Welcome to E-XFL.COM

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

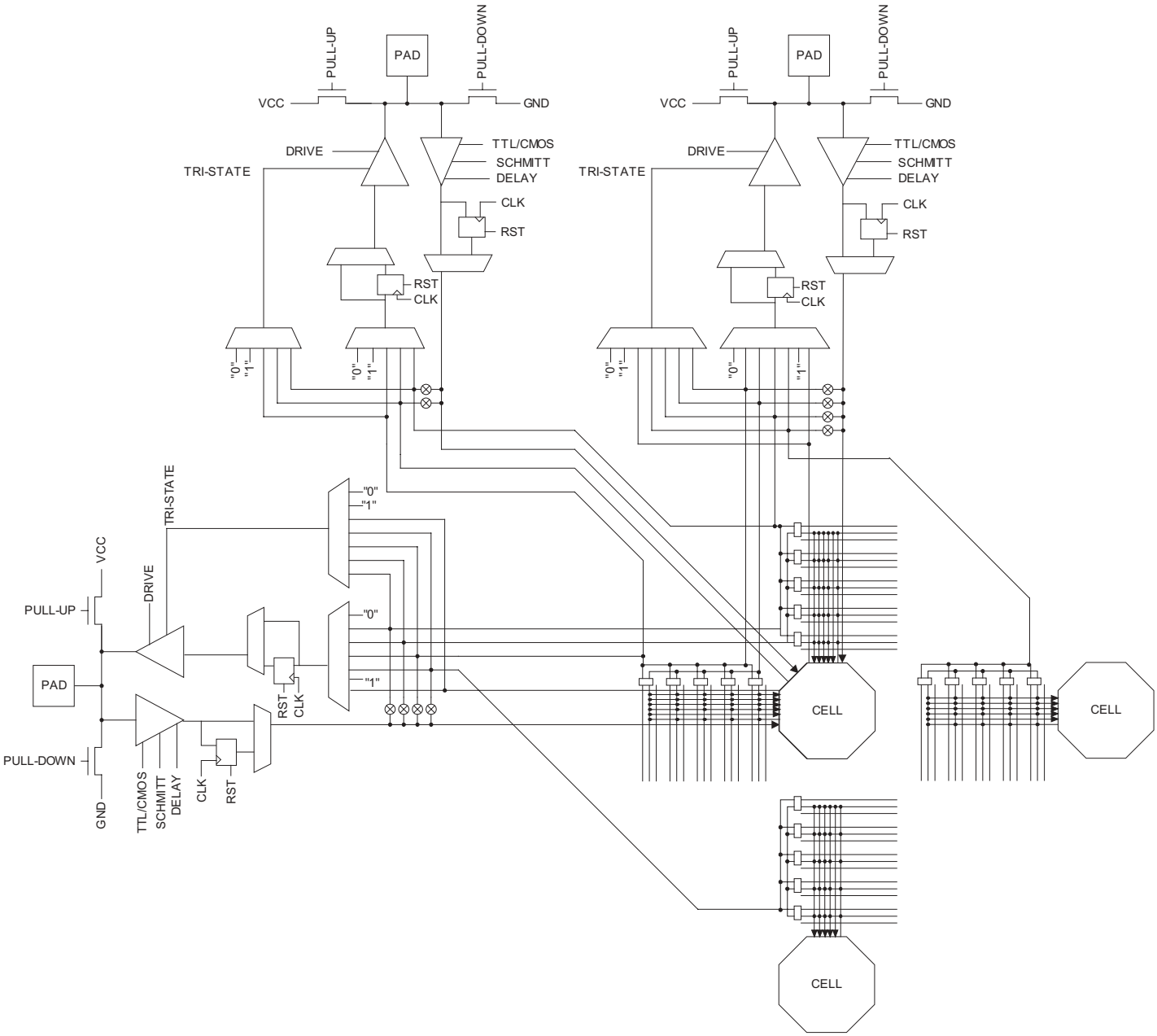
Embedded - FPGAs (Field Programmable Gate Arrays) with Microcontrollers represent a cutting-edge category of electronic components that combine the flexibility of FPGA technology with the processing power of integrated microcontrollers. This hybrid approach offers a versatile solution for designing and 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	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	0°C ~ 70°C
Package / Case	100-TQFP
Supplier Device Package	100-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at94k05al-25aqc

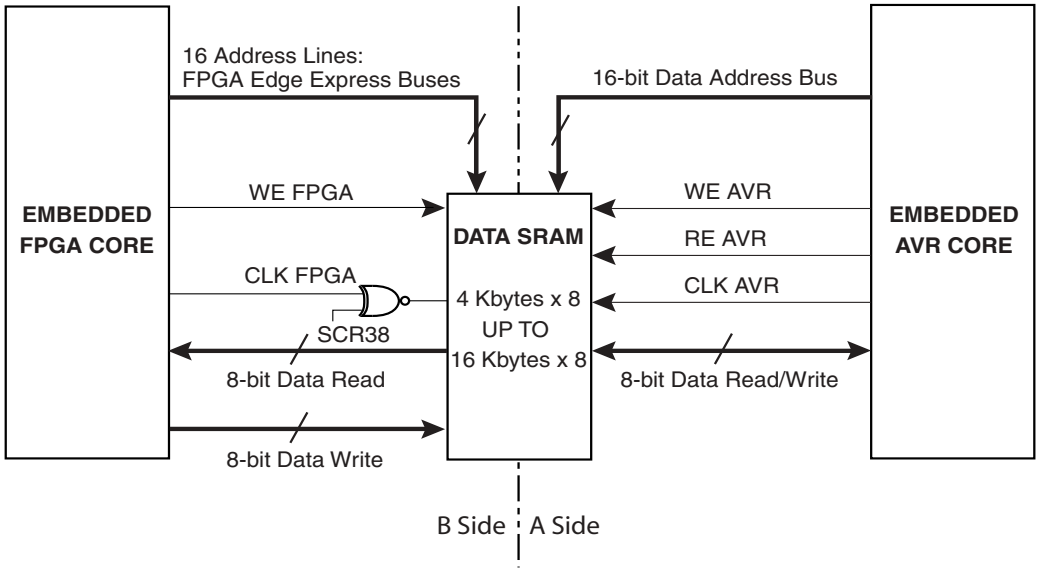
Figure 17. Corner I/Os



Data SRAM Access by FPGA – FPGAFrame Mode

The FPGA user logic has access to the data SRAM directly through the FPGA side of the dual-port memory, see Figure 20. A single bit in the configuration control register (SCR63 – see "System Control Register – FPGA/AVR" on page 30) enables this interface. The interface is disabled during configuration downloads. Express buses on the East edge of the array are used to interface the memory. Full read and write access is available. To allow easy implementation, the interface itself is dedicated in routing resources, and is controlled in the System Designer software suite using the AVR FPGA interface dialog.

Figure 20. Internal SRAM Access – Normal Use



Once the SCR63 bit is set there is no additional read enable from the FPGA side. This means that the read is always enabled. You can also perform a read or write from the AVR at the same time as an FPGA read or write. If there is a possibility of a write address being accessed by both devices at the same time, the designer should add arbitration to the FPGA Logic to control who has priority. In most cases the AVR would be used to restrict access by the FPGA using the FMXOR bit, see "Software Control Register – SFTCR" on page 51. You can read from the same location from both sides simultaneously.

SCR bit 38 controls the polarity of the clock to the SRAM from the AT40K FPGA.

SRAM Access by FPGA/AVR

This option is used to allow for code (Program Memory) changes.

Accessing and Modifying the Program Memory from the AVR

The FPSLIC SRAM is up to 36 x 8 Kbytes of dual port, see Figure 19):

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

Structurally, the [(n • 2) Kbytes 8] memory is built from (n)2 Kbytes 8 blocks, numbered SRAM0 through SRAM(n).

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.

Reset and Interrupt Handling

The embedded AVR and FPGA core provide 35 different interrupt sources. These interrupts and the separate reset vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits (masks) which must be set (one) together with the I-bit in the status register in order to enable the interrupt.

The lowest addresses in the program memory space must be defined as the Reset and Interrupt vectors. The complete list of vectors is shown in Table 15. The list also determines the priority levels of the different interrupts. The lower the address the higher the priority level. RESET has the highest priority, and next is FPGA_INT0 – the FPGA Interrupt Request 0 etc.

Table 15. Reset and Interrupt Vectors

Vector No. (hex)	Program Address	Source	Interrupt Definition
01	$0000	RESET	Reset Handle: Program Execution Starts Here
02	$0002	FPGA_INT0	FPGA Interrupt0 Handle
03	$0004	EXT_INT0	External Interrupt0 Handle
04	$0006	FPGA_INT1	FPGA Interrupt1 Handle
05	$0008	EXT_INT1	External Interrupt1 Handle
06	$000A	FPGA_INT2	FPGA Interrupt2 Handle
07	$000C	EXT_INT2	External Interrupt2 Handle
08	$000E	FPGA_INT3	FPGA Interrupt3 Handle
09	$0010	EXT_INT3	External Interrupt3 Handle
0A	$0012	TIM2_COMP	Timer/Counter2 Compare Match Interrupt Handle
0B	$0014	TIM2_OVF	Timer/Counter2 Overflow Interrupt Handle
0C	$0016	TIM1_CAPT	Timer/Counter1 Capture Event Interrupt Handle
0D	$0018	TIM1_COMPA	Timer/Counter1 Compare Match A Interrupt Handle
0E	$001A	TIM1_COMPB	Timer/Counter1 Compare Match B Interrupt Handle
0F	$001C	TIM1_OVF	Timer/Counter1 Overflow Interrupt Handle
10	$001E	TIM0_COMP	Timer/Counter0 Compare Match Interrupt Handle
11	$0020	TIM0_OVF	Timer/Counter0 Overflow Interrupt Handle
12	$0022	FPGA_INT4	FPGA Interrupt4 Handle
13	$0024	FPGA_INT5	FPGA Interrupt5 Handle
14	$0026	FPGA_INT6	FPGA Interrupt6 Handle
15	$0028	FPGA_INT7	FPGA Interrupt7 Handle
16	$002A	UART0_RXC	UART0 Receive Complete Interrupt Handle

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

Table 20. AVR I/O Boundary Scan – JTAG Instructions $0/$2

I/O Ports	Description	Bit	
XTAL	Clock In - XTAL1	8[1]	
	Enable Clock - XTAL 1	7	
TOSC	Clock In - TOSC 1	6[1]	
	Enable Clock - TOSC 1	5	
2-wire Serial	Data Out/In - SDA	4[1]	
	Enable Output - SDA	3	
	Clock Out/In - SCL	2[1]	
	Enable Output - SCL	1	
[2]	AVR Reset	0[1]	-> TDO

Notes: 1. Observe-only scan cell.
2. AVR Reset is High (one) if AVRResetn activated (Low) and enabled or the device is in general reset (Resetn or power-on) or configuration download.

Table 21. Bit EXTEST and SAMPLE_PRELOAD

Bit Type	EXTEST	SAMPLE_PRELOAD
Data Out/In - PXn	**Defines value driven if enabled**. Capture-DR grabs signal on pad.	Capture-DR grabs signal from pad if output disabled, or from the AVR if the output drive is enabled.
Enable Output - PXn	**1 = output drive enabled**. Capture-DR grabs output enable scan latch.	Capture-DR grabs output enable from the AVR.
Pull-up - PXn	**1 = pull-up disabled**. Capture-DR grabs pull-up control from the AVR.	Capture-DR grabs pull-up control from the AVR.
Input with Pull-up - INTPn	**Observe only**. Capture-DR grabs signal from pad.	Capture-DR grabs signal from pad.
Data Out - TXn	**Defines value driven if enabled**. Capture-DR grabs signal on pad.	Capture-DR always grabs “0” since Tx input is NC and tied to ground internally.
Enable Output - TXn	**1 = output drive enabled**. Capture-DR grabs output enable scan latch.	Capture-DR grabs output enable from the AVR.
Pull-up - TXn	**1 = pull-up disabled**. Capture-DR grabs pull-up control from the AVR.	Capture-DR grabs pull-up control from the AVR.
Input with Pull-up - RXn	**Observe only**. Capture-DR grabs signal from pad.	Capture-DR grabs signal from pad.
Clock In - XTAL1	**Observe only**. Capture-DR grabs signal from pad.	Capture-DR grabs signal from pad if clock is enabled, “1” if disabled.
Enable Clock - XTAL 1	**1 = clock disabled**. Capture-DR grabs clock enable from the AVR.	Capture-DR grabs enable from the AVR.

Figure 50. Timer/Counter0 Block Diagram

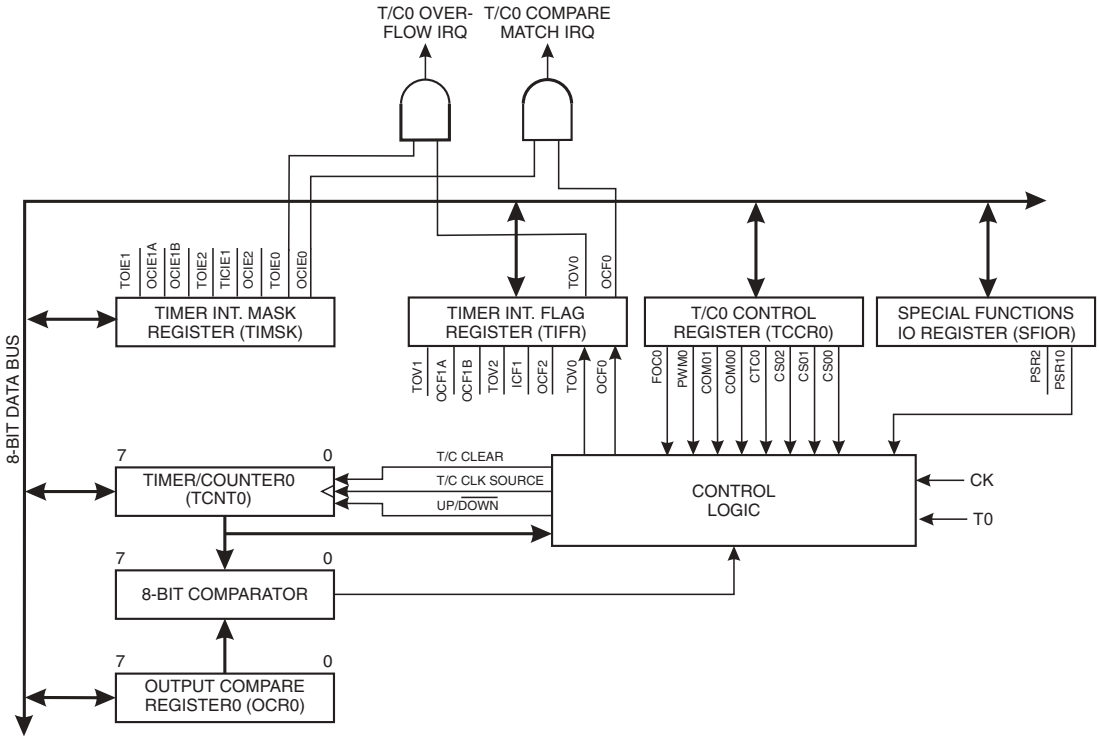
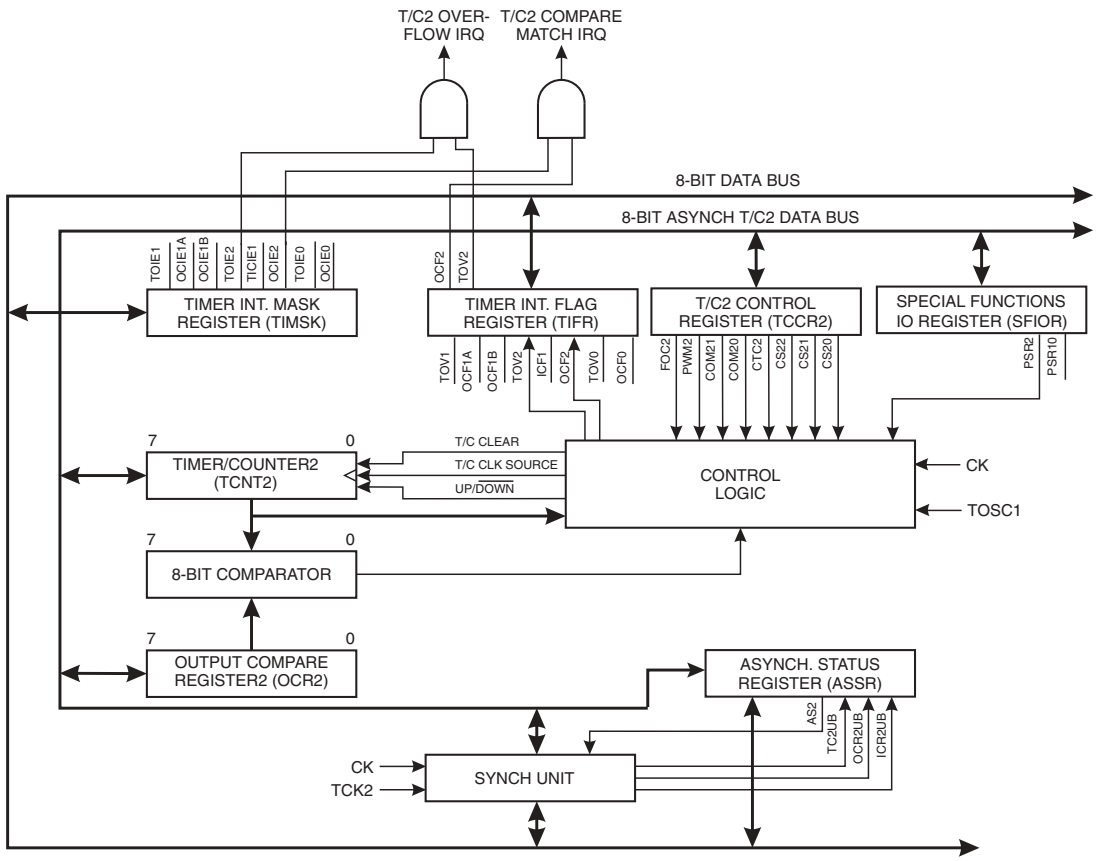


Figure 51. Timer/Counter2 Block Diagram



The 8-bit Timer/Counter0 can select the clock source from CK, prescaled CK, or an external pin.

The 8-bit Timer/Counter2 can select the clock source from CK, prescaled CK or external TOSC1.

Both Timers/Counters can be stopped as described in section “Timer/Counter0 Control Register – TCCR0” on page 88 and “Timer/Counter2 Control Register – TCCR2” on page 88.

The various status flags (overflow and compare match) are found in the Timer/Counter Interrupt Flag Register (TIFR). Control signals are found in the Timer/Counter Control Register (TCCR0 and TCCR2). The interrupt enable/disable settings are found in the Timer/Counter Interrupt Mask Register – TIMSK.

When Timer/Counter0 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 8-bit Timer/Counters feature both a high-resolution and a high-accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities make the Timer/Counter0 useful for lower speed functions or exact-timing functions with infrequent actions.

Timer/Counters 0 and 2 can also be used as 8-bit Pulse Width Modulators (PWM). In this mode, the Timer/Counter and the output compare register serve as a glitch-free, stand-alone PWM with centered pulses. See “Timer/Counter 0 and 2 in PWM Mode” on page 91 for a detailed description on this function.

Timer/Counter0 Control Register – TCCR0

Bit	7	6	5	4	3	2	1	0	
$33 ($53)	**FOC0**	**PWM0**	**COM01**	**COM00**	**CTC0**	**CS02**	**CS01**	**CS00**	**TCCR0**
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	

Timer/Counter2 Control Register – TCCR2

Bit	7	6	5	4	3	2	1	0	
$27 ($47)	**FOC2**	**PWM2**	**COM21**	**COM20**	**CTC2**	**CS22**	**CS21**	**CS20**	**TCCR2**
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 - FOC0/FOC2: Force Output Compare**

Writing a logic 1 to this bit forces a change in the compare match output pin PE1 (Timer/Counter0) and PE3 (Timer/Counter2) according to the values already set in COMn1 and COMn0. If the COMn1 and COMn0 bits are written in the same cycle as FOC0/FOC2, the new settings will not take effect until next compare match or Forced Output Compare match occurs. The Force Output Compare bit can be used to change the output pin without waiting for a compare match in the timer. The automatic action programmed in COMn1 and COMn0 happens as if a Compare Match had occurred, but no interrupt is generated and the Timer/Counters will not be cleared even if CTC0/CTC2 is set. The FOC0/FOC2 bits will always be read as zero. The setting of the FOC0/FOC2 bits has no effect in PWM mode.

- **Bit 6 - PWM0/PWM2: Pulse Width Modulator Enable**

When set (one) this bit enables PWM mode for Timer/Counter0 or Timer/Counter2. This mode is described on page 91.

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.

Table 30. Timer TOP Values and PWM Frequency

CTC1	PWM11	PWM10	PWM Resolution	Timer TOP Value	Frequency
0	0	1	8-bit	$00FF (255)	$f_{TCK1}/510$
0	1	0	9-bit	$01FF (511)	$f_{TCK1}/1022$
0	1	1	10-bit	$03FF(1023)	$f_{TCK1}/2046$
1	0	1	8-bit	$00FF (255)	$f_{TCK1}/256$
1	1	0	9-bit	$01FF (511)	$f_{TCK1}/512$
1	1	1	10-bit	$03FF(1023)	$f_{TCK1}/1024$

Table 31. Compare1 Mode Select in PWM Mode

CTC1(1)	COM1X1(1)	COM1X0(1)	Effect on OCX1
x(2)	0	x(2)	Not connected
0	1	0	Cleared on compare match, up-counting. Set on compare match, down-counting (non-inverted PWM)
0	1	1	Cleared on compare match, down-counting. Set on compare match, up-counting (inverted PWM)
1	1	0	Cleared on compare match, set on overflow
1	1	1	Set on compare match, set on overflow

Notes: 1. X = A or B
2. x = Don't care

In the PWM mode, the 8, 9 or 10 least significant OCR1A/OCR1B bits (depends of resolution), when written, are transferred to a temporary location. They are latched when Timer/Counter1 reaches the value TOP. This prevents the occurrence of odd-length PWM pulses (glitches) in the event of an unsynchronized OCR1A/OCR1B write. See Figure 56 and Figure 57 for an example in each mode.

Multiplier

The multiplier is capable of multiplying two 8-bit numbers, giving a 16-bit result using only two clock cycles. The multiplier can handle both signed and unsigned integer and fractional numbers without speed or code size penalty. Below are some examples of using the multiplier for 8-bit arithmetic.

To be able to use the multiplier, six new instructions are added to the AVR instruction set. These are:

- MUL, multiplication of unsigned integers
- MULS, multiplication of signed integers
- MULSU, multiplication of a signed integer with an unsigned integer
- FMUL, multiplication of unsigned fractional numbers
- FMULS, multiplication of signed fractional numbers
- FMULSU, multiplication of a signed fractional number and with an unsigned fractional number

The MULSU and FMULSU instructions are included to improve the speed and code density for multiplication of 16-bit operands. The second section will show examples of how to efficiently use the multiplier for 16-bit arithmetic.

The component that makes a dedicated digital signal processor (DSP) specially suitable for signal processing is the multiply-accumulate (MAC) unit. This unit is functionally equivalent to a multiplier directly connected to an arithmetic logic unit (ALU). The FPSLIC-based AVR Core is designed to give FPSLIC the ability to effectively perform the same multiply-accumulate operation.

The multiply-accumulate operation (sometimes referred to as *multiply-add operation*) has one critical drawback. When adding multiple values to one result variable, even when adding positive and negative values to some extent, cancel each other; the risk of the result variable to overrun its limits becomes evident, i.e. if adding 1 to a signed byte variable that contains the value +127, the result will be -128 instead of +128. One solution often used to solve this problem is to introduce fractional numbers, i.e. numbers that are less than 1 and greater than or equal to -1. Some issues regarding the use of fractional numbers are discussed.

A list of all implementations with key performance specifications is given in Table 34.

Table 34. Performance Summary

8-bit x 8-bit Routines:	Word (Cycles)
Unsigned Multiply 8 x 8 = 16 bits	1 (2)
Signed Multiply 8 x 8 = 16 bits	1 (2)
Fractional Signed/Unsigned Multiply 8 x 8 = 16 bits	1 (2)
Fractional Signed Multiply-accumulate 8 x 8 + = 16 bits	3 (4)
16-bit x 16-bit Routines:	**Word (Cycles)**
Signed/Unsigned Multiply 16 x 16 = 32 bits	6 (9)
UnSigned Multiply 16 x 16 = 32 bits	13 (17)
Signed Multiply 16 x 16 = 32 bits	15 (19)
Signed Multiply-accumulate 16 x 16 + = 32 bits	19 (23)
Fractional Signed Multiply 16 x 16 = 32 bits	16 (20)
Fractional Signed Multiply-accumulate 16 x 16 + = 32 bits	21 (25)

muls16x16_32

Description

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

Usage

R19:R18:R17:R16 = R23:R22 • R21:R20

Statistics

Cycles: 19 + ret

Words: 15 + ret

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

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

```
muls16x16_32:
   clr   r2
   muls  r23, r21             ; (signed)ah * (signed)bh
   movw  r19:r18, r1:r0
   mul   r22, r20             ; al * bl
   movw  r17:r16, r1:r0
   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
```

mac16x16_32

Description

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

Usage

R19:R18:R17:R16 += R23:R22 • R21:R20

Statistics

Cycles: 23 + ret

Words: 19 + ret

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

```
mac16x16_32:                  ; Register Usage Optimized
   clr   r2

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

   mul   r22, r20             ; al * bl
   add   r16, r0
   adc   r17, r1
   adc   r18, r2
   adc   r19, r2
```

UARTs

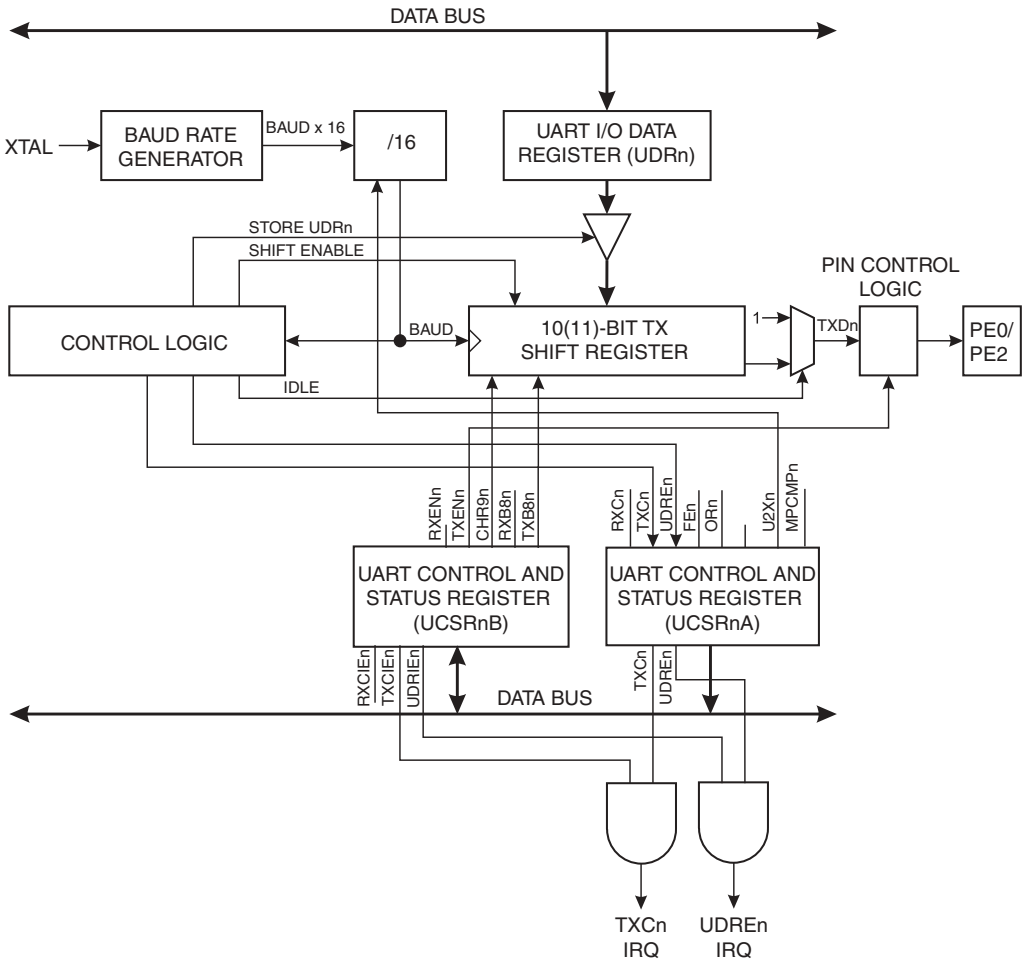
The FPSLIC features two full duplex (separate receive and transmit registers) Universal Asynchronous Receiver and Transmitter (UART). The main features are:

- Baud-rate Generator Generates any Baud-rate
- High Baud-rates at Low XTAL Frequencies
- 8 or 9 Bits Data
- Noise Filtering
- Overrun Detection
- Framing Error Detection
- False Start Bit Detection
- Three Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete
- Multi-processor Communication Mode
- Double Speed UART Mode

Data Transmission

A block schematic of the UART transmitter is shown in Figure 64. The two UARTs are identical and the functionality is described in general for the two UARTs.

Figure 64. UART Transmitter(1)



Note: 1. n = 0, 1

UART0 Baud-rate Register Low Byte – UBRR0

Bit	7	6	5	4	3	2	1	0	
$09 ($29)	MSB							LSB	UBRR0
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	

UART1 Baud-rate Register Low Byte – UBRR1

Bit	7	6	5	4	3	2	1	0	
$00 ($20)	MSB							LSB	UBRR1
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	

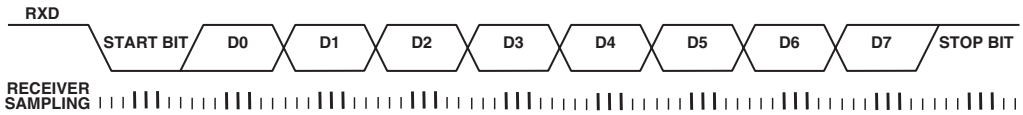
UBRRn stores the 8 least significant bits of the UART baud-rate register.

Double Speed Transmission

The FPSLIC provides a separate UART mode that allows the user to double the communication speed. By setting the U2X bit in UART Control and Status Register UCSRnA, the UART speed will be doubled. The data reception will differ slightly from normal mode. Since the speed is doubled, the receiver front-end logic samples the signals on the RXDn pin at a frequency 8 times the baud-rate. While the line is idle, one single sample of logic 0 will be interpreted as the falling edge of a start bit, and the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample. Following the 1-to-0 transition, the receiver samples the RXDn pin at samples 4, 5 and 6. If two or more of these three samples are found to be logic 1s, the start bit is rejected as a noise spike and the receiver starts looking for the next 1-to-0 transition.

If however, a valid start bit is detected, sampling of the data bits following the start bit is performed. These bits are also sampled at samples 4, 5 and 6. The logical value found in at least two of the three samples is taken as the bit value. All bits are shifted into the transmitter shift register as they are sampled. Sampling of an incoming character is shown in Figure 67.

Figure 67. Sampling Received Data when the Transmission Speed is Doubled



The Baud-rate Generator in Double UART Speed Mode

Note that the baud-rate equation is different from the equation[1] at page 126 when the UART speed is doubled:

$$BAUD = \frac{f_{CK}}{8(UBR + 1)}$$

- BAUD = Baud-rate
- f_{CK}= Crystal Clock Frequency
- UBR = Contents of the UBRRHI and UBRRn Registers, (0 - 4095)

Note: 1. This equation is only valid when the UART transmission speed is doubled.

For standard crystal frequencies, the most commonly used baud-rates can be generated by using the UBR settings in Table 36. UBR values which yield an actual baud-rate differing less than 1.5% from the target baud-rate, are bold in the table. However since the number of samples are reduced and the system clock might have some variance (this applies especially when using resonators), it is recommended that the baud-rate error is less than 0.5%. See Table 37 for the UBR settings at various crystal frequencies in double UART speed mode.

- **Bit 4 - TWSTO: 2-wire Serial Bus STOP Condition Flag**

TWSTO is a stop condition flag. In Master mode, setting the TWSTO bit in the control register will generate a STOP condition on the 2-wire Serial Bus. When the STOP condition is executed on the bus, the TWSTO bit is cleared automatically. In Slave mode, setting the TWSTO bit can be used to recover from an error condition. No stop condition is generated on the bus then, but the 2-wire Serial Interface returns to a well-defined unaddressed Slave mode.

- **Bit 3 - TWWC: 2-wire Serial Write Collision Flag**

Set when attempting to write to the 2-wire Serial Data Register – TWDR when TWINT is Low. This flag is updated at each attempt to write the TWDR register.

- **Bit 2 - TWEN: 2-wire Serial Interface Enable Flag**

The TWEN bit enables 2-wire serial operation. If this flag is cleared (zero), the bus outputs SDA and SCL are set to high impedance state and the input signals are ignored. The interface is activated by setting this flag (one).

- **Bit 1 - Res: Reserved Bit**

This bit is reserved in the AT94K and will always read as zero.

- **Bit 0 - TWIE: 2-wire Serial Interrupt Enable**

When this bit is enabled and the I-bit in SREG is set, the 2-wire Serial Interrupt will be activated for as long as the TWINT flag is High.

The TWCR is used to control the operation of the 2-wire Serial Interface. It is used to enable the 2-wire Serial Interface, to initiate a Master access, to generate a receiver acknowledge, to generate a stop condition, and control halting of the bus while the data to be written to the bus are written to the TWDR. It also indicates a write collision if data is attempted written to TWDR while the register is inaccessible.

The 2-wire Serial Status Register – TWSR

Bit	7	6	5	4	3	2	1	0	
$1D ($3D)	**TWS7**	**TWS6**	**TWS5**	**TWS4**	**TWS3**	**-**	**-**	**-**	**TWSR**
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	1	1	1	1	1	0	0	0	

- **Bits 7..3 - TWS: 2-wire Serial Status**

These 5 bits reflect the status of the 2-wire Serial Logic and the 2-wire Serial Bus.

- **Bits 2..0 - Res: Reserved Bits**

These bits are reserved in the AT94K and will always read as zero

TWSR is read only. It contains a status code which reflects the status of the 2-wire Serial Logic and the 2-wire Serial Bus. There are 26 possible status codes. When TWSR contains $F8, no relevant state information is available and no 2-wire Serial Interrupt is requested. A valid status code is available in TWSR one CPU clock cycle after the 2-wire Serial Interrupt flag (TWINT) is set by the hardware and is valid until one CPU clock cycle after TWINT is cleared by software. Table 41 to Table 45 give the status information for the various modes.

detailed in Table 41. The data must be loaded when TWINT is High only. If not, the access will be discarded, and the Write Collision bit, TWWC, will be set in the TWCR register. This scheme is repeated until a STOP condition is transmitted by writing a logic 1 to the TWSTO bit in the TWCR register.

After a repeated START condition (state $10) the 2-wire Serial Interface may switch to the Master Receiver mode by loading TWDR with SLA+R.

Master Receiver Mode

In the Master Receiver mode, a number of data bytes are received from a Slave Transmitter, see Figure 72. The transfer is initialized as in the Master Transmitter mode. When the START condition has been transmitted, the TWINT flag is set by the hardware. The software must then load TWDR with the 7-bit Slave address and the data direction bit (SLA+R). The 2-wire Serial Interrupt flag must then be cleared by software before the 2-wire Serial Transfer can continue.

When the Slave address and the direction bit have been transmitted and an acknowledgment bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Status codes $40, $48, or $38 apply to Master mode, and status codes $68, $78, or $B0 apply to Slave mode. The appropriate action to be taken for each of these status codes is detailed in Table 42. Received data can be read from the TWDR register when the TWINT flag is set High by the hardware. This scheme is repeated until a STOP condition is transmitted by writing a logic 1 to the TWSTO bit in the TWCR register.

After a repeated START condition (state $10), the 2-wire Serial Interface may switch to the Master Transmitter mode by loading TWDR with SLA+W.

Slave Receiver Mode

In the Slave Receiver mode, a number of data bytes are received from a Master Transmitter, see Figure 73. To initiate the Slave Receiver mode, TWAR and TWCR must be initialized as follows:

Table 39. TWAR: Slave Receiver Mode Initialization

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
value	Device's own Slave address							

The upper 7 bits are the address to which the 2-wire Serial Interface will respond when addressed by a Master. If the LSB is set, the 2-wire Serial Interface will respond to the general call address ($00), otherwise it will ignore the general call address.

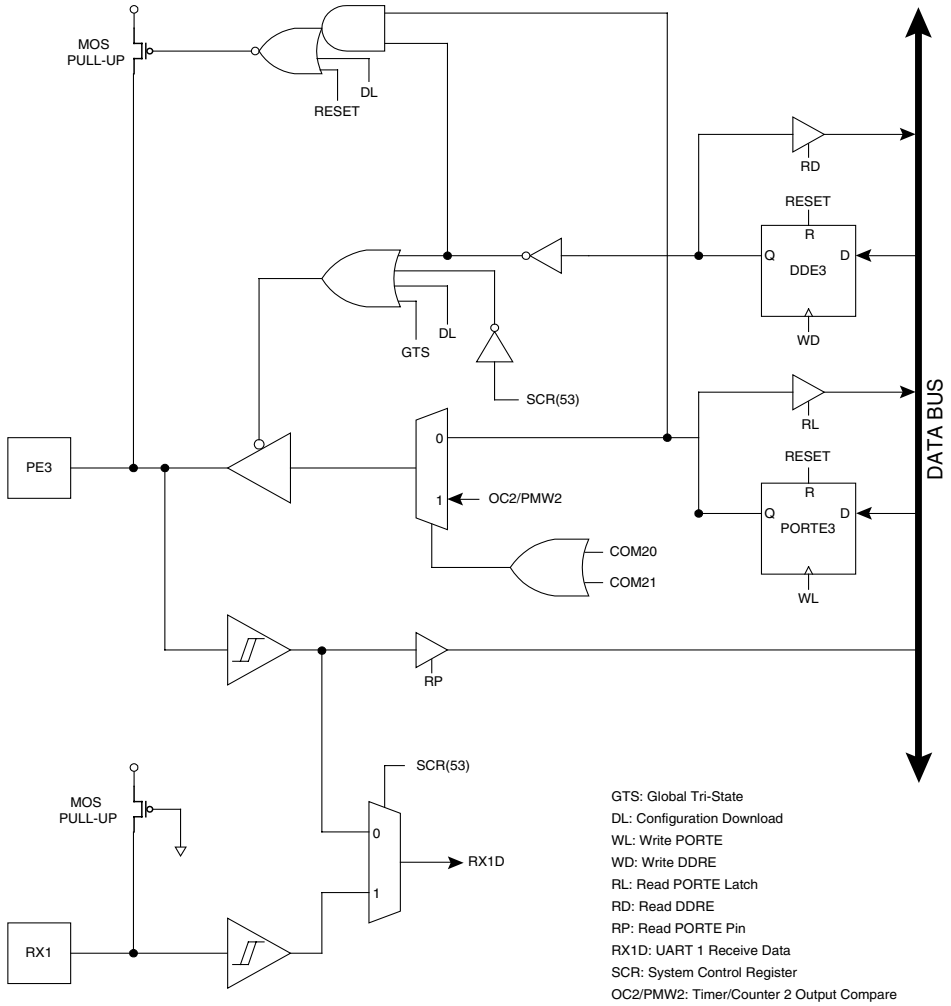
Table 40. TWCR: Slave Receiver Mode Initialization

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
value	0	1	0	0	0	1	0	X

TWEN must be set to enable the 2-wire Serial Interface. The TWEA bit must be set to enable the acknowledgment of the device's own Slave address or the general call address. TWSTA and TWSTO must be cleared.

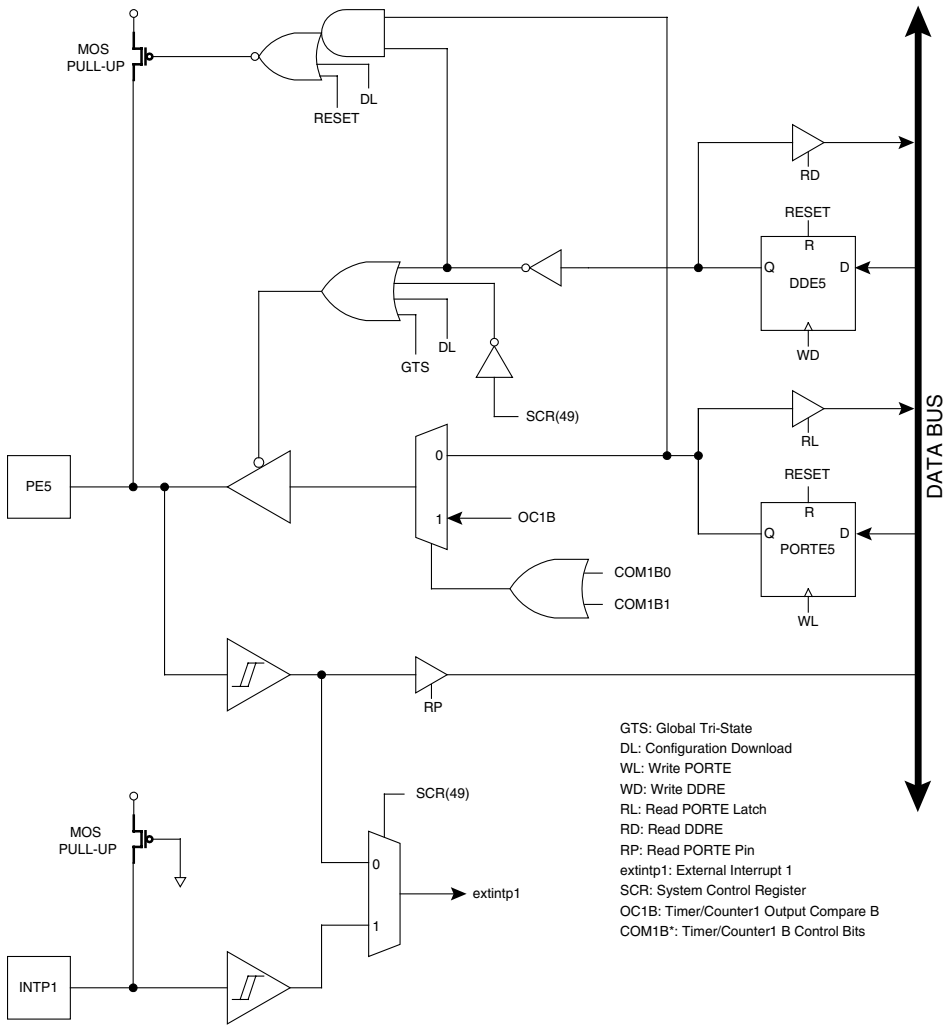
When TWAR and TWCR have been initialized, the 2-wire Serial Interface waits until it is addressed by its own Slave address (or the general call address if enabled) followed by the data direction bit which must be "0" (write) for the 2-wire Serial Interface to operate in the Slave Receiver mode. After its own Slave address and the write bit have been received, the 2-wire Serial Interrupt flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in Table 43. The Slave Receiver mode may also be entered if arbitration is lost while the 2-wire Serial Interface is in the Master mode (see states $68 and $78).

PortE Schematic Diagram (Pin PE3)

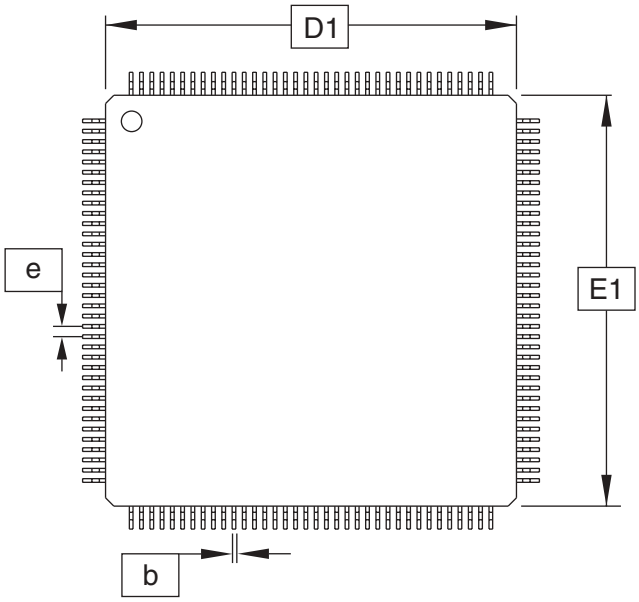


GTS: Global Tri-State
DL: Configuration Download
WL: Write PORTE
WD: Write DDRE
RL: Read PORTE Latch
RD: Read DDRE
RP: Read PORTE Pin
RX1D: UART 1 Receive Data
SCR: System Control Register
OC2/PMW2: Timer/Counter 2 Output Compare
COM2*: Timer/Counter2 Control Bits

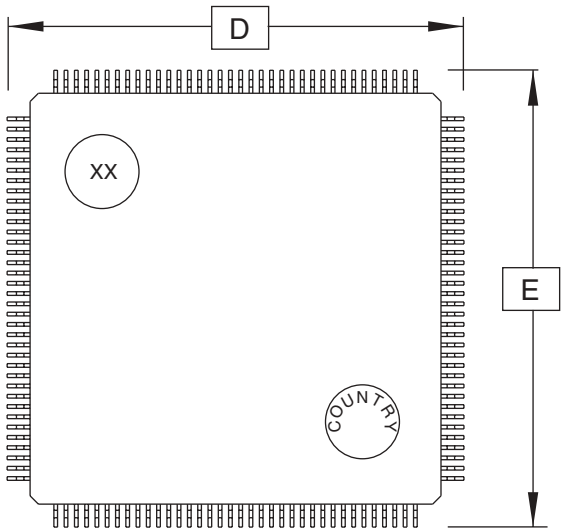
Figure 80. PortE Schematic Diagram (Pin PE5)



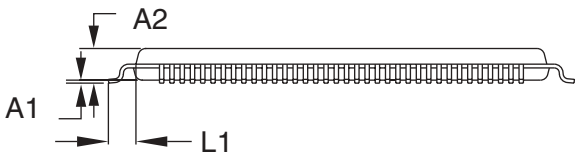
144L1 – LQFP



Top View



Bottom View



Side View

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