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	20K-32K
FPGA SRAM	4kb
EEPROM Size	-
Data SRAM Bytes	4K ~ 16K
FPGA Core Cells	576
FPGA Gates	10K
FPGA Registers	846
Voltage - Supply	3V ~ 3.6V
Mounting Type	Surface Mount
Operating Temperature	0°C ~ 70°C
Package / Case	84-LCC (J-Lead)
Supplier Device Package	84-PLCC (29.31x29.31)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at94k10al-25ajc

The FPGA clocks from the AVR are effected differently in the various sleep modes of the AVR, see Table 3.

The source clock into the FPGA GCK5 and GCK6 will determine what happens during the various power-down modes of the AVR.

If the XTAL clock input is used as an FPGA clock (GCK5 or GCK6) in Idle mode, it will still be running. In Power-down/save mode the XTAL clock input will be off.

If the TOSC clock input is used as an FPGA clock (GCK6) in Idle mode, it will still be running in Power-save mode but will be off in Power-down mode.

If the Watchdog Timer is used as an FPGA clock (GCK6) and was enabled in the AVR, it will be running in all sleep modes.

Table 3. Clock Activity in Various Modes

Mode	Clock Source	GCK5	GCK6
Idle	XTAL	Active	Active
	TOSC	Not Available	Active
	WDT	Not Available	Active
Power-save	XTAL	Inactive	Inactive
	TOSC	Not Available	Active
	WDT	Not Available	Active
Power-down	XTAL	Inactive	Inactive
	TOSC	Not Available	Inactive
	WDT	Not Available	Active

Figure 17. Corner I/Os

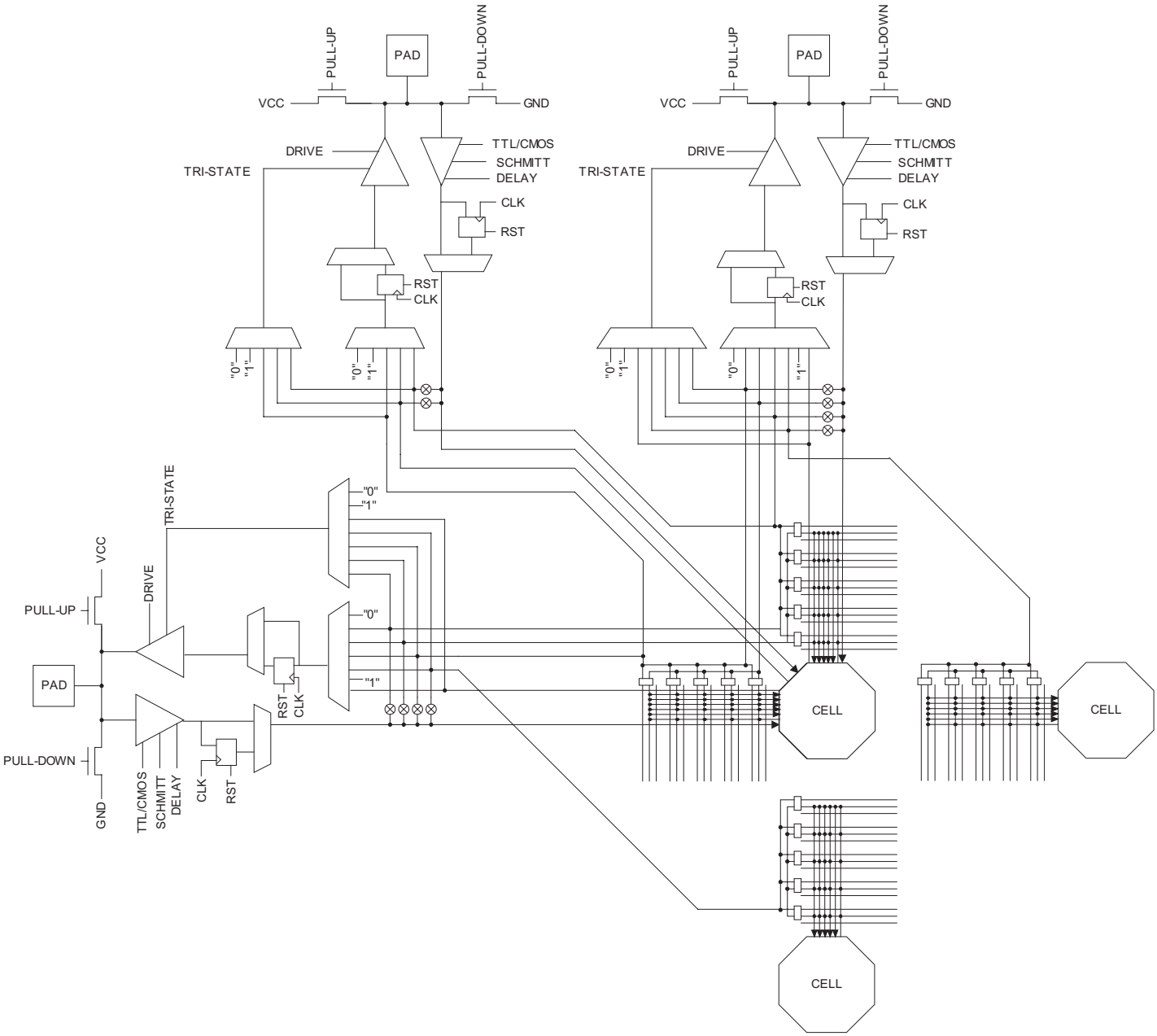


Table 7. Summary Table for AVR and FPGA SRAM Addressing (Continued)

SRAM	FPGA and AVR DBG Address Range	AVR Data Address Range	AVR PC Address Range
05[1]	$2800 - $2FFF	$2800 - $2FFF	$3000 - $37FF (MS Byte)
06[1]	$3000 - $37FF	$3000 - $37FF	$2800 - $2FFF (LS Byte)
07[1]	$3800 - $3FFF	$3800 - $3FFF	$2800 - $2FFF (MS Byte)
08	$4000 - $47FF		$2000 - $27FF (LS Byte)
09	$4800 - $4FFF		$2000 - $27FF (MS Byte)
10	$5000 - $57FF		$1800 - $1FFF (LS Byte)
11	$5800 - $5FFF		$1800 - $1FFF (MS Byte)
12	$6000 - $67FF		$1000 - $17FF (LS Byte)
13	$6800 - $6FFF		$1000 - $17FF (MS Byte)
14	$7000 - $77FF		$0800 - $0FFF (LS Byte)
15	$7800 - $7FFF		$0800 - $0FFF (MS Byte)
16	$8000 - $87FF		$0000 - $07FF (LS Byte)
17 = n	$8800 - $8FFF		$0000 - $07FF (MS Byte)

Note: 1. Whether these SRAMs are “Data” or “Program” depends on the SCR40 and SCR41 values.

Example: Frame (and AVR debug mode) write of instructions to associated AVR PC addresses, see Table 8 and Table 9.

Table 8. AVR PC Addresses

AVR PC	Instruction
0FFE	9B28
0FFF	CFFE
1000	B300
1001	9A39

Table 9. Frame Addresses

Frame Address	Frame Data
77FE	28
77FF	FE
6000	00
6001	39
7FFE	9B
7FFF	CF
6800	B3
6801	9A

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 1 = Disable Cascading 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 1 = Check Function Disabled 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 1 = Memory Lockout Enabled 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

Table 11. FPSLIC System Control Register

Bit	Description
SCR56	0 = Disable XTAL Pin ($R_{feedback}$) 1 = Enable XTAL Pin ($R_{feedback}$)
SCR57	0 = Disable TOSC2 Pin ($R_{feedback}$) 1 = Enable TOSC2 Pin ($R_{feedback}$)
SCR58 - SCR59	Reserved
SCR60 - SCR61	SCR61 = 0, SCR60 = 0 “1” SCR61 = 0, SCR60 = 1 AVR System Clock SCR61 = 1, SCR60 = 0 Timer Oscillator Clock (TOSC1)(1) SCR61 = 1, SCR60 = 1 Watchdog Clock Global Clock 6 mux select (set by using the AT94K Device Options in System Designer). Note: 1. The AS2 bit must be set in the ASSR register.
SCR62	0 = Disable CacheLogic Writes to FPGA by AVR 1 = Enable CacheLogic Writes to FPGA by AVR
SCR63	0 = Disable Access (Read and Write) to SRAM by FPGA 1 = Enable Access (Read and Write) to SRAM by FPGA

Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clock
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half-carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half-carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2
Data Transfer Instructions					
MOV	Rd, Rr	Copy Register	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Pair	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LDS	Rd, k	Load Direct from Data Space	Rd ← (k)	None	2
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Increment	Rd ← (X), X ← X + 1	None	2
LD	Rd, -X	Load Indirect and Pre-Decrement	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, -Y	Load Indirect and Pre-Decrement	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd, Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Decrement	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
STS	k, Rr	Store Direct to Data Space	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Increment	(X) ← Rr, X ← X + 1	None	2
ST	-X, Rr	Store Indirect and Pre-Decrement	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Increment	(Y) ← Rr, Y ← Y + 1	None	2

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

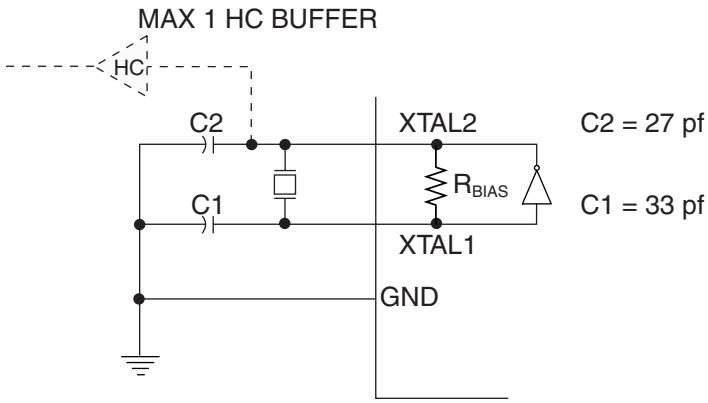
SDA 2-wire serial input/output data

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.

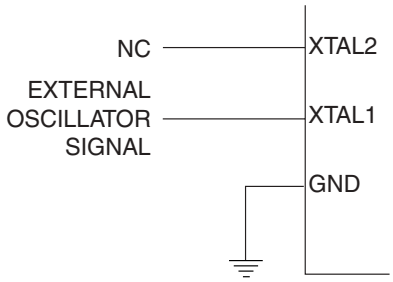
Figure 24. Oscillator Connections



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.

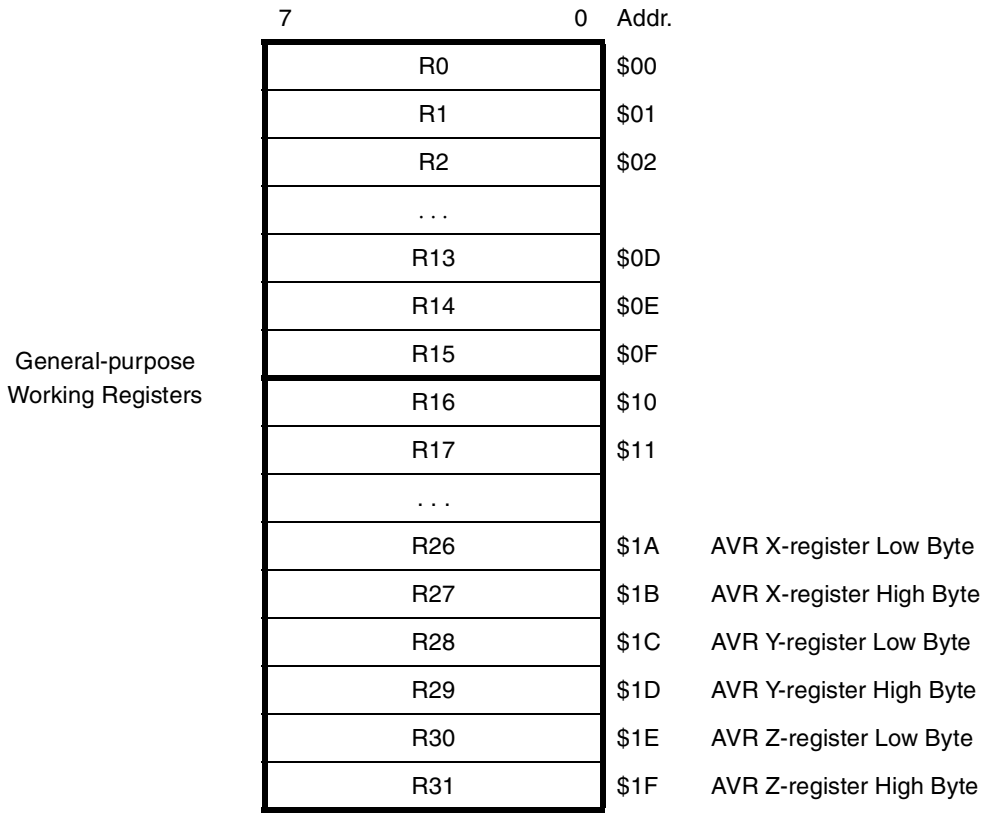
Figure 25. External Clock Drive Configuration



General-purpose Register File

Figure 28 shows the structure of the 32 x 8 general-purpose working registers in the CPU.

Figure 28. AVR CPU General-purpose Working Registers



All the register operating instructions in the instruction set have direct- and single-cycle access to all registers. The only exception is the five constant arithmetic and logic instructions SBCI, SUBI, CPI, ANDI and ORI between a constant and a register and the LDI instruction for load-immediate constant data. These instructions apply to the second half of the registers in the register file – R16..R31. The general SBC, SUB, CP, AND and OR and all other operations between two registers or on a single-register apply to the entire register file.

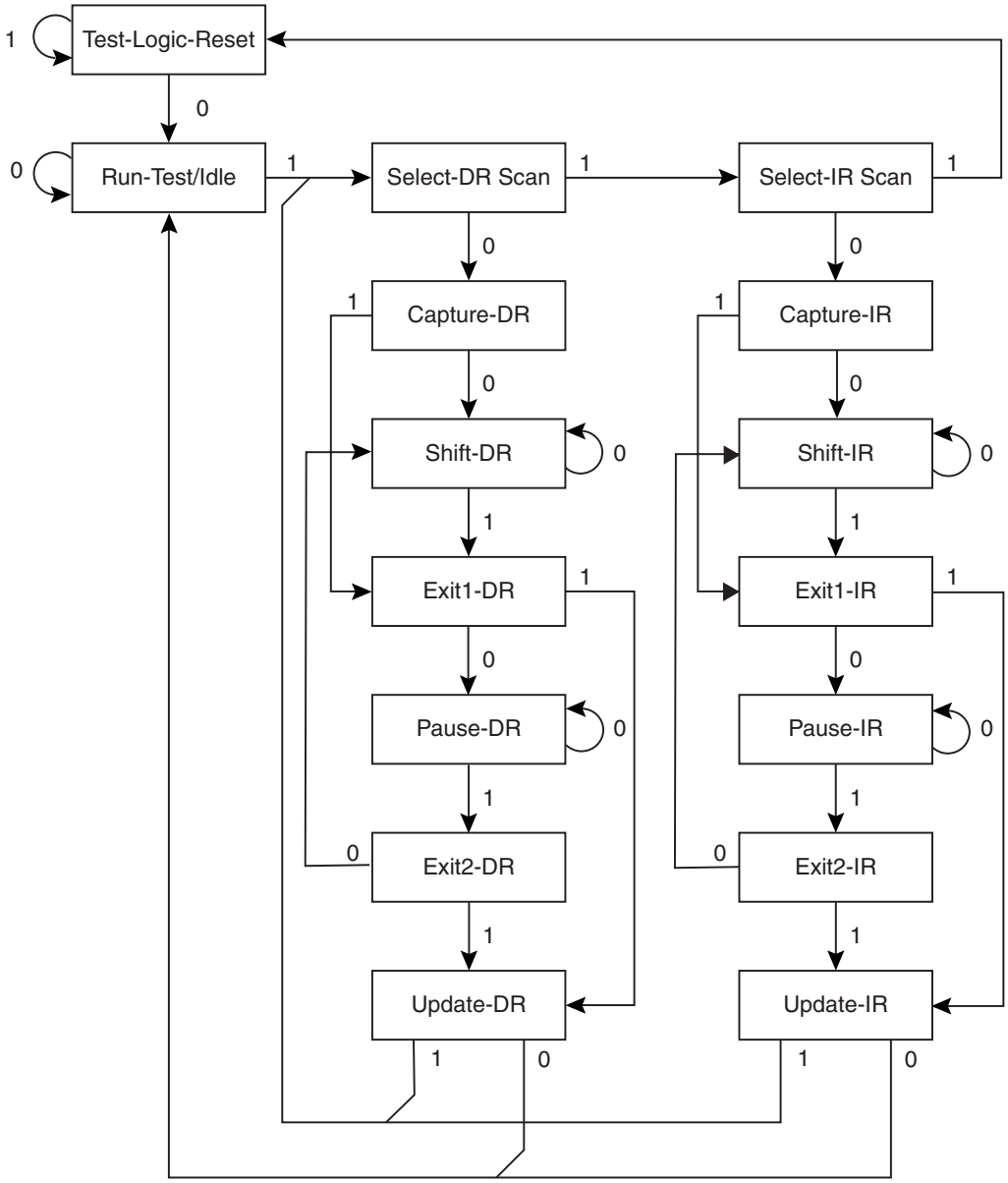
As shown in Figure 28 each register is also assigned a data memory address, mapping the registers directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X, Y and Z registers can be set to index any register in the file.

The 4 to 16 Kbytes of data SRAM, as configured during FPSLIC download, are available for general data are implemented starting at address $0060 as follows:

4 Kbytes	$0060 : $0FFF
8 Kbytes	$0060 : $1FFF
12 Kbytes	$0060 : $2FFF
16 Kbytes	$0060 : $3FFF

Addresses beyond the maximum amount of data SRAM are unavailable for write or read and will return unknown data if accessed. Ghost memory is not implemented.

Figure 40. TAP Controller State Diagram



TAP Controller

The TAP controller is a 16-state finite state machine that controls the operation of the Boundary-Scan circuitry and On-Chip Debug system. The state transitions depicted in Figure 40 depend on the signal present on TMS (shown adjacent to each state transition) at the time of the rising edge at TCK. The initial state after a Power-On Reset is Test-Logic-Reset.

As a definition in this document, the LSB is shifted in and out first for all shift registers.

The mechanisms for reading TCNT2, OCR2 and TCCR2 are different. When reading TCNT2, the actual timer value is read. When reading OCR2 or TCCR2, the value in the temporary storage register is read.

Asynchronous Operation of Timer/Counter2

When Timer/Counter2 operates asynchronously, some considerations must be taken:

- When switching between asynchronous and synchronous clocking of Timer/Counter2, the timer registers TCNT2, OCR2 and TCCR2 might get corrupted. A safe procedure for switching the clock source is:
  1. Disable the Timer/Counter2 interrupts by clearing OCIE2 and TOIE2.
  2. Select clock source by setting AS2 as appropriate.
  3. Write new values to TCNT2, OCR2 and TCCR2.
  4. To switch to asynchronous operation: Wait for TCN2UB, OCR2UB, and TCR2UB.
  5. Enable interrupts, if needed.
- The oscillator is optimized for use with a 32.768 kHz watch crystal. An external clock signal applied to this pin goes through the same amplifier having a bandwidth of 256 kHz. The external clock signal should therefore be in the interval
  0 Hz – 1 MHz. The frequency of the clock signal applied to the TOSC1 pin must be lower than one fourth of the CPU main clock frequency.
- When writing to one of the registers TCNT2, OCR2, or TCCR2, the value is transferred to a temporary register, and latched after two positive edges on TOSC1. The user should not write a new value before the contents of the temporary register have been transferred to its destination. Each of the three mentioned registers have their individual temporary register, which means that, e.g., writing to TCNT2 does not disturb an OCR2 write in progress. To detect that a transfer to the destination register has taken place, an Asynchronous Status Register – ASSR has been implemented.
- When entering Power-save mode after having written to TCNT2, OCR2, or TCCR2, the user must wait until the written register has been updated if Timer/Counter2 is used to wake-up the device. Otherwise, the MCU will go to sleep before the changes have had any effect. This is extremely important if the Output Compare2 interrupt is used to wake-up the device; Output compare is disabled during write to OCR2 or TCNT2. If the write cycle is not finished (i.e., the MCU enters Sleep mode before the OCR2UB bit returns to zero), the device will never get a compare match and the MCU will not wake-up.
- If Timer/Counter2 is used to wake-up the device from Power-save mode, precautions must be taken if the user wants to re-enter Power-save mode: The interrupt logic needs one TOSC1 cycle to be reset. If the time between wake-up and reentering Power-save mode is less than one TOSC1 cycle, the interrupt will not occur and the device will fail to wake up. If the user is in doubt whether the time before re-entering power-save is sufficient, the following algorithm can be used to ensure that one TOSC1 cycle has elapsed:
  1. Write a value to TCCR2, TCNT2, or OCR2.
  2. Wait until the corresponding Update Busy flag in ASSR returns to zero.
  3. Enter Power-save mode.
- When asynchronous operation is selected, the 32.768 kHz oscillator for Timer/Counter2 is always running, except in Power-down mode. After a power-up reset or wake-up from power-down, the user should be aware of the fact that this oscillator might take as long as one second to stabilize. Therefore, the contents of all Timer2 registers must be considered lost after a wake-up from power-down, due to the unstable clock signal. The user is advised to wait for at least one second before using Timer/Counter2 after power-up or wake-up from power-down.
- Description of wake-up from Power-save mode when the timer is clocked asynchronously. When the interrupt condition is met, the wake-up process is started on the following cycle

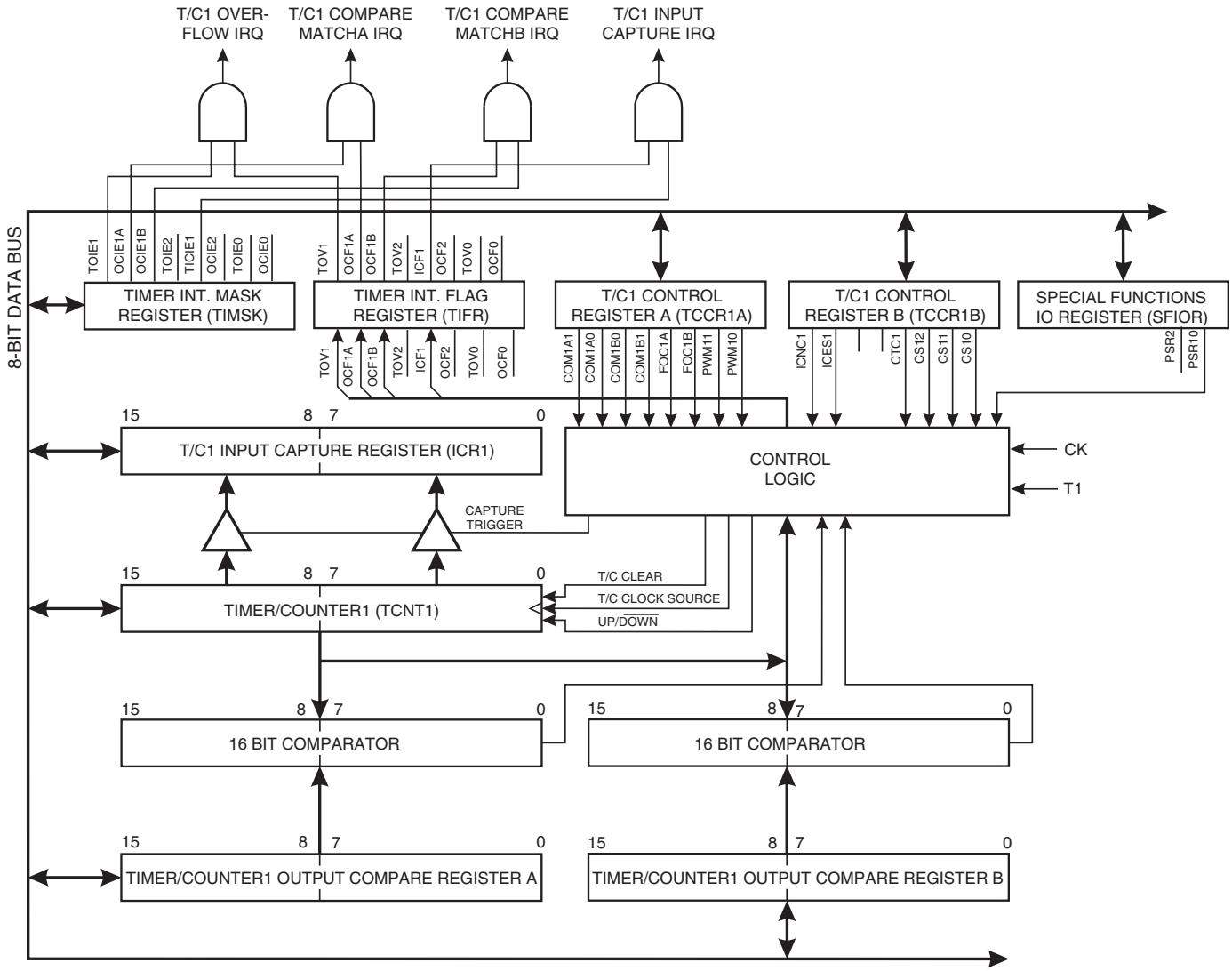
of the timer clock, that is, the timer is always advanced by at least one before the processor can read the counter value. The interrupt flags are updated three processor cycles after the processor clock has started. During these cycles, the processor executes instructions, but the interrupt condition is not readable, and the interrupt routine has not started yet.

- During asynchronous operation, the synchronization of the interrupt flags for the asynchronous timer takes three processor cycles plus one timer cycle. The timer is therefore advanced by at least one before the processor can read the timer value causing the setting of the interrupt flag. The output compare pin is changed on the timer clock and is not synchronized to the processor clock.

Timer/Counter1

Figure 54 shows the block diagram for Timer/Counter1.

Figure 54. Timer/Counter1 Block Diagram



- **Bits 2,1,0 - CS12, CS11, CS10: Clock Select1, Bits 2, 1 and 0**

The Clock Select1 bits 2,1 and 0 define the prescaling source of Timer/Counter1.

Table 29. Clock 1 Prescale Select

CS12	CS11	CS10	Description
0	0	0	Stop, the Timer/Counter1 is stopped
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	External pin PE4 (T1), falling edge
1	1	1	External pin PE4 (T1), rising edge

The Stop condition provides a Timer Enable/Disable function. The CK down-divided modes are scaled directly from the CK oscillator clock. If the external pin modes are used for Timer/Counter1, transitions on PE4/(T1) will clock the counter even if the pin is configured as an output. This feature can give the user SW control of the counting.

Timer/Counter1 Register – TCNT1H AND TCNT1L

Bit	15	14	13	12	11	10	9	8	
$2D ($4D)	MSB								TCNT1H
$2C ($4C)								LSB	TCNT1L
	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	

This 16-bit register contains the prescaled value of the 16-bit Timer/Counter1. To ensure that both the High and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary register (TEMP). This temporary register is also used when accessing OCR1A, OCR1B 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.

TCNT1 Timer/Counter1 Write

When the CPU writes to the high byte TCNT1H, the written data is placed in the TEMP register. Next, when the CPU writes the low byte TCNT1L, this byte of data is combined with the byte data in the TEMP register, and all 16 bits are written to the TCNT1 Timer/Counter1 register simultaneously. Consequently, the high byte TCNT1H must be accessed first for a full 16-bit register write operation.

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)

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.

To convert a negative fractional number, first add 2 to the number and then use the same algorithm as already shown.

16-bit fractional numbers use a format similar to that of 8-bit fractional numbers; the high 8 bits have the same format as the 8-bit format. The low 8 bits are only an increase of accuracy of the 8-bit format; while the 8-bit format has an accuracy of $\pm\ 2^{-8}$, the16-bit format has an accuracy of $\pm\ 2^{-16}$. Then again, the 32-bit fractional numbers are an increase of accuracy to the 16-bit fractional numbers. Note the important difference between integers and fractional numbers when extra byte(s) are used to store the number: while the accuracy of the numbers is increased when fractional numbers are used, the range of numbers that may be represented is extended when integers are used.

As mentioned earlier, using signed fractional numbers in the range [-1, 1> has one main advantage to integers: when multiplying two numbers in the range [-1, 1>, the result will be in the range [-1, 1], and an approximation (the highest byte(s)) of the result may be stored in the same number of bytes as the factors, with one exception: when both factors are -1, the product should be 1, but since the number 1 cannot be represented using this number format, the FMULS instruction will instead place the number -1 in R1:R0. The user should therefore assure that at least one of the operands is not -1 when using the FMULS instruction. The 16-bit x 16-bit fractional multiply also has this restriction.

Example 5 – Basic Usage 8-bit x 8-bit = 16-bit Signed Fractional Multiply

This example shows an assembly code that reads the port E input value and multiplies this value with a fractional constant (-0.625) before storing the result in register pair R17:R16.

```
in    r16,PINE      ; Read pin values
ldi   r17,$B0       ; Load -0.625 into r17
fmuls r16,r17       ; r1:r0 = r17 * r16
movw  r17:r16,r1:r0; Move the result to the r17:r16
                    ; register pair
```

Note that the usage of the FMULS (and FMUL) instructions is very similar to the usage of the MULS and MUL instructions.

Example 6 – Multiply-accumulate Operation

The example below uses data from the ADC. The ADC should be configured so that the format of the ADC result is compatible with the fractional two's complement format. For the ATmega83/163, this means that the ADLAR bit in the ADMUX I/O register is set and a differential channel is used. The ADC result is normalized to one.

```
ldi r23,$62         ; Load highbyte of
                    ; fraction 0.771484375
ldi r22,$C0         ; Load lowbyte of
                    ; fraction 0.771484375
in  r20,ADCL        ; Get lowbyte of ADC conversion
in  r21,ADCH        ; Get highbyte of ADC conversion
callfmac16x16_32    ;Call routine for signed fractional
                    ; multiply accumulate
```

The registers R19:R18:R17:R16 will be incremented with the result of the multiplication of 0.771484375 with the ADC conversion result. In this example, the ADC result is treated as a signed fraction number. We could also treat it as a signed integer and call it "mac16x16_32" instead of "fmac16x16_32". In this case, the 0.771484375 should be replaced with an integer.

UARTs

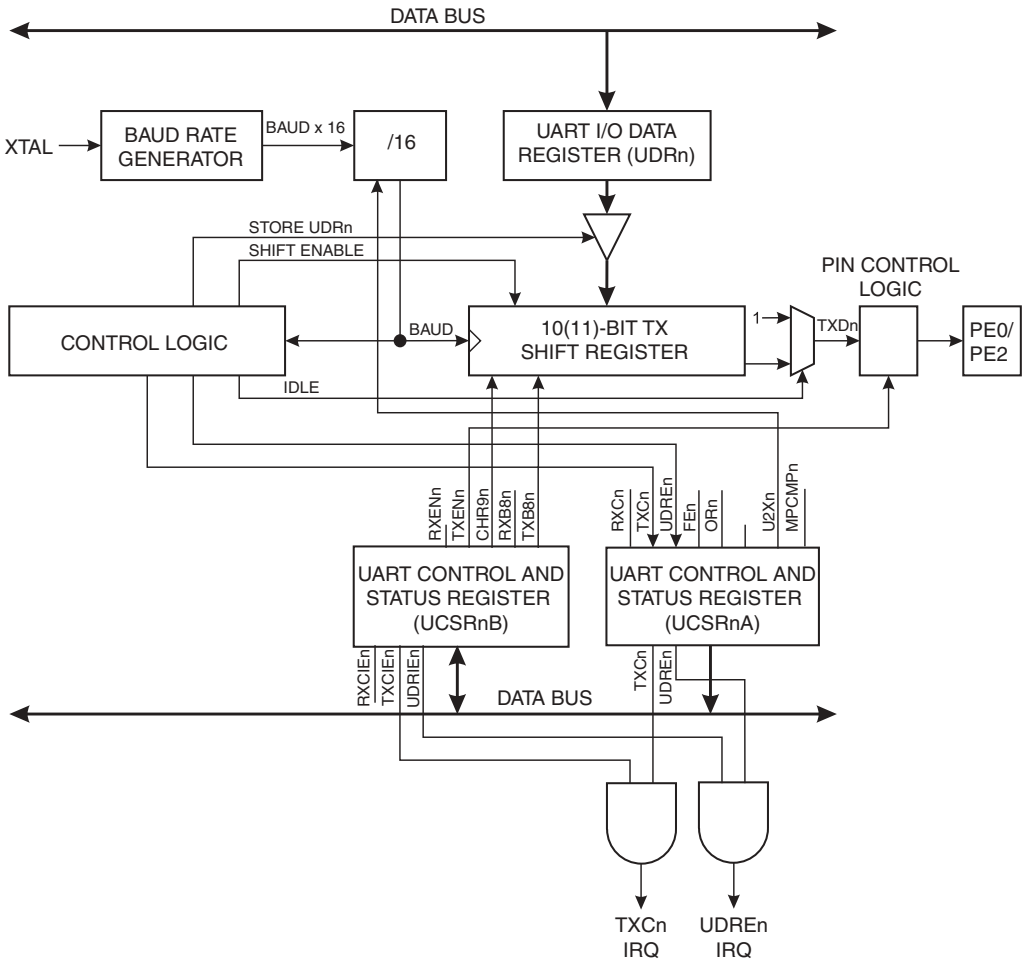
The FPSLIC features two full duplex (separate receive and transmit registers) Universal Asynchronous Receiver and Transmitter (UART). The main features are:

- 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

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

Figure 82. PortE Schematic Diagram (Pin PE7)

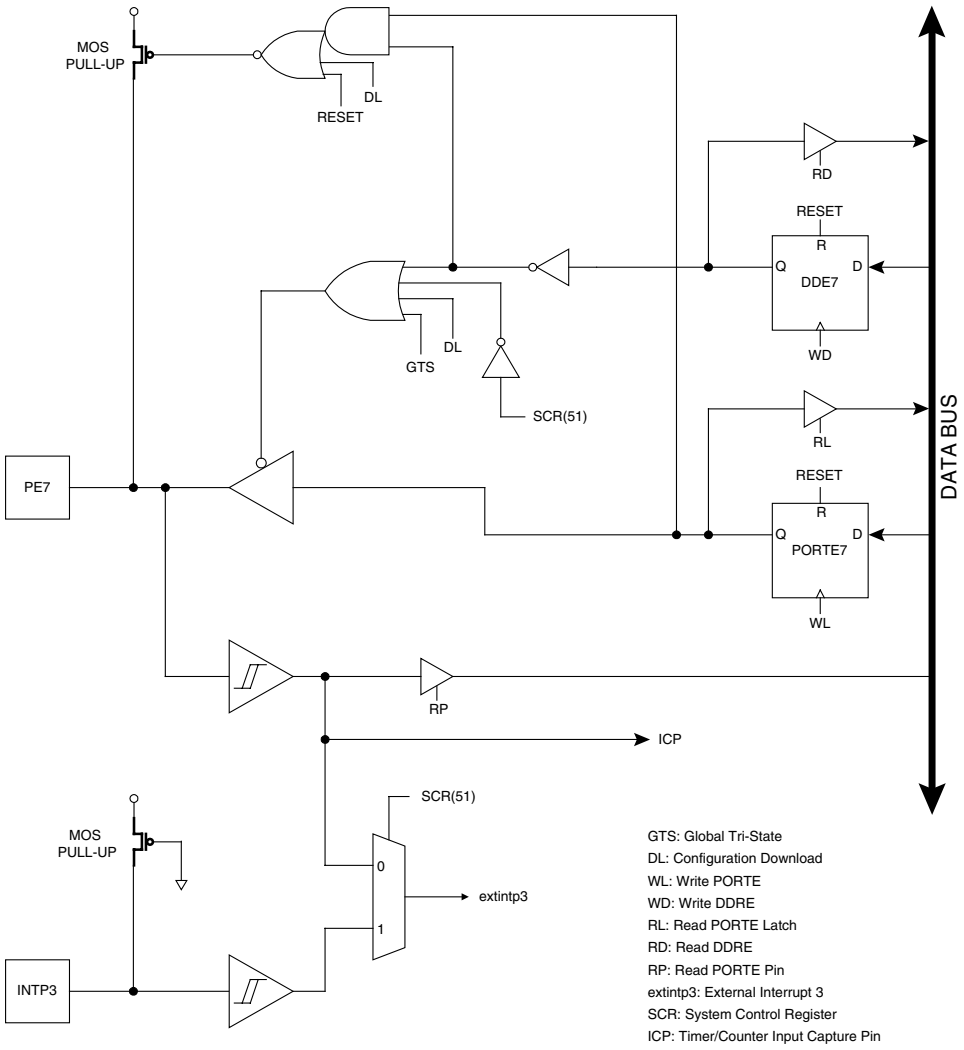


Table 52. FPSLIC Interface Timing Information[1]

Symbol	Parameter	3.3V Commercial ± 10%			3.3V Industrial ± 10%			Units
		Minimum	Typical	Maximum	Minimum	Typical	Maximum	
t_{IXG4}	Clock Delay From XTAL2 Pad to GCK_5 Access to FPGA Core	3.6	4.8	7.6	3.4	4.8	7.9	ns
t_{IXG5}	Clock Delay From XTAL2 Pad to GCK_6 Access to FPGA Core	3.9	5.2	8.1	3.6	5.2	8.8	ns
t_{IXC}	Clock Delay From XTAL2 Pad to AVR Core Clock	2.8	3.7	6.3	2.5	3.7	6.9	ns
t_{IXI}	Clock Delay From XTAL2 Pad to AVR I/O Clock	3.5	4.7	7.5	3.2	4.7	7.8	ns
t_{CFIR}	AVR Core Clock to FPGA I/O Read Enable	5.3	6.6	7.9	4.4	6.6	9.2	ns
t_{CFIW}	AVR Core Clock to FPGA I/O Write Enable	5.2	6.6	7.9	4.4	6.6	9.2	ns
t_{CFIS}	AVR Core Clock to FPGA I/O Select Active	6.3	7.8	9.4	5.3	7.8	11.0	ns
t_{FIRQ}	FPGA Interrupt Net Propagation Delay to AVR Core	0.2	0.2	0.3	0.1	0.2	0.3	ns
t_{IFS}	FPGA SRAM Clock to On-chip SRAM	6.1	7.7	7.7	4.9	7.7	7.7	ns
t_{FRWS}	FPGA SRAM Write Stobe to On-chip SRAM	4.4	5.5	5.5	3.7	5.5	5.5	ns
t_{FAS}	FPGA SRAM Address Valid to On-chip SRAM Address Valid	5.4	6.7	6.7	4.3	6.7	6.7	ns
t_{FDWS}	FPGA Write Data Valid to On-chip SRAM Data Valid	1.3	1.7	2.0	1.3	1.7	2.0	ns
t_{FDRS}	On-chip SRAM Data Valid to FPGA Read Data Valid	0.2	0.2	0.2	0.2	0.2	0.2	ns

Note: 1. Insertion delays are specified from XTAL2. These delays are more meaningful because the XTAL1-to-XTAL2 delay is sensitive to system loading on XTAL2. If it is necessary to drive external devices with the system clock, devices should use XTAL2 output pin. Remember that XTAL2 is inverted in comparison to XTAL1.

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.