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 | 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-25bqu |

## The Busing Network

**Figure 3.** Busing Network

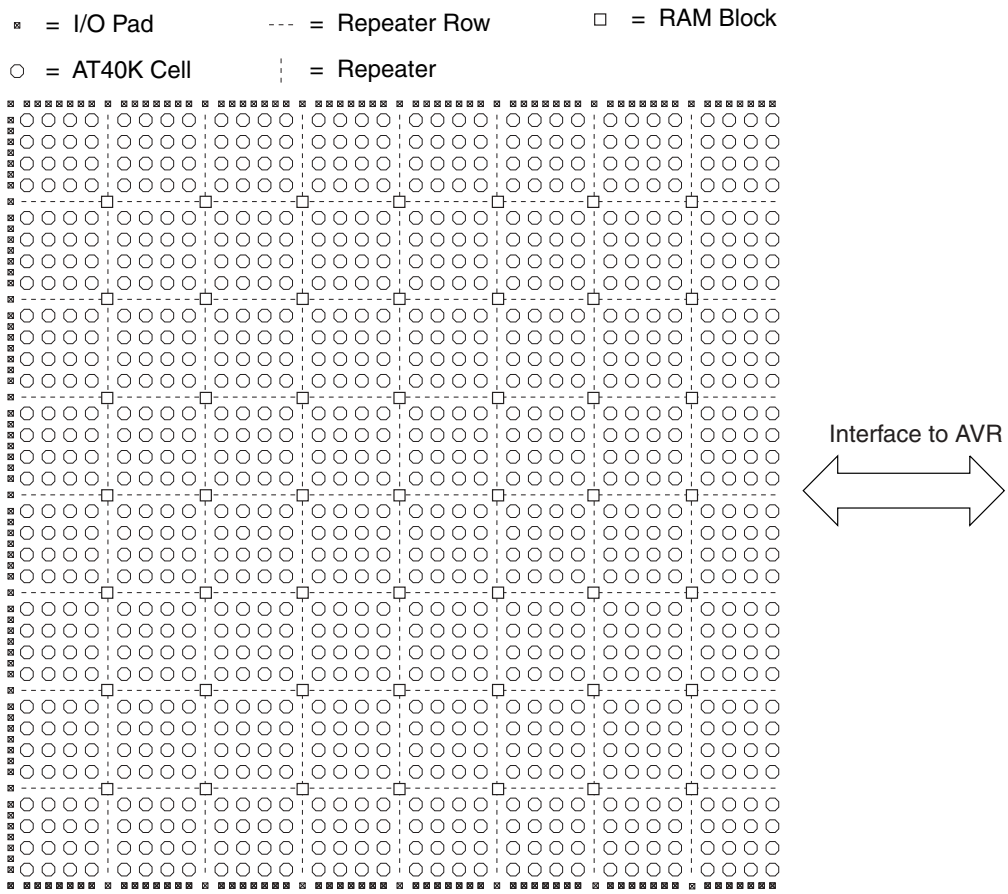



Figure 4 depicts one of five identical FPGA busing planes. Each plane has three bus resources: a local-bus resource (the middle bus) and two express-bus resources. Bus resources are connected via repeaters. Each repeater has connections to two adjacent local-bus segments and two express-bus segments. Each local-bus segment spans four cells and connects to consecutive repeaters. Each express-bus segment spans eight cells and bypasses a repeater. Repeaters regenerate signals and can connect any bus to any other bus (all pathways are legal) on the same plane. Although not shown, a local bus can bypass a repeater via a programmable pass gate, allowing long on-chip tri-state buses to be created. Local/local turns are implemented through pass gates in the cell-bus interface. Express/express turns are implemented through separate pass gates distributed throughout the array.

**Table 4.** Dual-function Buses

| Function | Type | Plane(s) | Direction | Comments |
|---|---|---|---|---|
| Cell Output Enable | Local | 5 | Horizontal and Vertical | |
| FreeRAM Output Enable | Express | 2 | Vertical | Bus full length at array edge bus in first column to left of RAM block |
| FreeRAM Write Enable | Express | 1 | Vertical | Bus full length at array edge bus in first column to left of RAM block |
| FreeRAM Address | Express | 1 - 5 | Vertical | Buses full length at array edge buses in second column to left of RAM block |
| FreeRAM Data In | Local | 1 | Horizontal | |
| FreeRAM Data Out | Local | 2 | Horizontal | |
| Clocking | Express | 4 | Vertical | Bus full length at array edge |
| Set/Reset | Express | 5 | Vertical | Bus full length at array edge |

**Figure 14.** Primary I/O

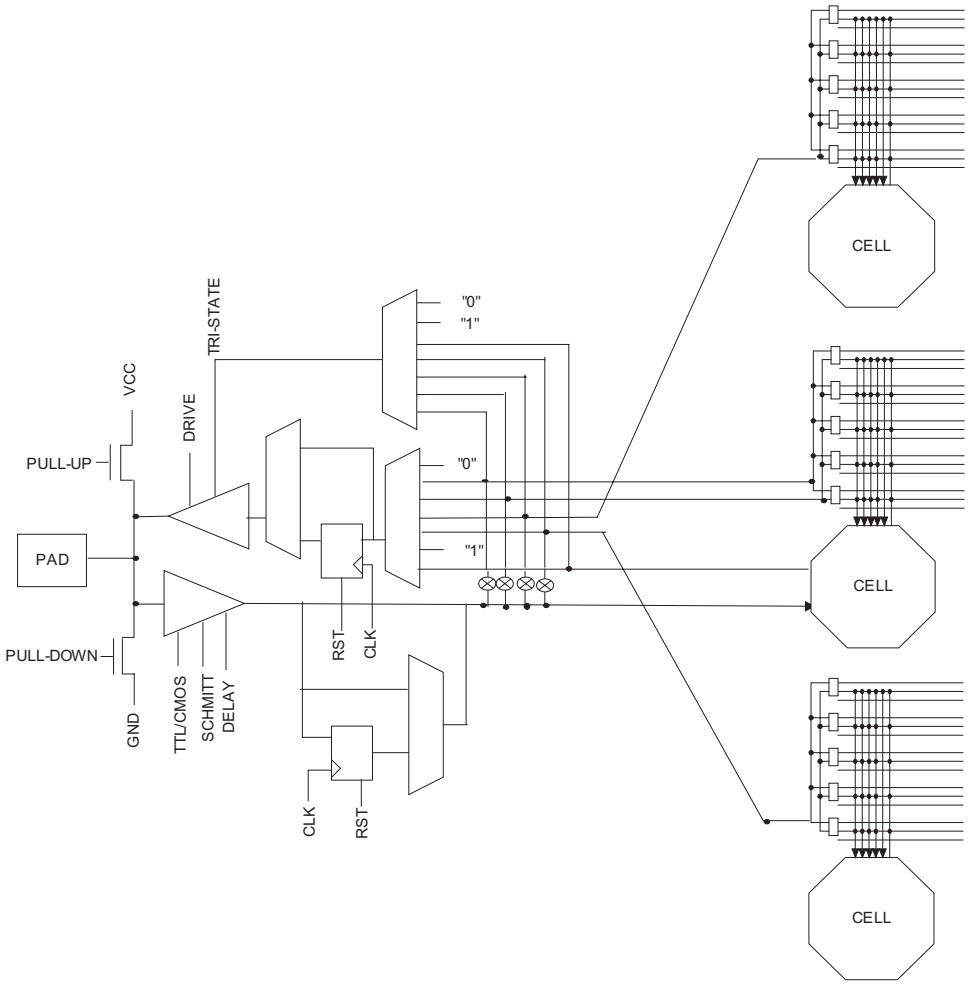


**Table 6.** AVR Data Decode for SRAM 0:17 (16K8)

| Address Range | SRAM | Comments |
|---|---|---|
| $07FF – $0000<br>$0FFF – $0800 | 00<br>01 | AVR Data Read/Write<br>AVR Data Read/Write |
| $17FF – $1000<br>$1FFF – $1800 | 02<br>03 | CR41:40 = 11,10,01 |
| $27FF – $2000<br>$2FFF – $2800 | 04<br>05 | CR41:40 = 11,10 |
| $37FF – $3000<br>$3FFF – $3800 | 06<br>07 | CR41:40 = 11 |

**B Side**

The B side is not partitioned; the FPGA (and AVR debug mode) views the memory space as 36 x 8 Kbytes.

- The B side is accessed by the FPGA/Configuration Logic.
- The B side is accessed by the AVR with ST and LD instructions in DBG mode for code self-modify.

  To activate the debug mode and allow the AVR to access the program code space (with ST – see Figure 21 – and LD – see Figure 22 – instructions), the DBG bit (bit 1) of the SFTCR $3A ($5A) register has to be set. When this bit is set, SCR36 and SCR37 are ignored – you can overwrite anything in the AVR program memory.

  The FPGA memory access interface should be disabled while in debug mode. This is to ensure that there is no contention between the FPGA address and data signals and the AVR-generated address and data signals. To ensure the AVR has control over the “B side” memory interface, the FMXOR bit (bit 3) of the SFTCR $3A ($5A) register should be used in conjunction with the SCR63 system control register bit.

  The FMXOR bit is XORed with the System Control Register’s Enable FPGA SRAM Interface bit (SCR63). The behavior when this bit is set to 1 is dependent on how the SCR was initialized. If the Enable FPGA SRAM Interface bit (SCR63) in the SCR is 0, the FMXOR bit enables the FPGA SRAM Interface when set to 1. If the Enable FPGA SRAM Interface bit in the SCR is 1, the FMXOR bit disables the FPGA SRAM Interface when set to 1. During AVR reset, the FMXOR bit is cleared by the hardware.

  Even though the FPGA (and AVR debug mode) views the memory space as 36 x 8 Kbytes, an awareness of the 2K x 8 partitions (or SRAM labels) is required if Frame (and AVR debug mode) read/writes are to be meaningful to the AVR.
- AVR data to FPGA addressing is 1:1 mapping.
- AVR program to FPGA addressing requires 16-bit to 8-bit mapping and an understanding of the partitions in Table 7.

**Table 7.** Summary Table for AVR and FPGA SRAM Addressing

| SRAM | FPGA and AVR DBG Address Range | AVR Data Address Range | AVR PC Address Range |
|---|---|---|---|
| 00 | $0000 - $07FF | $0000 - $07FF | |
| 01 | $0800 - $0FFF | $0800 - $0FFF | |
| 02(1) | $1000 - $17FF | $1000 - $17FF | $3800 - $3FFF (LS Byte) |
| 03(1) | $1800 - $1FFF | $1800 - $1FFF | $3800 - $3FFF (MS Byte) |
| 04(1) | $2000 - $27FF | $2000 - $27FF | $3000 - $37FF (LS Byte) |

## System Control

### Configuration Modes

The AT94K family has four configuration modes controlled by mode pins M0 and M2, see Table 10.

**Table 10.** Configuration Modes

| M2 | M0 | Name |
|---|---|---|
| 0 | 0 | Mode 0 - Master Serial |
| 0 | 1 | Mode 1 - Slave Serial Cascade |
| 1 | 0 | Mode 2 - Reserved |
| 1 | 1 | Mode 3 - Reserved |

Modes 2 and 3 are reserved and are used for factory test.

Modes 0 and 1 are pin-compatible with the appropriate AT40K counterpart. AVR I/O will be taken over by the configuration logic for the CHECK pin during both modes.

Refer to the “AT94K Series Configuration” application note for details on downloading bitstreams.

### System Control Register – FPGA/AVR

The configuration control register in the FPSLIC consists of 8 bytes of data, which are loaded with the FPGA/Prog. Code at power-up from external nonvolatile memory. FPSLIC System Control Register values, see Table 11, can be set in the System Designer software. Recommended defaults are included in the software.

**Table 11.** FPSLIC System Control Register

| Bit | Description |
|---|---|
| SCR0 - SCR1 | Reserved |
| SCR2 | 0 = Enable Cascading<br>1 = Disable Cascading<br>SCR2 controls the operation of the dual-function I/O CSOUT. When SCR2 is set, the CSOUT pin is not used by the configuration during downloads, set this bit for configurations where two or more devices are cascaded together. This applies for configuration to another FPSLIC device or to an FPGA. |
| SCR3 | 0 = Check Function Enabled<br>1 = Check Function Disabled<br>SCR3 controls the operation of the CHECK pin and enables the Check Function. When SCR3 is set, the dual use AVR I/O/CHECK pin is not used by the configuration during downloads, and can be used as AVR I/O. |
| SCR4 | 0 = Memory Lockout Disabled<br>1 = Memory Lockout Enabled<br>SCR4 is the Security Flag and controls the writing and checking of configuration memory during any subsequent configuration download. When SCR4 is set, any subsequent configuration download initiated by the user, whether a normal download or a CHECK function download, causes the INIT pin to immediately activate. CON is released, and no further configuration activity takes place. The download sequence during which SCR4 is set is NOT affected. The Control Register write is also prohibited, so bit SCR4 may only be cleared by a power-on reset or manual reset. |
| SCR5 | Reserved |

**Instruction Set Summary (Continued)**

| Mnemonics | Operands | Description | Operation | Flags | #Clock |
|---|---|---|---|---|---|
| CLS | | Clear Signed Test Flag | S ← 0 | S | 1 |
| SEV | | Set Two's Complement Overflow | V ← 1 | V | 1 |
| CLV | | Clear Two's Complement Overflow | V ← 0 | V | 1 |
| SET | | Set T in SREG | T ← 1 | T | 1 |
| CLT | | Clear T in SREG | T ← 0 | T | 1 |
| SEH | | Set Half-carry Flag in SREG | H ← 1 | H | 1 |
| CLH | | Clear Half-carry Flag in SREG | H ← 0 | H | 1 |
| NOP | | No Operation | | None | 1 |
| SLEEP | | Sleep | (See specific description for Sleep) | None | 1 |
| WDR | | Watchdog Reset | (See specific description for WDR) | None | 1 |
| BREAK | | Break | For on-chip debug only | None | N/A |

## Pin Descriptions

**$V_{CC}$** Supply voltage

**GND** Ground

**PortD (PD7..PD0)** Port D is an 8-bit bi-directional I/O port with internal programmable pull-up resistors. The Port D output buffers can be programmed to sink/source either 6 or 20 mA (SCR54 – see "System Control Register – FPGA/AVR" on page 30). As inputs, Port D pins that are externally pulled Low will source current if the programmable pull-up resistors are activated.

The Port D pins are input with pull-up when a reset condition becomes active, even if the clock is not running. On lower pin count packages Port D may not be available. Check the Pin List for details.

**PortE (PE7..PE0)** Port E is an 8-bit bi-directional I/O port with internal programmable pull-up resistors. The Port E output buffers can be programmed to sink/source either 6 or 20 mA (SCR55 – see "System Control Register – FPGA/AVR" on page 30). As inputs, Port E pins that are externally pulled Low will source current if the pull-up resistors are activated.

Port E also serves the functions of various special features. See Table 47 on page 149.

The Port E pins are input with pull-up when a reset condition becomes active, even if the clock is not running

**RX0** Input (receive) to UART(0) – See SCR52

**TX0** Output (transmit) from UART(0) – See SCR52

**RX1** Input (receive) to UART(1) – See SCR53

**TX1** Output (transmit) from UART(1) – See SCR53

**XTAL1** Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

#### *Data Indirect*

Operand address is the contents of the X-, Y- or the Z-register.

#### *Data Indirect with Pre-decrement*

The X-, Y- or the Z-register is decremented before the operation. Operand address is the decremented contents of the X, Y or the Z-register.

#### *Data Indirect with Post-increment*

The X-, Y- or the Z-register is incremented after the operation. The operand address is the content of the X-, Y- or the Z-register prior to incrementing.

#### *Direct Program Address, JMP and CALL*

Program execution continues at the address immediate in the instruction words.

#### *Indirect Program Addressing, IJMP and ICALL*

Program execution continues at address contained by the Z-register (i.e., the PC is loaded with the contents of the Z-register).

#### *Relative Program Addressing, RJMP and RCALL*

Program execution continues at address PC + k + 1. The relative address k is -2048 to 2047.

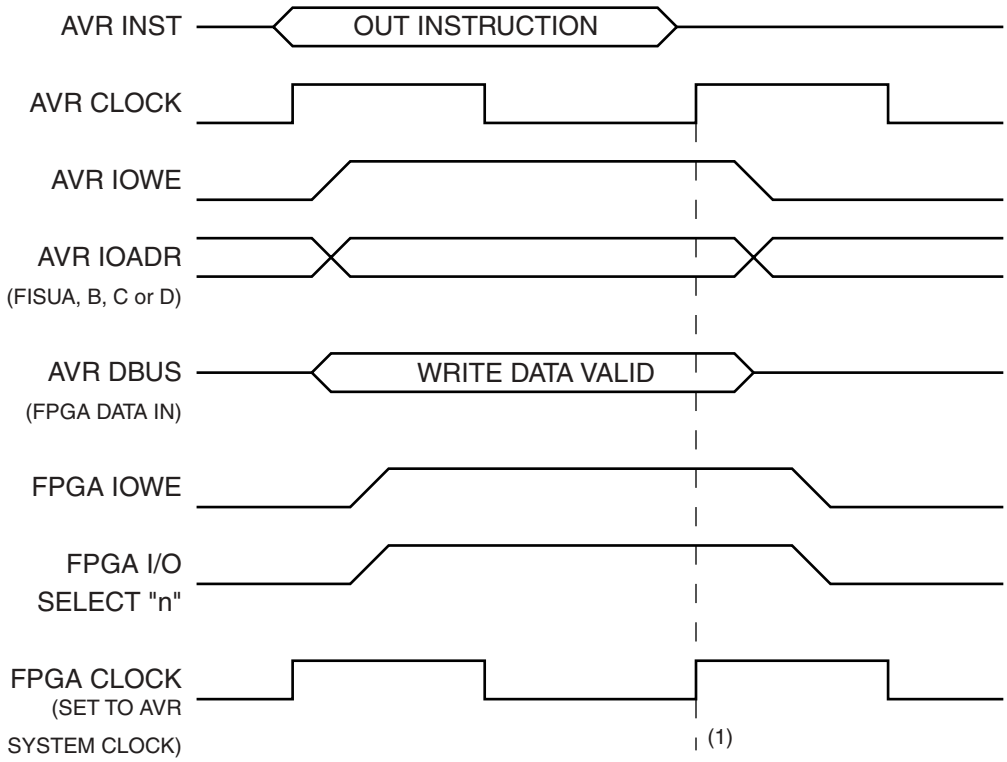
### Memory Access Times and Instruction Execution Timing

This section describes the general access timing concepts for instruction execution and internal memory access.

The AVR CPU is driven by the XTAL1 input directly generated from the external clock crystal for the chip. No internal clock division is used.

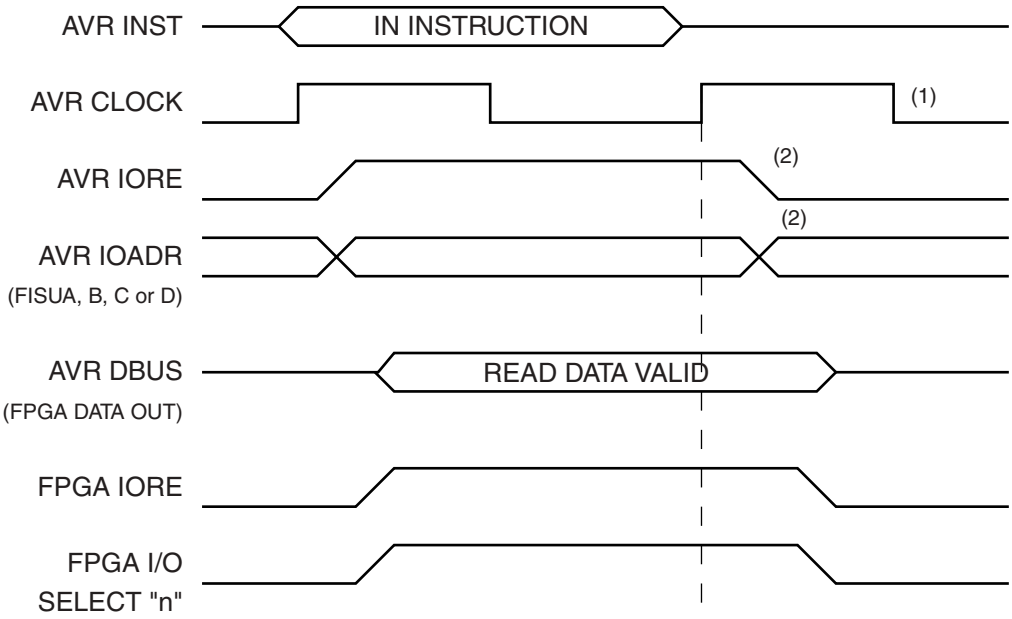
Figure 29 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access register file concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks and functions per power-unit.

**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.

## FPGA I/O Interrupt Control by AVR

This is an alternate memory space for the FPGA I/O Select addresses. If the FIADR bit in the FISCR register is set to logic 1, the four I/O addresses, FISUA - FISUD, are mapped to physical registers and provide memory space for FPGA interrupt masking and interrupt flag status. If the FIADR bit in the FISCR register is cleared to a logic 0, the I/O register addresses will be decoded into FPGA select lines.

All FPGA interrupt lines into the AVR are negative edge triggered. See page 58 for interrupt priority.

***Interrupt Control Registers – FISUA..D***

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $14 ($34) | **FIF3** | **FIF2** | **FIF1** | **FIF0** | **FINT3** | **FINT2** | **FINT1** | **FINT0** | **FISUA** |
| $15 ($35) | **FIF7** | **FIF6** | **FIF5** | **FIF4** | **FINT7** | **FINT6** | **FINT5** | **FINT4** | **FSUB** |
| $16 ($36) | **FIF11** | **FIF10** | **FIF9** | **FIF8** | **FINT11** | **FINT10** | **FINT9** | **FINT8** | **FISUC** |
| $17 ($37) | **FIF15** | **FIF14** | **FIF13** | **FIF12** | **FINT15** | **FINT14** | **FINT13** | **FINT12** | **FISUD** |
| 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 7..4 - FIF3 - 0: FPGA Interrupt Flags 3 - 0**

The 16 FPGA interrupt flag bits all work the same. Each is set (one) by a valid negative edge transition on its associated interrupt line from the FPGA. Valid transitions are defined as any change in state preceded by at least two cycles of the old state and succeeded by at least two cycles of the new state. Therefore, it is required that interrupt lines transition from 1 to 0 at least two cycles after the line is stable High; the line must then remain stable Low for at least two cycles following the transition. Each bit is cleared by the hardware when executing the corresponding interrupt handling vector. Alternatively, each bit will be cleared by writing a logic 1 to it. When the I-bit in the Status Register, the corresponding FPGA interrupt mask bit and the given FPGA interrupt flag bit are set (one), the associated interrupt is executed.

- **Bits 7..4 - FIF7 - 4: FPGA Interrupt Flags 7 - 4**

See *Bits 7..4 - FIF3 - 0: FPGA Interrupt Flags 3 - 0*.

- **Bits 7..4 - FIF11 - 8: FPGA Interrupt Flags 11 - 8**

See *Bits 7..4 - FIF3 - 0: FPGA Interrupt Flags 3 - 0*. Not available on the AT94K05.

- **Bits 7..4 - FIF15 - 12: FPGA Interrupt Flags 15 - 12**

See *Bits 7..4 - FIF3 - 0: FPGA Interrupt Flags 3 - 0*. Not available on the AT94K05.

- **Bits 3..0 - FINT3 - 0: FPGA Interrupt Masks 3 - 0**[1]

The 16 FPGA interrupt mask bits all work the same. When a mask bit is set (one) and the I-bit in the Status Register is set (one), the given FPGA interrupt is enabled. The corresponding interrupt handling vector is executed when the given FPGA interrupt flag bit is set (one) by a negative edge transition on the associated interrupt line from the FPGA.

Note: 1. FPGA interrupts 3 - 0 will cause a wake-up from the AVR Sleep modes. These interrupts are treated as low-level triggered in the Power-down and Power-save modes, see “Sleep Modes” on page 66.

- **Bits 3..0 - FINT7 - 4: FPGA Interrupt Masks 7 - 4**

See *Bits 3..0 - FINT3 - 0: FPGA Interrupt Masks 3 - 0.*

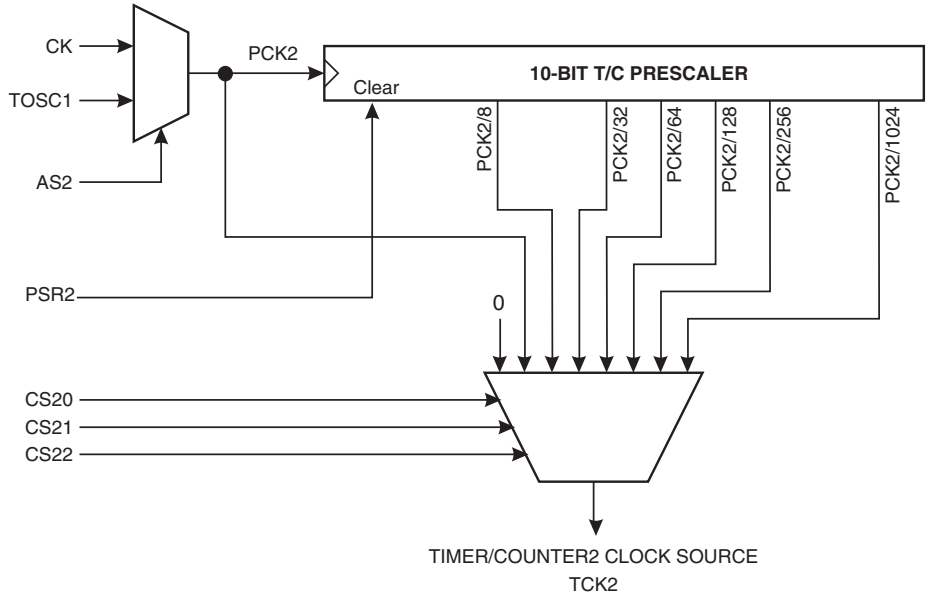
- **Bits 3..0 - FINT11 - 8: FPGA Interrupt Masks 11 - 8**

See *Bits 3..0 - FINT3 - 0: FPGA Interrupt Masks 3 - 0.* Not available on the AT94K05.

- **Bits 3..0 - FINT15 - 12: FPGA Interrupt Masks 15 -12**

See *Bits 3..0 - FINT3 - 0: FPGA Interrupt Masks 3 - 0.* Not available on the AT94K05.

**Figure 49.** Timer/Counter2 Prescaler




***Special Function I/O Register – SFIOR***

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $30 ($50) | - | - | - | - | - | - | PSR2 | PSR10 | SFIOR |
| Read/Write | R | R | R | R | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 7..2 - Res: Reserved Bits**

These bits are reserved bits in the FPSLIC and are always read as zero.

- **Bit 1 - PSR2: Prescaler Reset Timer/Counter2**

When this bit is set (one) the Timer/Counter2 prescaler will be reset. The bit will be cleared by the hardware after the operation is performed. Writing a zero to this bit will have no effect. This bit will always be read as zero if Timer/Counter2 is clocked by the internal CPU clock. If this bit is written when Timer/Counter2 is operating in asynchronous mode; however, the bit will remain as one until the prescaler has been reset. See “Asynchronous Operation of Timer/Counter2” on page 94 for a detailed description of asynchronous operation.

- **Bit 0 - PSR10: Prescaler Reset Timer/Counter1 and Timer/Counter0**

When this bit is set (one) the Timer/Counter1 and Timer/Counter0 prescaler will be reset. The bit will be cleared by the hardware after the operation is performed. Writing a zero to this bit will have no effect. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a reset of this prescaler will affect both timers. This bit will always be read as zero.

## 8-bit Timers/Counters T/C0 and T/C2

Figure 50 shows the block diagram for Timer/Counter0. Figure 51 shows the block diagram for Timer/Counter2.

*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

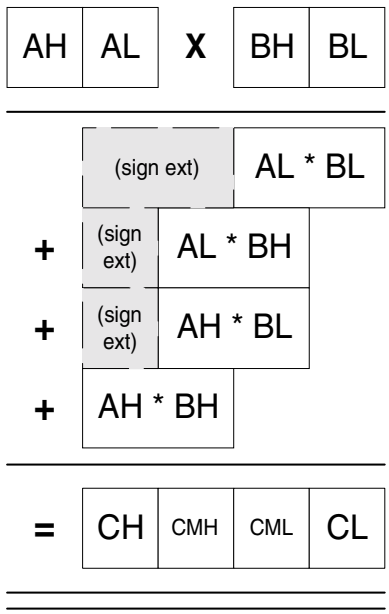


**Table 35.** Comparison of Integer and Fractional Formats

| Bit Number | Unsigned Integer Bit Significance | Unsigned Fractional Number Bit Significance |
|---|---|---|
| 7 | $2^7 = 128$ | $2^0 = 1$ |
| 6 | $2^6 = 64$ | $2^{-1} = 0.5$ |
| 5 | $2^5 = 32$ | $2^{-2} = 0.25$ |
| 4 | $2^4 = 16$ | $2^{-3} = 0.125$ |
| 3 | $2^3 = 8$ | $2^{-4} = 0.0625$ |
| 2 | $2^2 = 4$ | $2^{-5} = 0.3125$ |
| 1 | $2^1 = 2$ | $2^{-6} = 0.015625$ |
| 0 | $2^0 = 1$ | $2^{-7} = 0.0078125$ |

Using the FMUL, FMULS and FMULSU instructions should not be more complex than the MUL, MULS and MULSU instructions. However, one potential problem is to assign fractional variables right values in a simple way. The fraction 0.75 (= 0.5 + 0.25) will, for example, be “0110 0000” if 8 bits are used.

To convert a positive fractional number in the range [0, 2> (for example 1.8125) to the format used in the AVR, the following algorithm, illustrated by an example, should be used:

Is there a “1” in the number?

Yes, 1.8125 is higher than or equal to 1.

Byte is now “1xxx xxxx”

Is there a “0.5” in the rest?

0.8125 / 0.5 = 1.625

Yes, 1.625 is higher than or equal to 1.

Byte is now “11xx xxxx”

Is there a “0.25” in the rest?

0.625 / 0.5 = 1.25

Yes, 1.25 is higher than or equal to 1.

Byte is now “111x xxxx”

Is there a “0.125” in the rest?

0.25 / 0.5 = 0.5

No, 0.5 is lower than 1.

Byte is now “1110 xxxx”

Is there a “0.0625” in the rest?

0.5 / 0.5 = 1

Yes, 1 is higher than or equal to 1.

Byte is now “1110 1xxx”

Since we do not have a rest, the remaining three bits will be zero, and the final result is “1110 1000”, which is 1 + 0.5 + 0.25 + 0.0625 = 1.8125.

*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)

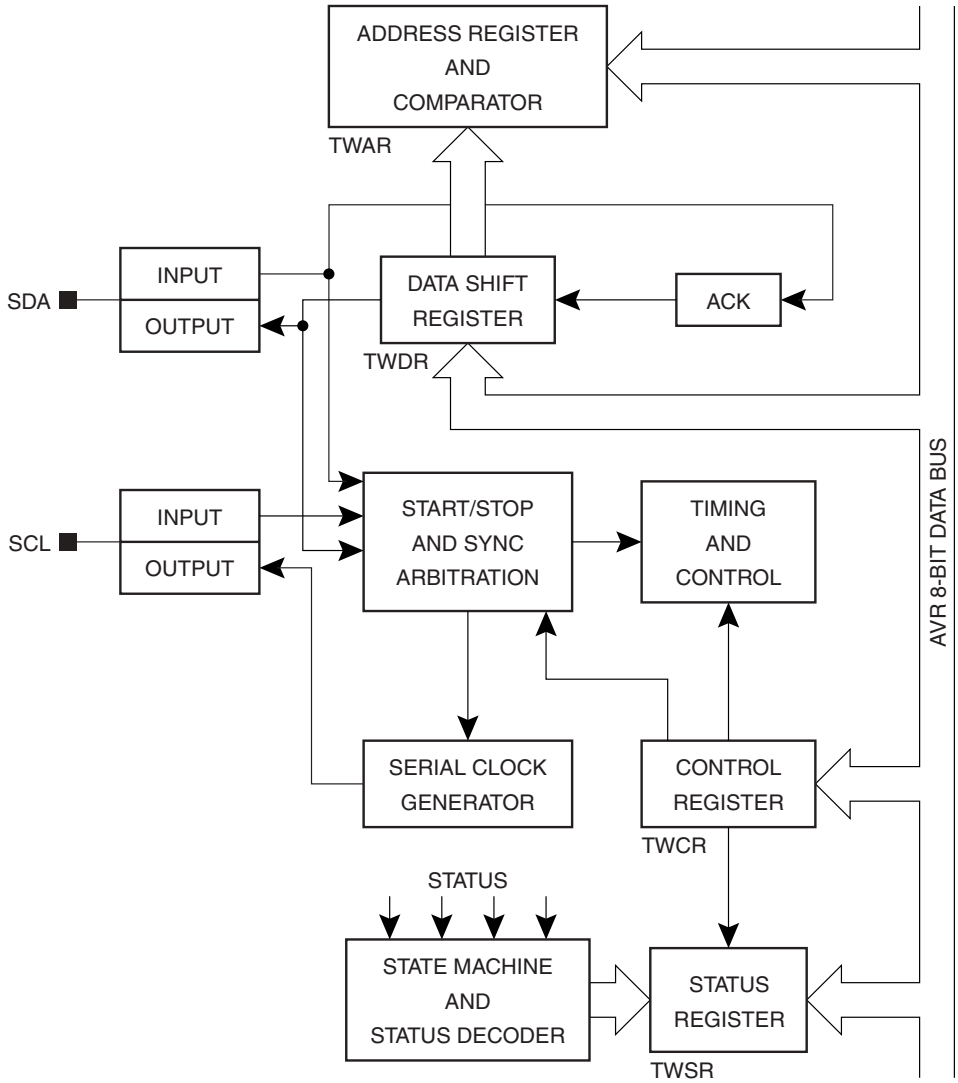
```
fmac16x16_32:                 ; Register usage optimized
   clr   r2

   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
```

```
   adc   r17, r1
   adc   r18, r2
   adc   r19, r2

   fmulsu   r23, r20          ; ( (signed)ah * bl ) << 1
   sbc   r19, r2
   add   r17, r0
   adc   r18, r1
   adc   r19, r2

   fmulsu   r21, r22          ; ( (signed)bh * al ) << 1
   sbc   r19, r2
   add   r17, r0
   adc   r18, r1
   adc   r19, r2

   ret

fmac16x16_32_method_B         ; uses two temporary registers (r4,r5), speed/Size
                              optimized
                              ; but reduces cycles/words by 2
   clr   r2

   fmuls r23, r21             ; ( (signed)ah * (signed)bh ) << 1
   movw  r5:r4,r1:r0
   fmul  r22, r20             ; ( al * bl ) << 1
   adc   r4, r2

add   r16, r0
   adc   r17, r1
   adc   r18, r4
   adc   r19, r5
fmulsu   r23, r20             ; ( (signed)ah * bl ) << 1
   sbc   r19, r2
   add   r17, r0
   adc   r18, r1
   adc   r19, r2
   fmulsu   r21, r22          ; ( (signed)bh * al ) << 1
   sbc   r19, r2
   add   r17, r0
   adc   r18, r1
   adc   r19, r2

   ret
```

*Comment on Implementations*

All 16-bit x 16-bit = 32-bit functions implemented here start by clearing the R2 register, which is just used as a “dummy” register with the “add with carry” (ADC) and “subtract with carry” (SBC) operations. These operations do not alter the contents of the R2 register. If the R2 register is not used elsewhere in the code, it is not necessary to clear the R2 register each time these functions are called, but only once prior to the first call to one of the functions.

**Figure 70.** Block diagram of the 2-wire Serial Bus Interface




The CPU interfaces with the 2-wire Serial Interface via the following five I/O registers: the 2-wire Serial Bit-rate Register (TWBR), the 2-wire Serial Control Register (TWCR), the 2-wire Serial Status Register (TWSR), the 2-wire Serial Data Register (TWDR), and the 2-wire Serial Address Register (TWAR, used in Slave mode).

### *The 2-wire Serial Bit-rate Register – TWBR*

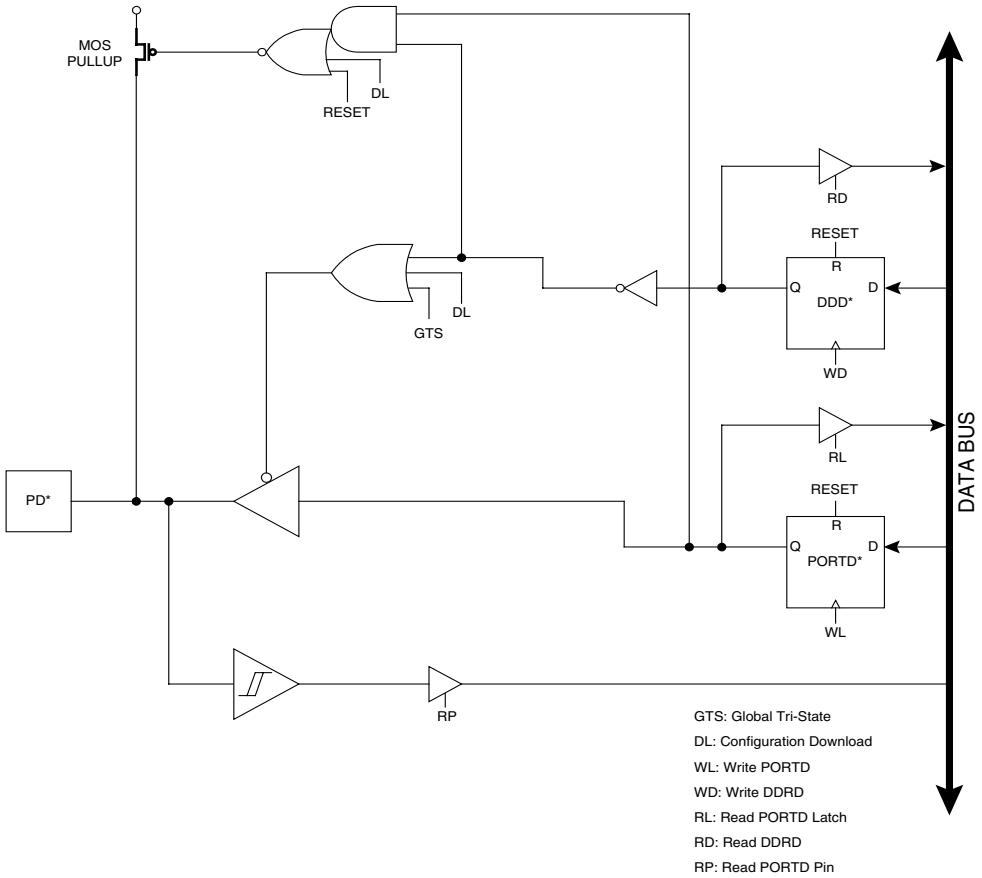
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| $1C ($3C) | **TWBR7** | **TWBR6** | **TWBR5** | **TWBR4** | **TWBR3** | **TWBR2** | **TWBR1** | **TWBR0** | **TWBR** |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**Table 46.** DDDn[1] Bits on PortD Pins

| DDDn[1] | PORTDn[1] | I/O | Pull-up | Comment |
|---|---|---|---|---|
| 0 | 0 | Input | No | Tri-state (High-Z) |
| 0 | 1 | Input | Yes | PDn 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

**Figure 75.** PortD Schematic Diagram




## PortE

PortE is an 8-bit bi-directional I/O port with internal pull-up resistors.

Three I/O memory address locations are allocated for the PortE, one each for the Data Register – PORTE, $07($27), Data Direction Register – DDRE, $06($26) and the PortE Input Pins – PINE, $05($25). The PortE Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

The PortE output buffers can sink 20 mA. As inputs, PortE pins that are externally pulled Low will source current if the pull-up resistors are activated.

All PortE pins have alternate functions as shown in Table 47.

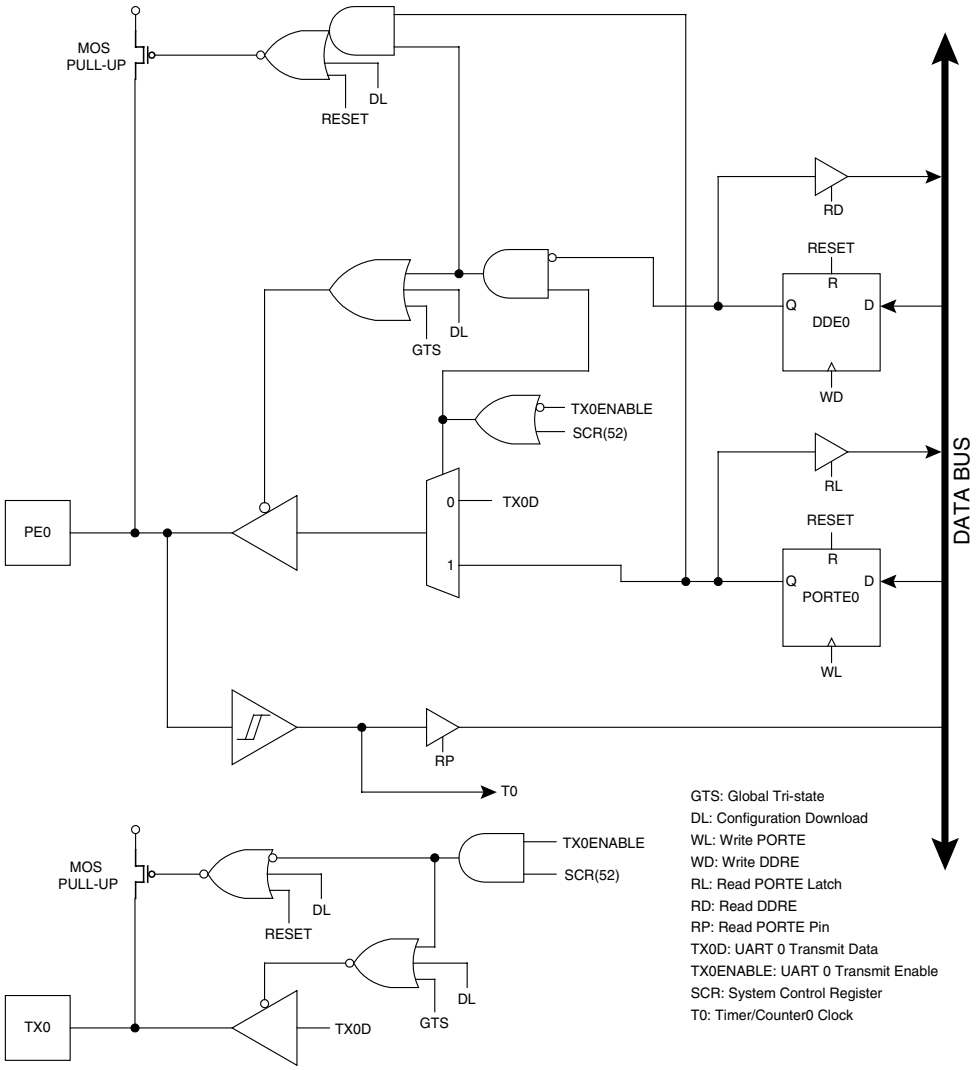
*Alternate I/O Functions of PortE*

PortE may also be used for various Timer/Counter functions, such as External Input Clocks (TC0 and TC1), Input Capture (TC1), Pulse Width Modulation (TC0, TC1 and TC2), and toggling upon an Output Compare (TC0, TC1 and TC2). For a detailed pinout description, consult Table 47 on page 149. For more information on the function of each pin, See “Timer/Counters” on page 85.

*PortE Schematics*

Note that all port pins are synchronized. The synchronization latches are, however, not shown in the figures.

**Figure 76.** PortE Schematic Diagram (Pin PE0)



# AC & DC Timing Characteristics

## Absolute Maximum Ratings*(1)

| | |
|---|---|
| Operating Temperature | -55°C to +125°C |
| Storage Temperature | -65°C to +150°C |
| Voltage(2) on Any Pin with Respect to Ground | -0.5V to +5.0V |
| Supply Voltage ($V_{CC}$) | -0.5V to +5.0V |
| Maximum Soldering Temp. (10 sec. @ 1/16 in.) | 250°C |
| ESD ($R_{ZAP}$ = 1.5K, $C_{ZAP}$ = 100 pF) | 2000V |

*NOTICE: Stresses beyond those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those listed under operating conditions is not implied. Exposure to Absolute Maximum Rating conditions for extended periods of time may affect device reliability.

Notes: 1. For AL parts only
2. Minimum voltage of -0.5V DC which may undershoot to -2.0V for pulses of less than 20 ns.

## DC and AC Operating Range – 3.3V Operation

| | | AT94K Commercial | AT94K Industrial |
|---|---|---|---|
| Operating Temperature (Case) | | 0°C - 70°C | -40°C - 85°C |
| $V_{CC}$ Power Supply | | 3.3V ± 0.3V | 3.3V ± 0.3V |
| Input Voltage Level (CMOS) | High ($V_{IHC}$) | 70% - 100% $V_{CC}$ | 70% - 100% $V_{CC}$ |
| | Low ($V_{ILC}$) | 0 - 30% $V_{CC}$ | 0 - 30% $V_{CC}$ |

**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.

**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/O173 | I/O345 | | | | 188 |
| | I/O174 | I/O346 | | | | 189 |
| I/O117 (A10) | I/O175 (A10) | I/O347 (A10) | 5 | 94 | 133 | 190 |
| I/O118 (A11) | I/O176 (A11) | I/O348 (A11) | 6 | 95 | 134 | 191 |
| | | VCC(1) | | | | |
| | | GND | | | | |
| | | I/O349 | | | | |
| | | I/O350 | | | | |
| | | I/O351 | | | | |
| | | I/O352 | | | | |
| | | I/O353 | | | | |
| | | I/O354 | | | | |
| | | GND | | | | |
| | | I/O355 | | | | |
| | | I/O356 | | | | |
| | VDD(2) | VDD(2) | | | | |
| | I/O177 | I/O357 | | | | |
| | I/O178 | I/O358 | | | | |
| I/O119 | I/O179 | I/O359 | | | 135 | 192 |
| I/O120 | I/O180 | I/O360 | | | 136 | 193 |
| GND | GND | GND | | | 137 | 194 |
| | | I/O361 | | | | |
| | | I/O362 | | | | |
| | I/O181 | I/O363 | | | | 195 |
| | I/O182 | I/O364 | | | | 196 |
| | | I/O365 | | | | |
| | | I/O366 | | | | |
| | | GND | | | | |
| | | I/O367 | | | | |
| | | I/O368 | | | | |
| I/O121 | I/O183 | I/O369 | | | | 197 |
| I/O122 | I/O184 | I/O370 | | | | 198 |
| I/O123 (A12) | I/O185 (A12) | I/O371 (A12) | 7 | 96 | 138 | 199 |
| I/O124 (A13) | I/O186 (A13) | I/O372 (A13) | 8 | 97 | 139 | 200 |

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.

## Ordering Information

| Usable Gates | Speed Grade | Ordering Code | Package | Operation Range |
|---|---|---|---|---|
| 5,000 | -25 MHz | AT94K05AL-25AJC<br>AT94K05AL-25AQC<br>AT94K05AL-25BQC<br>AT94K05AL-25DQC | 84J<br>100A<br>144L1<br>208Q1 | Commercial<br>(0°C - 70°C) |
| | | AT94K05AL-25AJI<br>AT94K05AL-25AQI<br>AT94K05AL-25BQI<br>AT94K05AL-25DQI | 84J<br>100A<br>144L1<br>208Q1 | Industrial<br>(-40°C - 85°C) |
| 10,000 | -25 MHz | AT94K10AL-25AJC<br>AT94K10AL-25AQC<br>AT94K10AL-25BQC<br>AT94K10AL-25DQC | 84J<br>100A<br>144L1<br>208Q1 | Commercial<br>(0°C - 70°C) |
| | | AT94K10AL-25AJI<br>AT94K10AL-25AQI<br>AT94K10AL-25BQI<br>AT94K10AL-25DQI | 84J<br>100A<br>144L1<br>208Q1 | Industrial<br>(-40°C - 85°C) |
| 40,000 | -25 MHz | AT94K40AL-25BQC<br>AT94K40AL-25DQC | 144L1<br>208Q1 | Commercial<br>(0°C - 70°C) |
| | | AT94K40AL-25BQI<br>AT94K40AL-25DQI | 144L1<br>208Q1 | Industrial<br>(-40°C - 85°C) |

| Package Type | |
|---|---|
| 84J | 84-lead, Plastic J-leaded Chip Carrier (PLCC) |
| 100A | 100-lead, Thin (1.0 mm) Plastic Quad Flat Package (TQFP) |
| 144L1 | 144-lead, Low Profile Plastic Gull Wing Quad Flat Package (LQFP) |
| 208Q1 | 208-lead, Plastic Gull Wing Quad Flat Package (PQFP) |