pro-Boc. $(Y - Rr, Y - Y + 1)$ NoneST-Y, RrStree Indirect and Pro-Boc. $(Y - Rr, Y - Y + 1)$ NoneST-Y, RrStree Indirect and Pro-Boc. $(Y - R, Y - Y + 1)$ NoneST-Y, RrStree Indirect and Pro-Boc. $(Y - R, Y - Y + 1)$ NoneST-Y, RrStree Indirect and Pro-Boc. $(Z - Rr, N - Z + Z + 1)$ NoneST-Z, RrStree Indirect and Pro-Boc. $(Z - Rr, N - Z + Z + 1)$ NoneST-Z, RrStree Indirect and Pro-Boc. $(Z - Rr, N - Z + Z + 1)$ NoneST-Z, RrStree Indirect and Pro-Boc. $(Z - Rr, N - Z + Z + 1)$ NoneST-Z, RrStore Indirect and Pro-Boc. $(Z - Rr, N - Z + Z + 1)$ NoneST-R, RrStore Indirect and Pro-Boc. $(Z - Rr, N - Z$			•			2
LD Rd. 2-q Load Indirect and Pro-Duc. 2 - 2 - 1, 16 - (2), None LDD Rd. 2-q Load Direct from SRAM Rd + (2 + q), None LDS Rd, k Load Direct from SRAM Rd + (2 + q), None ST X, Rr Store Indirect and Pros-Duc. (0) + Rr, X + X + 1 None ST X, Rr Store Indirect and Pros-Duc. X - X + 1, (1) + Rr None ST Y, Rr Store Indirect and Pros-Duc. X - X + 1, (1) + Rr None ST Y, Rr Store Indirect and Pros-Duc. Y - Y + 1, (1) + Rr None ST Y, Rr Store Indirect and Pros-Duc. Y - Y + 1, (1) + Rr None ST Y, Rr Store Indirect and Pros-Duc. Z - Z + 1, (2) + Rr None ST Z, Rr Store Indirect and Pros-Duc. Z - Z + 1, (2) + Rr None ST Z, Rr Store Indirect and Pros-Duc. Z - Z + 1, (2) + Rr None ST Z, Rr Store Indirect and Pros-Duc. Z - Z + 1, (2) + Rr None ST Z, Rr <						2
LDDR.d. 2r.q.Land Indirect with DisplacementRef $+(2+q)$ NoneLDSBd kLand Direct time SRAMRd $+(rb)$ NoneSTX, RrStore Indirect and Pac-Dac.(X) $+ R, X + X + 1$ NoneSTX, RrStore Indirect and Pac-Dac.X $+ X + 1, (X) + RrNoneSTY, RrStore Indirect and Pac-Dac.X + X + 1, (X) + RrNoneSTY, RrStore Indirect and Pac-Dac.(Y) + Rr, Y + Y + 1NoneSTY, RrStore Indirect and Pac-Dac.(Y) + Rr, Y - Y + 1NoneSTY, RrStore Indirect and Pac-Dac.(Y) + Rr, Y - Y + 1NoneSTY, RrStore Indirect and Pac-Dac.(Y) + Rr, Y - Y + 1NoneSTZ, RrStore Indirect and Pac-Dac.(Z) + Rr, Z - Z + 1NoneSTZ, RrStore Indirect and Pac-Dac.(Z) + Rr, Z - Z + 1NoneSTZ, RrStore Indirect and Pac-Dac.(Z) + Rr, Z - Z + 1NoneSTZ, RrStore Indirect and Pac-Dac.(Z) + Rr, Z - Z + 1NoneSTZ, RrStore Indirect and Pac-Dac.(Z) + Rr, Z - Z + 1NoneSTZ, RrStore Indirect and Pac-Dac.(Z) + Rr, Z - Z + 1NoneSTZ, RrStore Indirect and Pac-Dac.(Z) + Rr, Z - Z + 1NoneSTZ, RrStore Indirect and Pac-Dac.(Z) + Rr, Z - Z + 2NoneSTZ, RrStore Indirect and Pac-Dac.(Z) + Rr, Z - Z + 2NoneSTSto$						2
LDS R4, k Load Direct from SRAM R4 + (b) None ST X, Rr Store Indirect and Post-Inc. (b) + Rr, X + X + 1 None ST -X, Rr Store Indirect and Post-Inc. (b) + Rr, X + X + 1 None ST -X, Rr Store Indirect and Post-Inc. (b) + Rr, Y + Y + 1 None ST Y, Rr Store Indirect and Post-Inc. (Y) + Rr, Y + Y + 1 None ST Y, Rr Store Indirect and Post-Inc. (Y) + Rr, Y + Y + 1 None ST Y, Rr Store Indirect and Post-Inc. (Y) + Rr, Y + Y + 1 None ST Y, Rr Store Indirect with Displacement (Z) + Rr None ST Z, Rr Store Indirect and Post-Inc. (Z) + Rr None ST Z, Rr Store Indirect and Post-Inc. (Z) + Rr None ST Z, Rr Store Indirect and Post-Inc. (Z) + Rr None ST Z, Rr Store Indirect and Post-Inc. (Z) + Rr None ST Z, Rr Store Indinord Rd + Infor						2
STX, RrSince Indirect and Pre-Inc. $(0, + Rr, X + X + 1)$ NoneSTX, RrSince Indirect and Pre-Dec.X + X + 1, (λ) - RrNoneSTY, RrSince Indirect and Pre-Dec.Y + X + X + 1, (λ) - RrNoneSTY, RrSince Indirect and Pre-Dec.(Y) + Rr, Y - Y + 1NoneSTY, RrSince Indirect and Pre-Dec.Y + V + Y, (Y) + RrNoneSTY, RrSince Indirect and Pre-Dec.Y + V + Y, (Y) + RrNoneSTY, RrSince Indirect and Pre-Dec.Y + V + Y, (Y) + RrNoneSTZ, RrSince Indirect and Pre-Dec.(Z) + Rr, Z - Z + 1NoneSTZ, RrSince Indirect and Pre-Dec.(Z) + Rr, Z - Z + 1NoneSTZ, RrSince Indirect and Pre-Dec.Z - Z - Z + 1, (Z) - RrNoneSTZ, RrSince Indirect and Pre-Dec.Z - Z - Z + 1, (Z) - RrNoneSTZ, RrSince Indirect and Pre-Dec.Z - Z - Z + 1, (Z) - RrNoneSTZ, RrSince Indirect and Pre-Dec.Z - Z - Z + 1, (Z) - RrNoneSTK, RrSince Indirect and Pre-Dec.Z - Z + Z + 1, (Z) - RrNoneSTK, RrSince Indirect and Pre-Dec.Z - Z + Z + 1, (Z) - RrNoneSTLade Program MemoryR0 - (Z)NoneNoneSTSince Indirect and Pre-Dec.R - Z - Z + Z + Z + Z + Z + Z + Z + Z + Z						2
STX, RrStore indicet and Preschinc. $(2) \leftarrow P, X + X + 1$ NoneSTY, RrStore indicet and Prochem $X \leftarrow X + 1, (X) \leftarrow Rr$ NoneSTY, RrStore indicet and Posihinc. $(1') \leftarrow Rr, Y \leftarrow Y + 1$ NoneSTY, RrStore indicet and Posihinc. $(1') \leftarrow Rr, Y \leftarrow Y + 1$ NoneSTY, RrStore indicet and Posihinc. $(1') \leftarrow Rr, Y \leftarrow Y + 1$ NoneSTZ, RrStore indicet and Posihinc. $(2') \leftarrow Rr$ NoneSTZ, RrStore indicet and Procha $(2') \leftarrow Rr$ NoneSTZ, RrStore indicet and Proche $Z \leftarrow Z + 1, (Z) \leftarrow Rr$ NoneSTZ, RrStore indicet and Proche $Z \leftarrow Z + 1, (Z) \leftarrow Rr$ NoneSTSX, RrStore indicet and Proche $Z \leftarrow Z + 1, (Z) \leftarrow Rr$ NoneSTDZugRrStore indicet and Proche $Z \leftarrow Z + 1, (Z) \leftarrow Rr$ NoneLPMIncode Pogram MemoryR0 $\leftarrow (Z)$ NoneLPMRd, ZLoad Program MemoryR0 $\leftarrow (Z)$ NoneLPMRd, ZLoad Program MemoryRd $\leftarrow Rr$ NoneSPMRd, ZLoad Program MemoryRd $\leftarrow Rr$ NoneOUTP, RrOur PortRd $\leftarrow Rr$ NoneSPMRdPop Register from StackSTACK $\leftarrow Rr$ NoneOUTP, RrOur PortRd $\leftarrow Rr$ NoneDVTP, RrOur PortRd $\leftarrow Rr$ NoneCUTP, RrOur PortRd $\leftarrow Rr$ NoneDTADep Registe						2
ST X, Kr Store Indirect and Pos-Mc. $X \leftarrow X + 1, (Q) \in R^r$ NoneST Y, Rr Store Indirect and Post-Inc. $(Y) \leftarrow Rr, Y \leftarrow Y + 1$ NoneST Y, Rr Store Indirect and Post-Inc. $(Y \leftarrow Y + 1, (Y) \in R^r)$ NoneST Y, Rr Store Indirect and Post-Inc. $(Y \leftarrow Y + 1, (Y) \in R^r)$ NoneST Y, Rr Store Indirect and Post-Inc. $(Q) \leftarrow Rr$ NoneST Z, Rr Store Indirect and Post-Inc. $(Q) \leftarrow Rr, Z \leftarrow Z + 1.0$ NoneST Z, Rr Store Indirect and Post-Inc. $Z \leftarrow T + 1, (Q) \leftarrow Rr$ NoneST Z, Rr Store Indirect and Post-Inc. $Z \leftarrow T + 1, (Q) \leftarrow Rr$ NoneST Z, Rr Store Indirect and Post-Inc. $Z \leftarrow T + 1, (Q) \leftarrow Rr$ NoneST Z, Rr Store Indirect and Post-Inc. $Z \leftarrow T + 1, (Q) \leftarrow Rr$ NoneST Z, Rr Store Direct to SRAM $(R) \leftarrow Rr$ NoneST Z, Rr Store Direct to SRAM $(R) \leftarrow Rr$ NoneDPMRd, ZLoad Program MemoryRd \leftarrow QNoneDPMRd, ZLoad Program MemoryRd \leftarrow Q NoneNRd, PIn PortRd \leftarrow Q NoneNRd, PIn PortRd \leftarrow PNoneNRd, PDuri PortRd $\leftarrow Rr$ NoneDUTP, RrDup Register from StackSTACK $\leftarrow Rr$ NoneDUTP, RrDup Register from StackSTACK $\leftarrow Rr$ NoneDIT ALD BIT-TEST INSTRUCTIONSSTACK $\leftarrow Rr$ <t< td=""><td></td><td></td><td></td><td></td><td></td><td>2</td></t<>						2
ST V, Rr Store Indirect and Post-Inc. $(V) \leftarrow Rr$ NoneST V, Rr Store Indirect and Pre-Dec. $(V \leftarrow V + 1, (V) \leftarrow Rr)$ NoneSTD $Vq_R R$ Store Indirect and Pre-Dec. $(V \leftarrow V + 1, (V) \leftarrow Rr)$ NoneSTD Z, Rr Store Indirect and Pre-Dec. $(V \leftarrow V + 1, (V) \leftarrow Rr)$ NoneST Z, Rr Store Indirect and Pre-Dec. $(Q + Rr) < Z + 1$ NoneST Z, Rr Store Indirect and Pre-Dec. $(Z \leftarrow 2 - 1, IZ) \leftarrow Rr$ NoneST Z, Rr Store Indirect and Pre-Dec. $Z \leftarrow 2 - 1, IZ) \leftarrow Rr$ NoneSTS k, Rr Store Indirect and Pre-Dec. $Z \leftarrow 2 - 1, IZ) \leftarrow Rr$ NoneSTS k, Rr Store Indirect and Pre-Dec. $Z \leftarrow 2 - 1, IZ) \leftarrow Rr$ NoneDMLoad Frogram Memory $R0 \leftarrow IZ$ NoneNoneSTM k, Rr Store Program Memory and Posteric $Rd \leftarrow IZ$ NoneLPMRd, ZLoad Program Memory and Posteric $Rd \leftarrow IZ$ NoneSPMRdStore Program Memory and Posteric $Rd \leftarrow IZ$ NonePMRdPartNoneNoneNoneSPMRdPartPartNoneNonePMRdPartNoneNoneNonePMRdPartNoneNoneNoneSPMRdPartNoneNoneNonePDRdPartPartNoneNoneSPMRdPartPartNoneNone<						2
STY+RStore Indirect and Post-Inc.(Y)- $(R_1 \vee r) + 1$ NoneST-Y, R/rStore Indirect and Post-Dec.Y + $(Y_1, (Y) + Rr)$ NoneSTZ, R/rStore Indirect with Displacement(Y + $q) + Rr$ NoneSTZ, R/rStore Indirect and Post-Inc.(Z) - $(R, Z - Z + 1)$ NoneSTZ, R/rStore Indirect and Post-Inc.(Z) - $(R, Z - Z + 1)$ NoneSTZ, R/rStore Indirect and Post-Inc.(Z + $(R, Z - Z + 1)$ NoneSTZ-q, R/rStore Indirect and Post-Inc.(Z + $(R, Z - Z + 1)$ NoneSTDZ-q, R/rStore Indirect with Displacement(Z + $(Q + Rr)$ NoneSTSk, RrStore Direct to SRAMKlo + RrNoneLPMLoad Program MemoryRd + (Z)NoneLPMRd, ZLoad Program MemoryRd + (Z)NoneLPMRd, ZLoad Program MemoryRd + (Z)NoneNMRd, PIn PortRd + PNoneNMRd, PIn PortRd + PNoneNUTP, RrOut PortRd + STACKNonePCPRdPog Register for StackSTACK + RrNoneBIT AND BIT-TEST INSTRUCTIONSState In I/O RegisterUO(P, b) + 1NoneLSLRdLogical Shift LightRd(n) + Rd(n), Rd(n) - 0Z, C, NVROLRdRotate Left Through CarryRd(n) + Rd(n), Rd(n) - C, Z, C, NVSKRRdLogical Shift RightRd(n) + Rd(n), Rd(n) - C, Z, C, NV </td <td></td> <td></td> <td></td> <td></td> <td></td> <td>2</td>						2
ST···、KirStore Indirect and Pre-Dec. $Y + C_1(Y) - Rr$ NoneSTDY + Q, RrStore Indirect with Displacement $(Y + a) + Rr$ NoneSTZ, RrStore Indirect with Displacement $(Z) - Rr$ NoneSTZ, RrStore Indirect and Post-Inc. $(Z + Rr, Z - Z + 1)$ NoneSTZ, RrStore Indirect with Displacement $(Z + Cr, Z - 1, (Z) - Rr)$ NoneSTDZ, RrStore Indirect with Displacement $(Z + Q + Rr)$ NoneSTSk, RrStore Direct DSRAM $(k) + Rr$ NoneLPMTLoad Pogram MemoryRd - (Z)NoneLPMRd, ZLoad Pogram Memory and PostelncRd + (Z)NoneLPMRd, ZLoad Pogram Memory and PostelncRd + (Z)NoneSPMRdStore Program Memory and PostelncRd + RNoneOUTP, RrOul PortRd - PNonePUSHRrPug Register on StackSTACKNoneBIPug Register from StackRd + STACKNoneBIPubSet Bit in UO RegisterI/O(Pa) - 0NoneCBIP.bSet Bit in UO RegisterI/O(Pa) - 0NoneCBIRdLogical Shift HighRd(n) - Rd(n), Rd(n) - 0Z, NVROLRdLogical Shift HighRd(n - CR)(Rn(-), Rd(n))Z, NVROLRdRdAutimetic Shift RightRd(n - CR)(Rn(-), Rd(n+1), C-Rd(1))ROLRdRdSate Right Through CarryRd(n - C						2
STZ.RStore Indirect $(2) \leftarrow Rr$ NoneSTZ+, RrStore Indirect and Post-Inc. $(2) \leftarrow Rr, Z \leftarrow Z+1$ NoneSTZ, Q,RrStore Indirect and Pro-Dec.Z - Z - 1, (Z) - KrNoneSTDZ-Q,RrStore Indirect with Displacement $(Z + Q) \leftarrow Rr$ NoneSTSk, RrStore Direct to SRAM $(b) \leftarrow Rr$ NoneLPMLoad Program MemoryR0 - (Z)NoneLPMRd, ZLoad Program MemoryRd - (Z)NoneLPMRd, Z+Load Program MemoryRd + (Z)NoneSPMStore Program MemoryRd + (Z)NoneNoneSPMStore Program MemoryRd + (R)NoneNoneSPMStore Program MemoryRd + (R)NoneNoneOUTP, RrOut PortRd + PNoneNonePUSHRrPub Register on StackSTACK + RrNonePUSHRdPop Register from StackRd + STACK + RrNoneBIT AND BIT-TEST INSTRUCTIONSSet Bit in I/O RegisterI/O(P.b) + 0NoneCBIP,bClear Bit in U/O RegisterI/O(P.b) + 0NoneCBIP,bClear Bit in U/O RegisterI/O(P.b) + 0NoneCBIRdRdLogical Shift RightRd(n+1), Rd(n) + Rd(n+1), Rd(n) + Rd(n), Rd(n) + 0Z, C, N,VRORRdRdRotae Right Through CarryRd(1) + Rd(n+1), Rd(n), L, Rd(n) + Z, C, N,VRd(1) + Rd(n+1), Rd(n), Rd(n) + 0Z, C, N,VRORRd	ST		Store Indirect and Pre-Dec.		None	2
STZ+, RrStore Indirect and Post-Inc. $(Z) \leftarrow Rr, Z \leftarrow Z+1$ NoneST-Z, RrStore Indirect win Displacement $Z \leftarrow 2 + C + C r$ NoneSTDZ-q, RrStore Indirect win Displacement $(Z + q) \leftarrow -Rr$ NoneSTSk, RrStore Direct to SRAM(b) $\leftarrow Rr$ NoneLPMLoad Program MemoryR0 $\leftarrow (Z)$ NoneLPMRd, ZLoad Program MemoryRd $\leftarrow (Z)$ NoneLPMRd, ZLoad Program MemoryRd $\leftarrow (Z)$ NoneSPMStore Program MemoryRd $\leftarrow (Z)$ NoneNNRd, PIn PortRd $\leftarrow (Z)$ NoneOUTP, RrOut Port, and Memory and PoslencRd $\leftarrow (Z)$ NonePUSHRrPush Register on StackSTACK $\leftarrow Rr$ NonePOPRdPop Regram MemoryRd $\leftarrow STACK$ NonePUSHRrUot PortNoneNonePOPRdPop Register from StackSTACK $\leftarrow Rr$ NoneBIT AND BIT-TEST INSTRUCTIONSVO(P,b) $\leftarrow 1$ NoneNoneSISLP,bClear Bit in UO RegisterUO(P,b) $\leftarrow 1$ NoneLSLRdLogical Shift LeftRd(n+1) $\leftarrow Rd(n), Rd(0) \leftarrow 0$ Z,C,NVKNLRdAdmentes Shift RightRd(n) $\leftarrow Rd(n+1), Rd(n, C-Rd(T)$ Z,C,NVRORRdAdmentes Shift RightRd(n) $\leftarrow Rd(n+1), Rd(n, C-Rd(T)$ Z,C,NVRORRdAdmentes Shift RightRd(n) $\leftarrow Rd(n+1), Rd(n, C-Rd(T)$ Z,C,NVRORRdAdm	STD	Y+q,Rr	Store Indirect with Displacement	$(Y + q) \leftarrow Rr$	None	2
ST $2, Rr$ Store indirect with Displacement $Z \leftarrow 2 + 1, (Z) \leftarrow Rr$ NoneSTD $Z \leftarrow Q, Rr$ Store Direct to SRAM $(k) \leftarrow Rr$ NoneSTSk, RStore Direct to SRAM $(k) \leftarrow Rr$ NoneLPMMLoad Program MemoryR0 $\leftarrow (Z)$ NoneLPMRd, ZLoad Program MemoryRd $\leftarrow (Z), Z \leftarrow 241$ NoneLPMRd, Z+Load Program MemoryRd $\leftarrow (Z), Z \leftarrow 241$ NoneSPMStore Program MemoryRd $\leftarrow (Z), Z \leftarrow 241$ NoneSPMStore Program MemoryRd $\leftarrow P_Q, R k \in N$ NoneOUTP, RrOut PortRd $\leftarrow P_Q$ NonePUSHRrPush Register on StackSTACK $\leftarrow Rr$ NonePDPRdPope Register from StackRd $\leftarrow STACK$ NoneBIT AND BITTEST INSTUCTIONSSet Bit in UO RegisterUO(P,b) $\leftarrow 0$ NoneLSLRdLogical Shift RightRd(n+1) $\leftarrow Rd(n), Rd(0) \leftarrow 0$ Z,C,N/VRQP,bClear Bit in UO RegisterUO(P,b) $\leftarrow 0$ NoneLSLRdLogical Shift RightRd(n) $\leftarrow Rd(n+1), Rd(n), Ce-Rd(7)$ Z,C,N/VROLRdRdate Left Through CarryRd(n) $\leftarrow Rd(n+1), Rd(n), Ce-Rd(7)$ Z,C,N/VASRRdArithmetic Shift RightRd(n) $\leftarrow Rd(n+1), Rd(n), Ce-Rd(7)$ Z,C,N/VSWAPRdSwap NibblesRd(n) $\leftarrow Rd(n+1), Rd(n), Ce-Rd(7)$ Z,C,N/VASRRdArithmetic Shift RightRd(n) $\leftarrow Rd(n+1), Rd(n), Ce-Rd(1)$ Z,C,N/VSWAPRdR	ST	Z, Rr	Store Indirect	$(Z) \leftarrow Rr$	None	2
STDZ+q.RrStore Indicet with Displacement $(Z + q) \leftarrow Rr$ NoneSTSk, RrStore Direct to SRAM(k) $\leftarrow Rr$ NoneSTALoad Program MemoryR0 \leftarrow (Z)NoneLPMRd, ZLoad Program MemoryRd \leftarrow (Z)NoneLPMRd, ZLoad Program Memory and PositionRd \leftarrow (Z)NoneSTMTStore Program MemoryRd \leftarrow (Z) \leftarrow Rt R0NoneSPMTStore Program Memory(Z) \leftarrow Rt R0NoneOUTP, RrOut PortP \leftarrow RrNonePUSHRrPush Register on StackSTACk \leftarrow RrNonePUSHRrPush Register from StackRd \leftarrow STACkNoneBIT AND BIT-TEST INSTRUCTIONSStat In I/O RegisterI/O(P.b) \leftarrow 1NoneSISP.bClear Bit In I/O RegisterI/O(P.b) \leftarrow 0NoneLSLRdLogical Shift LightRd(n+1) \leftarrow Rd(n), Rd(n) \leftarrow 0Z,C,NVROLRdRdotate Light Intrough CarryRd(n+1) \leftarrow Rd(n), Rd(n), C+Rd(r), IZ, C,NVRORRdAttimetic Shift RightRd(r) \leftarrow Rd(n+1), C-Rd(n)Z,C,NVSWAPRdSwap NibblesRd(r) \leftarrow Rd(n+1), C-Rd(n), C \leftarrow Rd(n)Z,C,NVSWAPRdSwap NibblesRd(r) \leftarrow Rd(n+1), C-Rd(n), C \leftarrow Rd(n)Z,C,NVSWAPRdSwap NibblesRd(r) \leftarrow Rd(n+1), C-Rd(n), C \leftarrow Rd(n)Z,C,NVRORRdAttimetic Shift RightRd(r) \leftarrow Rd(n+1), C-Rd(n), C \leftarrow Rd(n)Z,C,NVSWAPRd <t< td=""><td>ST</td><td>Z+, Rr</td><td>Store Indirect and Post-Inc.</td><td>$(Z) \leftarrow Rr, Z \leftarrow Z + 1$</td><td>None</td><td>2</td></t<>	ST	Z+, Rr	Store Indirect and Post-Inc.	$(Z) \leftarrow Rr, Z \leftarrow Z + 1$	None	2
STSk, RrStore Direct to SRAM $(k) \leftarrow Rr$ NoneLPMLoad Program MemoryRd - (Z)NoneLPMRd, ZLoad Program MemoryRd - (Z)NoneLPMRd, Z+Load Program Memory and PoseIncRd - (Z)NoneStore Program MemoryRd - (Z)NoneNoneStore Program MemoryRd - (Z)NoneNoneStore Program MemoryRd - (Z)NoneNoneNRd, PIn PortRd - (Z)NoneOUTP, RrOut PortRd - PNonePUSHRrPus Register on StackSTACK - RrNonePDPRdPop Register from StackSTACK - RrNoneBIT AND BIT-TEST INSTRUCTIONSStatk - Rd(n), Rd(n) - 0Z, C, NVSILRdLogical Shift RightRd(n) - C, Rd(n+1), Rd(n), C, -dl(n)Z, C, NVLSLRdLogical Shift RightRd(n) - C, Rd(n+1), Rd(n), C, -Rd(n)Z, C, NVRORRdRotate Left Through CarryRd(n) - Rd(n), Al(r, Al), -Rd(n, C, Rd(n+1), C, Rd(n)Z, C, NVSWAPRdArithmeic Shift RightRd(n) - Rd(n), Al(r, Al), -Rd(n, C, Rd(n+1), C, Rd(n)Z, C, NVSWAPRdSwap NibblesSREG(s) - 1SREG(s)SREG(s)SESTsFlag SetSREG(s) -1SREG(s)SREG(s)SWAPRdSwap NibblesSREG(s) -1SREG(s)SREG(s)SESTsFlag SetSREG(s) -1SREG(s)SREG(s)SESTs	ST	-Z, Rr	Store Indirect and Pre-Dec.	$Z \leftarrow Z - 1, (Z) \leftarrow Rr$	None	2
LPMIcod Program Memory $R0 \leftarrow (2)$ NoneLPMRd, ZLoad Program Memory $Rd \leftarrow (2)$ NoneLPMRd, ZLoad Program Memory $Rd \leftarrow (2)$ $C \ge 1$ SPMImage: Comparison of the end of the en	STD	Z+q,Rr	Store Indirect with Displacement	$(Z + q) \leftarrow Rr$	None	2
LPMRd, ZLoad Program Memory and PosineRd $-(2)$ NoneLPMRd, Z+Load Program Memory and PosineRd $-(2)$, Z -2 21NoneSPMStore Program Memory and PosineRd $-(2)$, Z -2 21NoneSPMIn PortNoneRd $-P$ NoneOUTP, RrOut PortRd $-P$ NoneOUTP, RrOut PortP $-R$ NonePUSHRrPush Register on StackSTACK $-Rr$ NonePOPRdPop Register from StackSTACK $-Rr$ NoneBIT AND BIT-TEST INSTRUCTIONSSet Bit in I/O Register $I/O(P,b) \leftarrow 1$ NoneCBIP,bSet Bit in I/O Register $I/O(P,b) \leftarrow 0$ NoneLSLRdLogical Shift RightRd(n + CR(n+1), Rd(n) - 0Z,C,N/LSRRdLogical Shift RightRd(n + CR(n+1), Rd(n), Rd(n) - 0Z,C,N/RORRdRdRotate Left Through CarryRd(n + CR(n+1), C-Rd(n), Z,C,R/()Z,C,N/RORRdAnthreaic Shift RightRd(n - Rd(n+1), C-Rd(n), Z,C,R/()Z,C,N/RORRdAnthreaic Shift RightRd(n - Rd(n+1), C-Rd(n), Z,C,R/()Z,C,N/RORRdSate Right Through CarryRd(n - Rd(n+1), C-Rd(n), Z,C,R/()Z,C,N/RORRdSate Right Through CarryRd(n - Rd(n+1), C-Rd(n), C,C-Rd(n)Z,C,N/RORRdSate Right Through CarryRd(n - Rd(n+1), C-Rd(n), C,C-Rd(n)Z,C,N/SWAPRdSate Right Store from Register to TT - Rr(h)T -<	STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPMRd, Z+Load Program Memory and PoskIncRd \leftarrow (2), Z \leftarrow 2+1NoneSPMStore Program Memory(Z) \leftarrow Rt, R0NoneINRd, PIn PortRd \leftarrow PNoneOUTP, RrOut PortP \leftarrow RrNonePUSHRrPush Register on StackSTACK \leftarrow RrNonePOPRdPop Register from StackRd \leftarrow STACKNoneBIT AND BIT-TEST INSTRUCTIONSStablin I/O RegisterI/O(P,b) \leftarrow 1NoneSBIP,bClear Bit in I/O RegisterI/O(P,b) \leftarrow 0Z,C,N,VCBIP,bClear Bit in I/O RegisterI/O(P,b) \leftarrow 0Z,C,N,VROLRdLogical Shift RightRd(n) \leftarrow Rd(n+1), Rd(n), Ce-Rd(n)Z,C,N,VROLRdRotate Left Through CarryRd(n) \leftarrow Rd(n+1), Rd(n), Ce-Rd(n)Z,C,N,VASRRdArithmetic Shift RightRd(n) \leftarrow Rd(n+1), Rd(n), Ce-Rd(n)Z,C,N,VBSETsFlag ClearSREG(s) \leftarrow 0SREG(s)BSETsFlag ClearSREG(s) \leftarrow 0SREG(s)BSETsFlag ClearSREG(s) \leftarrow 0SREG(s)BSETs <t< td=""><td></td><td></td><td></td><td>$R0 \leftarrow (Z)$</td><td>None</td><td>3</td></t<>				$R0 \leftarrow (Z)$	None	3
SPMStore Program Memory $(2) \leftarrow R1:R0$ NoneINRd, PIn PortRd \leftarrow PNoneOUTP, RrOut PortP.e. RrNonePUSHRrPush Register on StackSTACK \leftarrow RrNonePOPRdPop Register from StackSTACK \leftarrow RrNonePDFRdPop Register from StackSTACKNonePOFRdPop Register from StackSTACKNonePOFRdPop Register from StackSTACKNoneDET AND BIT-TEST INSTUCTIONSUO(P.b) \leftarrow 1NoneNoneCBIP,bClear Bit in UO RegisterUO(P.b) \leftarrow 0NoneCBIP,bClear Bit in UO RegisterUO(P.b) \leftarrow 0NoneCBIRdLogical Shift RightRd(n) \leftarrow Rd(n+1) \leftarrow Rd(n), C.etAd(7)Z,C.N.VLSLRdLogical Shift RightRd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0Z,C.N.VRORRdRotate Right Through CarryRd(n) \leftarrow Rd(n+1), Rd(n), Z,-Ed(n)Z,C.N.VASRRdArithmetic Shift RightRd(n) \leftarrow Rd(n)+1, Rd(n), C,-Ed(n)Z,C.N.VSWAPRdSwap NibblesRd(3, O) \leftarrow Rd(7, A), Rd(7						3
INRd, PIn PortRd + PNoneOUTP, RrOut PortP+RNoneOUTP, RrOut PortP+RNonePUSHRrPush Register on StackSTACK - RrNonePOPRdPop Register from StackRd - STACK - RrNoneBIT AND BIT-TEST INSTRUCTIONSSet Bit in I/O RegisterI/O(P,b) $\leftarrow 1$ NoneCBIP,bClear Bit in I/O RegisterI/O(P,b) $\leftarrow 0$ NoneLSLRdLogical Shift LeftRd(n) $\leftarrow Rd(n+1)$, Rd(7) $\leftarrow 0$ Z,C,N,VLSRRdLogical Shift RightRd(n) $\leftarrow Rd(n+1)$, Rd(7) $\leftarrow 0$ Z,C,N,VRORRdRotate Left Through CarryRd(7) $\leftarrow C.Rd(n) \leftarrow Rd(n+1)$, Cd(7)Z,C,N,VRORRdRotate Right Through CarryRd(3) $\leftarrow Rd(n+1)$, Rd(7), $\leftarrow Rd(n)$ Z,C,N,VRSWAPRdArithmetic Shift RightRd(1) $\leftarrow Rd(n+1)$, Rd(7), $\leftarrow Rd(n-1)$, $\leftarrow Rd(n)$ Z,C,N,VSWAPRdArithmetic Shift RightRd(3) $\leftarrow Rd(n-1)$, $=Rd(3,0)$ NoneBSETsFlag SetSREG(s) $\leftarrow 0$ SREG(s)BCLRsFlag SetSREG(s) $\leftarrow 0$ SREG(s)BLDRd, bBit Store from Register to TT $\leftarrow R(t)$ NoneSECSet Legative FlagN $\leftarrow 0$ NSEZISet Legative FlagN $\leftarrow 0$ NSEZISet Legative FlagZ $\leftarrow 1$ ZCL2Clear Zaro FlagZ $\leftarrow 0$ ZSEISet Set Zero FlagZ $\leftarrow 0$ Z		Rd, Z+				3
OUTP, RrOut PortP 4 RrNonePUSHRrPush Register on StackSTACK $+$ RrNonePOPRdPop Register from StackRd $+$ STACKNoneBIT AND BIT-TEST-INSTRUCTIONSSet Bit in I/O RegisterI/O(P,b) -1 NoneCBIP,bSet Bit in I/O RegisterI/O(P,b) -1 NoneCBIP,bClear Bit in I/O RegisterI/O(P,b) -0 NoneCBIP,bClear Bit in I/O RegisterI/O(P,b) -0 X.O.VLSLRdLogical Shift LeftRd(n) $+ Rd(n), Rd(0) + 0$ Z.C.N.VRNRRdLogical Shift RightRd(n) $- Rd(n+1), Rd(n), C + Rd(n)$ Z.C.N.VROLRdRotate Left Through CarryRd(0) $- C(Rd(n+1), n=0.6)$ Z.C.N.VSWAPRdArithmetic Shift RightRd(n) $- Rd(n+1), n=0.6$ Z.C.N.VSWAPRdSwap NibblesRd(3.0) $- Rd(n-4), Rd(7.4), Rd(7.4), Rd(3.0)$ NoneSETsFlag SetSREG(s) $- 0$ SREG(s)BSTRr, bBit Store from Register to TT $- Rr(b)$ TBLDRd, bBit load from T to RegisterRd(b) $- T$ NoneSECsSet Negative FlagN $- 0$ NSELIClear XeryC $- 0$ CCLNSet Negative FlagN $- 0$ NNSECSet Negative FlagZ $- 1$ Z $- 0$ SECSet Negative FlagS $- 0$ NCSETGlobal Interrupt EnableI $- 0$ </td <td></td> <td></td> <td></td> <td></td> <td></td> <td>-</td>						-
PUSHRrPush Register on StackSTACK \leftarrow RrNonePOPRdPop Rgister from StackRd \leftarrow STACKNoneBIT AND BIT-TSTUCTIONSSBIP.bSet Bit in I/O RegisterI/O(P,b) \leftarrow 1NoneCBIP.bClear Bit in I/O RegisterI/O(P,b) \leftarrow 0NoneLSLRdLogical Shift RightRd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0Z,C,N/LSRRdLogical Shift RightRd(n) \leftarrow Rd(n+1), C,Rd(7)Z,C,N/ROLRdRotate Left Through CarryRd(0) \leftarrow C,Rd(n+1), C,-Rd(7)Z,C,N/RORRdAntimetic Shift RightRd(n) \leftarrow Rd(n+1), C,-Rd(0)Z,C,N/SWAPRdAntimetic Shift RightRd(n) \leftarrow Rd(n+1), C,-Rd(0)Z,C,N/SWAPRdSwap NibblesRd(3,0) \leftarrow Rd(n+1), C,-Rd(0)Z,C,N/SWAPRdSwap NibblesRd(1) \leftarrow Rd(n+1), C,-Rd(0)Z,C,N/SWAPRdSwap NibblesRd(1) \leftarrow Rd(n+1), C,-Rd(0)TBLDsFlag ClearSREG(s) \leftarrow 1SREG(s)BLDRd, bBit load from T to Register to TT \leftarrow Rd(b) \leftarrow TNoneSECSet Negative FlagN \leftarrow 1NC <t< td=""><td></td><td></td><td></td><td></td><td></td><td>1</td></t<>						1
POPRdPop Register from StackRd \leftarrow STACKNoneBIT AND BIT-TEST INSTRUCTIONSSBIP,bSet Bit in I/O RegisterI/O(P,b) \leftarrow 1NoneCBIP,bClear Bit in I/O RegisterI/O(P,b) \leftarrow 0NoneLSLRdLogical Shift RightRd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0Z,C,N,VLSRRdLogical Shift RightRd(n) \leftarrow Rd(n+1), Rd(n), C+Rd(r), Z,C,N,VRORRdROLRdRotate Left Through CarryRd(7) \leftarrow C,Rd(n+1), C+Rd(n), C-Rd(r), Z,C,N,VRORRdAntate Right Through CarryRd(7) \leftarrow C,Rd(n) \leftarrow Rd(n+1), C+Rd(3,0)Z,C,N,VASRRdAntimmetic Shift RightRd(n) \leftarrow Rd(n+1), C+Rd(3,0)NoneBSETsFlag SetSREG(s) \leftarrow 1SREG(s)BSETsFlag SetSREG(s) \leftarrow 1SREG(s)BCLRsFlag ClearSREG(s) \leftarrow 0SREG(s)BSTRr, bBit load from T to Register to TT \leftarrow Rr(b)TBLDRd, bBit load from T to Register to TC \leftarrow 1CCLCClear CarryC \leftarrow 0CCSECSet CarryC \leftarrow 0NSESETMSet Negative FlagN \leftarrow 0NCLDClear Argative FlagN \leftarrow 0NSETMGlobal Interrupt EnableI \leftarrow 11CLCClear Set Set OrSet Set Set OrSet Set Set Set OrSet Set Set Set Set Set Set Set Set Set						1
BIT AND BIT-TEST INSTRUCTIONSSBIP.bSet Bit in I/O RegisterI/O(P,b) $\leftarrow 1$ NoneCBIP.bClear Bit in I/O RegisterI/O(P,b) $\leftarrow 0$ NoneLSLRdLogical Shift LeftRd(n+1) $\leftarrow Rd(n), Rd(0) \leftarrow 0$ Z,C,N,VLSRRdLogical Shift RightRd(n) $\leftarrow Rd(n+1), Rd(7) \leftarrow 0$ Z,C,N,VROLRdRotate Left Through CarryRd(0) $\leftarrow C,Rd(n+1), -Rd(n), C,-Rd(7)$ Z,C,N,VRORRdRotate Right Through CarryRd(0) $\leftarrow C,Rd(n+1), -Rd(n), C,-Rd(7)$ Z,C,N,VSWAPRdSwap NibblesRd(3, 0) $\leftarrow Rd(n+1), R-Rd(3, 0)$ NoneBSETsFlag SetSREG(s) $\leftarrow 0$ SREG(s)BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)BSTRr, bBit Store from Register to TT $\leftarrow Rr(b)$ TBLDRd, bBit load from T to RegisterRd(b) $\leftarrow T$ NoneSECSet CarryC $\leftarrow 0$ CCCLUClear Kegative FlagN $\leftarrow 0$ NSEZSet Set Set FlagSet Zero FlagZ $\leftarrow 0$ ZSEZGlobal Interrupt EnableI $\leftarrow 0$ 1CLIGlobal Interrupt DisableI $\leftarrow 0$ 1CLIGlobal Interrupt DisableS $\leftarrow 0$ SSESSet Signed Test FlagS $\leftarrow 0$ SSEVSet Signed Test FlagS $\leftarrow 0$ SSEVClear Signed Test FlagS $\leftarrow 0$ SSEVClear Signed Test FlagS $\leftarrow 0$ SSEV						2
SBIP,bSet Bit in I/O RegisterI/O (P,b) $\leftarrow 1$ NoneCBIP,bClear Bit in I/O RegisterI/O (P,b) $\leftarrow 0$ NoneLSLRdLogical Shift LeftRd(n+1) \leftarrow Rd(n), Rd(0) $\leftarrow 0$ Z,C,N,VLSRRdLogical Shift RightRd(n) \leftarrow Rd(n+1), Rd(7) $\leftarrow 0$ Z,C,N,VROLRdRotate Left Through CarryRd(0) \leftarrow C,Rd(n+1), Rd(7) $\leftarrow 0$ Z,C,N,VRORRdRd Rotate Right Through CarryRd(7) \leftarrow C,Rd(n+1), C,C-Rd(0)Z,C,N,VASRRdArithmetic Shift RightRd(n) \leftarrow Rd(n+1), n=0.6Z,C,N,VASWAPRdSwap NibolesRd(3, O) \leftarrow Rd(7, 4), Rd(7, 4)-Rd(3, 0)NoneBSETsFlag SetSREG(s) \leftarrow 1SREG(s)BCLRsFlag ClearSREG(s) \leftarrow 0SREG(s)BCLRsFlag ClearSREG(s) \leftarrow 0SREG(s)BLDRd, bBit load from T to Register to TT \leftarrow Rt(b)TSECSet CarryC \leftarrow 0CCCLCClear CarryC \leftarrow 0CCSENSet Negative FlagN \leftarrow 0NSEZSet Zero FlagZ \leftarrow 0ZZCLZClear Zero FlagZ \leftarrow 0ZSESGlobal Interrupt DisableI \leftarrow 01SESIGlobal Interrupt DisableI \leftarrow 01SESClear Signed Test FlagS \leftarrow 0SSESClear Signed Test FlagS \leftarrow 0SSESClear Signed Test FlagS	-		Fup Register from Stack	Ru ← STACK	None	2
CBIP,bClear Bit in I/O RegisterI/O(P,b) $\leftarrow 0$ NoneLSLRdLogical Shift LeftRd(n+1) \leftarrow Rd(n), Rd(0) $\leftarrow 0$ Z,C,N,VLSRRdLogical Shift RightRd(n) \leftarrow Rd(n+1), Rd(7) $\leftarrow 0$ Z,C,N,VROLRdRotate Left Through CarryRd(0) \leftarrow C,Rd(n+1), C,C,Rd(n), C, \leftarrow Rd(7)Z,C,N,VRORRdRotate Left Through CarryRd(7) \leftarrow C,Rd(n+1), C,C,Rd(n), C, \leftarrow Rd(7)Z,C,N,VASRRdArithmetic Shift RightRd(n) \leftarrow Rd(n+1), n=0.6Z,C,N,VSWAPRdSwap NibblesRd(3,0) \leftarrow Rd(r,4), Rd(7,4), ed(3,0)NoneBSETsFlag SetSREG(s) $\leftarrow 1$ SREG(s)BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)BLDRd, bBit Store from Register to TT \leftarrow Rt(b) $\leftarrow T$ NoneSECSet CarryC $\leftarrow 1$ CCCLCSet Set Agative FlagN $\leftarrow 1$ NCLUClear Negative FlagN $\leftarrow 1$ NCLZGlobal Interrupt EnableI $\leftarrow 1$ ICLZGlobal Interrupt EnableI $\leftarrow 0$ ISESSet Signed Test FlagS $\leftarrow 1$ SCL3Global Interrupt EnableI $\leftarrow 0$ SSESGlobal Interrupt EnableI $\leftarrow 0$ SSESSet Signed Test FlagS $\leftarrow 0$ SSEVSet Signed Test FlagS $\leftarrow 0$ SSEVSet Signed Test FlagS $\leftarrow 0$ S			Set Bit in I/O Register	$I(O(P h) \leftarrow 1$	None	2
LSLRdLogical Shift Left $Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0$ Z,C,N,VLSRRdLogical Shift Right $Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0$ Z,C,N,VROLRdRotate Left Through Carry $Rd(0) \leftarrow C, Rd(n+1), Rd(7) \leftarrow 0$ Z,C,N,VRORRdRotate Right Through Carry $Rd(0) \leftarrow C, Rd(n+1), Rd(n), C \leftarrow Rd(7)$ Z,C,N,VRORRdAntithmetic Shift Right $Rd(0) \leftarrow Rd(n+1), Rd(n), C \leftarrow Rd(0)$ Z,C,N,VSWAPRdAntithmetic Shift Right $Rd(n) \leftarrow Rd(n+1), n=0.6$ Z,C,N,VSWAPRdSwap Nibbles $Rd(30) \leftarrow Rd(74), Rd(74) \leftarrow Rd(30)$ NoneBSETsFlag SetSREG(s) $\leftarrow 1$ SREG(s)BCLRsFlag ClearSREG(s) $\leftarrow 1$ SREG(s)BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)BCLsFlag Set form Register to TT $\leftarrow Rr(b)$ TBLDRd, bBit load from T to RegisterRd(b) $\leftarrow T$ NoneSECSet CarryC $\leftarrow 0$ CCCLCClear CarryC $\leftarrow 0$ NSELSet CarryC $\leftarrow 0$ NSEZSet Set Set Vergative FlagN $\leftarrow 0$ NCL1Clear Zero FlagZ $\leftarrow 0$ ZSEISet Zero FlagZ $\leftarrow 0$ ZSEIGlobal Interrupt TableI $\leftarrow 0$ ISESSet Signed Test FlagS $\leftarrow 1$ SCL2Clear Signed Test FlagS $\leftarrow 0$ SSEVSet Signed Test FlagS $\leftarrow 0$ S<						2
LSRRdLogical Shift RightRd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0Z, C, N, VROLRdRotate Left Through CarryRd(0) \leftarrow C, Rd(n+1), \leftarrow Rd(n), C, \leftarrow Rd(7)Z, C, N, VRORRdRotate Right Through CarryRd(0) \leftarrow C, Rd(n+1), \leftarrow Rd(n), C, \leftarrow Rd(7)Z, C, N, VASRRdArithmetic Shift RightRd(n) \leftarrow Rd(n+1), n=0.6Z, C, N, VSWAPRdSwap NibblesRd(3, O), \leftarrow Rd(7, 4), Rd(7, 4), \leftarrow Rd(3, O)NoneBSETsFlag SetSREG(s) \leftarrow 1SREG(s)BCLRsFlag ClearSREG(s) \leftarrow 0SREG(s)BSTRr, bBit Store from Register to TT \leftarrow Rt(b) \leftarrow TNoneSECSet CarryC \leftarrow 1CCCLCLear CarryC \leftarrow 0CCSENSet Negative FlagN \leftarrow 1NCLNClear CarryZ \leftarrow 1ZCLNSet Zero FlagZ \leftarrow 0ZSEIGlobal Interrupt EnableI \leftarrow 1ICLIGlobal Interrupt DisableS \leftarrow 1SSESSet Signed Test FlagS \leftarrow 0SCLSLear Xoc Complement Overflow.V \leftarrow 0VSETNClear Twos Complement Overflow.V \leftarrow 0VSETNClear Twos Complement Overflow.V \leftarrow 0V						1
ROLRdRotate Left Through Carry $Rd(0) \leftarrow C, Rd(n+1) \leftarrow Rd(n), C \leftarrow Rd(7)$ Z, C, N, V RORRdRotate Right Through Carry $Rd(7) \leftarrow C, Rd(n+1), -Rd(n), C \leftarrow Rd(0)$ Z, C, N, V ASRRdArithmetic Shift Right $Rd(n) \leftarrow Rd(n+1), n = .6$ Z, C, N, V SWAPRdSwap Nibbles $Rd(3, .0) \leftarrow Rd(n, .4), Rd(7, .4), -Rd(3, .0)$ NoneBSETsFlag Set $SREG(s) \leftarrow -1$ SREG(s)BCLRsFlag ClearSREG(s) < -1						1
RORRdRotate Right Through Carry $Rd(7)\leftarrow C.Rd(n)\leftarrow Rd(n+1), C\leftarrow Rd(0)$ Z, C, N, V ASRRdArithmetic Shift Right $Rd(n)\leftarrow Rd(n+1), n=0.6$ Z, C, N, V SWAPRdSwap Nibbles $Rd(30)\leftarrow Rd(74), Rd(74)\leftarrow Rd(30)$ NoneBSETsFlag SetSREG(s) $\leftarrow 1$ SREG(s)BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)BSTRr, bBit Store from Register to TT $\leftarrow R(h)$ TBLDRd, bBit load from T to RegisterRd(b) $\leftarrow T$ NoneSECSet CarryC $\leftarrow 1$ CCCLCSet CarryC $\leftarrow -0$ CSENSet Negative FlagN $\leftarrow 0$ NSEZSet Set Set Set Set Set Set Set Set Set						1
ASRRdArithmetic Shift RightRd(n) \leftarrow Rd(n+1), n=06Z,C,N,VSWAPRdSwap NibblesRd(30) \leftarrow Rd(n+1), n=06Z,C,N,VSWAPRdSwap NibblesRd(30) \leftarrow Rd(n+1), n=06X,REG(30)BSETsFlag SetSREG(s) \leftarrow 1SREG(s)BCLRsFlag ClearSREG(s) \leftarrow 0SREG(s)BSTRr, bBit Store from Register to TT \leftarrow Rr(b)TBLDRd, bBit load from T to RegisterRd(b) \leftarrow TNoneSECSet CarryC \leftarrow 0CCCLCClear CarryC \leftarrow 0CSENSet Negative FlagN \leftarrow 1NCLNClear Negative FlagN \leftarrow 0NSEZSet Zero FlagZ \leftarrow 1ZCLZClear Zero FlagZ \leftarrow 0ZSEIGlobal Interrupt EnableI \leftarrow 11CLIGlobal Interrupt DisableS \leftarrow 1SSESSet Signed Test FlagS \leftarrow 0SSESSet Signed Test FlagS \leftarrow 0SSEVISet Twos Complement Overflow.V \leftarrow 1VCLVIClear Twos Complement Overflow.V \leftarrow 0VSETSet T in SREGT \leftarrow 1TT						1
SWAPRdSwap NibblesRd(30) \leftarrow Rd(74),Rd(74) \leftarrow Rd(30)NoneBSETsFlag SetSREG(s)SREG(s) \leftarrow 0SREG(s)BCLRsFlag ClearSREG(s) \leftarrow 0SREG(s)BSTRr, bBit Store from Register to TT \leftarrow Rr(b)TBLDRd, bBit load from T to RegisterRd(b) \leftarrow TNoneSECSet CarryC \leftarrow 1C \leftarrow 1CCLCLear CarryC \leftarrow 0CSESENSet Negative FlagN \leftarrow 0NCLNClear Arguiter FlagN \leftarrow 0NSEZSet Zero FlagZ \leftarrow 1ZCLZSet Zero FlagZ \leftarrow 0ZSEIGlobal Interrupt EnableI \leftarrow 11CLIGlobal Interrupt DisableS \leftarrow 1SSESSet Signed Test FlagS \leftarrow 0SSEVASet Twos Complement Overflow.V \leftarrow 1VSEVISet Twos Complement Overflow.V \leftarrow 0VSETSet Tin SREGT \leftarrow 1TT						1
BSETsFlag SetSREG(s)SREG(s)SREG(s)BCLRsFlag ClearSREG(s)SREG(s)SREG(s)BSTRr, bBit Store from Register to TTTR(b)BLDRd, bBit Ioad from T to RegisterRd(b) \leftarrow TNoneSECSet CarryC<<1	SWAP	Rd	Swap Nibbles		None	1
BCLRsFlag ClearSREG(s) $\leftarrow 0$ SREG(s)BSTRr, bBit Store from Register to TT \leftarrow Rr(b)TBLDRd, bBit load from T to RegisterRd(b) \leftarrow TNoneSECSet CarryC $\leftarrow 1$ CCLCLear CarryC $\leftarrow 0$ CSENSet Negative FlagN $\leftarrow 1$ NCLNClear Negative FlagN $\leftarrow 0$ NSEZSet Zero FlagZ $\leftarrow 1$ ZCLZClear Setro FlagZ $\leftarrow 0$ ZCLZGlobal Interrupt EnableI $\leftarrow 1$ ICLIGlobal Interrupt DisableI $\leftarrow 0$ ISESSet Set Signed Test FlagS $\leftarrow 0$ SCLSLear Signed Test FlagS $\leftarrow 0$ SSEVSet Twos Complement Overflow.V $\leftarrow 0$ VSETSet T in SREGT $\leftarrow 1$ T						1
BLDRd, bBit load from T to RegisterRd(b) \leftarrow TNoneSECSet CarryC \leftarrow 1CCLCClear CarryC \leftarrow 0CSENSet Negative FlagN \leftarrow 1NCLNClear Negative FlagN \leftarrow 0NSEZSet Zero FlagZ \leftarrow 1ZCLZClear Zero FlagZ \leftarrow 0ZSEIGlobal Interrupt EnableI \leftarrow 1ICLIGlobal Interrupt EnableI \leftarrow 0ISESSet Signed Test FlagS \leftarrow 1SCLSClear Signed Test FlagS \leftarrow 0SSEVSet Twos Complement Overflow.V \leftarrow 0VSETSet T in SREGT \leftarrow 1T	BCLR	S	Flag Clear	$SREG(s) \leftarrow 0$	SREG(s)	1
SECSet Carry $C \leftarrow 1$ C CLCClear Carry $C \leftarrow 0$ CSENSet Negative Flag $N \leftarrow 1$ NCLNClear Negative Flag $N \leftarrow 0$ NSEZSet Zero Flag $Z \leftarrow 1$ ZCLZClear Zero Flag $Z \leftarrow 0$ ZSEIGlobal Interrupt Enable $I \leftarrow 1$ ICLIGlobal Interrupt Enable $I \leftarrow 0$ ISESSet Signed Test Flag $S \leftarrow 1$ SCLSClear Signed Test Flag $S \leftarrow 0$ SSEVSet Twos Complement Overflow. $V \leftarrow 0$ V CLVClear Twos Complement Overflow $V \leftarrow 0$ V SETSet T in SREG $T \leftarrow 1$ T	BST	Rr, b	Bit Store from Register to T		Т	1
CLCClear Carry $C \leftarrow 0$ CSENSet Negative Flag $N \leftarrow 1$ NCLNClear Negative Flag $N \leftarrow 0$ NSEZSet Zero Flag $Z \leftarrow 1$ ZCLZClear Zero Flag $Z \leftarrow 0$ ZSEIGlobal Interrupt Enable $I \leftarrow 1$ ICLIGlobal Interrupt Disable $I \leftarrow 0$ ISESSet Signed Test Flag $S \leftarrow 1$ SCLSClear Signed Test Flag $S \leftarrow 0$ SSEVSet Twos Complement Overflow. $V \leftarrow 0$ V CLVClear Twos Complement Overflow $V \leftarrow 0$ V SETSet T in SREG $T \leftarrow 1$ T	BLD	Rd, b	Bit load from T to Register	$Rd(b) \leftarrow T$		1
SENSet Negative Flag $N \leftarrow 1$ NCLNClear Negative Flag $N \leftarrow 0$ NSEZSet Zero Flag $Z \leftarrow 1$ ZCLZClear Zero Flag $Z \leftarrow 0$ ZSEIGlobal Interrupt Enable $I \leftarrow 1$ ICLIGlobal Interrupt Disable $I \leftarrow 0$ ISESSet Signed Test Flag $S \leftarrow 1$ SCLSClear Signed Test Flag $S \leftarrow 0$ SSEVSet Twos Complement Overflow. $V \leftarrow 1$ V CLVClear Twos Complement Overflow $V \leftarrow 0$ V SETSet T in SREG $T \leftarrow 1$ T						1
CLNClear Negative Flag $N \leftarrow 0$ NSEZSet Zero Flag $Z \leftarrow 1$ ZCLZClear Zero Flag $Z \leftarrow 0$ ZSEIGlobal Interrupt Enable $I \leftarrow 1$ ICLIGlobal Interrupt Disable $I \leftarrow 0$ ISESSet Signed Test Flag $S \leftarrow 1$ SCLSClear Signed Test Flag $S \leftarrow 0$ SSEVSet Twos Complement Overflow. $V \leftarrow 1$ V CLVClear Twos Complement Overflow $V \leftarrow 0$ V SETSet T in SREG $T \leftarrow 1$ T	CLC					1
SEZSet Zero Flag $Z \leftarrow 1$ ZCLZClear Zero Flag $Z \leftarrow 0$ ZSEIGlobal Interrupt Enable $I \leftarrow 1$ ICLIGlobal Interrupt Disable $I \leftarrow 0$ ISESSet Signed Test Flag $S \leftarrow 1$ SCLSClear Signed Test Flag $S \leftarrow 0$ SSEVSet Twos Complement Overflow. $V \leftarrow 1$ V CLVClear Twos Complement Overflow $V \leftarrow 0$ V SETSet T in SREG $T \leftarrow 1$ T						1
CLZClear Zero Flag $Z \leftarrow 0$ ZSEIGlobal Interrupt Enable $I \leftarrow 1$ ICLIGlobal Interrupt Disable $I \leftarrow 0$ ISESSet Signed Test Flag $S \leftarrow 1$ SCLSClear Signed Test Flag $S \leftarrow 0$ SSEVSet Twos Complement Overflow. $V \leftarrow 1$ V CLVClear Twos Complement Overflow $V \leftarrow 0$ V SETSet T in SREG $T \leftarrow 1$ T						1
SEIGlobal Interrupt Enable $I \leftarrow 1$ ICLIGlobal Interrupt Disable $I \leftarrow 0$ ISESSet Signed Test Flag $S \leftarrow 1$ SCLSClear Signed Test Flag $S \leftarrow 0$ SSEVSet Twos Complement Overflow. $V \leftarrow 1$ V CLVClear Twos Complement Overflow $V \leftarrow 0$ V SETSet T in SREG $T \leftarrow 1$ T						1
CLIGlobal Interrupt Disable $I \leftarrow 0$ ISESSet Signed Test Flag $S \leftarrow 1$ SCLSClear Signed Test Flag $S \leftarrow 0$ SSEVSet Twos Complement Overflow. $V \leftarrow 1$ V CLVClear Twos Complement Overflow $V \leftarrow 0$ V SETSet T in SREG $T \leftarrow 1$ T		-			Z	1
SESSet Signed Test Flag $S \leftarrow 1$ SCLSClear Signed Test Flag $S \leftarrow 0$ SSEVSet Twos Complement Overflow. $V \leftarrow 1$ V CLVClear Twos Complement Overflow $V \leftarrow 0$ V SETSet T in SREG $T \leftarrow 1$ T						1
CLSClear Signed Test Flag $S \leftarrow 0$ SSEVSet Twos Complement Overflow. $V \leftarrow 1$ V CLVClear Twos Complement Overflow $V \leftarrow 0$ V SETSet T in SREG $T \leftarrow 1$ T						1
SEV Set Twos Complement Overflow. V ← 1 V CLV Clear Twos Complement Overflow V ← 0 V SET Set T in SREG T ← 1 T						1
$\begin{tabular}{ c c c c c } \hline CLV & Clear Twos Complement Overflow & V \leftarrow 0 & V \\ \hline SET & Set T in SREG & T \leftarrow 1 & T \\ \hline \end{array}$						1
SET Set T in SREG T ← 1 T						1
						1
minemonics Operation Plage		Operanda				
	winemonics	Operations	Description	Operation	riays	#Clocks





Instruction Set Summary (Continued)

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

ATmega8(L)

Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package ⁽¹⁾	Operation Range
8	2.7 - 5.5	ATmega8L-8AU ⁽²⁾ ATmega8L-8PU ⁽²⁾ ATmega8L-8MU ⁽²⁾	32A 28P3 32M1-A	Industrial (-40°C to 85°C)
16	4.5 - 5.5	ATmega8-16AU ⁽²⁾ ATmega8-16PU ⁽²⁾ ATmega8-16MU ⁽²⁾	32A 28P3 32M1-A	Industrial (-40°C to 85°C)

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

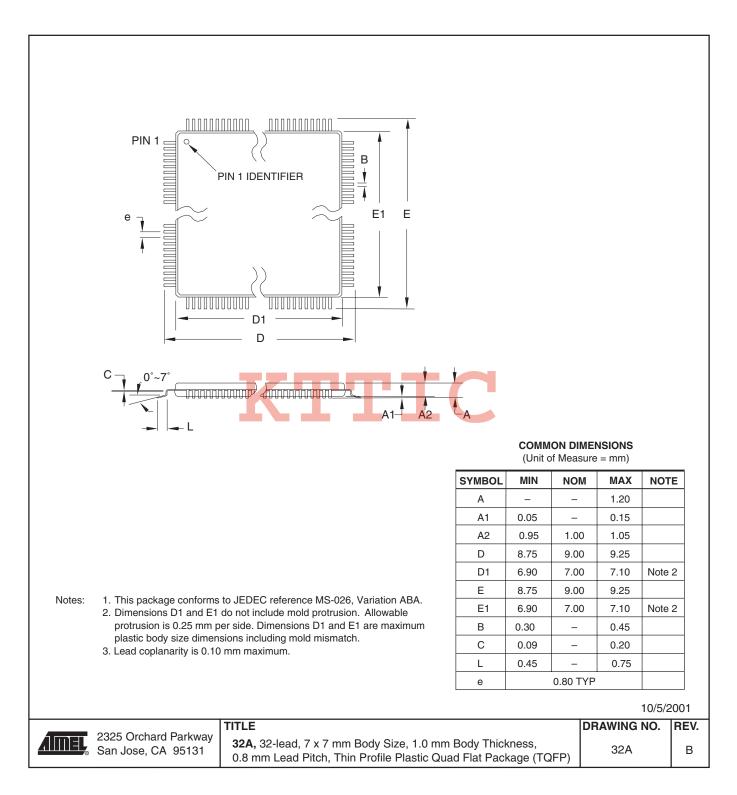
2. Pb-free packaging complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

	Package Type				
32A	32-lead, Thin (1.0 mm) Plastic Quad Flat Package (TQFP)				
28P3	28-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP)				
32M1-A	32-pad, 5 x 5 x 1.0 body, Lead Pitch 0.50 mm Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)				



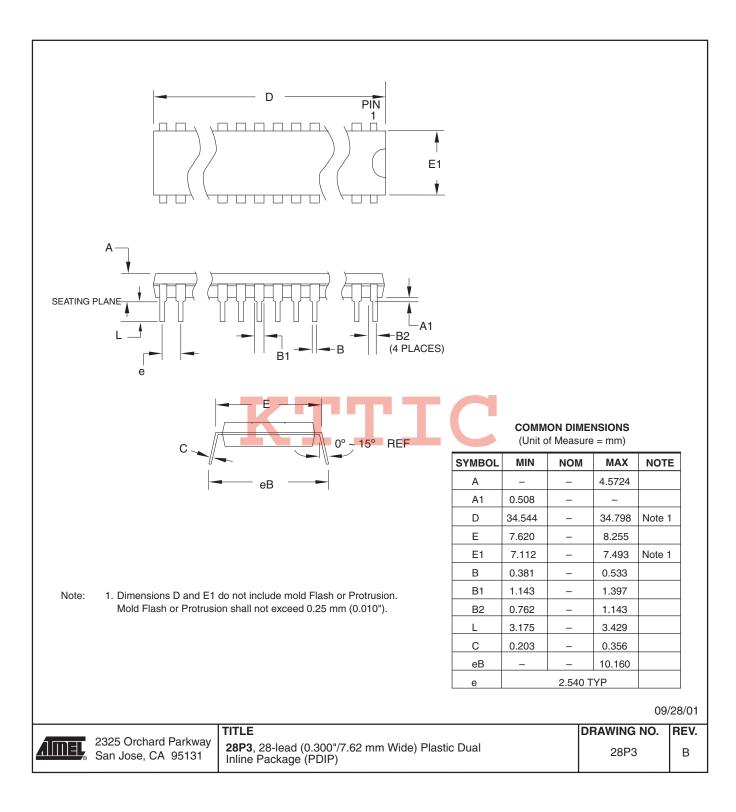
Packaging Information

32A



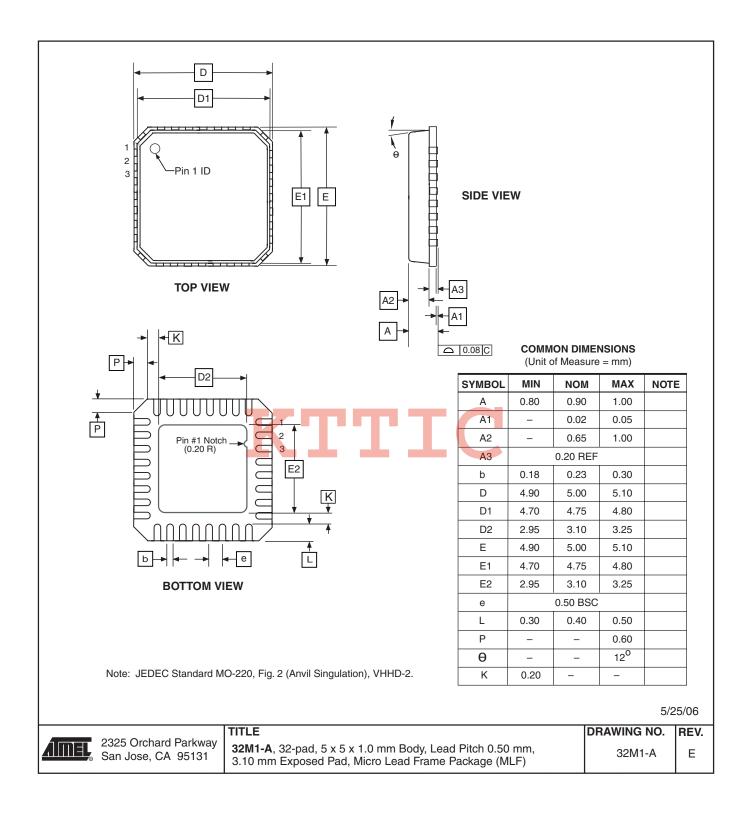
14 ATmega8(L)

28P3





32M1-A



Errata	The revision letter in this section refers to the revision of the ATmega8 device.
ATmega8	First Analog Comparator conversion may be delayed
Rev. D to I	 Interrupts may be lost when writing the timer registers in the asynchronous timer
	Signature may be Erased in Serial Programming Mode
	 CKOPT Does not Enable Internal Capacitors on XTALn/TOSCn Pins when 32 KHz Oscillator is Used to Clock the Asynchronous Timer/Counter2
	 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 theAnalog Comparator before the first conversion.

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

If one of the timer registers which is synchronized to the asynchronous timer2 clock is written in the cycle before a overflow interrupt occurs, the interrupt may be lost.

Problem Fix / Workaround

Always check that the Timer2 Timer/Counter register, TCNT2, does not have the value 0xFF before writing the Timer2 Control Register, TCCR2, or Output Compare Register, OCR2

3. Signature may be Erased in Serial Programming Mode

If the signature bytes are read before a chiperase command is completed, the signature may be erased causing the device ID and calibration bytes to disappear. This is critical, especially, if the part is running on internal RC oscillator.

Problem Fix / Workaround:

Ensure that the chiperase command has exceeded before applying the next command.

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

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

Problem Fix / Workaround

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

5. 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.





ATmega8(L)



Problem Fix / Workaround Always use OUT or SBI to set EERE in EECR.

KTTIC

18 ATmega8(L)

http://www.kttire.woom

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.	1. Updated Table 98 on page 240.
2486S- 08/07 to Rev. 2486T- 05/08	 2. Updated "Ordering Information" on page 292. - Commercial Ordering Code removed. - No Pb-free packaging option removed.
Changes from Rev.	1. Updated "Features" on page 1.
2486R- 07/07 to Rev. 2486S- 08/07	2. Added "Data Retention" on page 7.
	3. Updated "Errata" on page 17.
	4. Updated "Slave Mode" on page 129.
Changes from Rev.	1. Added text to Table 81 on page 218.
2486Q- 10/06 to Rev. 2486R- 07/07	2. Fixed typo in "Peripheral Features" on page 1.
	 Updated Table 16 on page 42. Updated Table 75 on page 206. Bernoved redundency and updated type in Notes section of "DC Characteristics" on
	5. Removed redundancy and updated typo in Notes section of "DC Characteristics" on page 242.
Changes from Rev.	1. Updated "Timer/Counter Oscillator" on page 32.
2486P- 02/06 to Rev. 2486Q- 10/06	2. Updated "Fast PWM Mode" on page 89.
	3. Updated code example in "USART Initialization" on page 138.
	4. Updated Table 37 on page 97, Table 39 on page 98, Table 42 on page 117, Table 44 on page 118, and Table 98 on page 240.
	5. Updated "Errata" on page 17.
Changes from Rev.	1. Added "Resources" on page 7.
2486O-10/04 to Rev. 2486P- 02/06	2. Updated "External Clock" on page 32.
	3. Updated "Serial Peripheral Interface – SPI" on page 124.
	4. Updated Code Example in "USART Initialization" on page 138.

ATmega8(L)

¹⁹ http://www.kttic.com



	5.	Updated Note in "Bit Rate Generator Unit" on page 170.
	6.	Updated Table 98 on page 240.
	7.	Updated Note in Table 103 on page 248.
	8.	Updated "Errata" on page 17.
Changes from Rev. 2486N-09/04 to Rev. 2486O-10/04	1.	Removed to instances of "analog ground". Replaced by "ground".
	2.	Updated Table 7 on page 29, Table 15 on page 38, and Table 100 on page 244.
	3.	Updated "Calibrated Internal RC Oscillator" on page 30 with the 1 MHz default value.
	4.	Table 89 on page 225 and Table 90 on page 225 moved to new section "Page Size" onpage 225.
	5.	Updated descripton for bit 4 in "Store Program Memory Control Register – SPMCR" on page 213.
	6.	Updated "Ordering Information" on page 13.
Changes from Rev.	1.	Added note to MLF package in "Pin Configurations" on page 2.
2486M-12/03 to Rev. 2486N-09/04	2.	Updated "Internal Voltage Reference Characteristics" on page 42.
	3.	Updated "DC Characteristics" on page 242.
	4.	ADC4 and ADC5 support 10-bit accuracy. Document updated to reflect this. Updated features in "Analog-to-Digital Converter" on page 196. Updated "ADC Characteristics" on page 248.
	5.	Removed reference to "External RC Oscillator application note" from "External RC Oscillator" on page 28.
Changes from Rev. 2486L-10/03 to Rev. 2486M-12/03	1.	Updated "Calibrated Internal RC Oscillator" on page 30.
Changes from Rev. 2486K-08/03 to Rev. 2486L-10/03	1.	Removed "Preliminary" and TBDs from the datasheet.
	2.	Renamed ICP to ICP1 in the datasheet.
	3.	Removed instructions CALL and JMP from the datasheet.
	4.	Updated t_{RST} in Table 15 on page 38, V_{BG} in Table 16 on page 42, Table 100 on page 244 and Table 102 on page 246.
	F	Persistent (VTAL1 and VTAL2 should be left unconnected (NC)" offer Table 0 in

5. Replaced text "XTAL1 and XTAL2 should be left unconnected (NC)" after Table 9 in "Calibrated Internal RC Oscillator" on page 30. Added text regarding XTAL1/XTAL2 and CKOPT Fuse in "Timer/Counter Oscillator" on page 32.

20 ATmega8(L)

http://www.kttirs.vpopp

http://www.kttic.com

	6.	Updated Watchdog Timer code examples in "Timed Sequences for Changing the Configuration of the Watchdog Timer" on page 45.
	7.	Removed bit 4, ADHSM, from "Special Function IO Register – SFIOR" on page 58.
	8.	Added note 2 to Figure 103 on page 215.
	9.	Updated item 4 in the "Serial Programming Algorithm" on page 238.
		Added t _{WD_FUSE} to Table 97 on page 239 and updated Read Calibration Byte, Byte 3, in Table 98 on page 240.
	11.	Updated Absolute Maximum Ratings* and DC Characteristics in "Electrical Character- istics" on page 242.
Changes from Rev.	1.	Updated V _{BOT} values in Table 15 on page 38.
2486J-02/03 to Rev. 2486K-08/03	2.	Updated "ADC Characteristics" on page 248.
	3.	Updated "ATmega8 Typical Characteristics" on page 249.
	4.	Updated "Errata" on page 17.
Changes from Rev.	1.	Improved the description of "Asynchronous Timer Clock – clkASY" on page 26.
2486I-12/02 to Rev. 2486J-02/03	2.	Removed reference to the "Multipurpose Oscillator" application note and the "32 kHz Crystal Oscillator" application note, which do not exist.
	3.	Corrected OCn waveforms in Figure 38 on page 90.
	4.	Various minor Timer 1 corrections.
	5.	Various minor TWI corrections.
	6.	Added note under "Filling the Temporary Buffer (Page Loading)" on page 216 about writing to the EEPROM during an SPM Page load.
	7.	Removed ADHSM completely.
	8.	Added section "EEPROM Write during Power-down Sleep Mode" on page 23.
	9.	Removed XTAL1 and XTAL2 description on page 5 because they were already described as part of "Port B (PB7PB0) XTAL1/XTAL2/TOSC1/TOSC2" on page 5.
	10.	Improved the table under "SPI Timing Characteristics" on page 246 and removed the table under "SPI Serial Programming Characteristics" on page 241.
	11.	Corrected PC6 in "Alternate Functions of Port C" on page 61.
	12.	Corrected PB6 and PB7 in "Alternate Functions of Port B" on page 58.
	13.	Corrected 230.4 Mbps to 230.4 kbps under "Examples of Baud Rate Setting" on page 159.



	14.	Added information about PWM symmetry for Timer 2 in "Phase Correct PWM Mode" on page 113.
	15.	Added thick lines around accessible registers in Figure 76 on page 169.
	16.	Changed "will be ignored" to "must be written to zero" for unused Z-pointer bits under "Performing a Page Write" on page 216.
	17.	Added note for RSTDISBL Fuse in Table 87 on page 223.
	18.	Updated drawings in "Packaging Information" on page 14.
Changes from Rev. 2486H-09/02 to Rev. 2486I-12/02	1.	Added errata for Rev D, E, and F on page 17.
Changes from Rev. 2486G-09/02 to Rev. 2486H-09/02	1.	Changed the Endurance on the Flash to 10,000 Write/Erase Cycles.
Changes from Rev. 2486F-07/02 to Rev. 2486G-09/02	1.	Updated Table 103, "ADC Characteristics," on page 248.
Changes from Rev.	1.	Changes in "Digital Input Enable and Sleep Modes" on page 55.
2486E-06/02 to Rev. 2486F-07/02	2.	Addition of OCS2 in "MOSI/OC2 – Port B, Bit 3" on page 59.
	3.	The following tables have been updated:
		Table 51, "CPOL and CPHA Functionality," on page 132, Table 59, "UCPOL Bit Settings," on page 158, Table 72, "Analog Comparator Multiplexed Input(1)," on page 195, Table 73, "ADC Conversion Time," on page 200, Table 75, "Input Channel Selections," on page 206, and Table 84, "Explanation of Different Variables used in Figure 103 and the Mapping to the Z-pointer," on page 221.
	4.	Changes in "Reading the Calibration Byte" on page 234.
	5.	Corrected Errors in Cross References.
Changes from Rev. 2486D-03/02 to Rev. 2486E-06/02	1.	Updated Some Preliminary Test Limits and Characterization Data The following tables have been updated: Table 15, "Reset Characteristics," on page 38, Table 16, "Internal Voltage Reference Char- acteristics," on page 42, DC Characteristics on page 242, Table , "ADC Characteristics," on
		page 248.
	2.	Changes in External Clock Frequency
		Added the description at the end of "External Clock" on page 32.
		Added period changing data in Table 99, "External Clock Drive," on page 244.
	3.	Updated TWI Chapter

http://www.kttig.voom

22 ATmega8(L)

More details regarding use of the TWI bit rate prescaler and a Table 65, "TWI Bit Rate Prescaler," on page 173.

Changes from Rev.	1.	Updated Typical Start-up Times.
2486C-03/02 to		The following tables has been updated:
Rev. 2486D-03/02		Table 5, "Start-up Times for the Crystal Oscillator Clock Selection," on page 28, Table 6, "Start-up Times for the Low-frequency Crystal Oscillator Clock Selection," on page 28, Table 8, "Start-up Times for the External RC Oscillator Clock Selection," on page 29, and Table 12, "Start-up Times for the External Clock Selection," on page 32.

2. Added "ATmega8 Typical Characteristics" on page 249.

Changes from Rev. 1. Updated TWI Chapter.

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

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

Added the note at the end of the "Bit Rate Generator Unit" on page 170.

Added the description at the end of "Address Match Unit" on page 170.

2. Updated Description of OSCCAL Calibration Byte.

In the datasheet, it was not explained how to take advantage of the calibration bytes for 2, 4, and 8 MHz Oscillator selections. This is now added in the following sections: Improved description of "Oscillator Calibration Register – OSCCAL" on page 31 and "Calibration Byte" on page 225.

3. Added Some Preliminary Test Limits and Characterization Data.

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

Table 3 on page 26, Table 15 on page 38, Table 16 on page 42, Table 17 on page 44, "TA = $-40 \times C$ to $85 \times C$, VCC = 2.7V to 5.5V (unless otherwise noted)" on page 242, Table 99 on page 244, and Table 102 on page 246.

4. Updated Programming Figures.

Figure 104 on page 226 and Figure 112 on page 237 are updated to also reflect that AV_{CC} must be connected during Programming mode.

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

Added a note in section "Enter Programming Mode" on page 228.

