Features

- High performance, low power AVR® 8-bit Microcontroller
- Advanced RISC architecture
 - 135 powerful instructions most single clock cycle execution
 - 32 × 8 general purpose working registers
 - Fully static operation
 - Up to 16MIPS throughput at 16MHz
 - On-chip 2-cycle multiplier
- Non-volatile program and data memories
 - 64/128Kbytes of in-system self-programmable flash
 - Endurance: 100,000 write/erase cycles
 - Optional Boot Code section with independent lock bits
 - USB boot loader programmed by default in the factory
 - In-system programming by on-chip boot program hardware activated after reset
 - True read-while-write operation
 - All supplied parts are pre-programed with a default USB bootloader
 - 2K/4K (64K/128K flash version) bytes EEPROM
 - Endurance: 100,000 write/erase cycles
 - 4K/8K (64K/128K flash version) bytes internal SRAM
 - Up to 64Kbytes optional external memory space
 - Programming lock for software security
- 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
- USB 2.0 full-speed/low-speed device and on-the-go module
 - Complies fully with:
 - Universal serial bus specification REV 2.0
 - On-the-go supplement to the USB 2.0 specification rev 1.0
 - Supports data transfer rates up to 12Mbit/s and 1.5Mbit/s
- USB full-speed/low speed device module with interrupt on transfer completion
 - Endpoint 0 for control transfers: up to 64-bytes
 - Six programmable endpoints with in or out directions and with bulk, interrupt or isochronous transfers
 - Configurable endpoints size up to 256bytes in double bank mode
 - Fully independent 832bytes USB DPRAM for endpoint memory allocation
 - Suspend/resume interrupts
 - Power-on reset and USB bus reset
 - 48MHz PLL for full-speed bus operation
 - USB bus disconnection on microcontroller request
- USB OTG reduced host:
 - Supports host negotiation protocol (HNP) and session request protocol (SRP) for OTG dual-role devices
 - Provide status and control signals for software implementation of HNP and SRP
 - Provides programmable times required for HNP and SRP
- Peripheral features
 - Two 8-bit timer/counters with separate prescaler and compare mode
 - Two16-bit timer/counter with separate prescaler, compare- and capture mode





8-bit Atmel Microcontroller with 64/128Kbytes of ISP Flash and USB Controller

AT90USB646 AT90USB647 AT90USB1286 AT90USB1287



- Real time counter with separate oscillator
- Four 8-bit PWM channels
- Six PWM channels with programmable resolution from 2 to 16 bits
- Output compare modulator
- 8-channels, 10-bit ADC
- Programmable serial USART
- Master/slave SPI serial interface
- Byte oriented 2-wire serial interface
- Programmable watchdog timer with separate on-chip oscillator
- On-chip analog comparator
- Interrupt and wake-up on pin change
- Special microcontroller features
 - Power-on reset and programmable brown-out detection
 - Internal calibrated oscillator
 - External and internal interrupt sources
 - Six sleep modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and packages
 - 48 programmable I/O lines
 - 64-lead TQFP and 64-lead QFN
- Operating voltages
 - 2.7 5.5V
- Operating temperature
- Industrial (-40°C to +85°C)
- Maximum frequency
 - 8MHz at 2.7V industrial range
 - 16MHz at 4.5V industrial range

1. Pin configurations



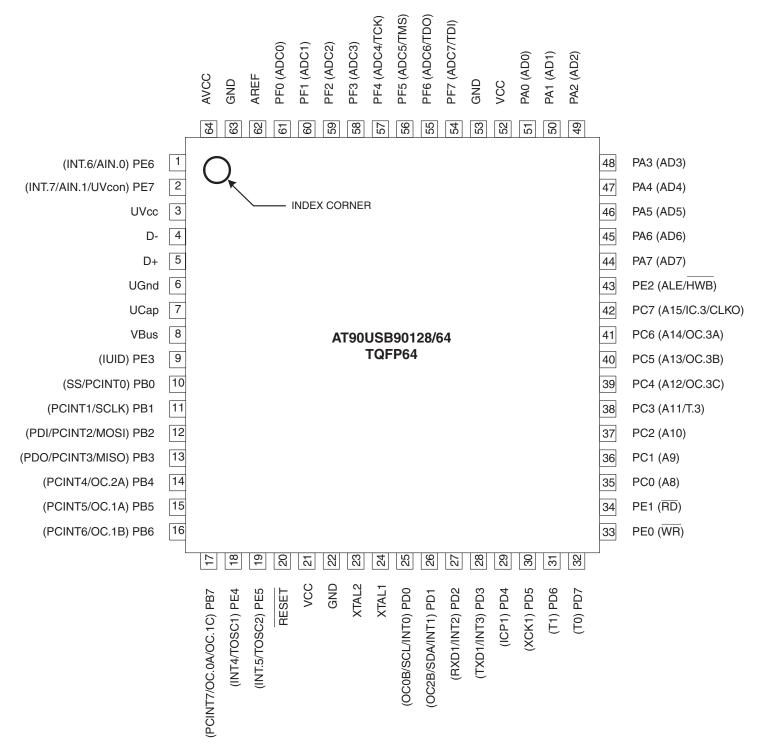
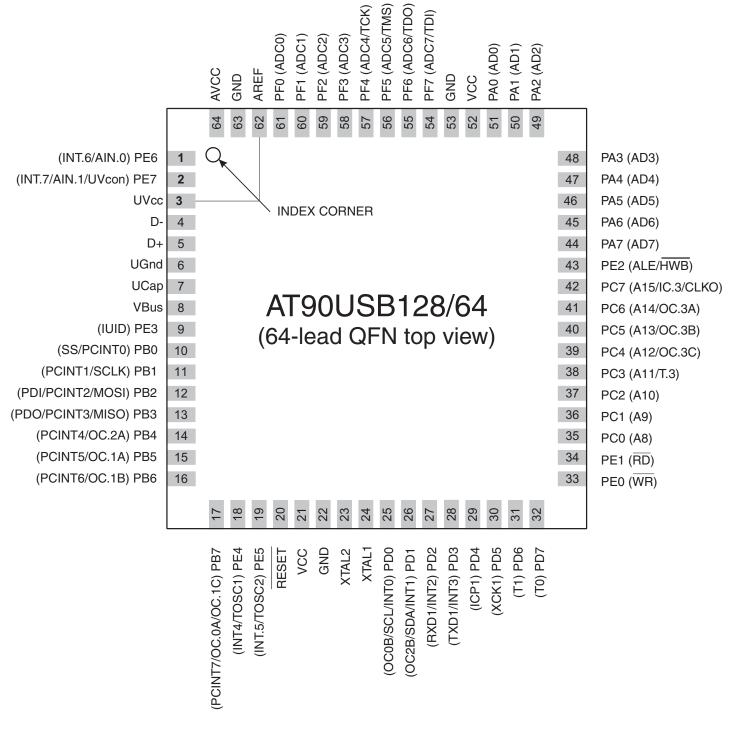






Figure 1-2. Pinout Atmel AT90USB64/128-QFN.



Note: The large center pad underneath the MLF packages is made of metal and internally connected to GND. It should be soldered or glued to the board to ensure good mechanical stability. If the center pad is left unconnected, the package might loosen from the board.

2. Overview

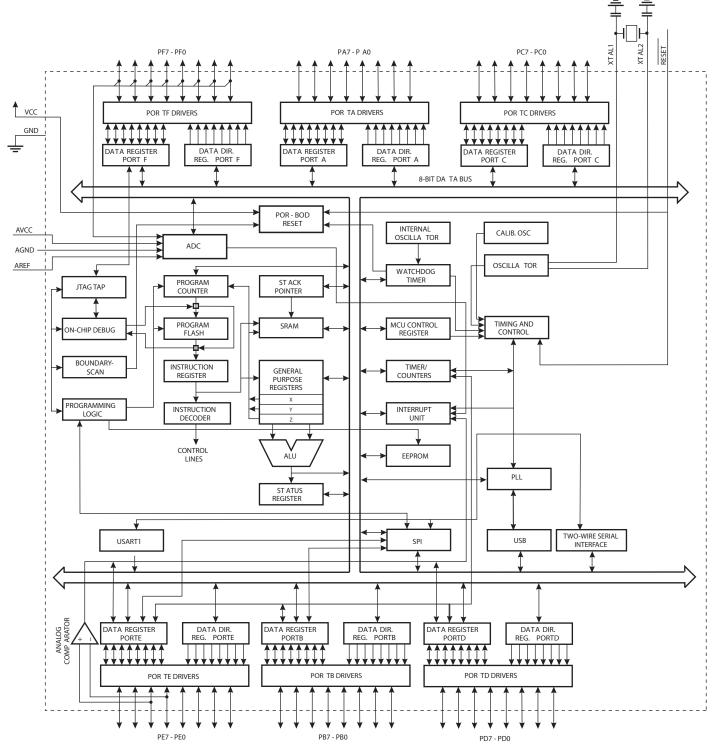
The Atmel® AVR® AT90USB64/128 is a low-power CMOS 8-bit microcontroller based on the Atmel® AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the AT90USB64/128 achieves throughputs approaching 1MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.





2.1 Block diagram

Figure 2-1. 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 Atmel AT90USB64/128 provides the following features: 64/128Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 2K/4Kbytes EEPROM, 4K/8K bytes SRAM, 48 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), four flexible Timer/Counters with compare modes and PWM, one USART, a byte oriented 2-wire Serial Interface, a 8-channels, 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 the Atmel high-density nonvolatile memory technology. The On-chip 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 AT90USB64/128 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The AT90USB64/128 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 Pin descriptions

- 2.2.1 VCC
 - Digital supply voltage.
- 2.2.2 GND

Ground.

2.2.3 AVCC

Analog supply voltage.

2.2.4 Port A (PA7..PA0)

Port A is an 8-bit bidirectional 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 Atmel AT90USB64/128 as listed on page 78.

2.2.5 Port B (PB7..PB0)

Port B is an 8-bit bidirectional 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 has better driving capabilities than the other ports.

Port B also serves the functions of various special features of the AT90USB64/128 as listed on page 79.

2.2.6 Port C (PC7..PC0)

Port C is an 8-bit bidirectional 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 AT90USB64/128 as listed on page 82.

2.2.7 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 AT90USB64/128 as listed on page 83.

2.2.8 Port E (PE7..PE0)

Port E is an 8-bit bidirectional 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 AT90USB64/128 as listed on page 86.

2.2.9 Port F (PF7..PF0)

Port F serves as analog inputs to the A/D Converter.

Port F also serves as an 8-bit bidirectional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port F output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port F pins that are externally pulled low will source current if the pull-up resistors are activated. The Port F pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PF7(TDI), PF5(TMS), and PF4(TCK) will be activated even if a reset occurs.

Port F also serves the functions of the JTAG interface.

2.2.10	D-	USB Full speed / Low Speed Negative Data Upstream Port. Should be connected to the USB D-connector pin with a serial 22 Ω resistor.
2.2.11	D+	USB Full speed / Low Speed Positive Data Upstream Port. Should be connected to the USB D+ connector pin with a serial 22 Ω resistor.
2.2.12	UGND	USB Pads Ground.
2.2.13	UVCC	USB Pads Internal Regulator Input supply voltage.
2.2.14	UCAP	USB Pads Internal Regulator Output supply voltage. Should be connected to an external capacitor (1 μ F).
2.2.15	VBUS	USB VBUS monitor and OTG negociations.
2.2.16	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 9-1 on page 58. Shorter pulses are not guaranteed to generate a reset.
2.2.17	XTAL1	Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.





2.2.18	XTAL2	Output from the inverting oscillator amplifier.
2.2.19	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.
2.2.20	AREF	This is the analog reference pin for the A/D Converter.

3. Resources

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

4. About code examples

This documentation contains simple code examples that briefly show how to use various parts of the device. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C compiler documentation for more details.

These code examples assume that the part specific header file is included before compilation. For I/O registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBRS", "SBRC", "SBR", and "CBR".

5. Register summary

AddressNameBit 7Bit 6Bit 5Bit 4Bit 3Bit 2Bit 1Bit 0Page(0x77)Nerves											
(Di-F) Failward · · <		Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(b, CP) Revnod · <			-	-	-	-	-	-	-	-	
00/FO Reserved - - -	(0xFE)	Reserved	-	-	-	-	-	-	-	-	
ON-POR (DR-F)PASEIIIIIII(DAF) (DR-F)OTTOCONPACEPACEPACEPACEPACEPACEPACEPACE(DAF) (DAF)UPECXIIIPACEPRITIPRITIPRITI(DAF) (DAF)UPECXIIIPRITIPRITIPRITI(DAF) (DAF)UPECXICOUNTRENDCRCNDPRITIDATAPICDATAPICDATAPIC(DAF) (DAF)UPECXICOUNTRENDCRCNDTMACOUTNATAPICDATAPICDATAPIC(DAF) (DAF)UPECXICOUNTRENDCRCNDPRITINAPRITINAPRITINA(DAF) (DAF)UPECXIIIIDATAPICDATAPICDATAPIC(DAF) (DAF)UPECXIIIIIIII(DAF) (DAF)UPECXIIIIIIII(DAF) (DAF)UPECXIIIIIIIIII(DAF) (DAF)UPECXII		Reserved	-	-	-	-	-	-	-	-	
(b)A.F.W. (b)A.F.W. (b)A.F.W.(b)A.F.W. (b)A.F.W.(b)A.F.W. (b)A.F.W.(b)A.F.W. (b)A.F.W.(b)A.F.W. (b)A.F.W.(b)A.F.W. (b)A.F.W.(b)A.F.W. (b)A.F.W.(b)A.F.W. (c)A.F.W. <th< td=""><td>(0xFC)</td><td>Reserved</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td></td></th<>	(0xFC)	Reserved	-	-	-	-	-	-	-	-	
(i) 0.76 TGC/DNPAGEPAGEPROFPROFVALUE(i) 0.76 TGC/DNVPROFPROF(i) 0.76 TGC/DNVPROFPROF(i) 0.76 TGC/DNV-PROFPROFNATATOLNATATOL(i) 0.76 TGC/DNVCOUNTERTOTIMEOUTPRONATATOLNATATOL(i) 0.76 TGC/DNVCOUNTERTONATATOLNATATOL(i) 0.76 TGC/DNVSTALLEONNATATOLNATATOLNATATOLNATATOL(i) 0.76 TGC/DNVNATATOLNATATOLNATATOL(i) 0.76 TGC/DNVNATATOLNATATOLNATATOLNATATOL(i) 0.76 TGC/DNV-NATATOLNATATOLNATATOL(i) 0.76 TGC/DNV-NATATOLNATATOLNATATOL(i) 0.76 TGC/DNV-NATATOLNATATOLNATATOL(i) 0.76 TGC/DNV-NATATOLNATATOLNATATOL(i) 0.76 TGC/DNV-NATATOLNATATOLNATATOL(i) 0.76 TGC/DNVFINTININATATOLNATATOLNATATOL(i) 0.76 TGC/DNNATATOLNATATOLNATATOLNATATOLNATATOL(i) 0.76 TGC/DNNATATOLNATATOLNATATOLNATATOLNATATOL(i) 0.76 TGC/DNNATATOLNATATOLNATATOLNATATOLNATATOL(i) 0.76 TGC/DNNATATOLNATATOLNATATOLNATATOL	(0xFB)	Reserved	-	-	-	-	-	-	-	-	
(b) (0) (7) UPNCY PRVT20 PRVT20 (0) (7) UPRCX PRVT20 PRVT20 PRVT20 (0) (6) UPRCX PRVT20 PRVT20 PRVT20 (0) (6) UPRCX CUNTER19 GC10 TIMEOUT PD DATARD DATARD (0) (6) UERCX STALLEDE EPNT20 PVT20 PVT20 (0) (7) UERCX STALLEDE TXME BVCT10 BVCT10 BVCT10 (0) (7) UERCX FLERRE NAKOUT FXXTPE BVCT10 CUURREND TXME (0) (7) UERCX CFCON VERTN ALLOC EVREND	(0xFA)	Reserved	-	-	-	-	-	-	-	-	
(intro) (intro) <t< td=""><td>(0xF9)</td><td>OTGTCON</td><td></td><td>PA</td><td>AGE</td><td></td><td></td><td></td><td>VA</td><td>LUE</td><td></td></t<>	(0xF9)	OTGTCON		PA	AGE				VA	LUE	
(b/b) UPECX UPECX UPECX DATAPL DATAPL DATAPL (b/b) UENT CUNTET10 FRC10 TINEOUT PU DATAPL DATAPL DATAPL (b/b) UERCX BYCT10 BYCT10 BYCT10 (b/b) UERCX NAKOUT FXXTPE BYCT10 BYCT10 (b/b) UERCX NAKOUT FXXTPE BYCT10 CUNRENCO (b/b) UERTX FLERRE NAKNE NAKOUT STALLED TXNE (b/b) UESTAX CTOCK OVERN UNCRN FRC10 ALLC EPPR10 (b/b) UEGTAX CTOCK OVERN STALLRO FSTD ALLC EPPR10 (b/b) UEGTAX CTOCK NERNIN RNAN NAKOUT FSTD FNUENCE EPPR10 (b/b) UENTX FSTALLRO FSTD ALLCC EPPR10 ALLCC TXIN	(0xF8)	UPINT				PI	NT7:0	-			
(a)GP3UPENK UPENOUNTEN 0COUNTEN 0COUNTEN 0POR 100DATATOL PUCPOR 100DATATOL DATATOL(b)GP3UERCLXPOR 100POR 100 <td>(0xF7)</td> <td>UPBCHX</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td></td> <td>PBYCT10:8</td> <td></td> <td></td>	(0xF7)	UPBCHX	-	-	-	-	-		PBYCT10:8		
Image: constraint of the const	(0xF6)	UPBCLX				PBY	'CT7:0	-		-	
(0.472) UEBCHX · · · PUCTIO (0.472) UEBCAX · · PARTO STALLED TANNOTE STALLED <td>(0xF5)</td> <td>UPERRX</td> <td>-</td> <td>COUN</td> <td>ITER1:0</td> <td>CRC16</td> <td>TIMEOUT</td> <td>PID</td> <td>DATAPID</td> <td>DATATGL</td> <td></td>	(0xF5)	UPERRX	-	COUN	ITER1:0	CRC16	TIMEOUT	PID	DATAPID	DATATGL	
(b/F) UBECLX UBECLX UBECLX Image: Destrict of the second o	(0xF4)	UEINT					EPINT6:0				
(inf) UERNX PLATE VANCUTE RNUTE STALED TALED	(0xF3)	UEBCHX	-	-	-	-	-		BYCT10:8		
(bF) UEINX FLEME NAMIE NAMOUTE RXDTE RXDUTE STALLEDE TAULE TAUE (0xE7) UESTAX GROK OVERFI UNDERFI DISC(1) CUENR(1) NULYPR(1) (0xE0) UESTAX GROK OVERFI UNDERFI DISC(1) ALCC PERMIN (0xE0) UECROX EFFVPE10 TAULROC RSTDT PERMIN PERMIN (0xE0) UECROX EFFVPE10 TAULROC RSTDT PENMIN (0xE0) UERMIN PIPCON NAKIN RVAL PERSTO PENMIN0 (0xE0) UENTX PIPCON NAKIN RVAL PENT PAUMI03 (0xE1) UDRNA PIPCON NAKIN RVALEUPE PORT PAUMI03 (0xE2) UDRNA ADDEN PIPCEAN VARCUPE EORSTI SOFE SUSPE (0xE1) UDRNA ADDEN PIPCEAN WARCUPE EORSTI SOFE SUSPE (0xE2) <t< td=""><td>(0xF2)</td><td>UEBCLX</td><td></td><td></td><td></td><td>BY</td><td>CT7:0</td><td></td><td></td><td></td><td></td></t<>	(0xF2)	UEBCLX				BY	CT7:0				
(INEE) UESTAXK C . CTRLOR CURRENTO CURRENTO CURRENTO (INEE) UESTAXK CFOK CVERTON EPSUZEO DTSECTO NALCO (INEE) UESCATIK EPTYPETO EPSUZEO EPSUZEO EPSUZEO (INEE) UESCATIK EPTYPETO STALLROC RSTDT EPNURACO (INEE) UESTAXK FETYPETO EPSUZEO EPRSTRO EPNURACO (INEG) UENTA FIFCOCM NAKINI RNAL NAKOUTI RXSTPI RXOUT EPNURACO (INEG) UENTA FIFCOCM NAKINI RNAL NAKOUTI RXSTPI RXOUT TAILEO TAINI (INEG) UDENUM EORSTME VAREUPE EORSTE SCPE SUSPE (INEG) UDENUM UPRSME EORSTME VAREUPE EORSTE SOPE SUSPE (INEG) UDENUM UPRSME EORSTME SOPE SUSPE SUSPE (INEG) UDENUM <t< td=""><td>(0xF1)</td><td>UEDATX</td><td></td><td></td><td></td><td>DA</td><td>AT7:0</td><td></td><td></td><td></td><td></td></t<>	(0xF1)	UEDATX				DA	AT7:0				
(abe) UESTAX CFGOK OVERFI UNDERFI INDERFI INDERFI NUMERFI NUMERFI ALLOS (bbc) UECPGAX EPYNPE10 V FRIT V PPNR (bbc) UECPGAX EPYNPE10 STALLEO RSTDT V PPNR (bbc) UESAT V STALLEO RSTDT PRSTP PRUM (bbc) UENAM V FRIT PRSTP PRUM108 TALLEO TALLEO (bbc) UENAM V PRUM108 FRUE PRUM108 TALLEO TALLEO (bbc) UDNFAM V PRUM108 FRUE PRUM108 TALLEO TALLEO (bbc) UDNFAM ADOEN VERSME EORSME VAREUPE EORSTI SOFE SUSPE (bbc) UDOCON VERSME EORSME VAREUPE EORSTI SOFE SUSPE (bbc) OTGOLM VERSME EORSTI SOFE SUSPE SUSPE	(0xF0)	UEIENX	FLERRE	NAKINE	-	NAKOUTE	RXSTPE	RXOUTE	STALLEDE	TXINE	
(bbc) UECFGX EPPKE10 EPPKE20 EPPK10 ALLOC Member 2000 (bbc) UECGGX STALLRQ STALLRQ RSTDT EPDR (bbc) UECGGX STALLRQ STALLRQ RSTDT EPN (bbc) UERST FFGCON NAKINI RWAL NAKOUTI RXSTPI RXOUTI STALLEDI TXINI (bbc) UENTX FFGCON NAKINI RWAL NAKOUTI RXSTPI RXOUTI STALLEDI TXINI (bbc) UDFNUM I <t< td=""><td>(0xEF)</td><td>UESTA1X</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>CTRLDIR</td><td>CURF</td><td>RBK1:0</td><td></td></t<>	(0xEF)	UESTA1X	-	-	-	-	-	CTRLDIR	CURF	RBK1:0	
(bCC) UECGXX EPTYFE10 Image: STALLROC RSTALLROC EPNUM120 TOTALLROC EPNUM120 (0x68) UERNTX FFCCON NAKINI RWAL NAKOUTI RXTFTI RXOUTI STALLROC FTALLROC TALLROC	(0xEE)	UESTA0X	CFGOK	OVERFI	UNDERFI	-	DTSE	EQ1:0	NBUS	YBK1:0	
(m.Ed)UECONKSTALLEOKSTALLEOKRETOTDEPR(m.Ed)UERSTNNNNACOUTIRXSTPIRXOUTISTALLEDITXINI(m.Ed)UENTKFIFOCONNAKINIRWALNAKOUTIRXSTPIRXOUTISTALLEDITXINI(m.Ed)UENTKFIFOCONNAKINIRWALNAKOUTIRXSTPIRXOUTISTALLEDITXINI(m.Ed)UDNUKFIFOCONNAKINIRWALNAKOUTIRXSTPISTALLEDITXINI(m.Ed)UDNUKUDNUKFIFOCONNAKINIFINUKIOSTT(m.Ed)UDNUKUDNUKUPSMEEORSMEMAKEUPEEORSTESOFISUSPE(m.Ed)UDNDRUPRSMEEORSMEWAKEUPEEORSTESOFISUSPE(m.Ed)UDNONUDNONUPRSMESTOIHNPERRIROLEEXIBCERRIVBERRISRPE(m.Ed)OTGINTSTOIHNPERRIROLEEXIBCERRIVBERRISRPE(m.DD)OTGINTSTOIHNPERRIROLEEXIBCERRIVBERRISRPE(m.DD)OTGINTSTOIHNPERRIROLEEXIBCERRIVBERRISRPE(m.DD)OTGINTSTOIHNPERRIROLEEXIBCERRIVBERRISRPE(m.DD)OTGINTSTOIHNPERRIROLEEXIBCERRIVBERRISRPE(m.DD)OTGINTSTOIHNPERRIROLEEXIBCERRIVBERRISRPE(m.DD)	(0xED)	UECFG1X			EPSIZE2:0		EPB	K1:0	ALLOC		
(b(E4) UENIX FIFOCON NAKINI RWAL NAKOUTI RXSTPI EVNUM2 EPNUM2:0 (b(66) UENIX FIFOCON NAKINI RWAL NAKOUTI RXSTPI FXOUTI STALLEDI TXINI (b(65) UDFNUM IDMAN IDMAN FNCERA FNUM108 FNUM108 (b(65) UDFNUM IDPNUM IDPNUM FNUM7:0 FNUM7:0 FNUM108 (b(65) UDFNUM UPRSME EORSME WAKEUPE EORSTE SOFE SUSPI (b(62) UDEN UPRSME EORSME WAKEUPE EORSTE SOFE SUSPI (b(62) UDEN UDRN UPRSME EORSME WAKEUPE EORSTE SOFI SUSPI (b(62) UDEN UDCON UPRSME EORSME WAKEUPE EORSTE SOFE SUSPI (b(62) OTGINT UPRSME EORSME WAKEUPE EORSTE SOFE SUSPI (b(60) OTGINT UPRSME	(0xEC)	UECFG0X	EPTY	'PE1:0				-	-	EPDIR	
(bE6)UENTXFIFCONNAKINIRWALNAKOUTRXSTPIRXOUTSTALLEDITXINI(0xE7)ReservedIIIIIIIIII(0xE6)UDINTNIII <td>(0xEB)</td> <td>UECONX</td> <td></td> <td></td> <td>STALLRQ</td> <td>STALLRQC</td> <td>RSTDT</td> <td></td> <td></td> <td>EPEN</td> <td>·</td>	(0xEB)	UECONX			STALLRQ	STALLRQC	RSTDT			EPEN	·
(bbs) UEINTX FIFOCON NAKINI FNAL NAKOUTI RXSTPI RXSUTI STALLEDI TXINI (0xE6) UDMPN -							EPRST6:0				
(b0:E) UEINTX FIFOCON NAKINI RWAL NAKOUTI RXSTPI RXSUTI STALLEDI TXINI (0xE6) UDMAN -									EPNUM2:0		
(bbf) Reserved Image: Second		UEINTX	FIFOCON	NAKINI	RWAL	NAKOUTI	RXSTPI	RXOUTI	STALLEDI	TXINI	
(bbc) UDRNIM Image: state of the second sec						-					
(br.E) UDFNUML Image: Constraint of the second sec						FNCERR					
(bc6) UDFNUML FNUM7.0 (wE3) UDADB ADDE UPASME EORSME WAKEUPE EORSTE SOFE SUSPE (bx62) UDINT UPASME EORSME WAKEUPE EORSTE SOFE SUSPE (bx61) UDINT UPSMI EORSME WAKEUPE EORSTE SOFI SUSPE (bx62) UDCON IPSMI EORSME WAKEUPE EORSTE SOFI SUSPE (bx0b) OTGIEN IPSMI STOE HNPERRI ROLEXX BCERRI VBERRE SRPE (bx0b) OTGIEN STOE HNPERE ROLEXX BCERRI VBERRE SRPE (bx0b) OTGIEN HNPERE SPEED IDTI VBUSRC VBUSRC (bx0b) UBSBIT IPSE SPEED ID VBUSTE (bx0b) UBSCN USBE HOST FR2CLK OTGPADE IDTE VBUSTE (bx0b) Beserved ID UVEGE ID </td <td>, ,</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>FNUM10:8</td> <td></td> <td></td>	, ,								FNUM10:8		
(IDE3) UDADBR ADDEN IVPRSME EORSME VARKEUPE EORSTE SOFE SUSPI (INE1) UDINT UPRSMI EORSME WAKEUP EORSTI SOFI SUSPI (INE1) UDINT UPRSMI EORSME WAKEUP EORSTI SOFI SUSPI (INE6) UDCON STOI HNPERRI ROLEEXI BCERRI VBERRI SRPI (INDE) OTGINT STOI HNPERRI ROLEEXI BCERRI VBERRE SRPE (INDD) OTGCON STOI HNPERRI ROLEEXI BCERRI VBERRE SRPE (INDD) OTGCON HNPERO SRPEQ SRPEQ VBUSHVC VBUSRQ VBUSRQ (INDD) OTGEON HNPERO SRPEQ SRPED IDT VBUSTI (INDD) Reserved IDT VBUSTI VBUSN IDT VBUSTI (INDD) UBBE HOST FRZCLK OTGPADE IDT VBUSTI (INDD) UBBE HOST FRZCLK OTGPADE IDT VBUSTE (INDD) UBBE HOST FRZCLK OTGPADE IDT VBUSTE (INDD) UBBERA IDT <						FNU	JM7:0				
(b/c2) UDEN UPRSME EORSME WAKEUPI EORSTI SOFI SUSPE (wE1) UDINT UPRSMI EORSMI WAKEUPI EORSTI SOFI SUSPE (wE0) UDCON ILSM RMWKUP DETACH (wDD) OTGIEN STOI HNPERRI ROLEEX BCERRI VBERRI SRPE (wDD) OTGIGON STOE HNPERRI ROLEEX BCERRI VBERRI SRPE (wDD) OTGGON HNPERQ SRPEQ SRPSEL VBUSHUC VBUSRQ VBUSRQ (wDD) Reserved ID HNPERQ SRPEQ SRPSEL VBUSHUC VBUSRQ (wDD) USSTA ID VBUS ID VBUS (wDA) USBCN USBE HOST FRZCLK OTGPADE ID VBUS (wDA) USBCN USBE HOST FRZCLK OTGPADE ID VBUS (wDA) USBCON USBE HOST F											
(bc1) UDINT UPRSMI EORSMI WAKEUPI EORSTI SOFI SUSPI (bc6) UDCON N RUMWLUP DETACH (bx6) OTGINT STOI HNPERRI ROLEEX BCERRI VBERRI SRPI (bx0b) OTGION STOI HNPERRI ROLEEXE BCERRI VBERRI SRPI (bx0b) OTGCON HNPERO SRPEQ SRPSLI VBERRI SRPE (bx0b) Reserved			ABBEN	LIPBSME	EOBSME	WAKELIPE	ř –	SOFE		SUSPE	
(b2E0) UDCON Image: Stress of the stress of											
(bxDF) OTGINT Image: STOI HNPERRI ROLEXI BCERN VBERRI SRPF (bxDD) OTGIN Image: STOI HNPERRE ROLEXE BCERRI VBERRI SRPF (bxDD) OTGCON Image: STOI HNPERO SRPEC VBUSRQ VBUSRQ (bxDC) Reserved Image: STOI HNPERO SRPEC VBUSNC VBUSRQ (bxDA) USBINT Image: STOI SRPEC IDT VBUSRQ VBUSRQ (bxD6) USBINT Image: STOI FRZCLK OTGPADE IDT VBUSRG (bxD6) USBCON UIBMO UDE IVCONE IDT VBUSRG (bxD6) Reserved Image: STRZLK OTGPADE IDTE VBUSRG				OFFICIAL	ECHOWI	WAREON	LONOTI		DMWKUD		
(bxDE) OTGLEN STOE HNPERPE ROLEXE BCERRE VBERRE SRPE (bxDD) OTGCON HNPRCO SRPREQ SRPSEL VBUSHWC VBUSHQC VBUSHQC (bxDD) Reserved Imprecision SRPEL VBUSHWC VBUSHQC VBUSHQC (bxDB) Reserved Imprecision Imprecision Imprecision Imprecision Imprecision Imprecision VBUSHWC VBUSHW					STOL	HNPERRI	BOLEEXI				
(bxDD) OTGCON Image: SRPEQ SRPEQ SRPSEL VBUSHWC VBUSRQC (bxDD) Reserved Image: SRPEQ SRPEQ SRPEQ VBUSHWC VBUSRQC (bxDB) Reserved Image: SRPEQ Image: SRPED											
(bxDC) Reserved Image: served served Image: served ser											
					HINFREQ	SHENEQ	SHFBEL	VBUSHWC	VBUSHEQ	VBUSHQU	
(bxDA)USBINTVIVIVIVIVIIIVIIIIVIIIIVIIIIVIIIIIVIIIIIVIIIII(bxDb)USBSTAIIIIIDVIIIIIDVIIIIIDVIIII(bxDb)USBCONUSBEHOSTFRZCLKOTGPADEIDTIDTVIIIIIVIIIIIVIIIIIIVIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII											
(bxD9) USBSTA USBE HOST FRZCLK OTGPADE ID VBUS (bxD8) USBCON USBE HOST FRZCLK OTGPADE IDTE VBUSTE (bxD6) Reserved UIMOD UIDE UVCONE IDTE VBUSTE (bxD5) Reserved IDTE VBUSTE IDTE VBUSTE (bxD5) Reserved IDTE VDREGE IDTE VDREGE (bxD4) Reserved IDTE IDTE VDREGE IDTE VDREGE (bxD4) Reserved IDTE IDTE IDTE IDTE IDTE (bxD4) Reserved IDTE IDTE IDTE IDTE IDTE (bxD2) Reserved IDTE									IDTI	VELICE	
(bxD8) USBCON USBE HOST FRZCLK OTGPADE IDTE VBUSTE (bxD7) UHWCON UIMOD UIDE UVCONE IDTE VVREGE (bxD6) Reserved IDTE VVREGE IVVREGE IVVREGE (bxD6) Reserved IDTE VVREGE IVVREGE IVVREGE (bxD6) Reserved IDTE VVREGE IVVREGE IVVREGE (bxD3) Reserved IDTE IVVREGE IVVREGE IVVREGE (bxD3) Reserved IDTE IVTREGE IVVREGE IVVREGE IVVREGE (bxD3) Reserved IDTE IVTREGE IVTREGE IVTREGE IVTREGE IVTREGE (bxD4) Reserved IDTE IVTREGE IVTREGE IVTREGE IVTREGE IVTREGE (bxD4) Reserved IDTE							00550				
(bxD7)UHWCONUIMODUIDEUVCONEImage: Constraint of the servedUVREGEUVREGE(bxD6)ReservedImage: Constraint of the servedImage:			LIODE	LIGOT	EDZOLIK	OTODADE	SPEED				
(bxD6) Reserved Image: constraint of the served Image: constr					FRZULK				IDTE		
(bxD5) Reserved Image: constraint of the served Image: constr			UIMOD	UIDE	-	UVCONE				UVREGE	
(0xD3) Reserved Image: constraint of the served Image: const											
(0xD2) Reserved · <											
(0xD0) Reserved · <			-	-	-	-	-	-	-	-	
(bxCF)Reserved <t< td=""><td></td><td></td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td></td></t<>			-	-	-	-	-	-	-	-	
(0xCE) UDR1 USART1 I/O Data Register (0xCD) UBR1H - - - USART1 Baud Rate Register High Byte (0xCC) UBR1L - - - USART1 Baud Rate Register Low Byte (0xCB) Reserved - - - - - - - (0xCB) Reserved - - - - - - - - (0xCA) UCSR1C UMSEL11 UMSEL10 UPM11 UPM10 USBS1 UCS211 UCSZ10 UCPOL1 (0xC3) UCSR1B RXCIE1 TXCIE1 UDRIE1 RXEN1 TXEN1 UCSZ12 RXB81 TXB81 (0xC8) UCSR1A RXC1 TXC1 UDRE1 FE1 DOR1 PE1 U2X1 MPCM1 (0xC6) Reserved - - - - - - - (0xC5) Reserved - - - - - - -	(0xD0)	Reserved	-	-	-	-	-	-	-	-	
(0xCD) UBR1H - - - USART1 Baud Rate Register High Byte (0xCC) UBR1L USART1 Baud Rate Register Low Byte USART1 Baud Rate Register Low Byte - (0xCB) Reserved -	(0xCF)		-	-	-	-	-	-	-	-	
(0xCC) UBR1L USARTI Baud Rate Register Low Byte (0xCB) Reserved -	(0xCE)	UDR1				USART1 I/C	Data Register				
(0xCB) Reserved - <	(0xCD)	UBRR1H	-	-	-	-	U	SART1 Baud Rat	e Register High E	Byte	
(NSC) NCSCI UMSEL11 UMSEL10 UPM11 UPM10 USBS1 UCSZ11 UCSZ10 UCPOL1 (0xC9) UCSR1B RXCIE1 TXCIE1 UDRIE1 RXEN1 TXEN1 UCSZ12 RXB81 TXB81 (0xC9) UCSR1A RXC1 TXCIE1 UDRIE1 RXEN1 TXEN1 UCSZ12 RXB81 TXB81 (0xC8) UCSR1A RXC1 TXC1 UDRE1 FE1 DOR1 PE1 U2X1 MPCM1 (0xC6) Reserved - <	(0xCC)	UBRR1L			l	JSART1 Baud Ra	ate Register Low I	Byte			
(0xC9) UCSR1B RXCIE1 TXCIE1 UDRIE1 RXEN1 TXEN1 UCSZ12 RXB81 TXB81 (0xC8) UCSR1A RXC1 TXC1 UDRE1 FE1 DOR1 PE1 U2X1 MPCM1 (0xC7) Reserved - - - - - - (0xC6) Reserved - - - - - - - (0xC6) Reserved - - - - - - - - (0xC6) Reserved -	(0xCB)	Reserved	-	-	-	-	-	-	-	-	· · · · · · · · · · · · · · · · · · ·
(0xC9) UCSR1B RXCIE1 TXCIE1 UDRIE1 RXEN1 TXEN1 UCSZ12 RXB81 TXB81 (0xC8) UCSR1A RXC1 TXC1 UDRE1 FE1 DOR1 PE1 U2X1 MPCM1 (0xC7) Reserved - - - - - - (0xC6) Reserved - - - - - - - (0xC6) Reserved - - - - - - - - (0xC6) Reserved -	(0xCA)	UCSR1C	UMSEL11	UMSEL10	UPM11	UPM10	USBS1	UCSZ11	UCSZ10	UCPOL1	
(0xC7) Reserved - <					UDRIE1	RXEN1	TXEN1		RXB81		
(0xC7) Reserved - <	. ,										
(0xC6) Reserved - <											
(0xC5) Reserved					-	-	-	-	-	-	
(0xC4) Reserved											
(0xC3) Reserved - <											
(0xC2) Reserved - <	, ,										
(0xC1) Reserved						1					
					1	1					
						1					
(0xBF) Reserved	(UXCU)								-		





Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xBE)	Reserved	-	-	-	-	-	-	-	-	
(0xBD)	TWAMR	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	-	
(0xBC)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	
(0xBB)	TWDR					erface Data Regis				
(0xBA)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	
(0xB9)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0	
(0xB8)	TWBR				-wire Serial Interf	ace Bit Rate Reg	ister		1	
(0xB7)	Reserved	-	-	-	-	-	-	-	-	
(0xB6)	ASSR	-	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB	
(0xB5)	Reserved	-	-	-	-	-	-	-	-	
(0xB4)	OCR2B					put Compare Reg				
(0xB3)	OCR2A			Tin		put Compare Reg	ister A			
(0xB2)	TCNT2					unter2 (8 Bit)				
(0xB1)	TCCR2B	FOC2A	FOC2B	-	-	WGM22	CS22	CS21	CS20	
(0xB0)	TCCR2A	COM2A1	COM2A0	COM2B1	COM2B0	-	-	WGM21	WGM20	
(0xAF)	UPDATX		1			AT7:0	1	1	1	
(0xAE)	UPIENX	FLERRE	NAKEDE	-	PERRE	TXSTPE	TXOUTE	RXSTALLE	RXINE	
(0xAD)	UPCFG2X		1	·	INTE	-RQ7:0		1		
(0xAC)	UPSTAX	CFGOK	OVERFI	UNDERFI			EQ1:0		SYBK1:0	
(0xAB)	UPCFG1X			PSIZE2:0		PBI	<1:0	ALLOC		
(0xAA)	UPCFG0X	PTY	PE1:0		EN1:0		PEPI	NUM3:0		
(0xA9)	UPCONX		PFREEZE	INMODE		RSTDT			PEN	
(0xA8)	UPRST					PRST6:0				
(0xA7)	UPNUM							PNUM2:0		
(0xA6)	UPINTX	FIFOCON	NAKEDI	RWAL	PERRI	TXSTPI	TXOUTI	RXSTALLI	RXINI	
(0xA5)	UPINRQX			•	INF	RQ7:0	•	•		
(0xA4)	UHFLEN				FLI	EN7:0				
(0xA3)	UHFNUMH							FNUM10:8		
(0xA2)	UHFNUML				FN	UM7:0				
(0xA1)	UHADDR					HADD6:0				
(0xA0)	UHIEN		HWUPE	HSOFE	RXRSME	RSMEDE	RSTE	DDISCE	DCONNE	
(0x9F)	UHINT		HWUPI	HSOFI	RXRSMI	RSMEDI	RSTI	DDISCI	DCONNI	
(0x9E)	UHCON						RESUME	RESET	SOFEN	
(0x9D)	OCR3CH			Timer/Co	unter3 - Output C	ompare Register				
(0x9C)	OCR3CL				· · ·	compare Register				
(0x9B)	OCR3BH					ompare Register				
(0x9A)	OCR3BL					Compare Register	* /			
(0x99)	OCR3AH					ompare Register				
(0x98)	OCR3AL					compare Register	• •			
(0x97)	ICR3H					Capture Register				
(0x96)	ICR3L					Capture Register				
(0x95)	TCNT3H					unter Register Hig	,			
(0x94)	TCNT3L					unter Register Lo				
(0x93)	Reserved	-	_	-	-		-	_	_	
(0x93) (0x92)	TCCR3C	- FOC3A	FOC3B	FOC3C	-	-	-	-	-	
						- WGM32				<u> </u>
(0x91) (0x90)	TCCR3B TCCR3A	ICNC3 COM3A1	ICES3 COM3A0	- COM3B1	WGM33 COM3B0	COM3C1	CS32 COM3C0	CS31 WGM31	CS30 WGM30	
, ,	Reserved	CONISAT	COWISAU	COWIGET	CONISBU	CONSCI	CONISCO	WGIVI31	VVGIVI30	<u> </u>
(0x8F)		-	-	-	-	-	-	-	-	
(0x8E)	Reserved	-	-		-	- compare Register		-	-	
(0x8D)	OCR1CH				•		0,			
(0x8C)	OCR1CL					Compare Register				
(0x8B)	OCR1BH					ompare Register				
(0x8A)	OCR1BL					Compare Register				
(0x89)	OCR1AH					ompare Register	• •			
(0x88)	OCR1AL					Compare Register				
(0x87)	ICR1H					Capture Register				
(0x86)	ICR1L					Capture Register				
(0x85)	TCNT1H					unter Register Hig				
(0x84)	TCNT1L				er/Counter1 - Co	unter Register Lo				
	Reserved	-	-	-	-	-	-	-	-	
(0x83)			FOC1B	FOC1C	-	-	-	-	-	
(0x82)	TCCR1C	FOC1A								
		FOC1A ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	
(0x82)	TCCR1C			- COM1B1	WGM13 COM1B0	WGM12 COM1C1	CS12 COM1C0	CS11 WGM11	CS10 WGM10	
(0x82) (0x81)	TCCR1C TCCR1B	ICNC1	ICES1	- COM1B1 -						
(0x82) (0x81) (0x80)	TCCR1C TCCR1B TCCR1A	ICNC1 COM1A1	ICES1 COM1A0		COM1B0	COM1C1	COM1C0	WGM11	WGM10	

			D ¹¹ 0		D :: 4	D ¹¹ 0	D ¹ 0	D !! 4	Dire	_
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x7C)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	
(0x7B)	ADCSRB	ADHSM	ACME	-	-	-	ADTS2	ADTS1	ADTS0	
(0x7A)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	
(0x79)	ADCH				ADC Data Re	egister High byte				
(0x78)	ADCL				ADC Data Re	egister Low byte				
(0x77)	Reserved	-	-	-	-	-	-	-	-	
(0x76)	Reserved	-	-	-	-	-	-	-	-	
(0x75)	XMCRB	XMBK	-	-	-	-	XMM2	XMM1	XMM0	
(0x74)	XMCRA	SRE	SRL2	SRL1	SRL0	SRW11	SRW10	SRW01	SRW00	
(0x73)	Reserved	-	-	-	-	-	-	-	-	
(0x72)	Reserved	-	-	-	-	-	-	-	-	
(0x71)	TIMSK3	-	-	ICIE3	-	OCIE3C	OCIE3B	OCIE3A	TOIE3	
(0x70)	TIMSK2	-	-	-	-	-	OCIE2B	OCIE2A	TOIE2	
(0x6F)	TIMSK1	-	-	ICIE1	-	OCIE1C	OCIE1B	OCIE1A	TOIE1	
(0x6E)	TIMSK0	-	-	-	-	-	OCIE0B	OCIE0A	TOIE0	
(0x6D)	Reserved	-	-	-	-	-	-	-	-	
(0x6C)	Reserved	-	-	-	-	-	-	-	-	
(0x6B)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	
(0x6A)	EICRB	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	
(0x69)	EICRA	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	
(0x68)	PCICR	-	-	-	-	-	-	-	PCIE0	
(0x67)	Reserved	-	-	-	-	-	-	-	-	
(0x66)	OSCCAL				Oscillator Cal	ibration Register				
(0x65)	PRR1	PRUSB	-	-	-	PRTIM3	-	-	PRUSART1	
(0x64)	PRR0	PRTWI	PRTIM2	PRTIM0	-	PRTIM1	PRSPI	-	PRADC	
(0x63)	Reserved	-	-	-	-	-	-	-	_	
(0x62)	Reserved	-	-	-	-	-	-	-	-	
(0x61)	CLKPR	CLKPCE	-	-	-	CLKPS3	CLKPS2	CLKPS1	CLKPS0	
(0x60)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	
0x3F (0x5F)	SREG		Т	Н	S	V	N	Z	C	
0x3E (0x5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	
0x3C (0x5C)	Reserved	-	-	-	-	-	-	-	-	
0x3B (0x5B)	RAMPZ	-	-	-	-	-	-	RAMPZ1	RAMPZ0	
0x3A (0x5A)	Reserved	-	-	-	-	-	-	-	-	
0x39 (0x59)	Reserved	-	-	-	-	-	-	-	-	
0x38 (0x58)	Reserved	-	-	-	-	-	-	-	-	
0x37 (0x57)	SPMCSR	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	
0x36 (0x56)	Reserved	-	-	-	-	-	-	-	-	
0x35 (0x55)	MCUCR	JTD		-	PUD			IVSEL	IVCE	
0x34 (0x54)	MCUSR	-	-	-	JTRF	WDRF	BORF	EXTRF	PORF	
0x33 (0x53)	SMCR	-	-	-	-	SM2	SM1	SM0	SE	
0x32 (0x52)	Reserved					ONIZ	0111	-	-	
0,02 (0,02)	OCDR/	OCDR7	OCDR6	OCDR5	OCDR4	OCDB3	OCDR2	OCDR1	OCDR0	
0x31 (0x51)	MONDR	CODITI	CODITO	CODITS		Data Register	CODITZ	CODITI	CODITIO	
0x30 (0x50)	ACSR		ACBG	400		ACIE		ACIS1	ACIS0	
0x30 (0x50) 0x2F (0x4F)	Reserved	ACD -	ACBG	ACO -	ACI	ACIE	ACIC	ACIS1	ACIS0	
0x2F (0x4F) 0x2E (0x4E)	SPDR	-	-	-	CDI Do	ta Register			-	
0x2E (0x4E) 0x2D (0x4D)	SPDR	SPIF	WCOL	-	SPI Da		-		SPI2X	
0x2D (0x4D) 0x2C (0x4C)	SPSR	SPIE	SPE	- DORD	- MSTR	- CPOL	- CPHA	- SPR1	SPI2X SPR0	
0x2C (0x4C) 0x2B (0x4B)	GPIOR2	GFIE	JFE	DOUD		use I/O Register 2		JITTI	Grinu	
0x2B (0x4B) 0x2A (0x4A)	GPIOR2 GPIOR1	l				*				
	PLLCSR	-	-	-	PLLP2	ose I/O Register 1 PLLP1	PLLP0	PLLE	PLOCK	
0x29 (0x49)		-	-					PLLE	PLUCK	
0x28 (0x48)	OCR0B				ner/Counter0 Out					
0x27 (0x47)	OCR0A			1 in	ner/Counter0 Out	put Compare Reg unter0 (8 Bit)	IISLEI A			
0x26 (0x46)	TCNT0	50004	FOOD			, ,	0000	0001	0000	
0x25 (0x45)	TCCR0B	FOC0A	FOCOB	-	-	WGM02	CS02	CS01	CS00	
0x24 (0x44)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0		-	WGM01	WGM00	
0x23 (0x43)	GTCCR	TSM	-	-	-	-	-	PSRASY	PSRSYNC	
0x22 (0x42)	EEARH	-	-	-	-			s Register High B	syte	
0x21 (0x41)	EEARL				EEPROM Addres	-	yte			
0x20 (0x40)	EEDR				1	Data Register				
0x1F (0x3F)	EECR	-	-	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	
0x1E (0x3E)	GPIOR0				1	ose I/O Register 0	1			
0x1D (0x3D)	EIMSK	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	
0x1C (0x3C)	EIFR	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0	





Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x1B (0x3B)	PCIFR	-	-	-	-	-	-	-	PCIF0	
0x1A (0x3A)	Reserved	-	-	-	-	-	-	-	-	
0x19 (0x39)	Reserved	-	-	-	-	-	-	-	-	
0x18 (0x38)	TIFR3	-	-	ICF3	-	OCF3C	OCF3B	OCF3A	TOV3	
0x17 (0x37)	TIFR2	-	-	-	-	-	OCF2B	OCF2A	TOV2	
0x16 (0x36)	TIFR1	-	-	ICF1	-	OCF1C	OCF1B	OCF1A	TOV1	
0x15 (0x35)	TIFR0	-	-	-	-	-	OCF0B	OCF0A	TOV0	
0x14 (0x34)	Reserved	-	-	-	-	-	-	-	-	
0x13 (0x33)	Reserved	-	-	-	-	-	-	-	-	
0x12 (0x32)	Reserved	-	-	-	-	-	-	-	-	
0x11 (0x31)	PORTF	PORTF7	PORTF6	PORTF5	PORTF4	PORTF3	PORTF2	PORTF1	PORTF0	
0x10 (0x30)	DDRF	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0	
0x0F (0x2F)	PINF	PINF7	PINF6	PINF5	PINF4	PINF3	PINF2	PINF1	PINF0	
0x0E (0x2E)	PORTE	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0	
0x0D (0x2D)	DDRE	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0	
0x0C (0x2C)	PINE	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0	
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	
0x08 (0x28)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	
0x07 (0x27)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	
0x06 (0x26)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	
0x02 (0x22)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	
0x01 (0x21)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	
0x00 (0x20)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	

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

2. I/O registers within the address range \$00 - \$1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.

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.

4. When using the I/O specific commands IN and OUT, the I/O addresses \$00 - \$3F must be used. When addressing I/O registers as data space using LD and ST instructions, \$20 must be added to these addresses. The Atmel AT90USB64/128 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from \$60 - \$1FF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

6. Instruction set summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
	ARITHMET	TIC AND LOGIC INSTRUCTIONS			1
ADD	Rd, Rr	Add two Registers	Rd ← 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 ← 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 ← 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	$Rd \gets Rd \lor K$	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ 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	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \bullet Rd$	Z,N,V	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	Z,N,V	1
SER	Rd	Set Register	$Rd \leftarrow 0xFF$	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x 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	$R1:R0 \leftarrow (Rd \times Rr) << 1$	Z,C	2
1 110200		RANCH INSTRUCTIONS		2,0	-
RJMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2
IJMP	ĸ	Indirect Jump to (Z)	PC ← Z	None	2
EIJMP		Extended Indirect Jump to (Z)	PC ←(EIND:Z)	None	2
JMP	k	Direct Jump	$PC \leftarrow k$	None	3
RCALL	k	Relative Subroutine Call	$PC \leftarrow PC + k + 1$	None	4
ICALL	ĸ	Indirect Call to (Z)	PC ← Z	None	4
EICALL		Extended Indirect Call to (Z)	PC ←(EIND:Z)	None	4
CALL	k	Direct Subroutine Call	$PC \leftarrow k$	None	5
RET	N	Subroutine Return	PC ← STACK	None	5
				None	5
RETI	D.(D)	Interrupt Return	$PC \leftarrow STACK$	Nerre	
CPSE	Rd,Rr	Compare, Skip if Equal	if $(Rd = Rr) PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC \leftarrow PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC \leftarrow PC + 2 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 \leftarrow PC + 2 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	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
	k	Branch if T Flag Cleared	if $(T = 0)$ then PC \leftarrow PC + k + 1	None	1/2
BRTC					





BRVC k Branch Interrupt Enabled If (I = 1) then PC + PC + BRID k Branch Interrupt Enabled If (I = 0) then PC + PC + BRID k Branch Interrupt Disabled If (I = 0) then PC + PC + BRID BRID Set Bit in IC Register IO(P.b) + -1 CBI P.b Set Bit in IC Register IO(P.b) + -1 CBI P.b Clear Bit in IC Register IO(P.b) + -1 CBI P.b Clear Bit in IC Register IO(P.b) + -1 CBI P.b Clear Bit in IC Register IO(P.b) + -1 CBI P.b Clear Bit in IC Register IO(P.b) + -1 LSL Rd Logical Shit Right Rd(n - Rd(n-h), Rd(n-L)) LSL Rd Rd Anthmole Shit Right Rd(n - Rd(n-h), Rd(n-L)) Start S Flig Set SREG(s) - 1 SREG(s) - 1 Start S Flig Set SREG(s) - 1 SREG(s) - 1 BLD Rd, b Bit Isot corn Register T T < Flight) SREG(s) - 1 SEC Set Cleary		
BRID k Branch Interrupt Deabled If (1=0) then PC + PC + BIT AND BIT-TEST INSTRUCTIONS I/O(PL) + 1 I/O(PL) + 1 CBI P.b Clear Bit in I/O Register I/O(PL) + 1 CBI P.b Clear Bit in I/O Register I/O(PL) + 1 CBI P.b Clear Bit in I/O Register I/O(PL) + 1 CBI Rd Logical Shift Left Rd(n) + Rd(n), Rd(n) LSL Rd Clear Bit In I/OR Register I/O(PL) + 1 DISL Rd Clear Bit In I/OR Register Rd(n) - Rd(n), Rd(n) ASR Rd Logical Shift Right Rd(n) - Rd(n), Rd(n) ASR Rd Anthretic Shift Right Rd(n) - Rd(n), Rd(n) BST Rr, b Bit Store from Register I T T - F. Rt(n) BLD Rd, b Bell Clear Carry C < + 0 SEC SEC Set Carry C < + 1 CL < 0 Clear Carry C < + 0 SEL Set Zarro Flag Z + - 1 Set Zarro Flag Z + - 1 CLA Clear Signed Test Flag	k + 1 None	1/2
BIT AND BIT-TEST INSTRUCTIONS I/O Pb SBI P.b Set Bit in I/O Register I/O(Pb) \leftarrow 1 CBI P.b Caure Bit in I/O Register I/O(Pb) \leftarrow 0 LSL Rd Logical Shift Hight Rd(n+1) \leftarrow Rd(n), Rd(n) SNM Rd Rotate Left Through Carry Rd(n) \leftarrow Rd(n+1), Rd(n) ASR Rd Rotate Left Through Carry Rd(n) \leftarrow Rd(n+1), Rd(n) ASR Rd Antimetic Shift Right Rd(n) \leftarrow Rd(n+1), Rd(n) SWAP Rd Swap Nbbbis Rd(n) \leftarrow Rd(n+1), Rd(n) SWAP Rd Swap Nbbbis Rd(n) \leftarrow Rd(n+1), Rd(n) BLD Rd Bd Street Tor Rd(n) \leftarrow Rd(n+1), Rd(n) BLD Rd () Bit Store Torn Register Tor T \leftarrow Rd(n) + T BLD Rd () Bit Store Torn Register Tor T \leftarrow Rd(n) + T CLC Clear Carry C \leftarrow 0 Store Carry C \leftarrow 1 CLD Clear Carry C \leftarrow 0 Store Carry C \leftarrow 1 CLI Clear Carro Flag Z \leftarrow 0 Store Carry C \leftarrow 1	k + 1 None	1/2
SBI P.b Set Bit in UD Register UO(P.b) \leftarrow 0 CBI P.b Clear Bit in UD Register UO(P.b) \leftarrow 0 LSL Rd Logical Shift Left Rd(n-1). Rd(n) LSR Rd Logical Shift Right Rd(n) \leftarrow Rd(n+1). Rd(n) RDR Rd Rotate Right Through Carry Rd(n) \leftarrow Rd(n+1). Rd(n) ASR Rd Anthrote Shift Right Rd(n) \leftarrow Rd(n+1). Rd(n) ASR Rd Anthrote Shift Right Rd(n) \leftarrow Rd(n+1). Rd(n) BSET s Flag Set SREG(a) \leftarrow 1 BCLR s Flag Set SREG(a) \leftarrow 1 BCLR s Flag Set SREG(a) \leftarrow 0 BST Rr, b Bit Isore from Register Rd(b) \leftarrow 1 T-c. Rt(b) BLD Rd, b Bit Isore from Register Rd(b) \leftarrow 1 C+ 0 SEC Set Carry C \leftarrow 1 C-c+ 0 SEN Set N Set Negative Flag N \leftarrow 0 SEI Global Interrupt Enable I \leftarrow 1 CLI Clear Zero Flag <	k + 1 None	1/2
CBI P.b Clear Bit in UD Register $UO(P.b) \leftarrow 0$ LSL Rd Logical Shift Laft Rd(n+1) ← Rd(n), Rd(0) LSR Rd Rotate Left Through Carry Rd(n) ← Rd(n+1), Rd(7) ROL Rd Rotate Left Through Carry Rd(n) ← Rd(n+1), Rd(7) RAR Rd Antonelic Shift Right Rd(n) ← Rd(n+1), Rd(7) ASR Rd Antonelic Shift Right Rd(n) ← Rd(n+1), Rd(7) BST S Flag Set SREG(s) ← 1 BCLR s Flag Clear SREG(s) ← 1 BLD Rd, b Bit Noot from Register to T T - Rr(b) BLD Rd, b Bit Noot from Register to T T - Rr(b) CLC Clear Carry C \leftarrow 0 Set Set Carry C \leftarrow 0 SEI Set Layor Plag N \leftarrow 0 Set Zarr Plag N \leftarrow 0 SEZ Set Zarr Plag X ← 1 Set Zarr Plag X ← 1 CL2 Clear Signed Test Flag X ← 1 Set Signed Test Flag X ← 1 CL3 Clear Tin SREG T ← 1 Set		
LSLRdLogical Shift LeftRd(n+1) - Ed(n), Rd(0)LSRRdLogical Shift RightRd(n+1), Rd(7)ROLRdRotate Loft Through CarryRd(0) - Rd(n+1), Rd(7)RORRdRdRotate Right Through CarryRd(0) - Rd(n+1), Rd(7)ASRRdAntimutel Shift RightRd(n) - Rd(n+1), Rd(7)ASRRdAntimutel Shift RightRd(n) - Rd(n+1), Rd(7)SWAPRdSwap NibblesRd(3, 0)-Rd(7, 4), Rd(7, 4),	None	2
LSRRdLogical Shit RightRd(n) - Rd(n+1), Rd(7)ROLRdRotate Left Trrough CarryRd(pC,Rd(n+1)- Rd(n))ASRRdRotate Left Trrough CarryRd(pC,Rd(n+1)- Rd(n))ASRRdAntimetic Shift RightRd(pC,Rd(n+1)- Rd(n))ASRRdAntimetic Shift RightRd(pC,Rd(n+1)- Rd(n))BSTsFlag ClearRd(R-C,Rd(n-1))BCLRsFlag ClearSREG(s) + 0BSTR, bBit Store from Register to TT + R(t)BLDRd, bBit load from T to RegisterRd(b) - TSECSet CarryC + 1CLCClear Vegative FlagN + 0CLClear Vegative FlagN + 0SEZSet Zero FlagZ + 0SEIGlobal Interrupt EnableI + 0SEIGlobal Interrupt EnableI + 0SESSet Signed Test FlagS + 1CL3Clear Vegative FlagS + 0SESSet Signed Test FlagS + 0SEVSet Half Carry Flag in SREGT + 0SEHSet Half Carry Flag in SREGT + 1CL1Clear Twos Complement OverflowV + 0SETSet Half Carry Flag in SREGH + 0CL2Clear Tin SREGT + 1CL4Clear Tin SREGH + 0SEHSet Half Carry Flag in SREGH + 0LVClear Tin SREGH + 0LVClear Tin SREGH + 0LDRd, XLoad Indirect and Pro-Dec.X + X + 1, Rd + (N)<	None	2
POL Rd Rotate Left Trough Carry Rd(0)—C.Rd(n)—Rd(n)=1)C. ROR Rd Natate Right Trough Carry Rd(0)—C.Rd(n)=Rd(n)=1)C. ASR Rd Anithmetic Shift Right Rd(n)—C.Rd(n)=Rd(n)=1)C. SWAP Rd Swap Nibbles Rd(3, 0)—Rd(7, A)Rd(7, A) BSET s Flag Set SREG(s) + 0 BST Rr, b Bit Store from Register to T T (~ Rd(n)) = C (~ 0) BLD Rd, b Bit Store from Register Rd(b) = T SEC Set Carry C (~ 0) C (~ 0) SEL Set Negative Flag N (~ 1) CLA Clear Xegative Flag N (~ 1) CLA Clear Carry C (~ 0) SEZ Set Lever Flag Z (~ 1) CLZ Clear Xegative Flag Z (~ 1) CLZ Global Interrupt Enable I (~ 1) CL3 Global Interrupt Enable I (~ 1) CL4 Global Interrupt Enable I (~ 1) CL3 Clear Xeg Signed Test Flag S (~ 0) SEV Set	← 0 Z,C,N,V	1
ROR Rd Retails Right Through Carry Rd(7)~C,Rd(r)~Rd(r+1), r=0 ASR Rd Antimetic Shift Right Rd(r)~C,Rd(r-A,Rd(r-1), r=0 SWAP Rd Swap Nibbles Rd(s,0)~Rd(r-A,Rd(r-A),Rd(r-A) BSET s Flag Set SREG(s) + 1 BCLR s Flag Clear SREE(s) + 1 BLD Rd, b Bit loor from Tic Register to T T = - Rr(t) BLD Rd, b Bit load from T to Register Rd(t) + T SEC Set Carry C < 1	← 0 Z,C,N,V	1
ASRRdAnthreets Shift RightRd(a) \leftarrow Rd(n+1), n=0SWAPRdSwap NubblesPd(a), \leftarrow Rd(n+1), n=0BSETsFlag GetSREG(s) \leftarrow 1BCLRsFlag ClearSREG(s) \leftarrow 1BLDRd, bBit Store from Register to TT \leftarrow Rr(b)BLDRd, bBit Store from Register to TT \leftarrow Rr(b)BLDRd, bBit load from T to RegisterRd(b) \leftarrow TSECSet CarryC \leftarrow 0CLCClear CarryC \leftarrow 0SENSet Negative FlagN \leftarrow 0SEZSet Zero FlagZ \leftarrow 1CLZClear Zero FlagZ \leftarrow 1CLIGlobal Interrupt EnableI \leftarrow 1CLIGlobal Interrupt EnableI \leftarrow 1CLIGlobal Interrupt EnableI \leftarrow 1CLSClear Signed Test FlagS \leftarrow 1CLSClear Signed Test FlagS \leftarrow 0SEVSet Twos Complement Overflow.V \leftarrow 0SETSet Twos Complement Overflow.V \leftarrow 0SETSet Tars Reg in SREGT \leftarrow 0CLHClear T in SREGT \leftarrow 0SEHSet Hall Carry Flag in SREGH \leftarrow 0CLHClear All Carry Flag in SREGH \leftarrow 0CLHClear Hall Carry Flag in SREGH \leftarrow 0LDRd, KLoad Indirect and Pro-Dro.X \leftarrow X + 1, Rd \leftarrow X)LDRd, KLoad Indirect and Post-Inc.Rd \leftarrow (Y) (\vee Y + 1LDRd, YLoad Indirect and Post-Inc.Rd \leftarrow (Y) + (Y + 1, IL) <td>⊷Rd(7) Z,C,N,V</td> <td>1</td>	⊷Rd(7) Z,C,N,V	1
SWAPRdSwap NibblesRd(3.0)-Rd(7.4),	C←Rd(0) Z,C,N,V	1
BSETsFlag SatSREG(s) \leftarrow 1BCLRsRag OlearSREG(s) \leftarrow 0BLDRr, bBit Stor from Register to TT \leftarrow Rr(tb)BLDRd, bBit load from T to RegisterRd(b) \leftarrow TSECSet CarryC \leftarrow 1CLCClear CarryC \leftarrow 0SENSet Negative FlagN \leftarrow 0SEZSet Vegative FlagN \leftarrow 0CLZClear Negative FlagZ \leftarrow 1CLZClear Stative FlagZ \leftarrow 1CLIGlobal Interrupt EnableI \leftarrow 1CLIGlobal Interrupt EnableI \leftarrow 1CLIGlobal Interrupt EnableI \leftarrow 1CLIGlobal Interrupt EnableI \leftarrow 0SESSet Signed Test FlagS \leftarrow 0SEVSet Twos Complement OverflowV \leftarrow 0SETSet Twos Complement OverflowV \leftarrow 0SETSet T in SREGT \leftarrow 1CLVClear Wes Complement OverflowV \leftarrow 0SETSet T in SREGH \leftarrow 1CLTClear Tim SREGH \leftarrow 1CLTClear Tim SREGH \leftarrow 1DATA TRANSFER INSTRUCTIONSNOVRd \leftarrow RrMOVRd, RrMove Between RegistersRd \leftarrow Rf \leftarrow (X), X \leftarrow X + 1LDRd, YLoad Indirect and Pre-Dec.X \leftarrow X + 1, Rd \leftarrow (X), X \leftarrow X + 1LDRd, YLoad Indirect and Pre-Dec.X \leftarrow X + 1, Rd \leftarrow (X), X \leftarrow X + 1LDRd, YLoad Indirect and Pre-Dec.X \leftarrow X + 1, Rd \leftarrow (Y)LDRd, YLoa	6 Z,C,N,V	1
BCLRsFlag ClearSREG(s) $\leftarrow 0$ BSTRr, bBit Store from Register to TT \leftarrow Ar(b)BLDRd, bBit Store from Register to TT \leftarrow Ar(b)SECSet CarryC $\leftarrow 0$ CLCClear CarryC $\leftarrow 0$ SENSet Regative FlagN $\leftarrow 1$ CLZClear Segative FlagN $\leftarrow 0$ SEZSet Zero FlagZ $\leftarrow 1$ CLZClear Segative FlagN $\leftarrow 1$ CLIGlobal Interrupt DisableI $\leftarrow 1$ CLIGlobal Interrupt DisableI $\leftarrow 0$ SESSet Signed Test FlagS $\leftarrow 0$ SEVSet Signed Test FlagS $\leftarrow 0$ SEVSet Ton Scomplement OverflowV $\leftarrow 1$ CLVClear Tin SREGT $\leftarrow 1$ CLVClear Tin SREGH $\leftarrow 1$ CLTClear Tin SREGH $\leftarrow 1$ CLTClear Tin SREGH $\leftarrow 1$ CLTClear Tin SREGH $\leftarrow 0$ SEHSet Hall Carry Flag in SREGH $\leftarrow 1$ DATA TRANSFER INSTRUCTIONSMOVRd, RrMOVRd, RrCopy Register WordRd $\leftarrow N$; N, X $\leftarrow X + 1$ LDRd, X +Load Indirect and Pro-Dec.X $\leftarrow X + X + 1$, Rd $\leftarrow (X)$ LDRd, Y +Load Indirect and Pro-Dec.X $\leftarrow X + 1$, Rd $\leftarrow (Y)$ LDRd, Y +Load Indirect and Pro-Dec.X $\leftarrow X + X + 1$, Rd $\leftarrow (Y)$ LDRd, Y +Load Indirect and Pro-Dec.X $\leftarrow X + X + 1$, Rd $\leftarrow (Y)$ LDRd, Y +Load Indirect and Pro-Dec.X $\leftarrow X + X + 1$, Rd $\leftarrow (Y)$ <td></td> <td>1</td>		1
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	SREG(s)	1
BLDRd, bBit load from T to RegisterRd(b) \leftarrow TSECSet CarryC \leftarrow 1CLCClear CarryC \leftarrow 0SENSet Negative FlagN \leftarrow 1CLNClear Negative FlagN \leftarrow 0SEZSet Zero FlagZ \leftarrow 1CLZClear Zero FlagZ \leftarrow 1CLIGlobal Interrupt Enable1 \leftarrow 1CLIGlobal Interrupt Enable1 \leftarrow 0SESSet Signed Test FlagS \leftarrow 0SESSet Signed Test FlagS \leftarrow 0SETSet Signed Test FlagS \leftarrow 0SETSet Signed Test FlagS \leftarrow 0SETSet Somplement OverflowV \leftarrow 1CLVClear Twos Complement OverflowV \leftarrow 0SETSet Half Carry Flag in SREGH \leftarrow 0SEHSet Half Carry Flag in SREGH \leftarrow 0DATA TRANSFER INSTRUCTIONSRd \leftarrow RfMOVWRd, RrMove Between RegistersRd \leftarrow KMOVRd, XLoad Indirect and Post-Inc.Rd \leftarrow K) $\times 1 + 1$ LDRd, XLoad Indirect and Post-Inc.Rd \leftarrow (X) $\times - X + 1$ LDRd, YLoad Indirect and Post-Inc.Rd \leftarrow (X) $\times - X + 1$ LDRd, YLoad Indirect and Post-Inc.Rd \leftarrow (X) $\times + 1 + 1$ LDRd, ZLoad Indirect and Post-Inc.Rd \leftarrow (X) $\leftarrow Y + 1$ LDRd, YLoad Indirect and Post-Inc.Rd \leftarrow (X) $\times - X + 1$ LDRd, ZLoad Indirect and Post-Inc.Rd \leftarrow (X) $\leftarrow Y + 1$ LDRd, ZLoad Indirect	SREG(s)	1
SECSet CarryC1CLCClear CarryC1CLCClear CarryC1CLNSet Negative FlagN1CLNClear Negative FlagN1CLZClear Zero FlagZ1CLZClear Zero FlagZ0SEIGlobal Interrupt Enable11CLIClosal Interrupt Disable11CLIClosal Interrupt Disable10SESSet Signed Test FlagS1CLSClear Twos Complement OverflowV0SETSet T in SREGT1CLVClear Tin SREGT1CLTClear Tin SREGH1CLHClear Half Carry Flag in SREGH1LDRd, KrCopy Register WordRd+1:Rd - Rr+1:RtMOWRd, RrCopy Register WordRd+1:Rd - Rr+1:RtLDRd, XLoad Indirect and Pre-Dec.X - X + X + 1.LDRd, YLoad Indirect and Pre-Dec.X - X + X + 1.LDRd, YLoad Indirect and Pre-Dec.X - X + 1.LDRd, YLoad Indirect and Pre-Dec.Z + 2LDRd, YLoad Indirect and Pre-Dec.Z + 2LDRd, YLoad Indirect and Pre-Dec.Z + 2LDRd, YLoad Indirect and Pre-Dec.Z + 2LDRd	Т	1
CLCClear CarryC + 0SENSet Negative FlagN + - 1CLNClear Negative FlagN + - 0SEZSet Zero FlagZ + -1CLZClear Zero FlagZ + -0SEIGlobal Interrupt Enable1 + -1CLIGlobal Interrupt Enable1 + -0SESSet Signed Test FlagS + -1CLSClear Signed Test FlagS + -0SEFSet Twos Complement Overflow.V + -1CLVClear Twos Complement Overflow.V + -1CLVClear Twos Complement Overflow.V + -1CLTClear Tins REGT + -0SETSet Tin SREGT + -0SEHSet Hall Carry Flag in SREGH + -1CLTClear Half Carry Flag in SREGH + -1CLHClear Half Carry Flag in SREGH + -1LDRd, RrCoopy Register WordRd+tRd + Err+1RrLDRd, KLoad Indirect and Pros-Inc.Rd - KLDRd, XLoad Indirect and Pros-Inc.Rd - (X), X < X + 1	None	1
$\begin{array}{ c c c c c c c c c c c c c c c c c c c$	С	1
$\begin{tabular}{ c c c c c c c c c c c c c c c c c c c$	С	1
SEZSet Zero Flag $Z \leftarrow 1$ CLZClear Zero Flag $Z \leftarrow 0$ SEIGlobal Interrupt Enable $1 \leftarrow 1$ CLIGlobal Interrupt Disable $1 \leftarrow 0$ SESSet Signed Test Flag $S \leftarrow 1$ CLSClear Signed Test Flag $S \leftarrow 0$ SEVSet Twos Complement Overflow. $V \leftarrow 1$ CLVClear Wos Complement Overflow $V \leftarrow 1$ CLVClear Twos Complement Overflow $V \leftarrow 0$ SETSet T in SREG $T \leftarrow 1$ CLTClear Twos Complement Overflow $V \leftarrow 0$ SEHSet Half Carry Flag in SREG $H \leftarrow 1$ CLHClear Thin SREG $H \leftarrow 0$ DATA TRANSFER INSTRUCTIONSMOVRd, RrMOVRd, RrMove Between RegistersRd $\leftarrow Rr$ MOWRd, KLoad IndirectRd $\leftarrow (X)$ LDRd, XLoad Indirect and Post-Inc.Rd $\leftarrow (X), X \leftarrow X + 1$ LDRd, XLoad Indirect and Pro-Dec. $X \leftarrow X + X + X + 1$. Ad IndirectLDRd, YLoad Indirect and Pro-Dec. $X \leftarrow X + 1$. Rd $\leftarrow (Y)$ LDRd, YLoad Indirect and Pro-Dec. $Y \leftarrow Y + 1$. I, Rd $\leftarrow (Y)$ LDRd, ZLoad Indirect and Pro-Dec. $Y \leftarrow Y + 1$. I, Rd $\leftarrow (Y)$ LDRd, ZLoad Indirect and Pro-Dec. $Y \leftarrow Y + 1$. I, Rd $\leftarrow (Y)$ LDRd, ZLoad Indirect and Pro-Dec. $Z \leftarrow Z - 1$. Rd $\leftarrow (Z)$ LDRd, ZLoad Indirect and Pro-Dec. $Z \leftarrow Z - 1$. Rd $\leftarrow (Z)$ LDRd, ZLoad Indirect and Pro-Dec. $Z \leftarrow Z - 1$. Rd $\leftarrow (X)$ <	N	1
CLZClear Zero Flag $Z \leftarrow 0$ SEIGlobal Interrupt Enable $I \leftarrow 1$ CLIGlobal Interrupt Disable $I \leftarrow 0$ SESSet Signed Test Flag $S \leftarrow 1$ CLSClear Signed Test Flag $S \leftarrow 0$ SEVSet Twos Complement Overflow. $V \leftarrow 1$ CLVClear Twos Complement Overflow $V \leftarrow 0$ SETSet T in SREG $T \leftarrow 1$ CLTClear Twos Complement Overflow $V \leftarrow 0$ SETSet T in SREG $T \leftarrow 1$ CLTClear Half Carry Flag in SREG $H \leftarrow 1$ CLHClear Half Carry Flag in SREG $H \leftarrow 0$ DATA TRANSFER INSTRUCTIONSDATA TRANSFER INSTRUCTIONSMOVRd, RrCopy Register WordRd+1Rd ← Rr+1RtLDRd, XLoad IndirectRd ← KLDRd, XLoad Indirect and Problec. $X \leftarrow X + 1, Rd \leftarrow (X), X \leftarrow X + 1$ LDRd, VLoad Indirect and Problec. $X \leftarrow X + 1, Rd \leftarrow (X)$ LDRd, YLoad Indirect and Problec. $X \leftarrow X + 1, Rd \leftarrow (Y)$ LDRd, VLoad Indirect and Problec. $Y \leftarrow Y + 1, Rd \leftarrow (Y)$ LDRd, VLoad Indirect and Pro-Dec. $X \leftarrow X + 1, Rd \leftarrow (Y)$ LDRd, QLoad Indirect and Problec. $Y \leftarrow Y + 1, Rd \leftarrow (Y)$ LDRd, ZLoad Indirect and Pro-Dec. $Y \leftarrow Y + 1, Rd \leftarrow (Y)$ LDRd, ZLoad Indirect and Pro-Dec. $Z \leftarrow Z + 1, Rd \leftarrow (Y)$ LDRd, ZLoad Indirect and Pro-Dec. $Z \leftarrow Z + 1, Rd \leftarrow (Y)$ LDRd, ZLoad Indirect and Pro-Dec.	N	1
SEIGiobal Interrupt Enable $I \leftarrow 1$ CLIGlobal Interrupt Disable $I \leftarrow 0$ SESSet Signed Test Flag $S \leftarrow 1$ CLSClear Signed Test Flag $S \leftarrow 0$ SEVSet Twos Complement Overflow $V \leftarrow 1$ CLVClear Tin SREG $T \leftarrow 1$ CLTSet Twos Complement Overflow $V \leftarrow 0$ SETSet Tin SREG $T \leftarrow 1$ CLTClear Tin SREG $T \leftarrow 0$ SEHSet Half Carry Flag in SREG $H \leftarrow 1$ CLHClear Half Carry Flag in SREG $H \leftarrow 0$ DATA TRANSFER INSTRUCTIONSMOVRd, RrCopy Register WordRd+1:Rd \leftarrow Rr+1:RrLDRd, XLoad IndirectRd $\leftarrow K$ LDRd, XLoad Indirect and Post-Inc.Rd $\leftarrow (X), X \leftarrow X + 1$ LDRd, YLoad Indirect and Pre-Dec. $X \leftarrow X + 1, Rd \leftarrow (X)$ LDRd, YLoad Indirect and Pre-Dec. $Y \leftarrow Y - 1, Rd \leftarrow (Y)$ LDRd, ZLoad Indirect and Pre-Dec. $Y \leftarrow Y - 1, Rd \leftarrow (Y)$ LDRd, ZLoad Indirect and Pre-Dec. $Y \leftarrow Y - 1, Rd \leftarrow (Y)$ LDRd, ZLoad Indirect and Pre-Dec. $Z \leftarrow Z + 1$ LDRd, ZLoad Indirect and Pre-Dec. $Z \leftarrow Z + 1$ LDRd, ZLoad Indirect and Post-Inc.Rd $\leftarrow (Y), Y \leftarrow Y + 1$ LDRd, ZLoad Indirect and Post-Inc.Rd $\leftarrow (Z), Z \leftarrow Z + 1$ LDRd, ZLoad Indirect and Post-Inc.Rd $\leftarrow (Z), Z \leftarrow Z + 1$ LDRd, ZLoad Indirect and Post-Inc.Rd $\leftarrow (Z), Q \leftarrow Z + 2$ <td>Z</td> <td>1</td>	Z	1
CLIGlobal Interrupt Disable $I \leftarrow 0$ SESSet Signed Test Flag $S \leftarrow -1$ CLSClear Signed Test Flag $S \leftarrow 0$ SEVSet Twos Complement Overflow. $V \leftarrow 1$ CLVClear Twos Complement Overflow. $V \leftarrow 1$ CLVClear Tin SREG $T \leftarrow 1$ CLTSet T in SREG $T \leftarrow 0$ SEHSet Hall Carry Flag in SREG $H \leftarrow 1$ CLHClear Half Carry Flag in SREG $H \leftarrow 0$ DATA TRANSFER INSTRUCTIONS $H \leftarrow 0$ DUVRd, RrMove Between Registers $Rd \leftarrow Rr$ MOVRd, RrCopy Register Word $Rd + 1:Rd \leftarrow Rr + 1:Rr$ LDRd, XLoad Indirect and Post-Inc. $Rd \leftarrow (X)$ LDRd, XLoad Indirect and Post-Inc. $Rd \leftarrow (X)$ LDRd, Y+Load Indirect and Post-Inc. $Rd \leftarrow (Y), Y \leftarrow Y + 1$ LDRd, Y+Load Indirect and Post-Inc. $Rd \leftarrow (Y), Y \leftarrow Y + 1$ LDRd, Y+Load Indirect and Post-Inc. $Rd \leftarrow (Y), Y \leftarrow Y + 1$ LDRd, Y+Load Indirect and Post-Inc. $Rd \leftarrow (Y), Y \leftarrow Y + 1$ LDRd, Y+Load Indirect and Post-Inc. $Rd \leftarrow (Z), Z \leftarrow Z + 1$ LDRd, Z+Load Indirect and Post-Inc. $Rd \leftarrow (Z)$ LDRd, Z+Load Indirect and Post-Inc. $Rd \leftarrow (Z), Z \leftarrow Z + 1$ LDRd, Z+Load Indirect and Post-Inc. $Rd \leftarrow (Z), Z \leftarrow Z + 1$ LDRd, Z+Load Indirect and Post-Inc. $Rd \leftarrow (Z), Z \leftarrow Z + 1$ LDRd, Z+Load Indirect and Post-Inc. $Rd \leftarrow (Z), C \leftarrow Z + 1$ <	Z	1
SESSet Signed Test Flag $S \leftarrow 1$ CLSClear Signed Test Flag $S \leftarrow 0$ SEVSet Twos Complement Overflow. $V \leftarrow 1$ CLVClear Twos Complement Overflow $V \leftarrow 0$ SETSet T in SREG $T \leftarrow 1$ CLTClear T in SREG $T \leftarrow 0$ SEHSet Hall Carry Flag in SREG $H \leftarrow 1$ CLHClear T in SRE $H \leftarrow 0$ DATA TRANSFER INSTRUCTIONS $H \leftarrow 0$ MOVRd, RrMove Between RegistersRd $\leftarrow Rr + 1$:RrMOWRd, RrCopy Register WordRd +1:Rd $\leftarrow Rr + 1$:RrLDRd, XLoad IndirectRd $\leftarrow (X)$ LDRd, XLoad IndirectRd $\leftarrow (X)$ LDRd, X+Load IndirectRd $\leftarrow (X), X \leftarrow X + 1$ LDRd, YLoad Indirect and Pro-Dec. $X \leftarrow X - 1, Rd \leftarrow (X)$ LDRd, YLoad Indirect and Pro-Dec. $Y \leftarrow Y - 1, Rd \leftarrow (Y)$ LDRd, YLoad Indirect and Pro-Dec. $Y \leftarrow Y - 1, Rd \leftarrow (Y)$ LDRd, YLoad Indirect and Pro-Dec. $Y \leftarrow Y - 1, Rd \leftarrow (Y)$ LDRd, ZLoad Indirect and Pro-Dec. $Y \leftarrow Y - 1, Rd \leftarrow (Y)$ LDRd, ZLoad Indirect and Pro-Dec. $Y \leftarrow Y - 1, Rd \leftarrow (Y), Y \leftarrow Y + 1$ LDRd, ZLoad Indirect and Pro-Dec. $Y \leftarrow Y - 1, Rd \leftarrow (Y)$ LDRd, ZLoad Indirect and Pro-Dec. $Y \leftarrow Y - 1, Rd \leftarrow (Y)$ LDRd, ZLoad Indirect and Pro-Dec. $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ LDRd, ZLoad Indirect and Pro-Dec. $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ LD	1	1
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	1	1
SEVSet Twos Complement Overflow. $V \leftarrow 1$ CLVClear Twos Complement Overflow $V \leftarrow 0$ SETSet T in SREG $T \leftarrow 1$ CLTClear Tin SREG $T \leftarrow 0$ SEHSet Half Carry Flag in SREG $H \leftarrow 1$ CLHClear Half Carry Flag in SREG $H \leftarrow 1$ CLHClear Half Carry Flag in SREG $H \leftarrow 0$ DATA TRANSFER INSTRUCTIONS $M \lor 0$ $Rd + 1:Rd \leftarrow Rr$ MOVRd, RrMove Between Registers $Rd \leftarrow Rr$ MOVWRd, RrCopy Register Word $Rd + 1:Rd \leftarrow Rr + 1:Rr$ LDRd, XLoad Indirect $Rd \leftarrow (X)$ LDRd, XLoad Indirect and Post-Inc. $Rd \leftarrow (X), X \leftarrow X + 1$ LDRd, Y+Load Indirect and Pre-Dec. $X \leftarrow X - 1, Rd \leftarrow (X)$ LDRd, Y+Load Indirect and Pre-Dec. $Y \leftarrow Y + 1, Rd \leftarrow (Y)$ LDRd, Y+Load Indirect and Pre-Dec. $Y \leftarrow Y + 1, Rd \leftarrow (Y)$ LDRd, Y+Load Indirect and Pre-Dec. $Y \leftarrow Y + 1, Rd \leftarrow (Y)$ LDRd, Z+Load Indirect and Pre-Dec. $Y \leftarrow Y + 1, Rd \leftarrow (Z)$ LDRd, Z+Load Indirect and Pre-Dec. $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ LDRd, Z+Load Indirect and Pre-Dec. $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ LDRd, Z+Load Indirect and Pre-Dec. $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ LDRd, Z+Load Indirect and Pre-Dec. $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ LDRd, Z+Load Indirect and Pre-Dec. $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ LDRd, Z+Load Indirect and Pre-Dec. $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ <tr< td=""><td>S</td><td>1</td></tr<>	S	1
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	S	1
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	V	1
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	V	1
SEHSet Half Carry Flag in SREG $H \leftarrow 1$ CLHClear Half Carry Flag in SREG $H \leftarrow 0$ DATA TRANSFER INSTRUCTIONSMOVRd, RrMove Between RegistersRd \leftarrow RrMOVWRd, RrCopy Register WordRd+1:Rd \leftarrow Rr+1:RrLDIRd, KLoad ImmediateRd \leftarrow (X)LDRd, XLoad Indirect and Post-Inc.Rd \leftarrow (X)LDRd, X+Load Indirect and Pro-Dec. $X \leftarrow X + 1$, Rd \leftarrow (X)LDRd, Y+Load Indirect and Post-Inc.Rd \leftarrow (Y), Y \leftarrow Y + 1LDRd, Y+Load Indirect and Post-Inc.Rd \leftarrow (Y), Y \leftarrow Y + 1LDRd, Y+Load Indirect and Post-Inc.Rd \leftarrow (Y), Y \leftarrow Y + 1LDRd, Y+Load Indirect and Post-Inc.Rd \leftarrow (Y), Y \leftarrow Y + 1LDRd, Y+Load Indirect and Post-Inc.Rd \leftarrow (Y), Y \leftarrow Y + 1LDRd, Y+Load Indirect and Post-Inc.Rd \leftarrow (Y), Y \leftarrow Y + 1LDRd, Z-Load Indirect and Post-Inc.Rd \leftarrow (Y), QLDRd, Z-Load Indirect and Post-Inc.Rd \leftarrow (Z), Z \leftarrow Z + 1, Rd \leftarrow (Z)LDRd, Z-Load Indirect and Post-Inc.Rd \leftarrow (Z), Z \leftarrow Z + 1, Rd \leftarrow (Z)LDRd, Z-Load Indirect and Post-Inc.Rd \leftarrow (Z), Z \leftarrow Z + 1, Rd \leftarrow (Z)LDRd, Z-Load Indirect tith DisplacementRd \leftarrow (X)LDRd, Z-Load Indirect tith DisplacementRd \leftarrow (X)LDRd, Z-Load Indirect and Post-Inc.(X) \leftarrow Rr, X + X + 1LDRd, Z-Load Indi	Т	1
CLHClear Haf Cary Flag in SREG $H \leftarrow 0$ DATA TRANSFER INSTRUCTIONSMOVRd, RrMove Between RegistersRd \leftarrow RrMOVWRd, RrCopy Register WordRd+1:Rd \leftarrow Rr+1:RrLDIRd, KLoad IndirectRd $\leftarrow (X)$ LDRd, XLoad IndirectRd $\leftarrow (X)$ LDRd, X+Load Indirect and Post-Inc.Rd $\leftarrow (X), X \leftarrow X + 1$ LDRd, Y+Load Indirect and Pre-Dec. $X \leftarrow X + 1, Rd \leftarrow (X)$ LDRd, Y+Load Indirect and Pre-Dec. $X \leftarrow X + 1, Rd \leftarrow (X)$ LDRd, Y+Load Indirect and Pre-Dec. $Y \leftarrow Y + 1, Rd \leftarrow (Y)$ LDRd, Y+Load Indirect and Pre-Dec. $Y \leftarrow Y + 1, Rd \leftarrow (Y)$ LDRd, Z+Load Indirect and Pre-Dec. $Y \leftarrow Y - 1, Rd \leftarrow (Y)$ LDRd, Z+Load Indirect and Pre-Dec. $Rd \leftarrow (Y + q)$ LDRd, ZLoad Indirect and Pre-Dec. $Rd \leftarrow (Z), Z \leftarrow Z+1$ LDRd, Z+Load Indirect and Pre-Dec. $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ LDRd, Z+Load Indirect and Pre-Dec. $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ LDRd, Z+Load Indirect and Pre-Dec. $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ LDRd, Z+Load Indirect and Pre-Dec. $Z \leftarrow Z - 1, Rd \leftarrow (X)$ LDRd, Z+Load Indirect and Pre-Dec. $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ LDRd, Z+Load Indirect and Pre-Dec. $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ LDRd, Z+Load Indirect and Pre-Dec. $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ LDRd, KLoad Direct from SRAMRd $\leftarrow (Z + q)$ <t< td=""><td>Т</td><td>1</td></t<>	Т	1
$\begin{array}{ c c c c c c c c c c c c c c c c c c c$	Н	1
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	Н	1
MOVWRd, RrCopy Register WordRd+1:Rd \leftarrow Rr+1:RrLDIRd, KLoad ImmediateRd \leftarrow KLDRd, XLoad IndirectRd \leftarrow (X)LDRd, X+Load Indirect and Post-Inc.Rd \leftarrow (X), X \leftarrow X + 1LDRd, -XLoad Indirect and Pre-Dec.X \leftarrow X - 1, Rd \leftarrow (X)LDRd, YLoad Indirect and Pre-Dec.X \leftarrow X - 1, Rd \leftarrow (Y)LDRd, YLoad Indirect and Post-Inc.Rd \leftarrow (Y), Y \leftarrow Y + 1LDRd, Y+Load Indirect and Post-Inc.Rd \leftarrow (Y), Y \leftarrow Y + 1LDRd, Y+Load Indirect and Pre-Dec.Y \leftarrow Y - 1, Rd \leftarrow (Y)LDRd, Y-qLoad Indirect with DisplacementRd \leftarrow (Y + q)LDRd, Z+Load Indirect and Post-Inc.Rd \leftarrow (Z), Z \leftarrow Z+1LDRd, Z+Load Indirect and Post-Inc.Rd \leftarrow (Z), Z \leftarrow Z+1LDRd, Z+Load Indirect and Post-Inc.Rd \leftarrow (Z), Z \leftarrow Z+1, Rd \leftarrow (Z)LDRd, Z+Load Indirect with DisplacementRd \leftarrow (Z), Z \leftarrow Z+1, Rd \leftarrow (Z)LDRd, Z+Load Indirect with DisplacementRd \leftarrow (K)STX, RrStore Indirect on Pre-Dec.Z \leftarrow Z - 1, Rd \leftarrow (K)STX, RrStore Indirect and Pre-Dec.X \leftarrow X - 1, (X) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.X \leftarrow X - 1, (X) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.X \leftarrow X - 1, (X) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.Y \leftarrow Y - 1, (Y) \leftarrow RrSTY, RrStore Indirect		
LDIRd, KLoad ImmediateRd \leftarrow KLDRd, XLoad IndirectRd \leftarrow (X)LDRd, XLoad Indirect and Post-Inc.Rd \leftarrow (X), X \leftarrow X + 1LDRd, -XLoad Indirect and Pre-Dec.X \leftarrow X - 1, Rd \leftarrow (X)LDRd, YLoad Indirect and Pre-Dec.X \leftarrow X - 1, Rd \leftarrow (Y)LDRd, YLoad Indirect and Post-Inc.Rd \leftarrow (Y), Y \leftarrow Y + 1LDRd, Y+Load Indirect and Pre-Dec.Y \leftarrow Y - 1, Rd \leftarrow (Y)LDRd, Y+Load Indirect with DisplacementRd \leftarrow (Y + q)LDDRd, Y-qLoad Indirect with DisplacementRd \leftarrow (Z), Z \leftarrow Z+1LDRd, ZLoad Indirect and Post-Inc.Rd \leftarrow (Z), Z \leftarrow Z+1LDRd, ZLoad Indirect and Post-Inc.Rd \leftarrow (Z), Z \leftarrow Z+1LDRd, Z-Load Indirect and Post-Inc.Rd \leftarrow (Z), Z \leftarrow Z+1LDRd, Z-Load Indirect with DisplacementRd \leftarrow (Z), Z \leftarrow Z+1, Rd \leftarrow (Z)LDDRd, Z-Load Indirect with DisplacementRd \leftarrow (X), C \leftarrow RrLDDRd, KLoad Direct from SRAMRd \leftarrow (k)STX, RrStore Indirect(X) \leftarrow RrSTX, RrStore Indirect and Pre-Dec.X \leftarrow X - 1, (X) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.Y \leftarrow Y - 1, (Y) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.Y \leftarrow Y - 1, (Y) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.Y \leftarrow Y - 1, (Y) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.Y \leftarrow Y - 1,	None	1
LDRd, XLoad IndirectRd \leftarrow (X)LDRd, X+Load Indirect and Post-Inc.Rd \leftarrow (X), X \leftarrow X + 1LDRd, -XLoad Indirect and Pre-Dec.X \leftarrow X - 1, Rd \leftarrow (X)LDRd, YLoad Indirect and Pre-Dec.X \leftarrow X - 1, Rd \leftarrow (Y)LDRd, Y+Load Indirect and Post-Inc.Rd \leftarrow (Y), Y \leftarrow Y + 1LDRd, -YLoad Indirect and Pre-Dec.Y \leftarrow Y - 1, Rd \leftarrow (Y)LDRd, -YLoad Indirect with DisplacementRd \leftarrow (Y + q)LDRd, ZLoad Indirect with OsplacementRd \leftarrow (Z), Z \leftarrow Z+1LDRd, ZLoad Indirect and Post-Inc.Rd \leftarrow (Z), Z \leftarrow Z+1LDRd, ZLoad Indirect with DisplacementRd \leftarrow (Z), Z \leftarrow Z+1LDRd, Z-Load Indirect with DisplacementRd \leftarrow (Z), Z \leftarrow Z+1LDRd, Z-Load Indirect and Pre-Dec.Z \leftarrow Z - 1, Rd \leftarrow (Z)LDDRd, Z-Load Indirect with DisplacementRd \leftarrow (Z), Q \leftarrow R-LDDRd, X-Load Indirect from SRAMRd \leftarrow (k)STX, RrStore Indirect and Post-Inc.(X) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.X \leftarrow X - 1, (X) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.Y \leftarrow X - 1, (X) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.Y \leftarrow Y - 1, (Y) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.Y \leftarrow Y - 1, (Y) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.Y \leftarrow Y - 1, (Y) \leftarrow RrSTY, RrStore Indirect		1
LDRd, X+Load Indirect and Post-Inc.Rd \leftarrow (X), X \leftarrow X + 1LDRd, -XLoad Indirect and Pre-Dec.X \leftarrow X - 1, Rd \leftarrow (X)LDRd, YLoad Indirect and Pre-Dec.X \leftarrow X - 1, Rd \leftarrow (Y)LDRd, Y+Load Indirect and Post-Inc.Rd \leftarrow (Y), Y \leftarrow Y + 1LDRd, -YLoad Indirect with DisplacementRd \leftarrow (Y + q)LDRd, ZLoad Indirect with DisplacementRd \leftarrow (Z), Z \leftarrow Z + 1LDRd, ZLoad Indirect and Post-Inc.Rd \leftarrow (Z), Z \leftarrow Z + 1LDRd, ZLoad Indirect and Post-Inc.Rd \leftarrow (Z), Z \leftarrow Z + 1LDRd, ZLoad Indirect with DisplacementRd \leftarrow (Z), Z \leftarrow Z + 1, Rd \leftarrow (Z)LDRd, ZLoad Indirect with DisplacementRd \leftarrow (Z), R \leftarrow Z + 1, Rd \leftarrow (Z)LDRd, ZLoad Indirect and Post-Inc.Rd \leftarrow (X), X \leftarrow X + 1LDRd, Z + qLoad Indirect with DisplacementRd \leftarrow (Z)LDDRd, Z + qLoad Indirect from SRAMRd \leftarrow (k)STX, RrStore Indirect and Post-Inc.(X) \leftarrow RrSTX, RrStore Indirect and Post-Inc.(X) \leftarrow Rr, X \leftarrow X + 1, (X) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.X \leftarrow X - 1, (X) \leftarrow RrSTY, RrStore Indirect and Post-Inc.(Y) \leftarrow Rr, Y \leftarrow Y + 1STY, RrStore Indirect and Post-Inc.(Y) \leftarrow RrSTY, RrStore Indirect and Post-Inc.(Y) \leftarrow Rr, Y \leftarrow Y + 1, (Y) \leftarrow RrSTY, RrStore Indirect and Post-Inc.(Y) \leftarrow Rr	None	1
LDRd, -XLoad Indirect and Pre-Dec. $X \leftarrow X - 1$, $Rd \leftarrow (X)$ LDRd, YLoad Indirect $Rd \leftarrow (Y)$ LDRd, Y+Load Indirect and Post-Inc. $Rd \leftarrow (Y), Y \leftarrow Y + 1$ LDRd, -YLoad Indirect and Pre-Dec. $Y \leftarrow Y - 1$, $Rd \leftarrow (Y)$ LDDRd, Y+qLoad Indirect with Displacement $Rd \leftarrow (Y+q)$ LDDRd, ZLoad Indirect and Post-Inc. $Rd \leftarrow (Z), Z \leftarrow Z+1$ LDRd, ZLoad Indirect and Post-Inc. $Rd \leftarrow (Z), Z \leftarrow Z+1$ LDRd, ZLoad Indirect and Post-Inc. $Rd \leftarrow (Z), Z \leftarrow Z+1$ LDRd, -ZLoad Indirect with Displacement $Rd \leftarrow (Z), Z \leftarrow Z+1$ LDRd, ZLoad Indirect with Displacement $Rd \leftarrow (Z), Z \leftarrow Z+1$ LDRd, ZLoad Indirect mad Post-Inc. $Z \leftarrow Z - 1, Rd \leftarrow (Z)$ LDDRd, Z+qLoad Indirect from SRAM $Rd \leftarrow (R)$ STX, RrStore Indirect $(X) \leftarrow Rr$ STX+, RrStore Indirect and Post-Inc. $(X) \leftarrow Rr, X \leftarrow X+1$ ST-X, RrStore Indirect and Pre-Dec. $X \leftarrow X - 1, (X) \leftarrow Rr$ STY, RrStore Indirect and Pre-Dec. $X \leftarrow X - 1, (X) \leftarrow Rr$ STY, RrStore Indirect and Pre-Dec. $Y \leftarrow Y - 1, (Y) \leftarrow Rr$ STY, RrStore Indirect and Post-Inc. $(Y) \leftarrow Rr, Y \leftarrow Y + 1$ ST-Y, RrStore Indirect and Pre-Dec. $Y \leftarrow Y - 1, (Y) \leftarrow Rr$	None	2
LDRd, YLoad IndirectRd \leftarrow (Y)LDRd, Y+Load Indirect and Post-Inc.Rd \leftarrow (Y), Y \leftarrow Y + 1LDRd, -YLoad Indirect and Pre-Dec.Y \leftarrow Y - 1, Rd \leftarrow (Y)LDDRd, Y+qLoad Indirect with DisplacementRd \leftarrow (Y + q)LDRd, ZLoad Indirect and Post-Inc.Rd \leftarrow (Z), Z \leftarrow Z+1LDRd, Z+Load Indirect and Post-Inc.Rd \leftarrow (Z), Z \leftarrow Z+1LDRd, -ZLoad Indirect with DisplacementRd \leftarrow (Z), Z \leftarrow Z+1, Rd \leftarrow (Z)LDRd, -ZLoad Indirect with DisplacementRd \leftarrow (Z), C \leftarrow Z + 1, Rd \leftarrow (Z)LDRd, Z+qLoad Indirect with DisplacementRd \leftarrow (Z + q)LDDRd, Z+qLoad Indirect mode Pre-Dec.Z \leftarrow Z - 1, Rd \leftarrow (Z)LDSRd, kLoad Direct from SRAMRd \leftarrow (X) \leftarrow RrSTX, RrStore Indirect and Post-Inc.(X) \leftarrow RrSTX+, RrStore Indirect and Pre-Dec.X \leftarrow X - 1, (X) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.X \leftarrow X - 1, (X) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.Y \leftarrow Y - 1, (Y) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.Y \leftarrow Y - 1, (Y) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.Y \leftarrow Y - 1, (Y) \leftarrow RrSTY, RrStore Indirect and Pre-Dec.Y \leftarrow Y - 1, (Y) \leftarrow Rr		2
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $		2
LDRd, - YLoad Indirect and Pre-Dec. $Y \leftarrow Y - 1$, Rd $\leftarrow (Y)$ LDDRd, Y+qLoad Indirect with DisplacementRd $\leftarrow (Y + q)$ LDRd, ZLoad Indirect and Post-Inc.Rd $\leftarrow (Z)$, $Z \leftarrow Z + 1$ LDRd, Z+Load Indirect and Pre-Dec. $Z \leftarrow Z - 1$, Rd $\leftarrow (Z)$ LDRd, -ZLoad Indirect with DisplacementRd $\leftarrow (Z)$, $Z \leftarrow Z + 1$ LDRd, -ZLoad Indirect and Pre-Dec. $Z \leftarrow Z - 1$, Rd $\leftarrow (Z)$ LDDRd, Z+qLoad Indirect with DisplacementRd $\leftarrow (Z + q)$ LDSRd, kLoad Direct from SRAMRd $\leftarrow (k)$ STX, RrStore Indirect $(X) \leftarrow Rr$ STX+, RrStore Indirect and Post-Inc. $(X) \leftarrow Rr, X \leftarrow X + 1$ ST-X, RrStore Indirect and Pre-Dec. $X \leftarrow X - 1$, $(X) \leftarrow Rr$ STY, RrStore Indirect and Post-Inc. $(Y) \leftarrow Rr, Y \leftarrow Y + 1$ STY+, RrStore Indirect and Post-Inc. $(Y) \leftarrow Rr, Y \leftarrow Y + 1$ ST-Y, RrStore Indirect and Post-Inc. $(Y) \leftarrow Rr, Y \leftarrow Y + 1$ ST-Y, RrStore Indirect and Post-Inc. $(Y) \leftarrow Rr, Y \leftarrow Y + 1$ ST-Y, RrStore Indirect and Post-Inc. $(Y) \leftarrow Rr, Y \leftarrow Y + 1$ ST-Y, RrStore Indirect and Pre-Dec. $Y \leftarrow Y - 1$, $(Y) \leftarrow Rr$	None	2
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $		2
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $		2
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	None	2
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	None	2
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	None	2
$\begin{tabular}{ c c c c c c c c c c c c c c c c c c c$	None	2
STX, RrStore Indirect $(X) \leftarrow Rr$ STX+, RrStore Indirect and Post-Inc. $(X) \leftarrow Rr, X \leftarrow X + 1$ ST-X, RrStore Indirect and Pre-Dec. $X \leftarrow X - 1, (X) \leftarrow Rr$ STY, RrStore Indirect $(Y) \leftarrow Rr$ STY+, RrStore Indirect and Post-Inc. $(Y) \leftarrow Rr, Y \leftarrow Y + 1$ ST-Y, RrStore Indirect and Post-Inc. $(Y) \leftarrow Rr, Y \leftarrow Y + 1$ ST-Y, RrStore Indirect and Pre-Dec. $Y \leftarrow Y - 1, (Y) \leftarrow Rr$	None	2
STX+, RrStore Indirect and Post-Inc. $(X) \leftarrow Rr, X \leftarrow X + 1$ ST- X, RrStore Indirect and Pre-Dec. $X \leftarrow X - 1, (X) \leftarrow Rr$ STY, RrStore Indirect $(Y) \leftarrow Rr$ STY+, RrStore Indirect and Post-Inc. $(Y) \leftarrow Rr, Y \leftarrow Y + 1$ ST- Y, RrStore Indirect and Pre-Dec. $Y \leftarrow Y - 1, (Y) \leftarrow Rr$	None	2
ST- X, RrStore Indirect and Pre-Dec. $X \leftarrow X - 1, (X) \leftarrow Rr$ STY, RrStore Indirect $(Y) \leftarrow Rr$ STY+, RrStore Indirect and Post-Inc. $(Y) \leftarrow Rr, Y \leftarrow Y + 1$ ST- Y, RrStore Indirect and Pre-Dec. $Y \leftarrow Y - 1, (Y) \leftarrow Rr$	None	2
STY, RrStore Indirect $(Y) \leftarrow Rr$ STY+, RrStore Indirect and Post-Inc. $(Y) \leftarrow Rr, Y \leftarrow Y + 1$ ST- Y, RrStore Indirect and Pre-Dec. $Y \leftarrow Y - 1, (Y) \leftarrow Rr$	None	2
STY+, RrStore Indirect and Post-Inc. $(Y) \leftarrow Rr, Y \leftarrow Y + 1$ ST- Y, RrStore Indirect and Pre-Dec. $Y \leftarrow Y - 1, (Y) \leftarrow Rr$		2
ST - Y, Rr Store Indirect and Pre-Dec. $Y \leftarrow Y - 1$, $(Y) \leftarrow Rr$	None	2
		2
		2
QT 7 Dr Store Indirect (7) Dr	None	
STZ, RrStore Indirect $(Z) \leftarrow Rr$ STZ, PrStore Indirect and Port Inc. $(Z) \leftarrow Pr, Z \leftarrow Z + 1$	None	2
STZ+, RrStore Indirect and Post-Inc. $(Z) \leftarrow Rr, Z \leftarrow Z + 1$ STZPrStore Indirect and Pro Dec. $Z \leftarrow Z + 1$	None	2
ST -Z, Rr Store Indirect and Pre-Dec. $Z \leftarrow Z - 1$, $(Z) \leftarrow Rr$		2
STD $Z+q,Rr$ Store Indirect with Displacement $(Z+q) \leftarrow Rr$ STDI/LI/LI/LI/LI/LSTDI/LI/LI/LI/LI/L	None	2
STS k, Rr Store Direct to SRAM (k) ← Rr	None	2
LPM Load Program Memory $R0 \leftarrow (Z)$	None	3
LPM Rd, Z Load Program Memory Rd \leftarrow (Z)	None	3
LPM Rd, Z+ Load Program Memory and Post-Inc $Rd \leftarrow (Z), Z \leftarrow Z+1$	None	3
ELPM Extended Load Program Memory R0 ← (RAMPZ:Z)	None	3
ELPM Rd, Z Extended Load Program Memory Rd ← (Z) ELPM Rd, Z+ Extended Load Program Memory Rd ← (RAMPZ:Z), RAMPZ:Z ← I	None RAMPZ:Z+1 None	3

Mnemonics	Operands	Description	Operation	Flags	#Clocks
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	$Rd \gets P$	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	$Rd \leftarrow STACK$	None	2
	MCU	CONTROL INSTRUCTIONS			
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-chip Debug Only	None	N/A





7. Ordering information

7.1 Atmel AT90USB646

Speed [MHz]	Power supply [V]	Ordering code ⁽²⁾	USB interface	Package ⁽¹⁾	Operating range
16 ⁽³⁾	2.7-5.5	AT90USB646-AU AT90USB646-MU	Device	MD PS	Industrial (-40° 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.

MD	64 - lead, 14×14 mm body size, 1.0mm body thickness 0.8mm lead pitch, thin profile plastic quad flat package (TQFP)
PS	64 - lead, 9 × 9mm body size, 0.50mm pitch Quad flat no lead package (QFN)

7.2 Atmel AT90USB647

Speed [MHz]	Power supply [V]	Ordering code ⁽²⁾	USB interface	Package ⁽¹⁾	Operating range
16 ⁽³⁾	2.7-5.5	AT90USB647-AU AT90USB647-MU	USB OTG	MD PS	Industrial (-40° 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.

MD	64 - lead, 14 × 14mm body size, 1.0mm body thickness 0.8mm lead pitch, thin profile plastic quad flat package (TQFP)
PS	64 - lead, 9 × 9mm body size, 0.50mm pitch Quad flat no lead package (QFN)





7.3 Atmel AT90USB1286

Speed [MHz]	Power supply [V]	Ordering code ⁽²⁾	USB interface	Package ⁽¹⁾	Operating range
16 ⁽³⁾	2.7-5.5	AT90USB1286-AU AT90USB1286-MU	Device	MD PS	Industrial (-40° 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.

MD	64 - lead, 14 × 14mm body size, 1.0mm body thickness 0.8mm lead pitch, thin profile plastic quad flat package (TQFP)	
PS	64 - lead, 9 × 9mm body size, 0.50mm pitch Quad flat no lead package (QFN)	

7.4 Atmel AT90USB1287

Speed [MHz]	Power supply [V]	Ordering code ⁽²⁾	USB interface	Package ⁽¹⁾	Operating range
16 ⁽³⁾	2.7-5.5	AT90USB1287-AU AT90USB1287-MU	Host (OTG)	MD PS	Industrial (-40° 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.

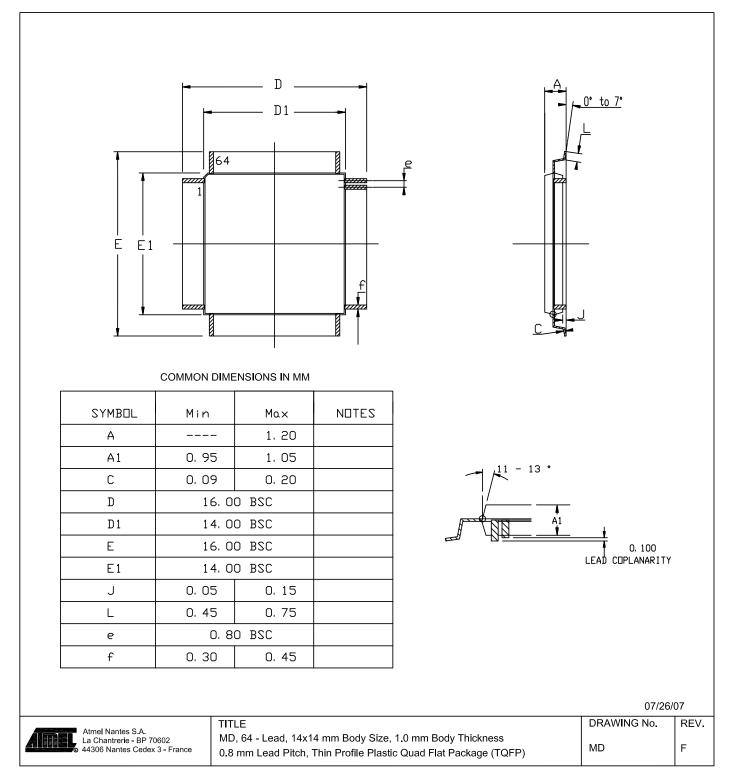
MD	64 - lead, 14 × 14mm body size, 1.0mm body thickness 0.8mm lead pitch, thin profile plastic quad flat package (TQFP)
PS	64 - lead, 9 × 9mm body size, 0.50mm pitch Quad flat no lead package (QFN)





8. Packaging information

8.1 TQFP64



NOTES: STANDARD NOTES FOR PQFP/VQFP/TQFP/DQFP

1. DIMENSIONING & TOLERANCING CONFORM TO ASME Y14.5M. - 1982.

2. "D1 AND E1" DIMENSIONS DO NOT INCLUDE MOLD PROTUSIONS MOLD PROTUSIONS SHALL NOT EXCEED 0.25 mm (0.010 INCH) . THE TOP PACKAGE BODY SIZE MAY BE SMALLER THAN THE BOTTOM PACKAGE BODY SIZE BY AS MUCH AS 0.15 mm.

3. DATUM PLANE "H" LOCATED AT MOLD PARTING LINE AND COINCIDENT WITH LEAD, WHERE LEAD EXISTS PLASTIC BODY AT BOTTOM OF PARTING LINE.

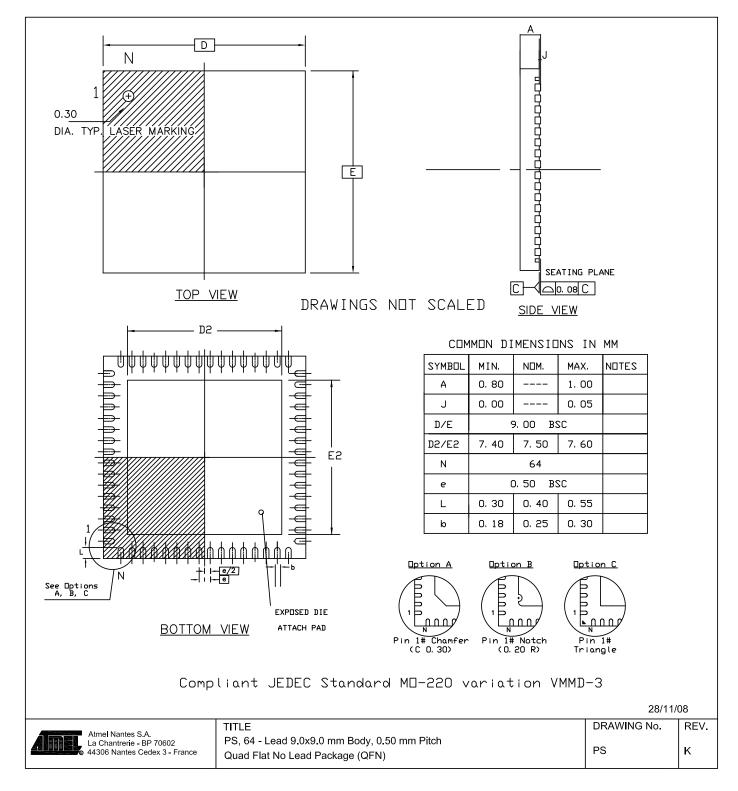
4. DATUM "A" AND "D" TO BE DETERMINED AT DATUM PLANE H.

5. DIMENSION "f" DOES NOT INCLUDE DAMBAR PROTUSION ALLOWABLE DAMBAR PROTUSION SHALL BE 0.08 mm/.003" TOTAL EXCESS OF THE "f" DIMENSION AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT.





8.2 QFN64



NOTES: QFN STANDARD NOTES

1. DIMENSIONING & TOLERANCING CONFORM TO ASME Y14.5M. - 1994.

2. DIMENSION & APPLIES TO METALLIZED TERMINAL AND IS MEASURED

BETWEEN 0.15 AND 0.30 mm FROM TERMINAL TIP. IF THE TERMINAL HAS THE OPTIONAL RADIUS ON THE OTHER END OF THE TERMINAL, THE DIMENSION & SHOULD NOT BE MEASURED IN THAT RADIUS AREA.

3. MAX. PACKAGE WARPAGE IS 0.05mm.

4. MAXIMUM ALLOWABLE BURRS IS 0.076 mm IN ALL DIRECTIONS.

5. PIN #1 ID ON TOP WILL BE LASER MARKED.

6. THIS DRAWING CONFORMES TO JEDEC REGISTERED OUTLINE MO-220.

7. A MAXIMUM 0.15mm PULL BACK (L1) MAY BE PRESENT.

L MINUS L1 TO BE EQUAL TO OR GREATER THAN 0.30 mm

8. THE TERMINAL #1 IDENTIFIER ARE OPTIONAL BUT MUST BE LOCATED WITHIN THE ZONE INDICATED. THE TERMINAL #1 IDENTIFIER BE EITHER A MOLD OR MARKED FEATURE





9. Errata

9.1 Atmel AT90USB1287/6 errata

9.1.1 AT90USB1287/6 errata history

Silicon Release	90USB1286-16MU	90USB1287-16AU	90USB1287-16MU
First Release	Date Code up to 0648	Date Code up to 0714 and lots 0735 6H2726 $^{(1)}$	Date Code up to 0701
Second Release	Date Code from 0709 to 0801 except lots 0801 7H5103 ⁽¹⁾	from Date Code 0722 to 0806 except lots 0735 6H2726 ⁽¹⁾	Date Code from 0714 to 0810 except lots 0748 7H5103 ⁽¹⁾
Third Release	Lots 0801 7H5103 ⁽¹⁾ and Date Code from 0814	Date Code from 0814	Lots 0748 7H5103 ⁽¹⁾ and Date Code from 0814
Fourth Release	TBD	TBD	TBD

Notes: 1. A blank or any alphanumeric string.

9.1.2 AT90USB1287/6 first release

- Incorrect CPU behavior for VBUSTI and IDTI interrupts routines
- USB Eye Diagram violation in low-speed mode
- Transient perturbation in USB suspend mode generates over consumption
- VBUS Session valid threshold voltage
- USB signal rate
- VBUS residual level
- Spike on TWI pins when TWI is enabled
- High current consumption in sleep mode
- · Async timer interrupt wake up from sleep generate multiple interrupts

9. Incorrect CPU behavior for VBUSTI and IDTI interrupts routines

The CPU core may incorrectly execute the interrupt vector related to the VBUSTI and IDTI interrupt flags.

Problem fix/workaround

Do not enable these interrupts, firmware must process these USB events by polling VBUSTI and IDTI flags.

8. USB Eye Diagram violation in low-speed mode

The low to high transition of D- violates the USB eye diagram specification when transmitting with low-speed signaling.

Problem fix/workaround

None.

7. Transient perturbation in USB suspend mode generates overconsumption

In device mode and when the USB is suspended, transient perturbation received on the USB lines generates a wake up state. However the idle state following the perturbation does

not set the SUSPI bit anymore. The internal USB engine remains in suspend mode but the USB differential receiver is still enabled and generates a typical 300µA extra-power consumption. Detection of the suspend state after the transient perturbation should be performed by software (instead of reading the SUSPI bit).

Problem fix/workaround

USB waiver allows bus powered devices to consume up to 2.5mA in suspend state.

6. VBUS session valid threshold voltage

The VSession valid threshold voltage is internally connected to VBus_Valid (4.4V approx.). That causes the device to attach to the bus only when Vbus is greater than VBusValid instead of V_Session Valid. Thus if VBUS is lower than 4.4V, the device is detached.

Problem fix/workaround

According to the USB power drop budget, this may require connecting the device toa root hub or a self-powered hub.

5. UBS signal rate

The average USB signal rate may sometime be measured out of the USB specifications $(12MHz \pm 30kHz)$ with short frames. When measured on a long period, the average signal rate value complies with the specifications. This bit rate deviation does not generates communication or functional errors.

Problem fix/workaround

None.

4. VBUS residual level

In USB device and host mode, once a 5V level has been detected to the VBUS pad, a residual level (about 3V) can be measured on the VBUS pin.

Problem fix/workaround

None.

3. Spike on TWI pins when TWI is enabled

100ns negative spike occurs on SDA and SCL pins when TWI is enabled.

Problem fix/workaround

No known workaround, enable Atmel AT90USB64/128 TWI first versus the others nodes of the TWI network.

2. High current consumption in sleep mode

If a pending interrupt cannot wake the part up from the selected mode, the current consumption will increase during sleep when executing the SLEEP instruction directly after a SEI instruction.

Problem fix/workaround

Before entering sleep, interrupts not used to wake up the part from the sleep mode should be disabled.





1. Asynchronous timer interrupt wake up from sleep generates multiple interrupts

If the CPU core is in sleep and wakes-up from an asynchronous timer interrupt and then go back in sleep again it may wake up multiple times.

Problem fix/workaround

A software workaround is to wait with performing the sleep instruction until TCNT2>OCR2+1.

9.1.3 Atmel AT90USB1287/6 second release

- Incorrect CPU behavior for VBUSTI and IDTI interrupts routines
- USB Eye Diagram violation in low-speed mode
- Transient perturbation in USB suspend mode generates over consumption
- VBUS Session valid threshold voltage
- Spike on TWI pins when TWI is enabled
- High current consumption in sleep mode
- · Async timer interrupt wake up from sleep generate multiple interrupts

7. Incorrect CPU behavior for VBUSTI and IDTI interrupts routines

The CPU core may incorrectly execute the interrupt vector related to the VBUSTI and IDTI interrupt flags.

Problem fix/workaround

Do not enable these interrupts, firmware must process these USB events by polling VBUSTI and IDTI flags.

6. USB Eye Diagram violation in low-speed mode

The low to high transition of D- violates the USB eye diagram specification when transmitting with low-speed signaling.

Problem fix/workaround

None.

5. Transient perturbation in USB suspend mode generates overconsumption

In device mode and when the USB is suspended, transient perturbation received on the USB lines generates a wake up state. However the idle state following the perturbation does not set the SUSPI bit anymore. The internal USB engine remains in suspend mode but the USB differential receiver is still enabled and generates a typical 300µA extra-power consumption. Detection of the suspend state after the transient perturbation should be performed by software (instead of reading the SUSPI bit).

Problem fix/workaround

USB waiver allows bus powered devices to consume up to 2.5mA in suspend state.

4. VBUS session valid threshold voltage

The VSession valid threshold voltage is internally connected to VBus_Valid (4.4V approx.). That causes the device to attach to the bus only when Vbus is greater than VBusValid instead of V_Session Valid. Thus if VBUS is lower than 4.4V, the device is detached.

Problem fix/workaround

According to the USB power drop budget, this may require connecting the device toa root hub or a self-powered hub.

3. Spike on TWI pins when TWI is enabled

100ns negative spike occurs on SDA and SCL pins when TWI is enabled.





Problem fix/workaround

No known workaround, enable Atmel AT90USB64/128 TWI first versus the others nodes of the TWI network.

2. High current consumption in sleep mode

If a pending interrupt cannot wake the part up from the selected mode, the current consumption will increase during sleep when executing the SLEEP instruction directly after a SEI instruction.

Problem fix/workaround

Before entering sleep, interrupts not used to wake up the part from the sleep mode should be disabled.

1. Asynchronous timer interrupt wake up from sleep generates multiple interrupts

If the CPU core is in sleep and wakes-up from an asynchronous timer interrupt and then go back in sleep again it may wake up multiple times.

Problem fix/workaround

A software workaround is to wait with performing the sleep instruction until TCNT2>OCR2+1.

9.1.4 Atmel AT90USB1287/6 Third Release

- Incorrect CPU behavior for VBUSTI and IDTI interrupts routines
- Transient perturbation in USB suspend mode generates over consumption
- Spike on TWI pins when TWI is enabled
- High current consumption in sleep mode
- · Async timer interrupt wake up from sleep generate multiple interrupts

5. Incorrect CPU behavior for VBUSTI and IDTI interrupts routines

The CPU core may incorrectly execute the interrupt vector related to the VBUSTI and IDTI interrupt flags.

Problem fix/workaround

Do not enable these interrupts, firmware must process these USB events by polling VBUSTI and IDTI flags.

4. Transient perturbation in USB suspend mode generates overconsumption

In device mode and when the USB is suspended, transient perturbation received on the USB lines generates a wake up state. However the idle state following the perturbation does not set the SUSPI bit. The internal USB engine remains in suspend mode but the USB differential receiver is still enabled and generates a typical 300µA extra-power consumption. Detection of the suspend state after the transient perturbation should be performed by software (instead of reading the SUSPI bit).

Problem fix/workaround

USB waiver allows bus powered devices to consume up to 2.5mA in suspend state.

3. Spike on TWI pins when TWI is enabled

100ns negative spike occurs on SDA and SCL pins when TWI is enabled.

Problem fix/workaround

No known workaround, enable AT90USB64/128 TWI first, before the others nodes of the TWI network.

2. High current consumption in sleep mode

If a pending interrupt cannot wake the part up from the selected mode, the current consumption will increase during sleep when executing the SLEEP instruction directly after a SEI instruction.

Problem fix/workaround

Before entering sleep, interrupts not used to wake up the part from sleep mode should be disabled.

1. Asynchronous timer interrupt wake up from sleep generates multiple interrupts If the CPU core is in sleep mode and wakes-up from an asynchronous timer interrupt and then goes back into sleep mode, it may wake up multiple times.





Problem fix/workaround

A software workaround is to wait before performing the sleep instruction: until TCNT2>OCR2+1.

9.1.5 Atmel AT90USB1287/6 Fourth Release

- Transient perturbation in USB suspend mode generates over consumption
- Spike on TWI pins when TWI is enabled
- High current consumption in sleep mode
- Async timer interrupt wake up from sleep generate multiple interrupts

4. Transient perturbation in USB suspend mode generates overconsumption

In device mode and when the USB is suspended, transient perturbation received on the USB lines generates a wake up state. However the idle state following the perturbation does not set the SUSPI bit. The internal USB engine remains in suspend mode but the USB differential receiver is still enabled and generates a typical 300μ A extra-power consumption. Detection of the suspend state after the transient perturbation should be performed by software (instead of reading the SUSPI bit).

Problem fix/workaround

USB waiver allows bus powered devices to consume up to 2.5mA in suspend state.

3. Spike on TWI pins when TWI is enabled

100ns negative spike occurs on SDA and SCL pins when TWI is enabled.

Problem fix/workaround

No known workaround, enable Atmel AT90USB64/128 TWI first, before the others nodes of the TWI network.

2. High current consumption in sleep mode

If a pending interrupt cannot wake the part up from the selected mode, the current consumption will increase during sleep when executing the SLEEP instruction directly after a SEI instruction.

Problem fix/workaround

Before entering sleep, interrupts not used to wake up the part from sleep mode should be disabled.

1. Asynchronous timer interrupt wake up from sleep generates multiple interrupts

If the CPU core is in sleep mode and wakes-up from an asynchronous timer interrupt and then goes back into sleep mode, it may wake up multiple times.

Problem fix/workaround

A software workaround is to wait before performing the sleep instruction: until TCNT2>OCR2+1.





9.2 Atmel AT90USB646/7 errata

9.2.1 AT90USB646/7 errata history TBD

Silicon Release	90USB646-16MU	90USB647-16AU	90USB647-16MU
First Release			
Second Release			

Note '*' means a blank or any alphanumeric string.

9.2.2 AT90USB646/7 first release.

- Incorrect interrupt routine execution for VBUSTI, IDTI interrupts flags
- USB Eye Diagram violation in low-speed mode
- Transient perturbation in USB suspend mode generates over consumption
- Spike on TWI pins when TWI is enabled
- High current consumption in sleep mode
- Async timer interrupt wake up from sleep generate multiple interrupts

6. Incorrect CPU behavior for VBUSTI and IDTI interrupts routines

The CPU core may incorrectly execute the interrupt vector related to the VBUSTI and IDTI interrupt flags.

Problem fix/workaround

Do not enable these interrupts, firmware must process these USB events by polling VBUSTI and IDTI flags.

5. USB Eye Diagram violation in low-speed mode

The low to high transition of D- violates the USB eye diagram specification when transmitting with low-speed signaling.

Problem fix/workaround

None.

4. Transient perturbation in USB suspend mode generates overconsumption

In device mode and when the USB is suspended, transient perturbation received on the USB lines generates a wake up state. However the idle state following the perturbation does not set the SUSPI bit anymore. The internal USB engine remains in suspend mode but the USB differential receiver is still enabled and generates a typical 300µA extra-power consumption. Detection of the suspend state after the transient perturbation should be performed by software (instead of reading the SUSPI bit).

Problem fix/workaround

USB waiver allows bus powered devices to consume up to 2.5mA in suspend state.

3. Spike on TWI pins when TWI is enabled

100ns negative spike occurs on SDA and SCL pins when TWI is enabled.

Problem fix/workaround

No known workaround, enable Atmel AT90USB64/128 TWI first versus the others nodes of the TWI network.

2. High current consumption in sleep mode

If a pending interrupt cannot wake the part up from the selected mode, the current consumption will increase during sleep when executing the SLEEP instruction directly after a SEI instruction.

Problem fix/workaround

Before entering sleep, interrupts not used to wake up the part from the sleep mode should be disabled.

1. Asynchronous timer interrupt wake up from sleep generates multiple interrupts

If the CPU core is in sleep and wakes-up from an asynchronous timer interrupt and then go back in sleep mode again it may wake up several times.

Problem fix/workaround

A software workaround is to wait with performing the sleep instruction until TCNT2>OCR2+1.





9.2.3 Atmel AT90USB646/7 Second Release.

- USB Eye Diagram violation in low-speed mode
- Transient perturbation in USB suspend mode generates over consumption
- Spike on TWI pins when TWI is enabled
- High current consumption in sleep mode
- Async timer interrupt wake up from sleep generate multiple interrupts

5. USB Eye Diagram violation in low-speed mode

The low to high transition of D- violates the USB eye diagram specification when transmitting with low-speed signaling.

Problem fix/workaround

None.

4. Transient perturbation in USB suspend mode generates overconsumption

In device mode and when the USB is suspended, transient perturbation received on the USB lines generates a wake up state. However the idle state following the perturbation does not set the SUSPI bit anymore. The internal USB engine remains in suspend mode but the USB differential receiver is still enabled and generates a typical 300µA extra-power consumption. Detection of the suspend state after the transient perturbation should be performed by software (instead of reading the SUSPI bit).

Problem fix/workaround

USB waiver allows bus powered devices to consume up to 2.5mA in suspend state.

3. Spike on TWI pins when TWI is enabled

100ns negative spike occurs on SDA and SCL pins when TWI is enabled.

Problem fix/workaround

No known workaround, enable Atmel AT90USB64/128 TWI first versus the others nodes of the TWI network.

2. High current consumption in sleep mode

If a pending interrupt cannot wake the part up from the selected mode, the current consumption will increase during sleep when executing the SLEEP instruction directly after a SEI instruction.

Problem fix/workaround

Before entering sleep, interrupts not used to wake up the part from the sleep mode should be disabled.

1. Asynchronous timer interrupt wake up from sleep generates multiple interrupts

If the CPU core is in sleep and wakes-up from an asynchronous timer interrupt and then go back in sleep mode again it may wake up several times.

Problem fix/workaround

A software workaround is to wait with performing the sleep instruction until TCNT2>OCR2+1.

10. Datasheet revision history for Atmel AT90USB64/128

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 Changes from 7593A to 7593B

- 1. Changed default configuration for fuse bytes and security byte.
- 2. Suppression of timer 4,5 registers which does not exist.
- 3. Updated typical application schematics in USB section

10.2 Changes from 7593B to 7593C

1. Update to package drawings, MQFP64 and TQFP64.

10.3 Changes from 7593C to 7593D

 For further product compatibility, changed USB PLL possible prescaler configurations. Only 8MHz and 16MHz crystal frequencies allows USB operation (see Table 7-11 on page 50).

10.4 Changes from 7593D to 7593E

- 1. Updated PLL Prescaler table: configuration words are different between AT90USB64x and AT90USB128x to enable the PLL with a 16MHz source.
- 2. Cleaned up some bits from USB registers, and updated information about OTG timers, remote wake-up, reset and connection timings.
- 3. Updated clock distribution tree diagram (USB prescaler source and configuration register).
- 4. Cleaned up register summary.
- 5. Suppressed PCINT23:8 that do not exist from External Interrupts.
- 6. Updated Electrical Characteristics.
- 7. Added Typical Characteristics.
- 8. Update Errata section.

10.5 Changes from 7593E to 7593F

- 1. Removed 'Preliminary' from document status.
- 2. Clarification in Stand by mode regarding USB.

10.6 Changes from 7593F to 7593G

1. Updated Errata section.

10.7 Changes from 7593G to 7593H

- 1. Added Signature information for 64K devices.
- 2. Fixed figure for typical bus powered application
- 3. Added min/max values for BOD levels
- 4. Added ATmega32U6 product
- 5. Update Errata section
- 6. Modified descriptions for HWUPE and WAKEUPE interrupts enable (these interrupts should be enabled only to wake up the CPU core from power down mode).





7. Added description to access unique serial number located in Signature Row see "Reading the Signature Row from software" on page 354.

10.8 Changes from 7593H to 7593I

1. Updated Table 9-2 in "Brown-out detection" on page 60. Unused BOD levels removed.

10.9 Changes from 7593I to 7593J

- 1. Updated Table 9-2 in "Brown-out detection" on page 60. BOD level 100 removed.
- 2. Updated "Ordering information" on page 18.
- 3. Removed ATmega32U6 errata section.

10.10 Changes from 7593J to 7593K

- 1. Corrected Figure 6-7 on page 34, Figure 6-8 on page 34 and Figure 6-9 on page 35.
- Corrected ordering information for Section 7.3 "Atmel AT90USB1286" on page 20, Section 7.4 "Atmel AT90USB1287" on page 21 andSection 7.2 "Atmel AT90USB647" on page 19.
- 3. Removed the ATmega32U6 device and updated the datasheet accordingly.
- 4. Updated Assembly Code Example in "Watchdog reset" on page 61.

10.11 Changes from 7593K to 7593L

- 1. Updated the "Ordering information" on page 18. Changed the speed from 20MHz to 16MHz.
- 2. Replaced ATmegaAT90USBxxxx by AT90USBxxxx through the datasheet.
- 3. Updated the first paragraph of "Overview" on page 307. Port A replaced by Port F.
- 4. Updated ADC equation in "ADC conversion result" on page 318. The equation has 1024 instead of 1023.
- 5. Created "Packaging Information" chapter.
- 6. Replaced the "QFN64" Packaging by an updated QFN64 Packaging drawing.
- 7. Updated "Errata" on page 26. AT90USB1286/7 has a fourth release, while AT90USB646/7 updated with a second release.
- 8. In Section "Overview" on page 307, "Port A" has been replaced by "Port F" in the first section.
- 9. In Section "Atmel AT90USB647" on page 19 the USB interface has been changed to USB OTG.
- 10. In Section "Atmel AT90USB1286" on page 20 the USB interface has been changed to Device.
- 11. In Section "Atmel AT90USB1287" on page 21 the USB interface has been changed to Host OTG.
- 12. General update according to new template.



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

16F, Shin Osaki Kangyo Bldg. 1-6-4 Osaki Shinagawa-ku Tokyo 104-0032 JAPAN Tel: (+81) 3-6417-0300 Fax: (+81) 3-6417-0370

© 2012 Atmel Corporation. All rights reserved.

Atmel[®], Atmel logo and combinations thereof, AVR[®], AVR Studio[®], and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Windows[®] is a registered trademark 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 DIRECT, 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.