Microchip Technology - AT94K05AL-25BQI Datasheet

Welcome to E-XFL.COM

Embedded - FPGAs (Field Programmable Gate Array) with Microcontrollers: Enhancing Flexibility and Performance

Embedded - FPGAs (Field Programmable Gate Arrays) with Microcontrollers represent a cutting-edge category of electronic components that combine the flexibility of FPGA technology with the processing power of integrated microcontrollers. This hybrid approach offers a versatile solution for designing and implementing complex digital systems that require both programmable logic and embedded processing capabilities.

What Are Embedded - FPGAs with Microcontrollers?

Details	
Product Status	Obsolete
Core Type	8-Bit AVR
Speed	25 MHz
Interface	I²C, UART
Program SRAM Bytes	4K-16K
FPGA SRAM	2kb
EEPROM Size	-
Data SRAM Bytes	4K ~ 16K
FPGA Core Cells	256
FPGA Gates	5K
FPGA Registers	436
Voltage - Supply	3V ~ 3.6V
Mounting Type	Surface Mount
Operating Temperature	-40°C ~ 85°C
Package / Case	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at94k05al-25bqi

Email: info@E-XFL.COM Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong

Table 11. FPSLIC System Control Register

Bit	Description
SCR56	0 = Disable XTAL Pin ($R_{feedback}$) 1 = Enable XTAL Pin ($R_{feedback}$)
SCR57	0 = Disable TOSC2 Pin ($R_{feedback}$) 1 = Enable TOSC2 Pin ($R_{feedback}$)
SCR58 - SCR59	Reserved

SCR60 - SCR61	SCR61 = 0, SCR60 = 0 “1”
	SCR61 = 0, SCR60 = 1 AVR System Clock
	SCR61 = 1, SCR60 = 0 Timer Oscillator Clock (TOSC1)(1)
	SCR61 = 1, SCR60 = 1 Watchdog Clock
	Global Clock 6 mux select (set by using the AT94K Device Options in System Designer).
	Note: 1. The AS2 bit must be set in the ASSR register.

SCR62	0 = Disable CacheLogic Writes to FPGA by AVR 1 = Enable CacheLogic Writes to FPGA by AVR
SCR63	0 = Disable Access (Read and Write) to SRAM by FPGA 1 = Enable Access (Read and Write) to SRAM by FPGA

Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clock
CBR	Rd, K	Clear Bit(s) in Register	Rd ← Rd • ($FFh - K)	Z,N,V,S	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V,S	1
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V,S	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V,S	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V,S	1
SER	Rd	Set Register	Rd ← $FF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd × Rr (UU)	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd × Rr (SS)	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd × Rr (SU)	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd × Rr)<<1 (UU)	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd × Rr)<<1 (SS)	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd × Rr)<<1 (SU)	Z,C	2
		Branch Instructions			
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC(15:0) ← Z	None	2
JMP	k	Jump	PC ← k	None	3
RCALL	k	Relative Call Subroutine	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC(15:0) ← Z	None	3
CALL	k	Call Subroutine	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd, Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2 / 3
CP	Rd, Rr	Compare	Rd - Rr	Z,C,N,V,S,H	1
CPC	Rd, Rr	Compare with Carry	Rd - Rr - C	Z,C,N,V,S,H	1
CPI	Rd, K	Compare with Immediate	Rd - K	Z,C,N,V,S,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b) = 0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBRS	Rr, b	Skip if Bit in Register Set	if (Rr(b) = 1) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIC	A, b	Skip if Bit in I/O Register Cleared	if(I/O(A,b) = 0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIS	A, b	Skip if Bit in I/O Register Set	If(I/O(A,b) = 1) PC ← PC + 2 or 3	None	1 / 2 / 3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC ←PC+k+1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC ←PC+k+1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2

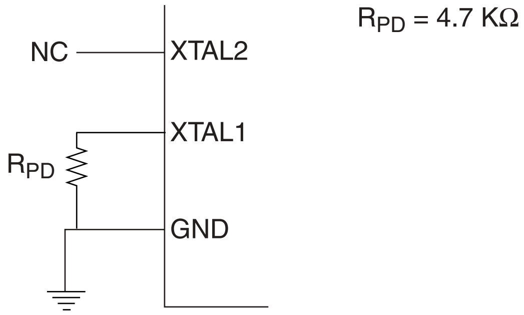
Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clock
ST	-Y, Rr	Store Indirect and Pre-Decrement	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q, Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Increment	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Decrement	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q, Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Increment	Rd ← (Z), Z ← Z + 1	None	3
IN	Rd, A	In From I/O Location	Rd ← I/O(A)	None	1
OUT	A, Rr	Out To I/O Location	I/O(A) ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
		Bit and Bit-test Instructions			
LSL	Rd	Logical Shift Left	Rd(n+1)←Rd(n),Rd(0)←0,C←Rd(7)	Z,C,N,V,H	1
LSR	Rd	Logical Shift Right	Rd(n)←Rd(n+1),Rd(7)←0,C←Rd(0)	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)←Rd(n),C←Rd(7)	Z,C,N,V,H	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)←Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0) ↔ Rd(7..4)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
SBI	A, b	Set Bit in I/O Register	I/O(A, b) ← 1	None	2
CBI	A, b	Clear Bit in I/O Register	I/O(A, b) ← 0	None	2
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1

No Clock/Oscillator Source

When not in use, for low static IDD, add a pull-down resistor to XTAL1.

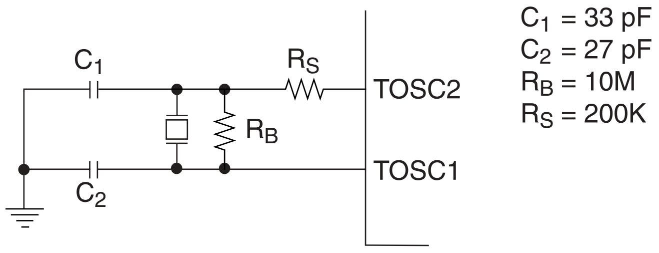
Figure 26. No Clock/Oscillator Connections



Timer Oscillator

For the timer oscillator pins, TOSC1 and TOSC2, the crystal is connected directly between the pins. The oscillator is optimized for use with a 32.768 kHz watch crystal. An external clock signal applied to this pin goes through the same amplifier having a bandwidth of 1 MHz. The external clock signal should therefore be in the range
0 Hz – 1 MHz.

Figure 27. Time Oscillator Connections



Architectural Overview

The AVR uses a Harvard architecture concept – with separate memories and buses for program and data. The program memory is accessed with a single level pipeline. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock-cycle. The program memory is in-system programmable SRAM memory. With a few exceptions, AVR instructions have a single 16-bit word format, meaning that every program memory address contains a single 16-bit instruction.

During interrupts and subroutine calls, the return address program counter (PC) is stored on the stack. The stack is effectively allocated in the general data SRAM, as a consequence, the stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the Stack Pointer (SP) in the reset routine (before subroutines or interrupts are executed). The 16-bit stack pointer is read/write accessible in the I/O space.

The data SRAM can be easily accessed through the five different addressing modes supported in the AVR architecture.

A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the status register. All the different interrupts have a separate interrupt vector in the interrupt vector table at the beginning of the program memory. The different interrupts have priority in accordance with their interrupt vector position. The lower the interrupt vector address, the higher the priority.

The memory spaces in the AVR architecture are all linear and regular memory maps.

X-register, Y-register and Z-register

Registers R26..R31 have some added functions to their general-purpose usage. These registers are address pointers for indirect addressing of the SRAM. The three indirect address registers X, Y and Z have functions as fixed displacement, automatic increment and decrement (see the descriptions for the different instructions).

ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general-purpose working registers. Within a single clock cycle, ALU operations between registers in the register file are executed. The ALU operations are divided into three main categories – arithmetic, logical and bit-functions.

Multiplier Unit

The high-performance AVR Multiplier operates in direct connection with all the 32 general-purpose working registers. This unit performs 8 x 8 multipliers every two clock cycles. See multiplier details on page 106.

SRAM Data Memory

External data SRAM (or program) cannot be used with the FPSLIC AT94K family.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement and Indirect with Post-increment. In the register file, registers R26 to R31 feature the indirect addressing pointer registers.

The Indirect with Displacement mode features a 63 address locations reach from the base address given by the Y- or Z-register.

When using register indirect addressing modes with automatic Pre-decrement and Post-increment, the address registers X, Y and Z are decremented and incremented.

The entire data address space including the 32 general-purpose working registers and the 64 I/O registers are all accessible through all these addressing modes. See the next section for a detailed description of the different addressing modes.

Program and Data Addressing Modes

The embedded AVR core supports powerful and efficient addressing modes for access to the program memory (SRAM) and data memory (SRAM, Register File and I/O Memory). This section describes the different addressing modes supported by the AVR architecture.

Register Direct, Single-register Rd

The operand is contained in register d (Rd).

Register Direct, Two Registers Rd and Rr

Operands are contained in register r (Rr) and d (Rd). The result is stored in register d (Rd).

I/O Direct

Operand address is contained in 6 bits of the instruction word. *n* is the destination or source register address.

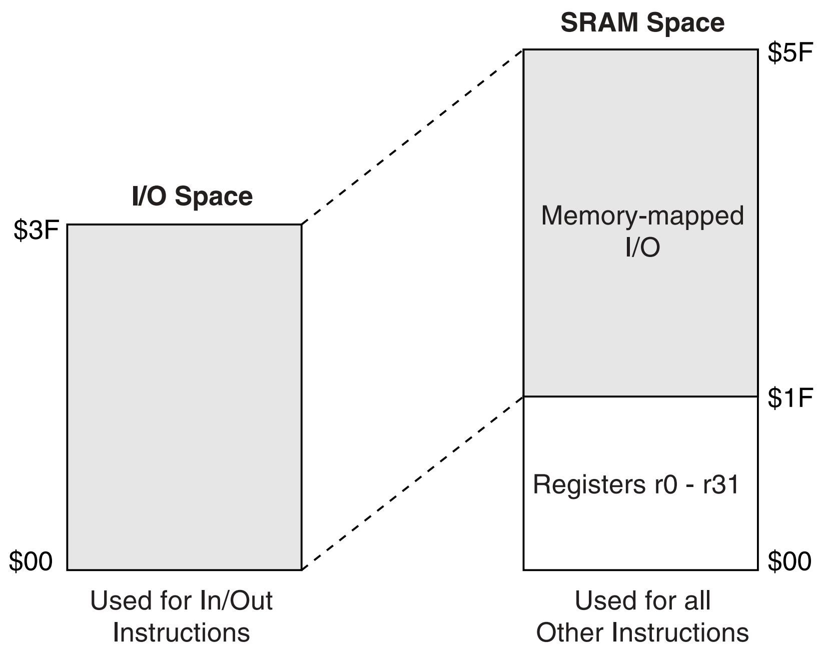
Data Direct

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

Data Indirect with Displacement

Operand address is the result of the Y- or Z-register contents added to the address contained in 6 bits of the instruction word.

Figure 32. Memory-mapped I/O



For single-cycle access (In/Out Commands) to I/O, the instruction has to be less than 16 bits:

opcode	register	address
	r0 - 31 ($1F)	r0 - 63 ($3F)
5 bits	5 bits	6 bits

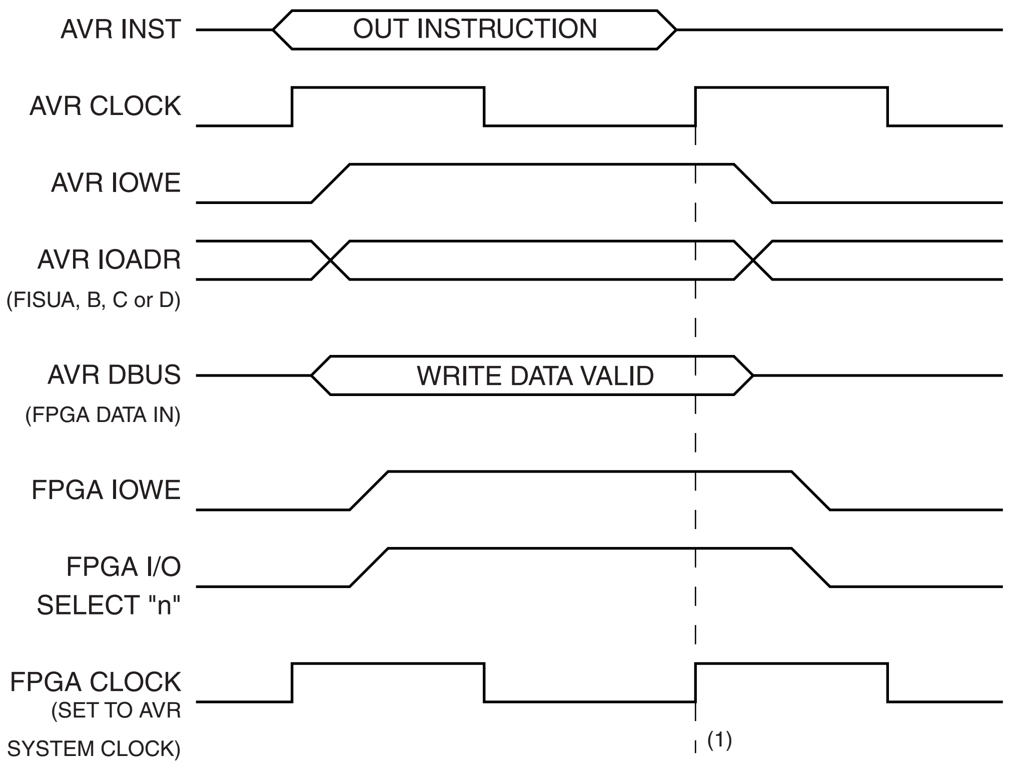
In the data SRAM, the registers are located at memory addresses $00 - $1F and the I/O space is located at memory addresses $20 - $5F.

As there are only 6 bits available to refer to the I/O space, the address is shifted down 2 bits. This means the In/Out commands access $00 to $3F which goes directly to the I/O and maps to $20 to $5F in SRAM. All other instructions access the I/O space through the $20 - $5F addressing.

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

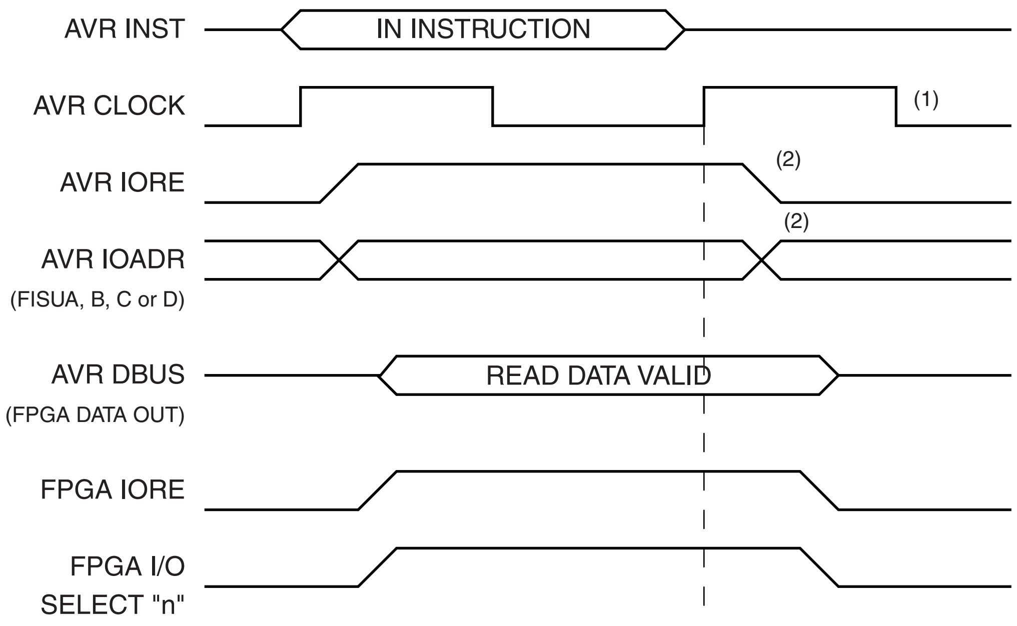
The status flags are cleared by writing a logic 1 to them. Note that the CBI and SBI instructions will operate on all bits in the I/O register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.

Figure 33. Out Instruction – AVR Writing to the FPGA



Note: 1. AVR expects Write to be captured by the FPGA upon posedge of the AVR clock.

Figure 34. In Instruction – AVR Reading FPGA



Notes: 1. AVR captures read data upon posedge of the AVR clock.
2. At the end of an FPGA read cycle, there is a chance for the AVR data bus contention between the FPGA and another peripheral to start to drive (active IORE at new address versus FPGAIORE + Select “n”), but since the AVR clock would have already captured the data from AVR DBUS (= FPGA Data Out), this is a “don’t care” situation.

External Interrupt Mask/Flag Register – EIMF

Bit	7	6	5	4	3	2	1	0	
$3B ($5B)	INTF3	INTF2	INTF1	INTF0	INT3	INT2	INT1	INT0	EIMF
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 3..0 - INT3, 2, 1, 0: External Interrupt Request 3, 2, 1, 0 Enable**

When an INT3 - INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the corresponding external pin interrupt is enabled. The external interrupts are always negative edge triggered interrupts, see "Sleep Modes" on page 66.

- **Bits 7..4 - INTF3, 2, 1, 0: External Interrupt 3, 2, 1, 0 Flags**

When a falling edge is detected on the INT3, 2, 1, 0 pins, an interrupt request is triggered. The corresponding interrupt flag, INTF3, 2, 1, 0 becomes set (one). If the I-bit in SREG and the corresponding interrupt enable bit, INT3, 2, 1, 0 in EIMF, are set (one), the MCU will jump to the interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag is cleared by writing a logic 1 to it.

Timer/Counter Interrupt Mask Register – TIMSK

Bit	7	6	5	4	3	2	1	0	
$39 ($39)	TOIE1	OCIE1A	OCIE1B	TOIE2	TICIE1	OCIE2	TOIE0	OCIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - TOIE1: Timer/Counter1 Overflow Interrupt Enable**

When the TOIE1 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter1 occurs, i.e., when the TOV1 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

- **Bit 6 - OCIE1A: Timer/Counter1 Output CompareA Match Interrupt Enable**

When the OCIE1A bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 CompareA Match interrupt is enabled. The corresponding interrupt is executed if a CompareA match in Timer/Counter1 occurs, i.e., when the OCF1A bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

- **Bit 5 - OCIE1B: Timer/Counter1 Output CompareB Match Interrupt Enable**

When the OCIE1B bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 CompareB Match interrupt is enabled. The corresponding interrupt is executed if a CompareB match in Timer/Counter1 occurs, i.e., when the OCF1B bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

- **Bit 4 - TOIE2: Timer/Counter2 Overflow Interrupt Enable**

When the TOIE2 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter2 overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter2 occurs, i.e., when the TOV2 bit is set in the Timer/Counter interrupt flag register – TIFR.

- **Bit 3 - TICIE1: Timer/Counter1 Input Capture Interrupt Enable**

When the TICIE1 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 input capture event interrupt is enabled. The corresponding interrupt is executed if a capture-triggering event occurs on pin 29, (IC1), i.e., when the ICF1 bit is set in the Timer/Counter interrupt flag register – TIFR.

Assuming Run-Test/Idle is the present state, a typical scenario for using the JTAG interface is

- At the TMS input, apply the sequence 1, 1, 0, 0 at the rising edges of TCK to enter the Shift Instruction Register - Shift-IR state. While TMS is Low, shift the 4 bit JTAG instructions into the JTAG instruction register from the TDI input at the rising edge of TCK, while the captured IR-state 0x01 is shifts out on the TDO pin. The JTAG Instruction selects a particular Data Register as path between TDI and TDO and controls the circuitry surrounding the selected Data Register.
- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. The instruction is latched onto the parallel output from the shift register path in the Update-IR state. The Exit-IR, Pause-IR, and Exit2-IR states are only used for navigating the state machine.
- At the TMS input, apply the sequence 1, 0, 0 at the rising edges of TCK to enter the Shift Data Register - Shift-DR state. While TMS is Low, upload the selected Data Register (selected by the present JTAG instruction in the JTAG Instruction Register) from the TDI input at the rising edge of TCK. At the same time, the parallel inputs to the Data Register captured in the Capture-DR state shifts out on the TDO pin.
- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. If the selected Data Register has a latched parallel-output, the latching takes place in the Update-DR state. The Exit-DR, Pause-DR, and Exit2-DR states are only used for navigating the state machine.

As shown in Figure 40 on page 70, the Run-Test/Idle(1) state need not be entered between selecting JTAG instruction and using Data Registers, and some JTAG instructions may select certain functions to be performed in the Run-Test/Idle, making it unsuitable as an Idle state.

Note: 1. Independent of the initial state of the TAP Controller, the Test-Logic-Reset state can always be entered by holding TMS High for 5 TCK clock periods.

Using the Boundary-scan Chain

A complete description of the Boundary-Scan capabilities are given in the section "IEEE 1149.1 (JTAG) Boundary-scan" on page 73.

Using the On-chip Debug System

As shown in Figure 39, the hardware support for On-Chip Debugging consists mainly of

- A scan chain on the interface between the internal AVR CPU and the internal peripheral units
- A breakpoint unit
- A communication interface between the CPU and JTAG system
- A scan chain on the interface between the internal AVR CPU and the FPGA
- A scan chain on the interface between the internal Program/Data SRAM and the FPGA

All read or modify/write operations needed for implementing the Debugger are done by applying AVR instructions via the internal AVR CPU Scan Chain. The CPU sends the result to an I/O memory mapped location which is part of the communication interface between the CPU and the JTAG system.

undetermined state when exiting the test mode. If needed, the BYPASS instruction can be issued to make the shortest possible scan chain through the device. The AVR can be set in the reset state either by pulling the external $\overline{\text{AVR RESET}}$ pin Low, or issuing the AVR_RESET instruction with appropriate setting of the Reset Data Register.

The EXTEST instruction is used for sampling external pins and loading output pins with data. The data from the output latch will be driven out on the pins as soon as the EXTEST instruction is loaded into the JTAG IR-register. Therefore, the SAMPLE/PRELOAD should also be used for setting initial values to the scan ring, to avoid damaging the board when issuing the EXTEST instruction for the first time. SAMPLE/PRELOAD can also be used for taking a snapshot of the AVR's external pins during normal operation of the part.

The JTAG Enable bit must be programmed and the JTD bit in the I/O register MCUR must be cleared to enable the JTAG Test Access Port.

When using the JTAG interface for Boundary-Scan, using a JTAG TCK clock frequency higher than the internal chip frequency is possible. The chip clock is not required to run.

Data Registers

The Data Registers are selected by the JTAG instruction registers described in section "Boundary-scan Specific JTAG Instructions" on page 75. The data registers relevant for Boundary-Scan operations are:

- Bypass Register
- Device Identification Register
- AVR Reset Register
- AVR Boundary-Scan Chain

Bypass Register

The Bypass register consists of a single shift-register stage. When the Bypass register is selected as path between TDI and TDO, the register is reset to 0 when leaving the Capture-DR controller state. The Bypass register can be used to shorten the scan chain on a system when the other devices are to be tested.

Device Identification Register

Figure 41 shows the structure of the Device Identification register.

Figure 41. The format of the Device Identification Register

	MSB			LSB
Bit	31 28	27 12	11 1	0
Device ID	**Version**	**Part Number**	**Manufacturer ID**	**1**
	4 bits	16 bits	11 bits	1 bit

Version

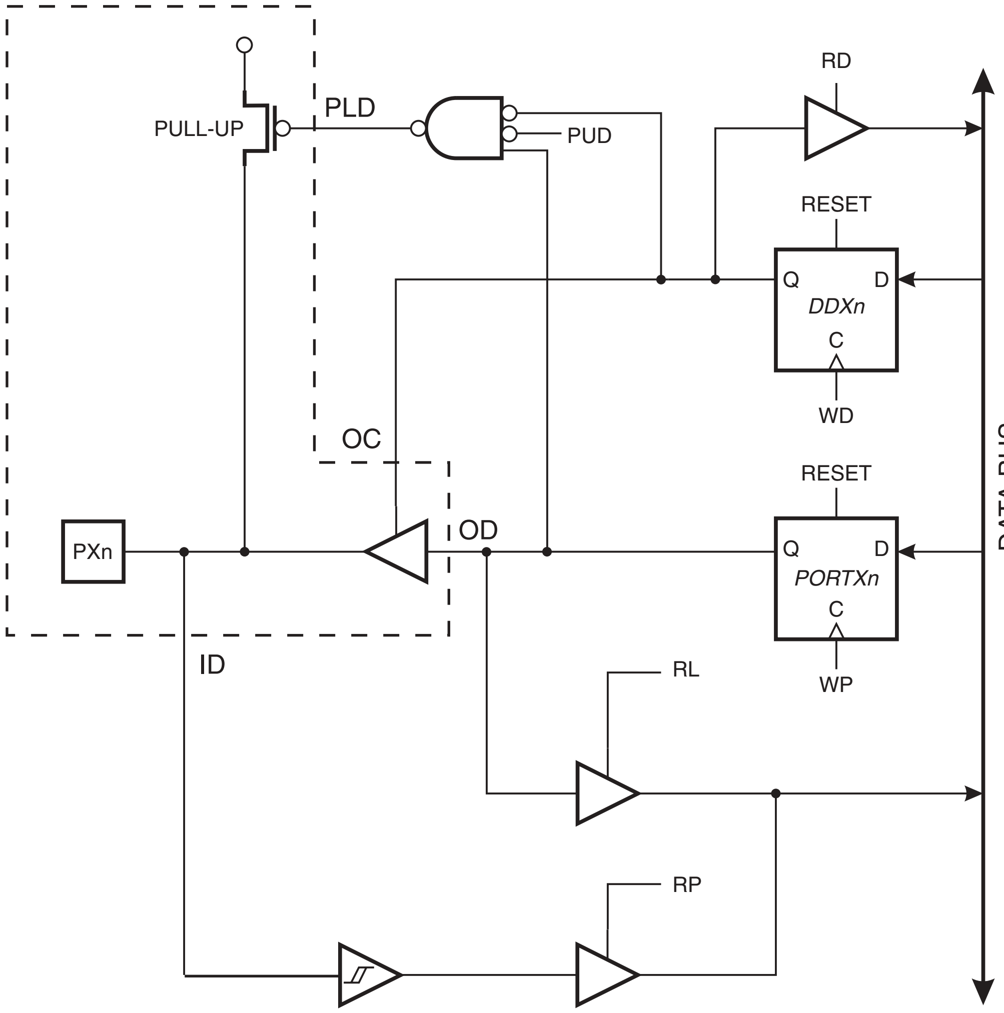
Version is a 4-bit number identifying the revision of the component. The relevant version numbers are shown in Table 18.

Table 18. JTAG Part Version

Device	Version (Binary Digits)
AT94K05	–
AT94K10	0010
AT94K40	–

Figure 44 shows a simple digital Port Pin as described in the section "I/O Ports" on page 147. The Boundary-Scan details from Figure 43 replaces the dashed box in Figure 44.

Figure 44. General Port Pin Schematic Diagram



WP: WRITE PORTX
WD: WRITE DDRX
RL: READ PORTX LATCH
RP: READ PORTX PIN
RD: READ DDRX
n: 0-7
PUD: PULL-UP DISABLE
PuD: JTAG PULL-UP DISABLE
OC: JTAG OUTPUT CONTROL
OD: JTAG OUTPUT DATA
ID: JTAG INPUT DATA

The 16-bit Timer/Counter1 can select the clock source from CK, prescaled CK, or an external pin. In addition it can be stopped as described in section "Timer/Counter1 Control Register B – TCCR1B" on page 98. The different status flags (overflow, compare match and capture event) are found in the Timer/Counter Interrupt Flag Register – TIFR. Control signals are found in the Timer/Counter1 Control Registers – TCCR1A and TCCR1B. The interrupt enable/disable settings for Timer/Counter1 are found in the Timer/Counter Interrupt Mask Register – TIMSK.

When Timer/Counter1 is externally clocked, the external signal is synchronized with the oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

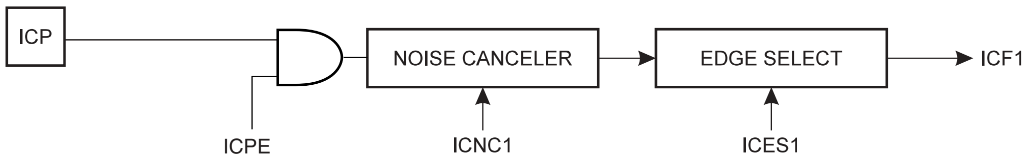
The 16-bit Timer/Counter1 features both a high-resolution and a high-accuracy usage with the lower prescaling opportunities. Similarly, the high-prescaling opportunities makes the Timer/Counter1 useful for lower speed functions or exact-timing functions with infrequent actions.

The Timer/Counter1 supports two Output Compare functions using the Output Compare Register 1 A and B – OCR1A and OCR1B as the data sources to be compared to the Timer/Counter1 contents. The Output Compare functions include optional clearing of the counter on compareA match, and actions on the Output Compare pins on both compare matches.

Timer/Counter1 can also be used as a 8-, 9- or 10-bit Pulse Width Modulator. In this mode, the counter and the OCR1A/OCR1B registers serve as a dual-glitch-free stand-alone PWM with centered pulses. Alternatively, the Timer/Counter1 can be configured to operate at twice the speed in PWM mode, but without centered pulses. Refer to page 101 for a detailed description on this function.

The Input Capture function of Timer/Counter1 provides a capture of the Timer/Counter1 contents to the Input Capture Register – ICR1, triggered by an external event on the Input Capture Pin – PE7(ICP). The actual capture event settings are defined by the Timer/Counter1 Control Register – TCCR1B.

Figure 55. ICP Pin Schematic Diagram



ICPE: Input Capture Pin Enable

If the noise canceler function is enabled, the actual trigger condition for the capture event is monitored over four samples, and all four must be equal to activate the capture flag.

TCNT1 Timer/Counter1 Read

When the CPU reads the low byte TCNT1L, the data of the low byte TCNT1L is sent to the CPU and the data of the high byte TCNT1H is placed in the TEMP register. When the CPU reads the data in the high byte TCNT1H, the CPU receives the data in the TEMP register. Consequently, the low byte TCNT1L must be accessed first for a full 16-bit register read operation.

The Timer/Counter1 is realized as an up or up/down (in PWM mode) counter with read and write access. If Timer/Counter1 is written to and a clock source is selected, the Timer/Counter1 continues counting in the timer clock-cycle after it is preset with the written value.

Timer/Counter1 Output Compare Register – OCR1AH AND OCR1AL

Bit	15	14	13	12	11	10	9	8	
$2B ($4B)	MSB								OCR1AH
$2A ($4A)								LSB	OCR1AL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Timer/Counter1 Output Compare Register – OCR1BH AND OCR1BL

Bit	15	14	13	12	11	10	9	8	
$29 ($49)	MSB								OCR1BH
$28 ($48)								LSB	OCR1BL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The output compare registers are 16-bit read/write registers.

The Timer/Counter1 Output Compare Registers contain the data to be continuously compared with Timer/Counter1. Actions on compare matches are specified in the Timer/Counter1 Control and Status register. A compare match does only occur if Timer/Counter1 counts to the OCR value. A software write that sets TCNT1 and OCR1A or OCR1B to the same value does not generate a compare match.

A compare match will set the compare interrupt flag in the CPU clock cycle following the compare event.

Since the Output Compare Registers – OCR1A and OCR1B – are 16-bit registers, a temporary register TEMP is used when OCR1A/B are written to ensure that both bytes are updated simultaneously. When the CPU writes the high byte, OCR1AH or OCR1BH, the data is temporarily stored in the TEMP register. When the CPU writes the low byte, OCR1AL or OCR1BL, the TEMP register is simultaneously written to OCR1AH or OCR1BH. Consequently, the high byte OCR1AH or OCR1BH must be written first for a full 16-bit register write operation.

The TEMP register is also used when accessing TCNT1, and ICR1. If the main program and also interrupt routines perform access to registers using TEMP, interrupts must be disabled during access from the main program and interrupt routines.

Example 3 – Multiply-accumulate Operation

The final example of 8-bit multiplication shows a multiply-accumulate operation. The general formula can be written as:

$$c(n) = a(n) \times b + c(n-1)$$

```
                ; r17:r16 = r18 * r19 + r17:r16

in    r18,PINB  ; Get the current pin value on port B
ldi   r19,b     ; Load constant b into r19
muls  r19,r18   ; r1:r0 = variable A * variable B
add   r16,r0    ; r17:r16 += r1:r0
adc   r17,r1
```

Typical applications for the multiply-accumulate operation are FIR (Finite Impulse Response) and IIR (Infinite Impulse Response) filters, PID regulators and FFT (Fast Fourier Transform). For these applications the FMULS instruction is particularly useful. The main advantage of using the FMULS instruction instead of the MULS instruction is that the 16-bit result of the FMULS operation always may be approximated to a (well-defined) 8-bit format, see "Using Fractional Numbers" on page 111.

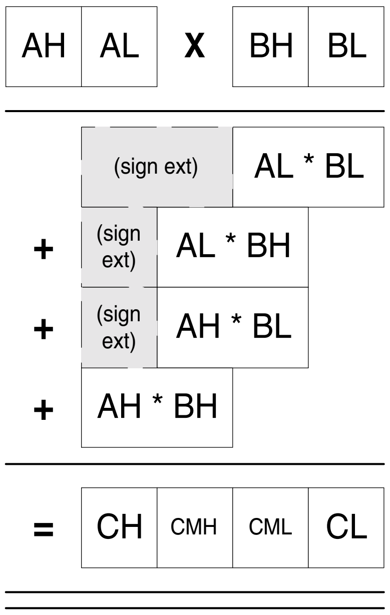
16-bit Multiplication

The new multiply instructions are specifically designed to improve 16-bit multiplication. This section presents solutions for using the hardware multiplier to do multiplication with 16-bit operands.

Figure 60 schematically illustrates the general algorithm for multiplying two 16-bit numbers with a 32-bit result (C = A • B). AH denotes the high byte and AL the low byte of the A operand. CMH denotes the middle high byte and CML the middle low byte of the result C. Equal notations are used for the remaining bytes.

The algorithm is basic for all multiplication. All of the partial 16-bit results are shifted and added together. The sign extension is necessary for signed numbers only, but note that the carry propagation must still be done for unsigned numbers.

Figure 60. 16-bit Multiplication, General Algorithm



UART0 Control and Status Registers – UCSR0A

Bit	7	6	5	4	3	2	1	0	
$0B ($2B)	**RXC0**	**TXC0**	**UDRE0**	**FE0**	**OR0**	**-**	**U2X0**	**MPCM0**	**UCSR0A**
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

UART1 Control and Status Registers – UCSR1A

Bit	7	6	5	4	3	2	1	0	
$02 ($22)	**RXC1**	**TXC1**	**UDRE1**	**FE1**	**OR1**	**-**	**U2X1**	**MPCM1**	**UCSR1A**
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 - RXC0/RXC1: UART Receive Complete**

This bit is set (one) when a received character is transferred from the Receiver Shift register to UDRn. The bit is set regardless of any detected framing errors. When the RXCIEn bit in UCS-RnB is set, the UART Receive Complete interrupt will be executed when RXCn is set (one). RXCn is cleared by reading UDRn. When interrupt-driven data reception is used, the UART Receive Complete Interrupt routine must read UDRn in order to clear RXCn, otherwise a new interrupt will occur once the interrupt routine terminates.

- **Bit 6 - TXC0/TXC1: UART Transmit Complete**

This bit is set (one) when the entire character (including the stop bit) in the Transmit Shift register has been shifted out and no new data has been written to UDRn. This flag is especially useful in half-duplex communications interfaces, where a transmitting application must enter receive mode and free the communications bus immediately after completing the transmission.

When the TXCIEn bit in UCSRnB is set, setting of TXCn causes the UART Transmit Complete interrupt to be executed. TXCn is cleared by the hardware when executing the corresponding interrupt handling vector. Alternatively, the TXCn bit is cleared (zero) by writing a logic 1 to the bit.

- **Bit 5 - UDRE0/UDRE1: UART Data Register Empty**

This bit is set (one) when a character written to UDRn is transferred to the Transmit shift register. Setting of this bit indicates that the transmitter is ready to receive a new character for transmission.

When the UDRIEn bit in UCSRnB is set, the UART Transmit Complete interrupt will be executed as long as UDREn is set and the global interrupt enable bit in SREG is set. UDREn is cleared by writing UDRn. When interrupt-driven data transmittal is used, the UART Data Register Empty Interrupt routine must write UDRn in order to clear UDREn, otherwise a new interrupt will occur once the interrupt routine terminates.

UDREn is set (one) during reset to indicate that the transmitter is ready.

- **Bit 4 - FE0/FE1: Framing Error**

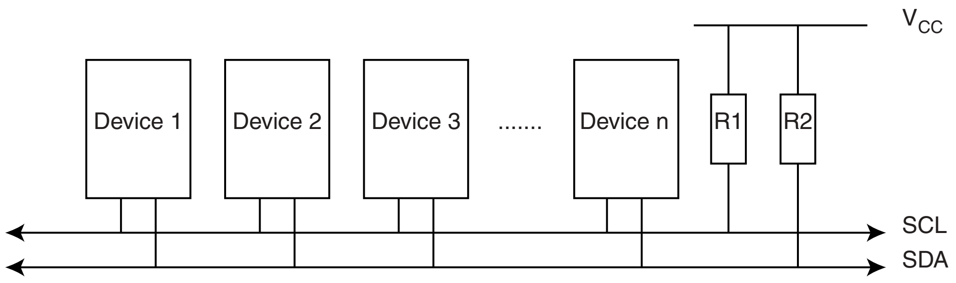
This bit is set if a Framing Error condition is detected, i.e., when the stop bit of an incoming character is zero.

The FEn bit is cleared when the stop bit of received data is one.

2-wire Serial Interface (Byte Oriented)

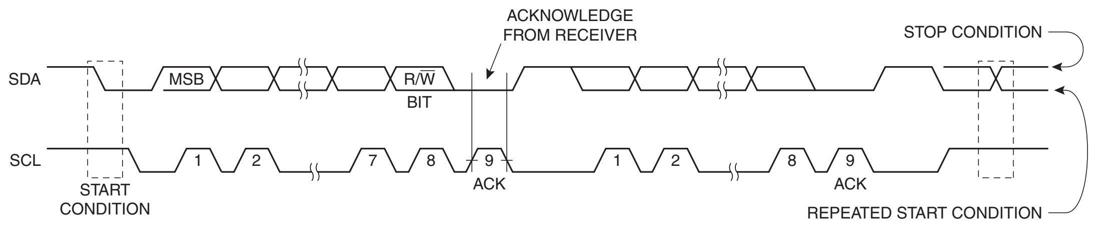
The 2-wire Serial Bus is a bi-directional two-wire serial communication standard. It is designed primarily for simple but efficient integrated circuit (IC) control. The system is comprised of two lines, SCL (Serial Clock) and SDA (Serial Data) that carry information between the ICs connected to them. Various communication configurations can be designed using this bus. Figure 68 shows a typical 2-wire Serial Bus configuration. Any device connected to the bus can be Master or Slave.

Figure 68. 2-wire Serial Bus Configuration



The 2-wire Serial Interface provides a serial interface that meets the 2-wire Serial Bus specification and supports Master/Slave and Transmitter/Receiver operation at up to 400 kHz bus clock rate. The 2-wire Serial Interface has hardware support for the 7-bit addressing, but is easily extended to 10-bit addressing format in software. When operating in 2-wire Serial mode, i.e., when TWEN is set, a glitch filter is enabled for the input signals from the pins SCL and SDA, and the output from these pins are slew-rate controlled. The 2-wire Serial Interface is byte oriented. The operation of the serial 2-wire Serial Bus is shown as a pulse diagram in Figure 69, including the START and STOP conditions and generation of ACK signal by the bus receiver.

Figure 69. 2-wire Serial Bus Timing Diagram



The block diagram of the 2-wire Serial Bus interface is shown in Figure 70.

DC Characteristics – 3.3V Operation – Commercial/Industrial (Preliminary)

T_A = -40°C to 85°C, V_{CC} = 2.7V to 3.6V (unless otherwise noted(1))

Symbol	Parameter	Conditions	Minimum(3)	Typical	Maximum(2)	Units
V_{IH}	High-level Input Voltage	CMOS	0.7 V_{CC}	–	5.5	V
V_{IH1}	Input High-voltage	XTAL	0.7 V_{CC}(3)	–	V_{CC} + 0.5	V
V_{IH2}	Input High-voltage	$\overline{\text{RESET}}$	0.85 V_{CC}(3)	–	V_{CC} + 0.5	V
V_{IL}	Low-level Input Voltage	CMOS	-0.3	–	30% V_{CC}	V
V_{IL1}	Input Low-voltage	XTAL	-0.5	–	0.1(2)	V
V_{OH}	High-level Output Voltage	I_{OH} = 4 mA V_{CC} = V_{CC} Minimum	2.1	–	–	V
		I_{OH} = 12 mA V_{CC} = 3.0V	2.1	–	–	V
		I_{OH} = 16 mA V_{CC} = 3.0V	2.1	–	–	V
V_{OL}	Low-level Output Voltage	I_{OL} = -4 mA V_{CC} = 3.0V	–	–	0.4	V
		I_{OL} = -12 mA V_{CC} = 3.0V	–	–	0.4	V
		I_{OL} = -16 mA V_{CC} = 3.0V	–	–	0.4	V
RRST	Reset Pull-up		100	–	500	kΩ
$R_{I/O}$	I/O Pin Pull-up		35	–	120	kΩ
I_{IH}	High-level Input Current	V_{IN} = V_{CC} Maximum	–	–	10	µA
		With Pull-down, V_{IN} = V_{CC}	75	150	300	µA
I_{IL}	Low-level Input Current	V_{IN} = V_{SS}	-10	–	–	µA
		With Pull-up, V_{IN} = V_{SS}	-300	-150	-75	µA
I_{OZH}	High-level Tri-state Output Leakage Current	Without Pull-down, V_{IN} = V_{CC} Maximum			10	µA
		With Pull-down, V_{IN} = V_{CC} Maximum	75	150	300	µA
I_{OZL}	Low-level Tri-state Output Leakage Current	Without Pull-up, V_{IN} = V_{SS}	-10			µA
		With Pull-up, V_{IN} = V_{SS}	-300	-150	-75	µA
I_{CC}	Standby Current Consumption	Standby, Unprogrammed	–	0.6	0.5	mA
	Power Supply Current	Active, V_{CC} = 3V(1) 25 MHz	–	80(4)	–	mA
		Idle, V_{CC} = 3V(1)	–	–	1.0	mA
		Power-down, V_{CC} = 3V(1) WDT Enable	–	60	500	µA
		Power-down, V_{CC} = 3V(1) WDT Disable	–	30	200	µA
		Power-save, V_{CC} = 3V(1) WDT Disable	–	50	400	µA
	FPGA Core Current Consumption		–	2	–	mA/MHz
C_{IN}	Input Capacitance	All Pins	–	–	10	pF

Notes: 1. Complete FPSLIC device with static FPGA core (no clock in FPGA active).
2. "Maximum" is the highest value where the pin is guaranteed to be read as Low.
3. "Minimum" is the lowest value where the pin is guaranteed to be read as High.
4. 54 mA for AT94K05 devices.

Packaging and Pin List Information

FPSLIC devices should be laid out to support a split power supply for both AL and AX families. Please refer to the “Designing in Split Power Supply Support for AT94KAL and AT94SAL Devices” application note, available on the Atmel web site.

Table 54. Part and Package Combinations Available

Part #	Package	AT94K05	AT94K10	AT94K40
PLCC 84	AJ	46	46	
TQ 100	AQ	58	58	
LQ144	BQ	82	84	84
PQ 208	DQ	96	116	120

Table 55. AT94K JTAG ICE Pin List

Pin	AT94K05 96 FPGA I/O	AT94K10 192 FPGA I/O	AT94K40 384 FPGA I/O
TDI	IO34	IO50	IO98
TDO	IO38	IO54	IO102
TMS	IO43	IO63	IO123
TCK	IO44	IO64	IO124

Table 56. AT94K Pin List

AT94K05 96 FPGA I/O	AT94K10 192 FPGA I/O	AT94K40 384 FPGA I/O	Packages			
			PC84	TQ100	PQ144	PQ208
West Side						
GND	GND	GND	12	1	1	2
I/O1, GCK1 (A16)	I/O1, GCK1 (A16)	I/O1, GCK1 (A16)	13	2	2	4
I/O2 (A17)	I/O2 (A17)	I/O2 (A17)	14	3	3	5
I/O3	I/O3	I/O3			4	6
I/O4	I/O4	I/O4			5	7
I/O5 (A18)	I/O5 (A18)	I/O5 (A18)	15	4	6	8
I/O6 (A19)	I/O6 (A19)	I/O6 (A19)	16	5	7	9
		GND				
		I/O7				
		I/O8				
		I/O9				

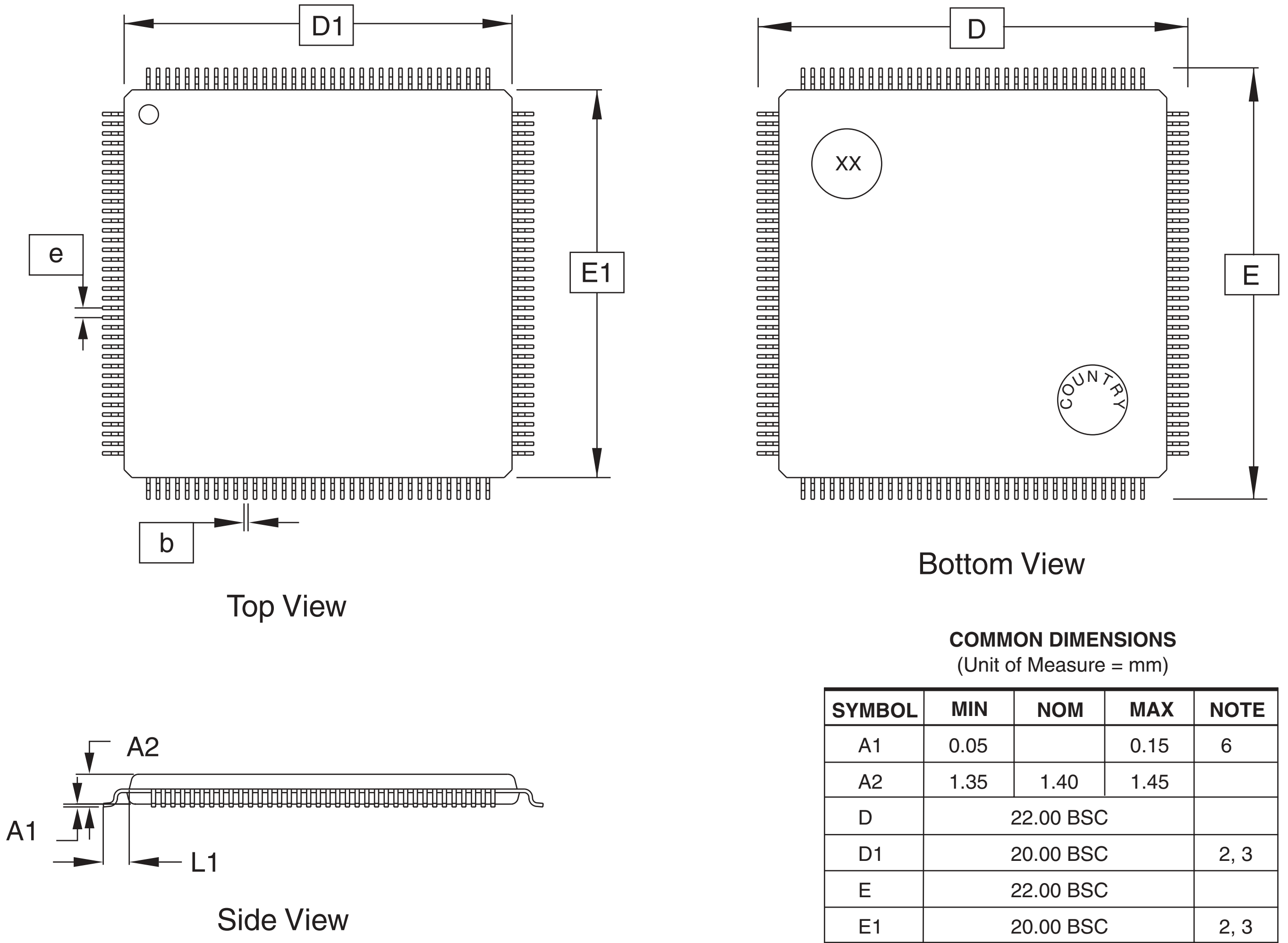
Notes:
1. VCC is I/O high voltage. Please refer to the “Designing in Split Power Supply Support for AT94KAL and AT94SAL Devices” application note.
2. VDD is core high voltage. Please refer to the “Designing in Split Power Supply Support for AT94KAL and AT94SAL Devices” application note.
3. Unbonded pins are No Connects.

Table 56. AT94K Pin List (Continued)

AT94K05 96 FPGA I/O	AT94K10 192 FPGA I/O	AT94K40 384 FPGA I/O	Packages			
			PC84	TQ100	PQ144	PQ208
I/O100	I/O148	I/O292			114	164
		I/O293				
		I/O294				
		GND				
		I/O295				
		I/O296				
I/O101 ($\overline{CS1}$, A2)	I/O149 ($\overline{CS1}$, A2)	I/O297 ($\overline{CS1}$, A2)	79	80	115	165
I/O102 (A3)	I/O150 (A3)	I/O298 (A3)	80	81	116	166
		I/O299				
		I/O300				
		VCC(1)				
		GND				
I/O104	I/O151	I/O301	Shorted to Testclock	Shorted to Testclock	Shorted to Testclock	Shorted to Testclock
	I/O152	I/O302				
I/O103	I/O153	I/O303			117	167
	I/O154	I/O304				168
		I/O305				
		I/O306				
		GND				
		I/O307				
		I/O308				
	I/O155	I/O309				169
	I/O156	I/O310				170
		I/O311				
		I/O312				
GND	GND	GND			118	171
I/O105	I/O157	I/O313			119	172
I/O106	I/O158	I/O314			120	173
	I/O159	I/O315				
	I/O160	I/O316				
	VCC(1)	VCC(1)				
		I/O317				
		I/O318				

Notes:
1. VCC is I/O high voltage. Please refer to the “Designing in Split Power Supply Support for AT94KAL and AT94SAL Devices” application note.
2. VDD is core high voltage. Please refer to the “Designing in Split Power Supply Support for AT94KAL and AT94SAL Devices” application note.
3. Unbonded pins are No Connects.

144L1 – LQFP



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A1	0.05		0.15	6
A2	1.35	1.40	1.45	
D	22.00 BSC			
D1	20.00 BSC			2, 3
E	22.00 BSC			
E1	20.00 BSC			2, 3
e	0.50 BSC			
b	0.17	0.22	0.27	4, 5
L1	1.00 REF			

Notes:
1. This drawing is for general information only; refer to JEDEC Drawing MS-026 for additional information.
2. The top package body size may be smaller than the bottom package size by as much as 0.15 mm.
3. Dimensions D1 and E1 do not include mold protrusions. Allowable protrusion is 0.25 mm per side. D1 and E1 are maximum plastic body size dimensions including mold mismatch.
4. Dimension b does not include Dambar protrusion. Allowable Dambar protrusion shall not cause the lead width to exceed the maximum b dimension by more than 0.08 mm. Dambar cannot be located on the lower radius or the foot. Minimum space between protrusion and an adjacent lead is 0.07 mm for 0.4 and 0.5 mm pitch packages.
5. These dimensions apply to the flat section of the lead between 0.10 mm and 0.25 mm from the lead tip.
6. A1 is defined as the distance from the seating place to the lowest point on the package body.

11/30/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**144L1**, 144-lead (20 x 20 x 1.4 mm Body), Low Profile Plastic Quad Flat Pack (LQFP)	144L1	A