Welcome to E-XFL.COM

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

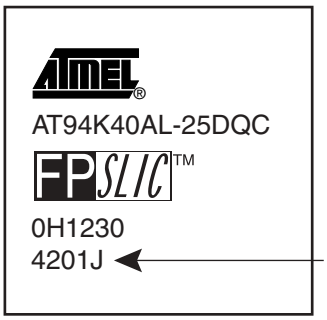
Embedded - FPGAs (Field Programmable Gate Arrays) with Microcontrollers represent a cutting-edge category of electronic components that combine the flexibility of FPGA technology with the processing power of integrated microcontrollers. This hybrid approach offers a versatile solution for designing and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

What Are Embedded - FPGAs with Microcontrollers?

At their core, **FPGAs** are semiconductor devices that can

Details	
Product Status	Active
Core Type	8-Bit AVR
Speed	18 MHz
Interface	I²C, UART
Program SRAM Bytes	20K-32K
FPGA SRAM	18kb
EEPROM Size	-
Data SRAM Bytes	4K ~ 16K
FPGA Core Cells	-
FPGA Gates	40K
FPGA Registers	-
Voltage - Supply	3V ~ 3.6V
Mounting Type	Surface Mount
Operating Temperature	-40°C ~ 85°C
Package / Case	208-BFQFP
Supplier Device Package	208-PQFP (28x28)
Purchase URL	https://www.e-xfl.com/product-detail/atmel/at94k40al-25dqu

Figure 1. FPSLIC Device Date Code with JTAG ICE Support



The AT94K series architecture is shown in Figure 2.

Figure 2. AT94K Series Architecture

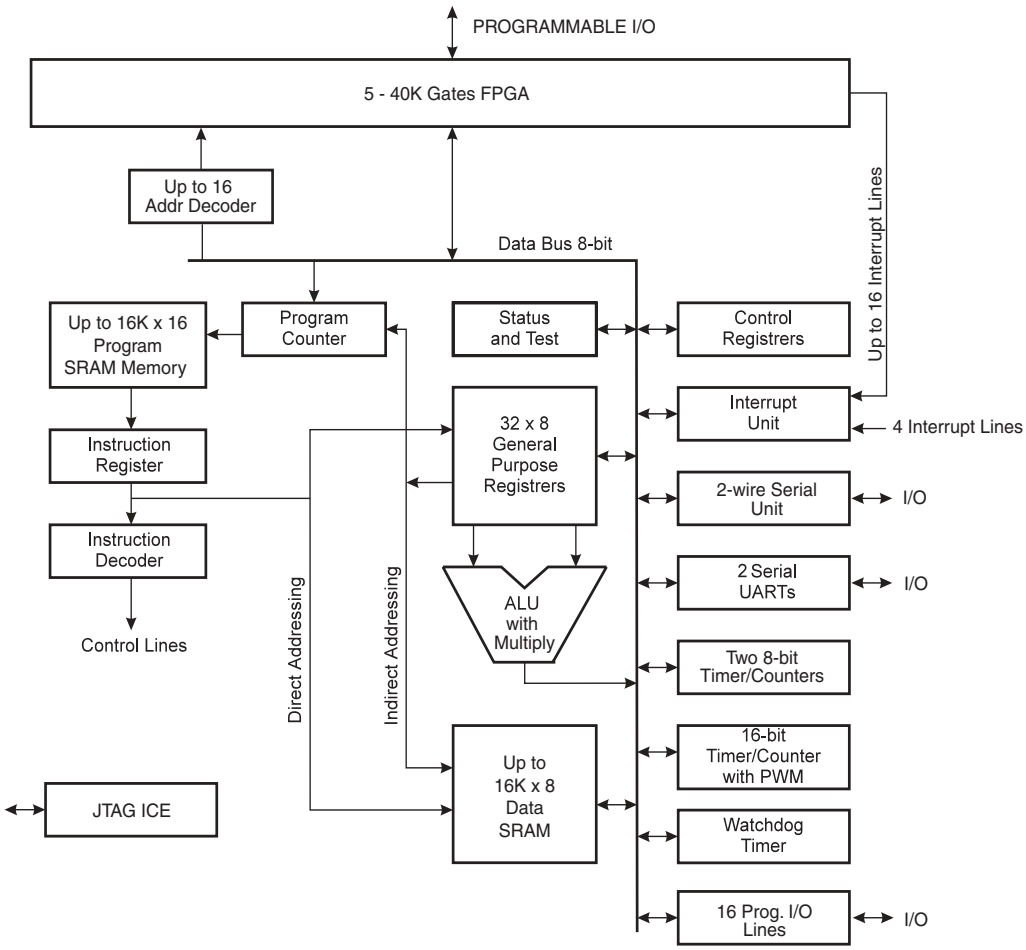
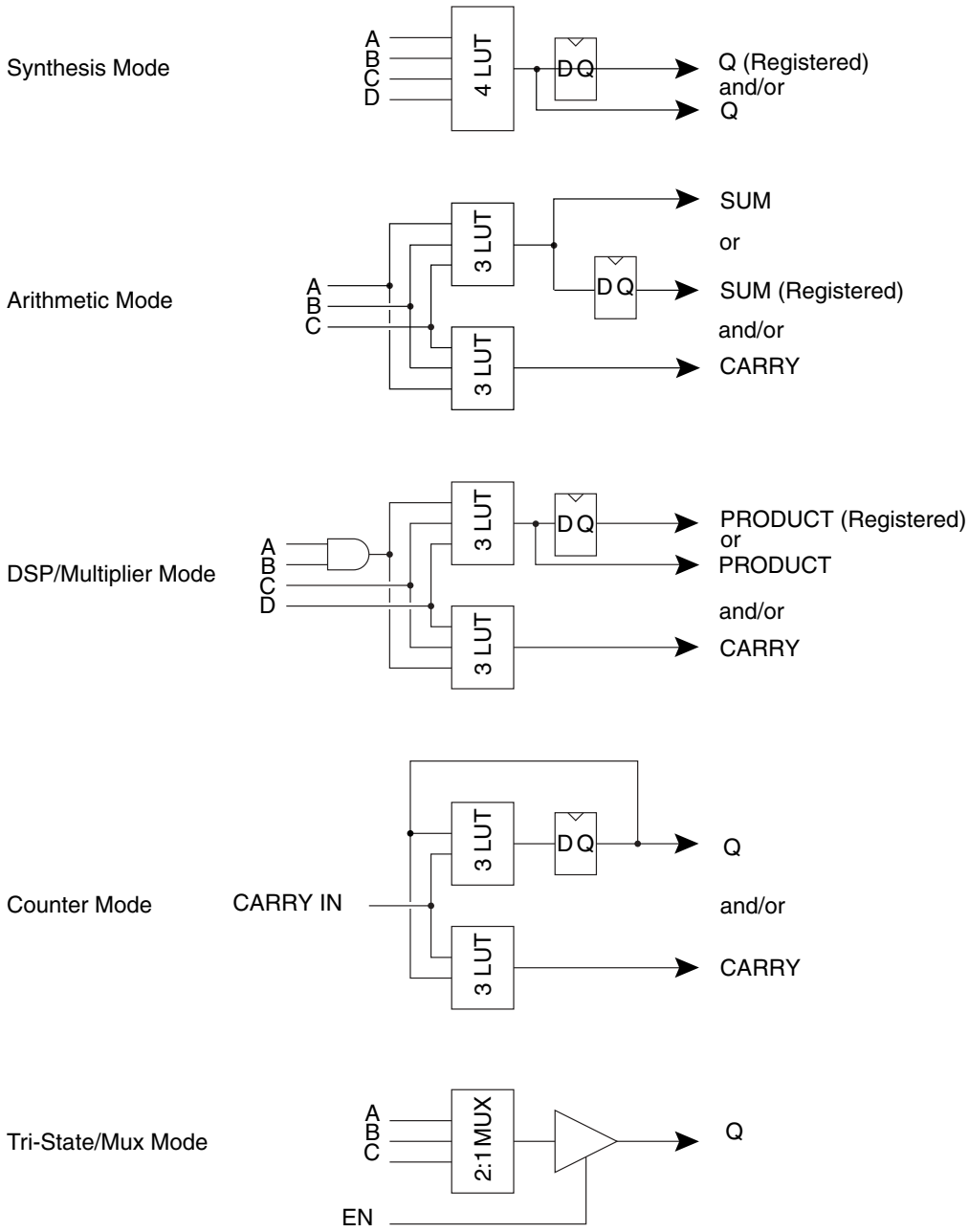


Figure 7. Some Single Cell Modes



RAM

There are two types of RAM in the FPSLIC device: the FreeRAM distributed through the FPGA Core and the SRAM shared by the AVR and FPGA. The SRAM is described in "FPGA/AVR Interface and System Control" on page 21. The 32 x 4 dual-ported FPGA FreeRAM blocks are dispersed throughout the array and are connected in each sector as shown in Figure 8. A four-bit Input Data bus connects to four horizontal local buses (Plane 1) distributed over four sector rows. A four-bit Output Data bus connects to four horizontal local buses (Plane 2) distributed over four sector rows. A five-bit Input-address bus connects to five vertical express buses in the same sector column (column 3). A five-bit Output-address bus connects to five vertical express buses in the same column. WAddr (Write Address) and RAddr (Read Address) alternate positions in horizontally aligned RAM blocks. For the left-

Instruction Set Nomenclature (Summary)

The complete “AVR Instruction Set” document is available on the Atmel web site, at http://www.atmel.com/atmel/acrobat/doc0856.pdf.

Status Register (SREG)

SREG:	Status register
C:	Carry flag in status register
Z:	Zero flag in status register
N:	Negative flag in status register
V:	Two’s complement overflow indicator
S:	N ⊕ V, For signed tests
H:	Half-carry flag in the status register
T:	Transfer bit used by BLD and BST instructions
I:	Global interrupt enable/disable flag

Registers and Operands

Rd:	Destination (and source) register in the register file
Rr:	Source register in the register file
R:	Result after instruction is executed
K:	Constant data
k:	Constant address
b:	Bit in the register file or I/O register ($0 \le b \le 7$)
s:	Bit in the status register ($0 \le s \le 2$)
X,Y,Z:	Indirect address register (X = R27:R26, Y = R29:R28 and Z = R31:R30)
A:	I/O location address
q:	Displacement for direct addressing ($0 \le q \le 63$)

I/O Registers

Stack

STACK:	Stack for return address and pushed registers
SP:	Stack Pointer to STACK

Flags

⇔:	Flag affected by instruction
0:	Flag cleared by instruction
1:	Flag set by instruction
-:	Flag not affected by instruction

The instructions EIJMP, EICALL, ELPM, GPM, ESPM (from the megaAVR Instruction Set) are not supported in the FPSLIC device.

XTAL2 Output from the inverting oscillator amplifier

TOSC1 Input to the inverting timer/counter oscillator amplifier

TOSC2 Output from the inverting timer/counter oscillator amplifier

SCL 2-wire serial input/output clock

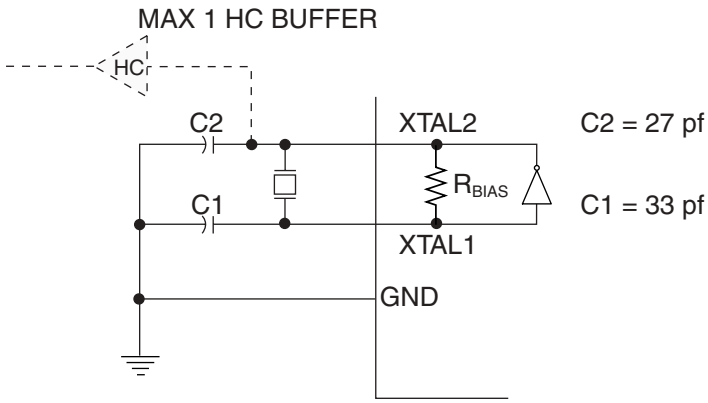
SDA 2-wire serial input/output data

Clock Options

Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier, which can be configured for use as an on-chip oscillator, as shown in Figure 24. Either a quartz crystal or a ceramic resonator may be used.

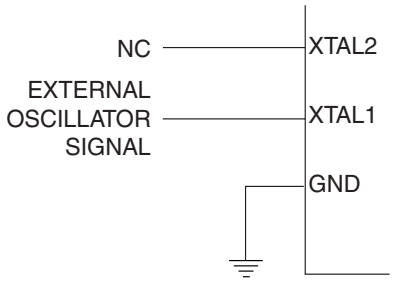
Figure 24. Oscillator Connections



External Clock

To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 25.

Figure 25. External Clock Drive Configuration



General AVR/FPGA I/O Select Procedure

I/O select depends on the FISCR register setup and the FISUA..D register written to or read from.

The following FISCR setups and writing data to the FISUA..D registers will result in the shown I/O select lines and data presented on the 8-bit AVR–FPGA data bus.

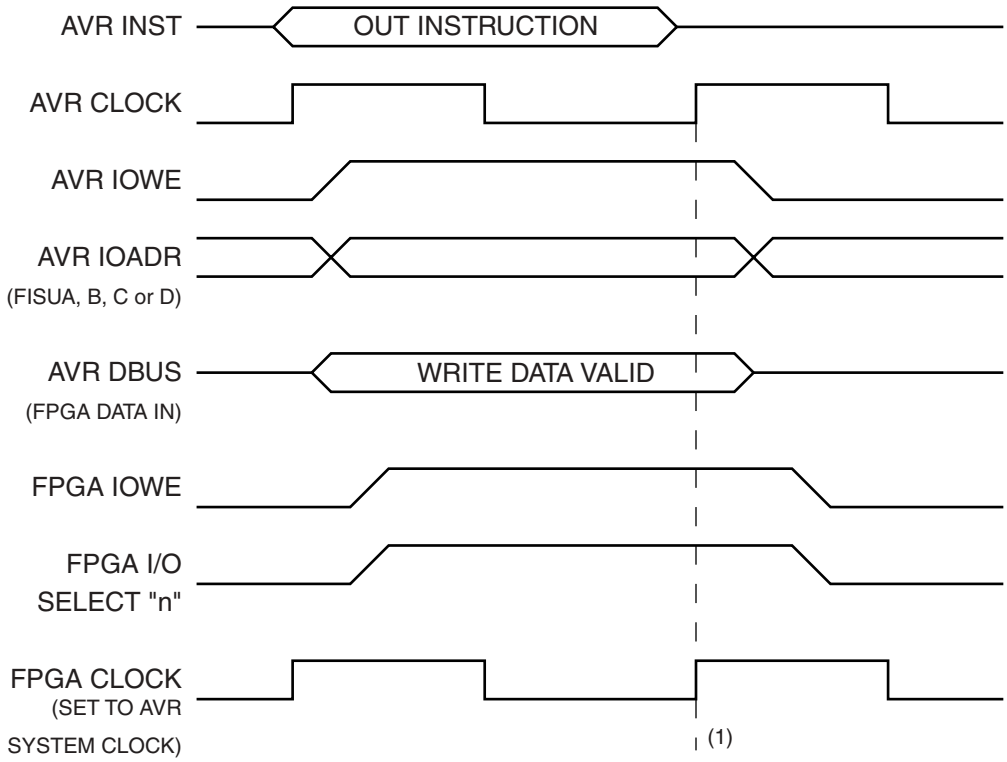
Table 14. FISCR Register Setups and I/O Select Lines.

FISCR Register				I/O Select Lines[1]			
FIADR(b7)	b6-2	XFIS1(b1)	XFIS0(b0)	FISUA	FISUB	FISUC	FISUD
0	-	0	0	IOSEL 0	IOSEL 4	IOSEL 8	IOSEL 12
0	-	0	1	IOSEL 1	IOSEL 5	IOSEL 9	IOSEL 13
0	-	1	0	IOSEL 2	IOSEL 6	IOSEL 10	IOSEL 14
0	-	1	1	IOSEL 3	IOSEL 7	IOSEL 11	IOSEL 15

Note: 1. IOSEL 15..8 are not available on AT94K05.

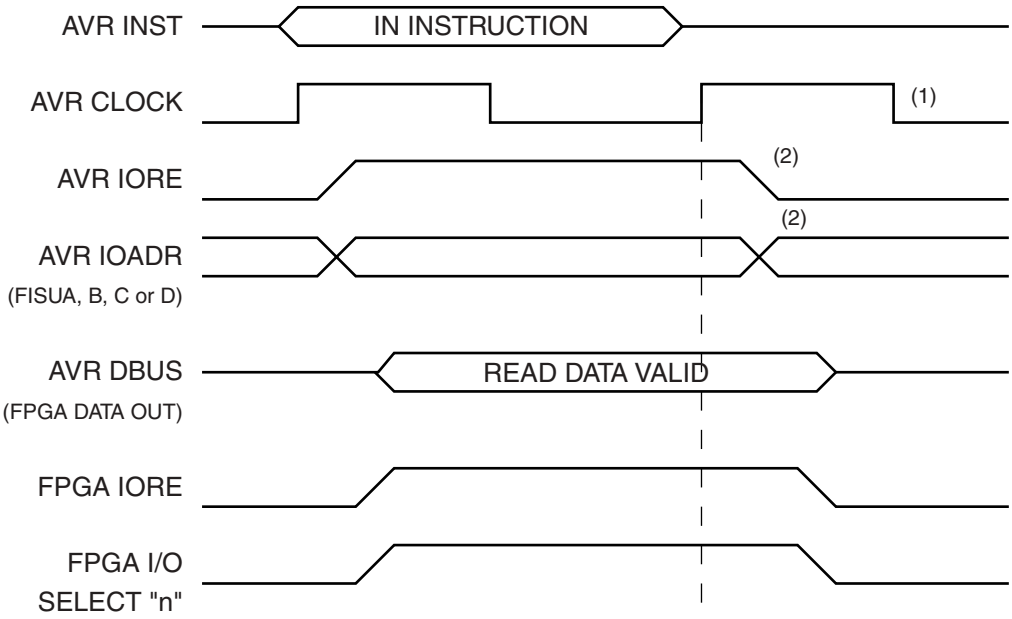
```
;-------------------------------------------
io_select0_write:
  ldi r16,0x00         ;FIADR=0,XFIS1=0,XFIS0=0 ->I/O select line=0
  out FISCR,r16        ;load I/O select values into FISCR register
  out FISUA,r17;       ;select line 0 high. Place data on AVR<->FPGA bus
                       ; from r17 register. (out going data is assumed
                       ; to be present in r17 before calling this subroutine)
  ret

;-------------------------------------------

io_select13_read:
  ldi r16,0x01         ;FIADR=0,XFIS1=0,XFIS0=1 ->I/O select line=13
  out FISCR,r16        ;load I/O select values into FISCR register
  in r18,FISUD         ;select line 13 high. Read data on AVR<->FPGA bus
                       ;which was placed into register FISUD.
  ret
```

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.

Table 15. Reset and Interrupt Vectors (Continued)

Vector No. (hex)	Program Address	Source	Interrupt Definition
17	$002C	UART0_DRE	UART0 Data Register Empty Interrupt Handle
18	$002E	UART0_TXC	UART0 Transmit Complete Interrupt Handle
19	$0030	FPGA_INT8	FPGA Interrupt8 Handle (not available on AT94K05)
1A	$0032	FPGA_INT9	FPGA Interrupt9 Handle (not available on AT94K05)
1B	$0034	FPGA_INT10	FPGA Interrupt10 Handle (not available on AT94K05)
1C	$0036	FPGA_INT11	FPGA Interrupt11 Handle (not available on AT94K05)
1D	$0038	UART1_RXC	UART1 Receive Complete Interrupt Handle
1E	$003A	UART1_DRE	UART1 Data Register Empty Interrupt Handle
1F	$003C	UART1_TXC	UART1 Transmit Complete Interrupt Handle
20	$003E	FPGA_INT12	FPGA Interrupt12 Handle (not available on AT94K05)
21	$0040	FPGA_INT13	FPGA Interrupt13 Handle (not available on AT94K05)
22	$0042	FPGA_INT14	FPGA Interrupt14 Handle (Not Available on AT94K05)
23	$0044	FPGA_INT15	FPGA Interrupt15 Handle (not available on AT94K05)
24	$0046	TWS_INT	2-wire Serial Interrupt

The most typical program setup for the Reset and Interrupt Vector Addresses are:

```
Address     Labels     Code              Comments
$0000       jmp        RESET             Reset Handle: Program Execution Starts Here
$0002       jmp        FPGA_INT0         ; FPGA Interrupt0 Handle
$0004       jmp        EXT_INT0          ; External Interrupt0 Handle
$0006       jmp        FPGA_INT1         ; FPGA Interrupt1 Handle
$0008       jmp        EXT_INT1          ; External Interrupt1 Handle
$000A       jmp        FPGA_INT2         ; FPGA Interrupt2 Handle
$000C       jmp        EXT_INT2          ; External Interrupt2 Handle
$000E       jmp        FPGA_INT3         ; FPGA Interrupt3 Handle
$0010       jmp        EXT_INT3          ; External Interrupt3 Handle
$0012       jmp        TIM2_COMP         ; Timer/Counter2 Compare Match Interrupt Handle
$0014       jmp        TIM2_OVF          ; Timer/Counter2 Overflow Interrupt Handle
$0016       jmp        TIM1_CAPT         ; Timer/Counter1 Capture Event Interrupt Handle
$0018       jmp        TIM1_COMPA        ; Timer/Counter1 Compare Match A Interrupt Handle
$001A       jmp        TIM1_COMPB        ; Timer/Counter1 Compare Match B Interrupt Handle
$001C       jmp        TIM1_OVF          ; Timer/Counter1 Overflow Interrupt Handle
$001E       jmp        TIM0_COMP         ; Timer/Counter0 Compare Match Interrupt Handle
$0020       jmp        TIM0_OVF          ; Timer/Counter0 Overflow Interrupt Handle
$0022       jmp        FPGA_INT4         ; FPGA Interrupt4 Handle
$0024       jmp        FPGA_INT5         ; FPGA Interrupt5 Handle
$0026       jmp        FPGA_INT6         ; FPGA Interrupt6 Handle
$0028       jmp        FPGA_INT7         ; FPGA Interrupt7 Handle
$002A       jmp        UART0_RXC         ; UART0 Receive Complete Interrupt Handle
$002C       jmp        UART0_DRE         ; UART0 Data Register Empty Interrupt Handle
$002E       jmp        UART0_TXC         ; UART0 Transmit Complete Interrupt Handle
$0030       jmp        FPGA_INT8         ; FPGA Interrupt8 Handle(1)
$0032       jmp        FPGA_INT9         ; FPGA Interrupt9 Handle(1)
$0034       jmp        FPGA_INT10        ; FPGA Interrupt10 Handle(1)
$0036       jmp        FPGA_INT11        ; FPGA Interrupt11 Handle(1)
$0038       jmp        UART1_RXC         ; UART1 Receive Complete Interrupt Handle
$003A       jmp        UART1_DRE         ; UART1 Data Register Empty Interrupt Handle
$003C       jmp        UART1_TXC         ; UART1 Transmit Complete Interrupt Handle
$003E       jmp        FPGA_INT12        ; FPGA Interrupt12 Handle(1)
$0040       jmp        FPGA_INT13        ; FPGA Interrupt13 Handle(1)
$0042       jmp        FPGA_INT14        ; FPGA Interrupt14 Handle(1)
$0044       jmp        FPGA_INT15        ; FPGA Interrupt15 Handle(1)
$0046       jmp        TWS_INT           ; 2-wire Serial Interrupt
;
RESET:
$0048       ldi        r16,high(RAMEND)  ; Main program start
$0049       out        SPH,r16

$004A       ldi        r16,low(RAMEND)
$004B       out        SPL,r16
$004C       <instr>    xxx
  ...         ...        ...
```

Note: 1. Not Available on AT94K05. However, the vector jump table positions must be maintained for appropriate UART and 2-wire serial interrupt jumps.

- **Bit 3 - ICF1: Input Capture Flag 1**

The ICF1 bit is set (one) to flag an input capture event, indicating that the Timer/Counter1 value has been transferred to the input capture register – ICR1. ICF1 is cleared by the hardware when executing the corresponding interrupt handling vector. Alternatively, ICF1 is cleared by writing a logic 1 to the flag. When the SREG I-bit, and TICIE1 (Timer/Counter1 Input Capture Interrupt Enable), and ICF1 are set (one), the Timer/Counter1 Capture Interrupt is executed.

- **Bit 2 - OCF2: Output Compare Flag 2**

The OCF2 bit is set (one) when compare match occurs between Timer/Counter2 and the data in OCR2 – Output Compare Register 2. OCF2 is cleared by the hardware when executing the corresponding interrupt handling vector. Alternatively, OCF2 is cleared by writing a logic 1 to the flag. When the I-bit in SREG, and OCIE2 (Timer/Counter2 Compare Interrupt Enable), and the OCF2 are set (one), the Timer/Counter2 Output Compare Interrupt is executed.

- **Bit 1 - TOV0: Timer/Counter0 Overflow Flag**

The TOV0 bit is set (one) when an overflow occurs in Timer/Counter0. TOV0 is cleared by the hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic 1 to the flag. When the SREG I-bit, and TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set (one), the Timer/Counter0 Overflow interrupt is executed. In PWM mode, this bit is set when Timer/Counter0 advances from $00.

- **Bit 0 - OCF0: Output Compare Flag 0**

The OCF0 bit is set (one) when compare match occurs between Timer/Counter0 and the data in OCR0 – Output Compare Register 0. OCF0 is cleared by the hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0 is cleared by writing a logic 1 to the flag. When the I-bit in SREG, and OCIE0 (Timer/Counter2 Compare Interrupt Enable), and the OCF0 are set (one), the Timer/Counter0 Output Compare Interrupt is executed.

Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. Four clock cycles after the interrupt flag has been set, the program vector address for the actual interrupt handling routine is executed. During this four clock-cycle period, the Program Counter (2 bytes) is pushed onto the Stack, and the Stack Pointer is decremented by 2. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is serviced.

A return from an interrupt handling routine (same as for a subroutine call routine) takes four clock cycles. During these four clock cycles, the Program Counter (2 bytes) is popped back from the Stack, and the Stack Pointer is incremented by 2. When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is serviced.

Sleep Modes

To enter any of the three Sleep modes, the SE bit in MCUR must be set (one) and a SLEEP instruction must be executed. The SM1 and SM0 bits in the MCUR register select which Sleep mode (Idle, Power-down, or Power-save) will be activated by the SLEEP instruction, see Table 12 on page 52.

In Power-down and Power-save modes, the four external interrupts, EXT_INT0...3, and FPGA interrupts, FPGA INT0...3, are triggered as low level-triggered interrupts. If an enabled interrupt occurs while the MCU is in a Sleep mode, the MCU awakes, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the register file, SRAM, and I/O memory are unaltered. If a reset occurs during Sleep mode, the MCU wakes up and executes from the Reset vector

On-chip Debug Specific JTAG Instructions

The On-Chip debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected third-party vendors only. Table 17 lists the instruction opcode.

Table 17. JTAG Instruction and Code

JTAG Instruction	4-bit Code	Selected Scan Chain	# Bits
EXTEST	$0 (0000)	AVR I/O Boundary	69
IDCODE	$1 (0001)	Device ID	32
SAMPLE_PRELOAD	$2 (0010)	AVR I/O Boundary	69
RESERVED	$3 (0011)	N/A	–
PRIVATE	$4 (0100)	FPSLIC On-chip Debug System	–
PRIVATE	$5 (0101)	FPSLIC On-chip Debug System	–
PRIVATE	$6 (0110)	FPSLIC On-chip Debug System	–
RESERVED	$7 (0111)	N/A	–
PRIVATE	$8 (1000)	FPSLIC On-chip Debug System	–
PRIVATE	$9 (1001)	FPSLIC On-chip Debug System	–
PRIVATE	$A (1010)	FPSLIC On-chip Debug System	–
PRIVATE	$B (1011)	FPSLIC On-chip Debug System	–
AVR_RESET	$C (1100)	AVR Reset	1
RESERVED	$D (1101)	N/A	–
RESERVED	$E (1110)	N/A	–
BYPASS	$F (1111)	Bypass	1

IEEE 1149.1 (JTAG) Boundary-scan

Features

- **JTAG (IEEE std. 1149.1 compliant) Interface**
- **Boundary-scan Capabilities According to the JTAG Standard**
- **Full Scan of All Port Functions**
- **Supports the Optional IDCODE Instruction**
- **Additional Public AVR_RESET Instruction to Reset the AVR**

System Overview

The Boundary-Scan chain has the capability of driving and observing the logic levels on the AVR's digital I/O pins. At system level, all ICs having JTAG capabilities are connected serially by the TDI/TDO signals to form a long shift register. An external controller sets up the devices to drive values at their output pins, and observe the input values received from other devices. The controller compares the received data with the expected result. In this way, Boundary-Scan provides a mechanism for testing interconnections and integrity of components on Printed Circuits Boards by using the 4 TAP signals only.

The four IEEE 1149.1 defined mandatory JTAG instructions IDCODE, BYPASS, SAMPLE/PRELOAD, and EXTEST, as well as the AVR specific public JTAG instruction AVR_RESET can be used for testing the Printed Circuit Board. Initial scanning of the data register path will show the ID-code of the device, since IDCODE is the default JTAG instruction. It may be desirable to have the AVR device in reset during test mode. If not reset, inputs to the device may be determined by the scan operations, and the internal software may be in an

EXTEST; $0

Mandatory JTAG instruction for selecting the Boundary-Scan Chain as Data Register for testing circuitry external to the AVR package. For port-pins, Pull-up Disable, Output Control, Output Data, and Input Data are all accessible in the scan chain. For Analog circuits having off-chip connections, the interface between the analog and the digital logic is in the scan chain. The contents of the latched outputs of the Boundary-Scan chain are driven out as soon as the JTAG IR-register is loaded by the EXTEST instruction.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-Scan Chain.
- Shift-DR: The Internal Scan Chain is shifted by the TCK input.
- Update-DR: Data from the scan chain is applied to output pins.

IDCODE; $1

Optional JTAG instruction selecting the 32-bit ID register as Data Register. The ID register consists of a version number, a device number and the manufacturer code chosen by JEDEC. This is the default instruction after power-up.

The active states are:

- Capture-DR: Data in the IDCODE register is sampled into the Boundary-Scan Chain.
- Shift-DR: The IDCODE scan chain is shifted by the TCK input.

SAMPLE_PRELOAD; $2

Mandatory JTAG instruction for pre-loading the output latches and taking a snap-shot of the input/output pins without affecting the system operation. However, the output latches are not connected to the pins. The Boundary-Scan Chain is selected as Data Register.

The active states are:

- Capture-DR: Data on the external pins are sampled into the Boundary-Scan Chain.
- Shift-DR: The Boundary-Scan Chain is shifted by the TCK input.
- Update-DR: Data from the Boundary-Scan chain is applied to the output latches. However, the output latches are not connected to the pins.

AVR_RESET; $C

The AVR specific public JTAG instruction for forcing the AVR device into the Reset Mode or releasing the JTAG reset source. The TAP controller is not reset by this instruction. The one bit Reset Register is selected as Data Register. Note that the reset will be active as long as there is a logic “1” in the Reset Chain. The output from this chain is not latched.

The active state is:

- Shift-DR: The Reset Register is shifted by the TCK input.

BYPASS; $F

Mandatory JTAG instruction selecting the Bypass Register for Data Register.

The active states are:

- Capture-DR: Loads a logic “0” into the Bypass Register.
- Shift-DR: The Bypass Register cell between TDI and TDO is shifted.

Boundary-scan Chain

The Boundary-Scan chain has the capability of driving and observing the logic levels on the AVR’s digital I/O pins.

Scanning the Digital Port Pins

Figure 43 shows the boundary-scan cell for bi-directional port pins with pull-up function. The cell consists of a standard boundary-scan cell for the pull-up function, and a bi-directional pin cell that combines the three signals Output Control (OC), Output Data (OD), and Input Data (ID), into only a two-stage shift register.

Figure 50. Timer/Counter0 Block Diagram

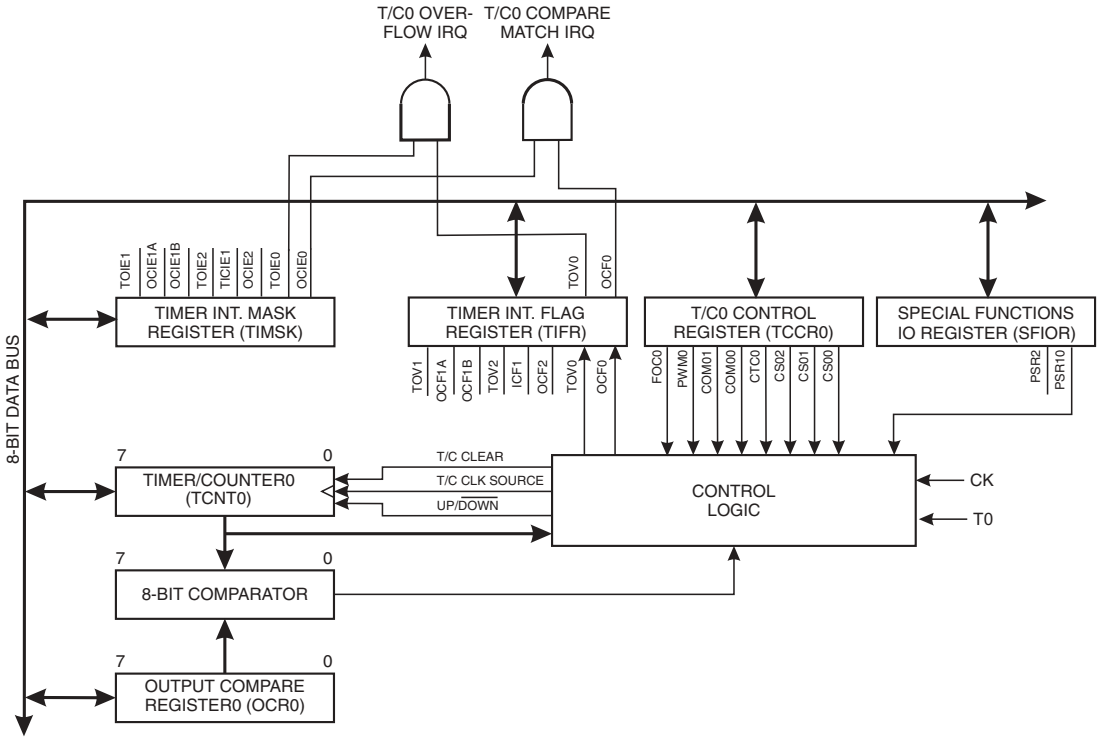
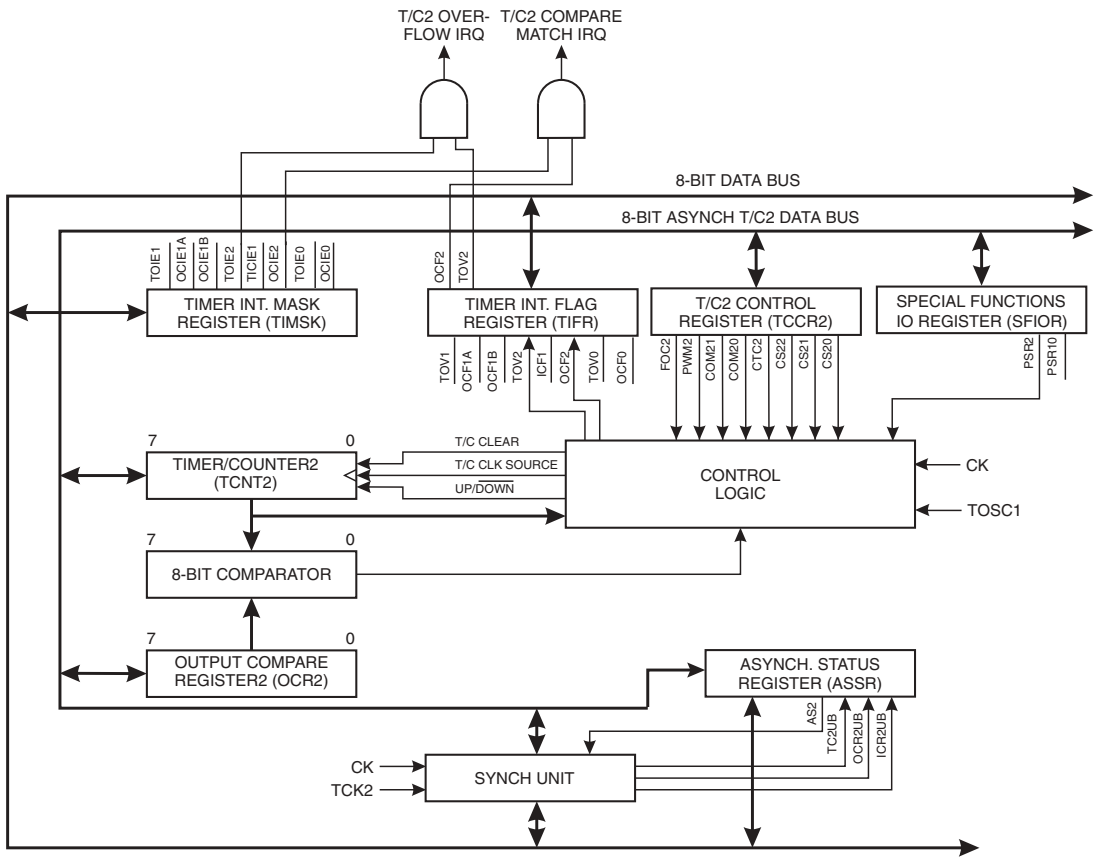


Figure 51. Timer/Counter2 Block Diagram



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.

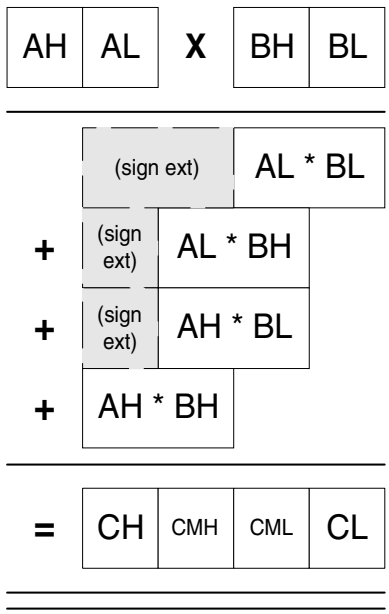
16-bit Multiplication

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



```
   mulsu r23, r20            ; (signed)ah * bl
   sbc   r19, r2
   add   r17, r0
   adc   r18, r1
   adc   r19, r2

   mulsu r21, r22            ; (signed)bh * al
   sbc   r19, r2             ; Sign extend
   add   r17, r0
   adc   r18, r1
   adc   r19, r2

   ret

mac16x16_32_method_B:        ; uses two temporary registers (r4,r5), Speed / Size
                             Optimized
                             ; but reduces cycles/words by 1
   clr   r2

   muls  r23, r21            ; (signed)ah * (signed)bh
   movw  r5:r4,r1:r0

   mul   r22, r20            ; al * bl

   add   r16, r0
   adc   r17, r1
   adc   r18, r4
   adc   r19, r5

mulsu r23, r20               ; (signed)ah * bl
   sbc   r19, r2             ; Sign extend
   add   r17, r0
   adc   r18, r1
   adc   r19, r2

   mulsu r21, r22            ; (signed)bh * al
   sbc   r19, r2             ; Sign extend
   add   r17, r0
   adc   r18, r1
   adc   r19, r2

   ret
```

fmuls16x16_32

Description

Signed fractional multiply of two 16-bit numbers with a 32-bit result.

Usage

R19:R18:R17:R16 = (R23:R22 • R21:R20) << 1

Statistics

Cycles: 20 + ret

Words: 16 + ret

Register usage: R0 to R2 and R16 to R23 (11 registers)[1]

Note: 1. The routine is non-destructive to the operands.

```
fmuls16x16_32:
   clr   r2
   fmuls r23, r21             ; ( (signed)ah * (signed)bh ) << 1
   movw  r19:r18, r1:r0
   fmul  r22, r20             ; ( al * bl ) << 1
   adc   r18, r2
   movw  r17:r16, r1:r0
   fmulsu   r23, r20          ; ( (signed)ah * bl ) << 1
   sbc   r19, r2              ; Sign extend
   add   r17, r0
   adc   r18, r1
   adc   r19, r2
   fmulsu   r21, r22          ; ( (signed)bh * al ) << 1
   sbc   r19, r2              ; Sign extend
   add   r17, r0
   adc   r18, r1
   adc   r19, r2
   ret
```

fmac16x16_32

Description

Signed fractional multiply-accumulate of two 16-bit numbers with a 32-bit result.

Usage

R19:R18:R17:R16 += (R23:R22 • R21:R20) << 1

Statistics

Cycles: 25 + ret

Words: 21 + ret

Register usage: R0 to R2 and R16 to R23 (11 registers)

```
fmac16x16_32:                 ; Register usage optimized
   clr   r2

   fmuls r23, r21             ; ( (signed)ah * (signed)bh ) << 1
   add   r18, r0
   adc   r19, r1

   fmul  r22, r20             ; ( al * bl ) << 1
   adc   r18, r2
   adc   r19, r2
   add   r16, r0
```

Table 36. UBR Settings at Various Crystal Frequencies

Clock MHz	UBRRHI 7:4 or 3:0	UBRRn	UBR HEX	UBR	Actual Freq	Desired Freq.	% Error
1	**0000**	**00011001**	**019**	**25**	**2404**	**2400**	**0.2**
	0000	**00001100**	**00C**	**12**	**4808**	**4800**	**0.2**
	0000	00000110	006	6	8929	9600	7.5
	0000	00000011	003	3	15625	14400	7.8
	0000	00000010	002	2	20833	19200	7.8
	0000	00000001	001	1	31250	28880	7.6
	0000	00000001	001	1	31250	38400	22.9
	0000	00000000	000	0	62500	57600	7.8
	0000	00000000	000	0	62500	76800	22.9
	0000	00000000	000	0	62500	115200	84.3

Clock MHz	UBRRHI 7:4 or 3:0	UBRRn	UBR HEX	UBR	Actual Freq	Desired Freq.	% Error
1.8432	**0000**	**00101111**	**02F**	**47**	**2400**	**2400**	**0.0**
	0000	**00010111**	**017**	**23**	**4800**	**4800**	**0.0**
	0000	**00001011**	**00B**	**11**	**9600**	**9600**	**0.0**
	0000	**00000111**	**007**	**7**	**14400**	**14400**	**0.0**
	0000	**00000101**	**005**	**5**	**19200**	**19200**	**0.0**
	0000	**00000011**	**003**	**3**	**28800**	**28880**	**0.3**
	0000	**00000010**	**002**	**2**	**38400**	**38400**	**0.0**
	0000	**00000001**	**001**	**1**	**57600**	**57600**	**0.0**
	0000	00000001	001	1	57600	76800	33.3
	0000	**00000000**	**000**	**0**	**115200**	**115200**	**0.0**

Clock MHz	UBRRHI 7:4 or 3:0	UBRRn	UBR HEX	UBR	Actual Freq	Desired Freq.	% Error
9.216	**0000**	**11101111**	**0EF**	**239**	**2400**	**2400**	**0.0**
	0000	**01110111**	**077**	**119**	**4800**	**4800**	**0.0**
	0000	**00111011**	**03B**	**59**	**9600**	**9600**	**0.0**
	0000	**00100111**	**027**	**39**	**14400**	**14400**	**0.0**
	0000	**00011101**	**01D**	**29**	**19200**	**19200**	**0.0**
	0000	**00010011**	**013**	**19**	**28800**	**28880**	**0.3**
	0000	**00001110**	**00E**	**14**	**38400**	**38400**	**0.0**
	0000	**00001001**	**009**	**9**	**57600**	**57600**	**0.0**
	0000	00000111	007	7	72000	76800	6.7
	0000	**00000100**	**004**	**4**	**115200**	**115200**	**0.0**
	0000	00000001	001	1	288000	230400	20.0
	0000	00000000	000	0	576000	460800	20.0
	0000	00000000	000	0	576000	912600	58.4

Clock MHz	UBRRHI 7:4 or 3:0	UBRRn	UBR HEX	UBR	Actual Freq	Desired Freq.	% Error
18.432	**0001**	**11011111**	**1DF**	**479**	**2400**	**2400**	**0.0**
	0000	**11101111**	**0EF**	**239**	**4800**	**4800**	**0.0**
	0000	**01110111**	**077**	**119**	**9600**	**9600**	**0.0**
	0000	**01001111**	**04F**	**79**	**14400**	**14400**	**0.0**
	0000	**00111011**	**03B**	**59**	**19200**	**19200**	**0.0**
	0000	**00100111**	**027**	**39**	**28800**	**28880**	**0.3**
	0000	**00011101**	**01D**	**29**	**38400**	**38400**	**0.0**
	0000	**00010011**	**013**	**19**	**57600**	**57600**	**0.0**
	0000	**00001110**	**00E**	**14**	**76800**	**76800**	**0.0**
	0000	**00001001**	**009**	**9**	**115200**	**115200**	**0.0**
	0000	**00000100**	**004**	**4**	**230400**	**230400**	**0.0**
	0000	00000001	001	1	576000	460800	20.0
	0000	00000000	000	0	1152000	912600	20.8

Clock MHz	UBRRHI 7:4 or 3:0	UBRRn	UBR HEX	UBR	Actual Freq	Desired Freq.	% Error
25.576	**0010**	**10011001**	**299**	**665**	**2400**	**2400**	**0.0**
	0001	**01001100**	**14C**	**332**	**4800**	**4800**	**0.0**
	0000	**10100110**	**0A6**	**166**	**9572**	**9600**	**0.3**
	0000	**01101110**	**06E**	**110**	**14401**	**14400**	**0.0**
	0000	**01010010**	**052**	**82**	**19259**	**19200**	**0.3**
	0000	**00110110**	**036**	**54**	**29064**	**28880**	**0.6**
	0000	**00101001**	**029**	**41**	**38060**	**38400**	**0.9**
	0000	**00011011**	**01B**	**27**	**57089**	**57600**	**0.9**
	0000	**00010100**	**014**	**20**	**76119**	**76800**	**0.9**
	0000	**00001101**	**00D**	**13**	**114179**	**115200**	**0.9**
	0000	**00000110**	**006**	**6**	**228357**	**230400**	**0.9**
	0000	00000011	003	3	399625	460800	15.3
	0000	00000001	001	1	799250	912600	14.2

Clock MHz	UBRRHI 7:4 or 3:0	UBRRn	UBR HEX	UBR	Actual Freq	Desired Freq.	% Error
40	**0100**	**00010001**	**411**	**1041**	**2399**	**2400**	**0.0**
	0010	**00001000**	**208**	**520**	**4798**	**4800**	**0.0**
	0001	**00000011**	**103**	**259**	**9615**	**9600**	**0.2**
	0000	**10101100**	**0AC**	**172**	**14451**	**14400**	**0.4**
	0000	**10000001**	**081**	**129**	**19231**	**19200**	**0.2**
	0000	**01010110**	**056**	**86**	**28736**	**28880**	**0.5**
	0000	**01000000**	**040**	**64**	**38462**	**38400**	**0.2**
	0000	**00101010**	**02A**	**42**	**58140**	**57600**	**0.9**
	0000	00100000	020	32	75758	76800	1.4
	0000	00010101	015	21	113636	115200	1.4
	0000	00001010	00A	10	227273	230400	1.4
	0000	00000100	004	4	500000	460800	7.8
	0000	00000010	002	2	833333	912600	9.5

UART0 and UART1 High Byte Baud-rate Register UBRRHI

Bit	7	6	5	4	3	2	1	0	
$20 ($40)	**MSB1**			**LSB1**	**MSB0**			**LSB0**	**UBRRHI**
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	

The UART baud register is a 12-bit register. The 4 most significant bits are located in a separate register, UBRRHI. Note that both UART0 and UART1 share this register. Bit 7 to bit 4 of UBRRHI contain the 4 most significant bits of the UART1 baud register. Bit 3 to bit 0 contain the 4 most significant bits of the UART0 baud register.

Table 43. Status Codes for Slave Receiver Mode

Status Code (TWSR)	Status of the 2-wire Serial Bus and 2-wire Serial Hardware	Application Software Response					Next Action Taken by 2-wire Serial Hardware
		To/From TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$60	Own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$68	Arbitration lost in SLA+R/W as Master; own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$70	General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$78	Arbitration lost in SLA+R/W as Master; General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$80	Previously addressed with own SLA+W; data has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$88	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if GC = “1”
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if GC = “1”; a START condition will be transmitted when the bus becomes free

LowPortE as General Digital I/O

PEn, General I/O pin: The DDEn bit in the DDRE register selects the direction of this pin. If DDEn is set (one), PEn is configured as an output pin. If DDEn is cleared (zero), PEn is configured as an input pin. If PEn is set (one) when configured as an input pin, the MOS pull-up resistor is activated. To switch the pull-up resistor off the PEn has to be cleared (zero) or the pin has to be configured as an output pin. The port pins are input with pull-up when a reset condition becomes active, even if the clock is not running.

Table 48. DDEn(1) Bits on PortE Pins

DDEn(1)	PORTEn(1)	I/O	Pull-up	Comment
0	0	Input	No	Tri-state (High-Z)
0	1	Input	Yes	PDn(1) will source current if external pulled Low (default).
1	0	Output	No	Push-pull zero output
1	1	Output	No	Push-pull one output

Note: 1. n: 7,6...0, pin number

Alternate Functions of PortE

- **PortE, Bit 0**

UART0 Transmit Pin.

- **PortE, Bit 1**

UART0 Receive Pin. Receive Data (Data input pin for the UART0). When the UART0 receiver is enabled this pin is configured as an input regardless of the value of DDRE0. When the UART0 forces this pin to be an input, a logic 1 in PORTE0 will turn on the internal pull-up.

- **PortE, Bit 2**

UART1 Transmit Pin. The alternate functions of Port E as UART0 pins are enabled by setting bit SCR52 in the FPSLIC System Control Register. This is necessary only in smaller pinout packages where the UART signals are not bonded out. The alternate functions of Port E as UART1 pins are enabled by setting bit SCR53 in the FPSLIC System Control Register.

- **PortE, Bit 3**

UART1 Receive Pin. Receive Data (Data input pin for the UART1). When the UART1 receiver is enabled this pin is configured as an input regardless of the value of DDRE2. When the UART1 forces this pin to be an input, a logic 1 in PORTE2 will turn on the internal pull-up.

- **PortE, Bit 4-7**

External Interrupt sources 0/1/2/3: The PE4 – PE7 pins can serve as external interrupt sources to the MCU. Interrupts can be triggered by low-level on these pins. The internal pull-up MOS resistors can be activated as described above.

The alternate functions of PortE as Interrupt pins by setting a bit in the System Control Register. INT0 is controlled by SCR48. INT1 is controlled by SCR49. INT2 is controlled by SCR50. INT3 is controlled by SCR51.

PortE, Bit 7 also shares a pin with the configuration control signal CHECK. Lowering CON to initiate an FPSLIC download (whether for loading or Checking) causes the PE7/CHECK pin to immediately tri-state. This function happens only if the Check pin has been enabled in the system control register. The use of the Check pin will NOT disable the use of that pin as an input to PE7 nor as an input as alternate INT3.

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.

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
VCC(1)	VCC(1)	VCC(1)	54	51	73	106
$\overline{\text{RESET}}$	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$	55	52	74	108
PE0	PE0	PE0	56	53	75	109
PE1	PE1	PE1	57	54	76	110
PD0	PD0	PD0			77	111
PD1	PD1	PD1			78	112
		GND				
		VCC(1)				
		GND				
PE2	PE2	PE2	58	55	79	113
PD2	PD2	PD2		56	80	114
		GND				
No Connect	No Connect	No Connect			81	119
PD3	PD3	PD3			82	120
PD4	PD4	PD4			83	121
	VCC(1)	VCC(1)				
PE3	PE3	PE3	59	57	84	122
$\overline{\text{CS0}}$, Cs0n	$\overline{\text{CS0}}$, Cs0n	$\overline{\text{CS0}}$, Cs0n	60	58	85	123
		GND				
		GND				
		VCC(1)				
SDA	SDA	SDA				124
SCL	SCL	SCL				125
		GND				
PD5	PD5	PD5		59	86	126
PD6	PD6	PD6		60	87	127
PE4	PE4	PE4	61	61	88	128
PE5	PE5	PE5	62	62	89	129
VDD(2)	VDD(2)	VDD(2)	63	63	90	130
GND	GND	GND	64	64	91	131
PE6	PE6	PE6	65	65	92	132
PE7 ($\overline{\text{CHECK}}$)	PE7 ($\overline{\text{CHECK}}$)	PE7 ($\overline{\text{CHECK}}$)	66	66	93	133
PD7	PD7	PD7		67	94	134

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.