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	-40°C ~ 85°C
Package / Case	100-TQFP
Supplier Device Package	100-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at94k10al-25aqi

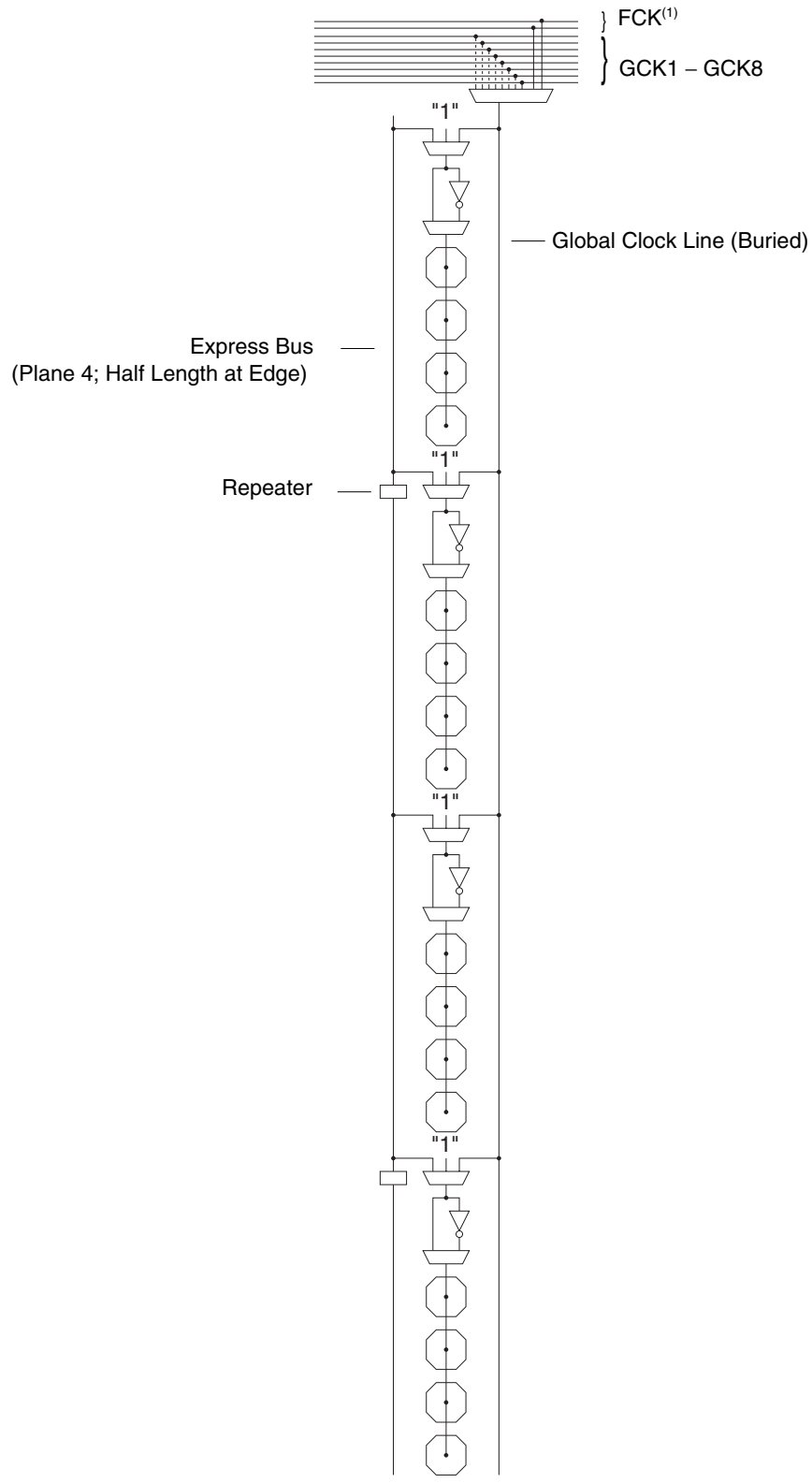
The embedded AVR core achieves throughputs approaching 1 MIPS per MHz by executing powerful instructions in a single-clock cycle, and allows system designers to optimize power consumption versus processing speed. The AVR core is based on an enhanced RISC architecture that combines a rich instruction set with 32 general-purpose working registers. All 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code-efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers at the same clock frequency. The AVR executes out of on-chip SRAM. Both the FPGA configuration SRAM and the AVR instruction code SRAM can be automatically loaded at system power-up using Atmel's In-System Programmable (ISP) AT17 Series EEPROM Configuration Memories or ATFS FPSLIC Support Devices.

State-of-the-art FPSLIC design tools, System Designer™, were developed in conjunction with the FPSLIC architecture to help reduce overall time-to-market by integrating microcontroller development and debug, FPGA development and Place and Route, and complete system co-verification in one easy-to-use software tool.

Table 2. ATFS FPSLIC Support Devices

FPSLIC Device	FPSLIC Support Device	Configuration Data	Spare Memory
AT94K05	ATFS05	226520 Bits	35624 Bits
AT94K10	ATFS10	430488 Bits	93800 Bits
AT94K40	ATFS40	815382 Bits	233194 Bits

Figure 12. Clocking (for One Column of Cells)



Note: 1. Two on left edge column of the embedded FPGA array only.

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

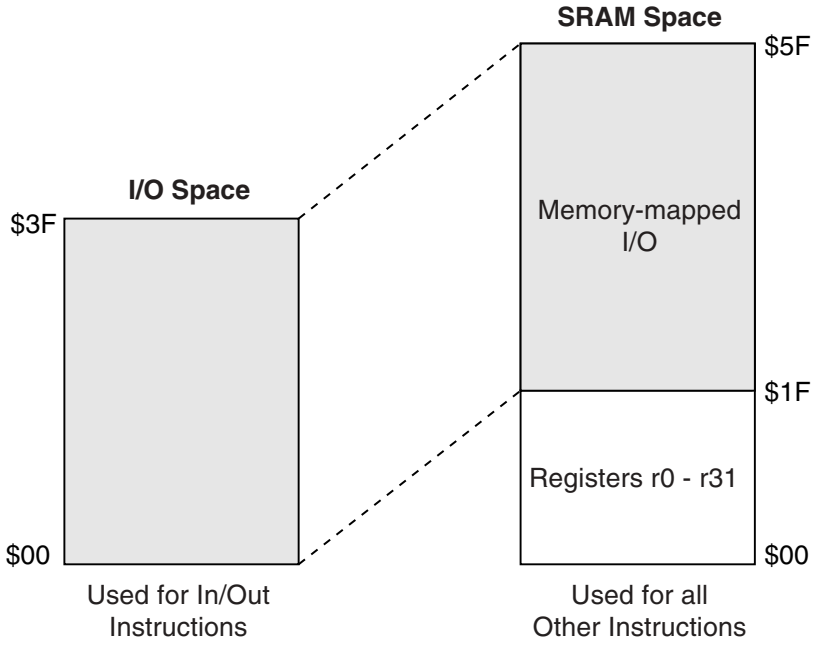
Table 11. FPSLIC System Control Register

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

Figure 32. Memory-mapped I/O



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

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

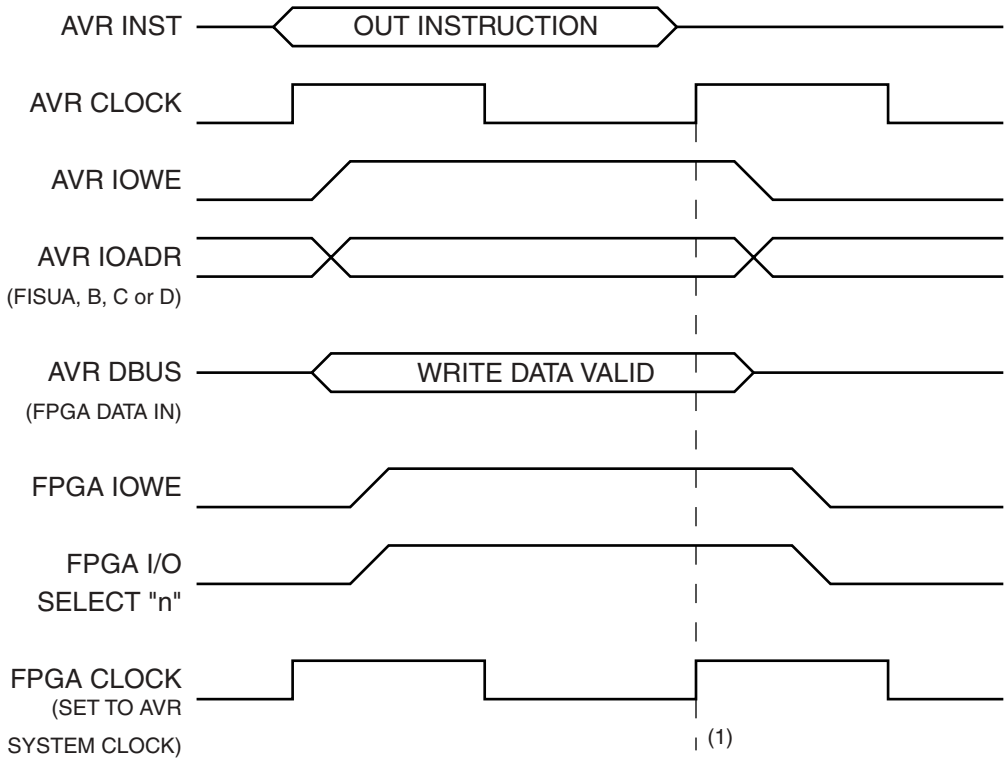
In the data SRAM, the registers are located at memory addresses $00 - $1F and the I/O space is located at memory addresses $20 - $5F.

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

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

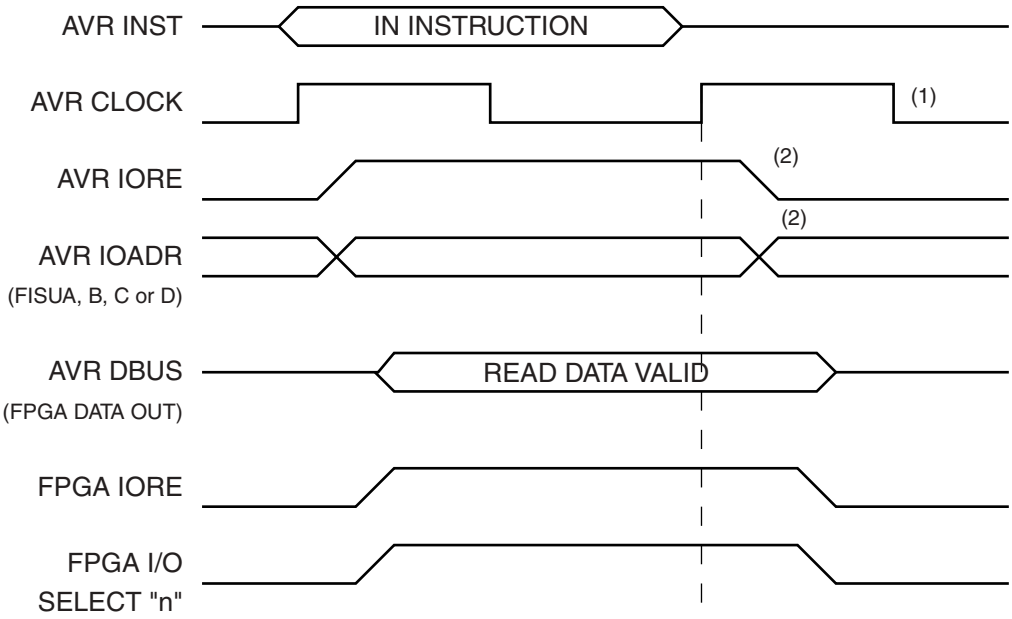
The status flags are cleared by writing a logic 1 to them. Note that the CBI and SBI instructions will operate on all bits in the I/O register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.

Figure 33. Out Instruction – AVR Writing to the FPGA



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

Figure 34. In Instruction – AVR Reading FPGA



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

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

- **Bit 2 - OCIE2: Timer/Counter2 Output Compare Interrupt Enable**

When the OCIE2 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter2 Compare Match interrupt is enabled. The corresponding interrupt is executed if a Compare match in Timer/Counter2 occurs, i.e., when the OCF2 bit is set in the Timer/Counter interrupt flag register – TIFR.

- **Bit 1 - TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

- **Bit 0 - OCIE0: Timer/Counter0 Output Compare Interrupt Enable**

When the OCIE0 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter0 Compare Match interrupt is enabled. The corresponding interrupt is executed if a Compare match in Timer/Counter0 occurs, i.e., when the OCF0 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

Timer/Counter Interrupt Flag Register – TIFR

Bit	7	6	5	4	3	2	1	0	
$38 ($58)	**TOV1**	**OCF1A**	**OCF1B**	**TOV2**	**ICF1**	**OCF2**	**TOV0**	**OCF0**	**TIFR**
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 - TOV1: Timer/Counter1 Overflow Flag**

The TOV1 is set (one) when an overflow occurs in Timer/Counter1. TOV1 is cleared by the hardware when executing the corresponding interrupt handling vector. Alternatively, TOV1 is cleared by writing a logic 1 to the flag. When the I-bit in SREG, and TOIE1 (Timer/Counter1 Overflow Interrupt Enable), and TOV1 are set (one), the Timer/Counter1 Overflow Interrupt is executed. In PWM mode, this bit is set when Timer/Counter1 advances from $0000.

- **Bit 6 - OCF1A: Output Compare Flag 1A**

The OCF1A bit is set (one) when compare match occurs between the Timer/Counter1 and the data in OCR1A – Output Compare Register 1A. OCF1A is cleared by the hardware when executing the corresponding interrupt handling vector. Alternatively, OCF1A is cleared by writing a logic 1 to the flag. When the I-bit in SREG, and OCIE1A (Timer/Counter1 Compare Interrupt Enable), and the OCF1A are set (one), the Timer/Counter1 Compare A match Interrupt is executed.

- **Bit 5 - OCF1B: Output Compare Flag 1B**

The OCF1B bit is set (one) when compare match occurs between the Timer/Counter1 and the data in OCR1B – Output Compare Register 1B. OCF1B is cleared by the hardware when executing the corresponding interrupt handling vector. Alternatively, OCF1B is cleared by writing a logic 1 to the flag. When the I-bit in SREG, and OCIE1B (Timer/Counter1 Compare match Interrupt Enable), and the OCF1B are set (one), the Timer/Counter1 Compare B match Interrupt is executed.

- **Bit 4 - TOV2: Timer/Counter2 Overflow Flag**

The TOV2 bit is set (one) when an overflow occurs in Timer/Counter2. TOV2 is cleared by the hardware when executing the corresponding interrupt handling vector. Alternatively, TOV2 is cleared by writing a logic 1 to the flag. When the I-bit in SREG, and TOIE2 (Timer/Counter1 Overflow Interrupt Enable), and TOV2 are set (one), the Timer/Counter2 Overflow Interrupt is executed. In PWM mode, this bit is set when Timer/Counter2 advances from $00.

Figure 43. Boundary-scan Cell For Bi-directional Port Pin with Pull-up Function

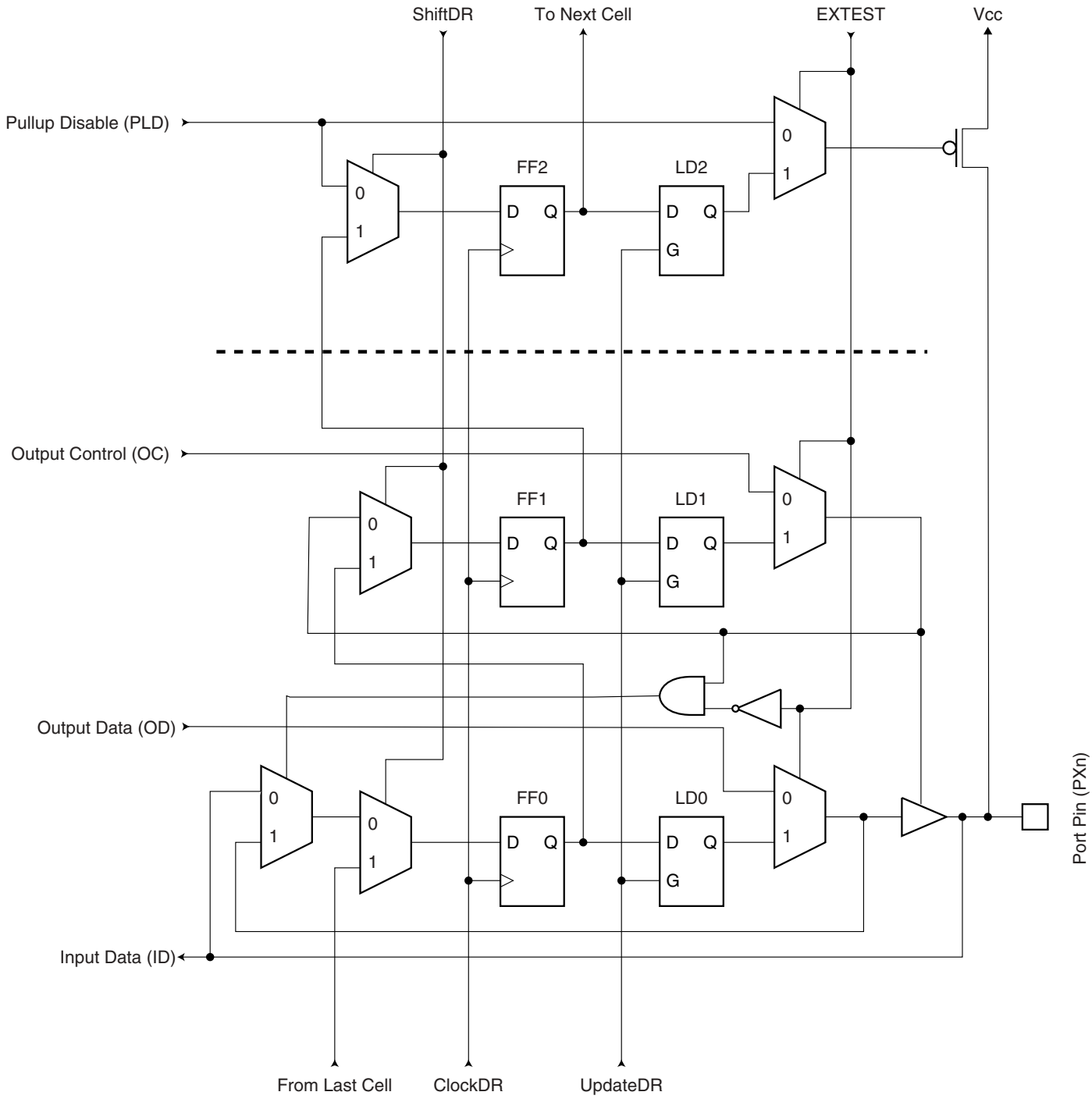
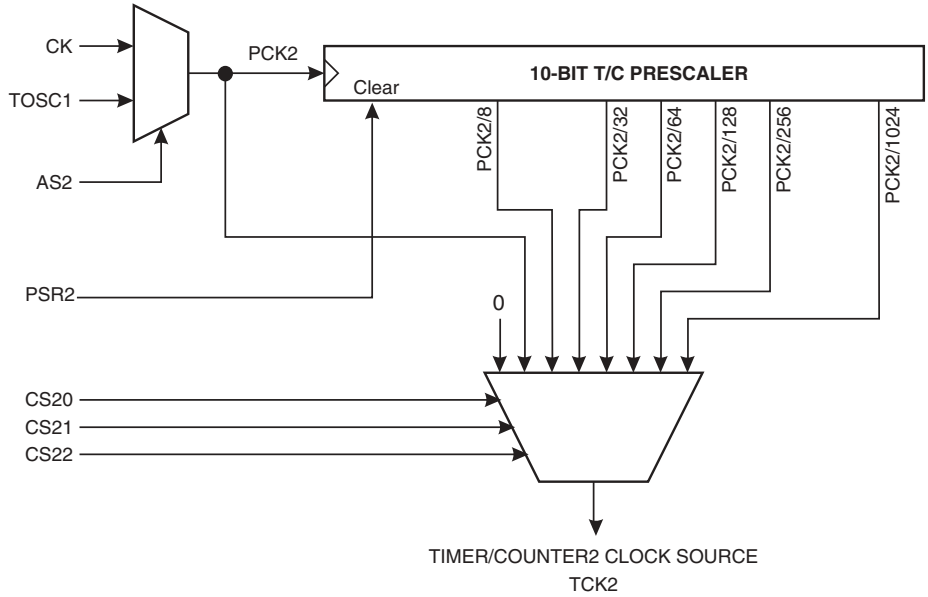


Table 20. AVR I/O Boundary Scan – JTAG Instructions $0/$2

I/O Ports	Description	Bit
PORTD	Data Out/In - PD7	44
	Enable Output - PD7	43
	Pull-up - PD7	42
	Data Out/In - PD6	41
	Enable Output - PD6	40
	Pull-up - PD6	39
	Data Out/In - PD5	38
	Enable Output - PD5	37
	Pull-up - PD5	36
	Data Out/In - PD4	35
	Enable Output - PD4	34
	Pull-up - PD4	33
	Data Out/In - PD3	32
	Enable Output - PD3	31
	Pull-up - PD3	30
	Data Out/In - PD2	29
	Enable Output - PD2	28
	Pull-up - PD2	27
	Data Out/In - PD1	26
	Enable Output - PD1	25
	Pull-up - PD1	24
	Data Out/In - PD0	23
	Enable Output - PD0	22
	Pull-up - PD0	21
EXT. INTERRUPTS	Input with Pull-up - INTP3	20[1]
	Input with Pull-up - INTP2	19[1]
	Input with Pull-up - INTP1	18[1]
	Input with Pull-up - INTP0	17[1]
UART1	Data Out/In - TX1	16
	Enable Output - TX1	15
	Pull-up - TX1	14
	Input with Pull-up - RX1	13[1]
UART0	Data Out/In - TX0	12
	Enable Output - TX0	11
	Pull-up - TX0	10
	Input with Pull-up - RX0	9[1]

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.

Multi-processor Communication Mode

The Multi-processor Communication Mode enables several Slave MCUs to receive data from a Master MCU. This is done by first decoding an address byte to find out which MCU has been addressed. If a particular Slave MCU has been addressed, it will receive the following data bytes as normal, while the other Slave MCUs will ignore the data bytes until another address byte is received.

For an MCU to act as a Master MCU, it should enter 9-bit transmission mode (CHR9n in UCSRnB set). The 9-bit must be one to indicate that an address byte is being transmitted, and zero to indicate that a data byte is being transmitted.

For the Slave MCUs, the mechanism appears slightly different for 8-bit and 9-bit Reception mode. In 8-bit Reception mode (CHR9n in UCSRnB cleared), the stop bit is one for an address byte and zero for a data byte. In 9-bit Reception mode (CHR9n in UCSRnB set), the 9-bit is one for an address byte and zero for a data byte, whereas the stop bit is always High.

The following procedure should be used to exchange data in Multi-processor Communication mode:

1. All Slave MCUs are in Multi-processor Communication Mode (MPCMn in UCSRnA is set).
2. The Master MCU sends an address byte, and all Slaves receive and read this byte. In the Slave MCUs, the RXCn flag in UCSRnA will be set as normal.
3. Each Slave MCU reads the UDRn register and determines if it has been selected. If so, it clears the MPCMn bit in UCSRnA, otherwise it waits for the next address byte.
4. For each received data byte, the receiving MCU will set the receive complete flag (RXCn in UCSRnA. In 8-bit mode, the receiving MCU will also generate a framing error (FEn in UCSRnA set), since the stop bit is zero. The other Slave MCUs, which still have the MPCMn bit set, will ignore the data byte. In this case, the UDRn register and the RXCn, FEn, or flags will not be affected.
5. After the last byte has been transferred, the process repeats from step 2.

UART Control

UART0 I/O Data Register – UDR0

Bit	7	6	5	4	3	2	1	0	
$0C ($2C)	MSB							LSB	UDR0
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 I/O Data Register – UDR1

Bit	7	6	5	4	3	2	1	0	
$03 ($23)	MSB							LSB	UDR1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The UDRn register is actually two physically separate registers sharing the same I/O address. When writing to the register, the UART Transmit Data register is written. When reading from UDRn, the UART Receive Data register is read.

- **Bit 3 - TXEN0/TXEN1: Transmitter Enable**

This bit enables the UART transmitter when set (one). When disabling the transmitter while transmitting a character, the transmitter is not disabled before the character in the shift register plus any following character in UDRn has been completely transmitted.

- **Bit 2 - CHR90/CHR91: 9-bit Characters**

When this bit is set (one) transmitted and received characters are 9-bit long plus start and stop bits. The 9-bit is read and written by using the RXB8n and TXB8n bits in UCSRnB, respectively. The 9th data bit can be used as an extra stop bit or a parity bit.

- **Bit 1 - RXB80/RXB81: Receive Data Bit 8**

When CHR9n is set (one), RXB8n is the 9th data bit of the received character.

- **Bit 0 - TXB80/TXB81: Transmit Data Bit 8**

When CHR9n is set (one), TXB8n is the 9th data bit in the character to be transmitted.

Baud-rate Generator

The baud-rate generator is a frequency divider which generates baud-rates according to the following equation[1]:

$$BAUD = \frac{f_{CK}}{16(UBR + 1)}$$

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

Note: 1. This equation is not valid when the UART transmission speed is doubled. See “Double Speed Transmission” on page 128 for a detailed description.

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 2% from the target baud-rate, are bold in the table. However, using baud-rates that have more than 1% error is not recommended. High error ratings give less noise resistance.

Table 36. UBR Settings at Various Crystal Frequencies

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

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

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

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

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

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

UART0 and UART1 High Byte Baud-rate Register UBRRHI

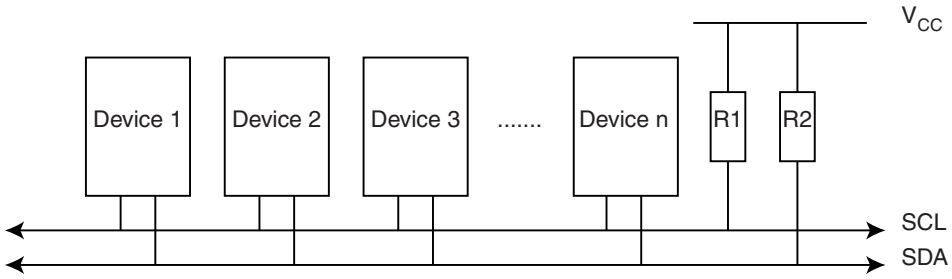
Bit	7	6	5	4	3	2	1	0	
$20 ($40)	**MSB1**			**LSB1**	**MSB0**			**LSB0**	**UBRRHI**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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

2-wire Serial Interface (Byte Oriented)

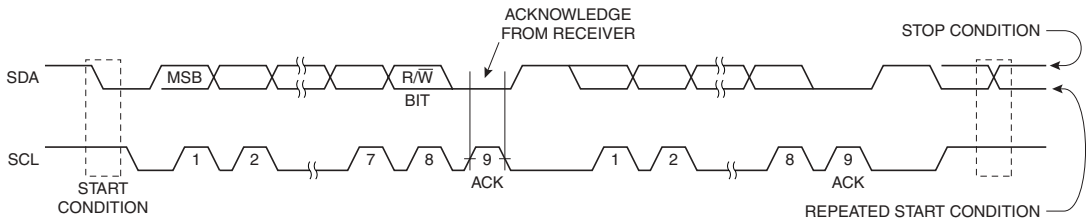
The 2-wire Serial Bus is a bi-directional two-wire serial communication standard. It is designed primarily for simple but efficient integrated circuit (IC) control. The system is comprised of two lines, SCL (Serial Clock) and SDA (Serial Data) that carry information between the ICs connected to them.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

Figure 68. 2-wire Serial Bus Configuration



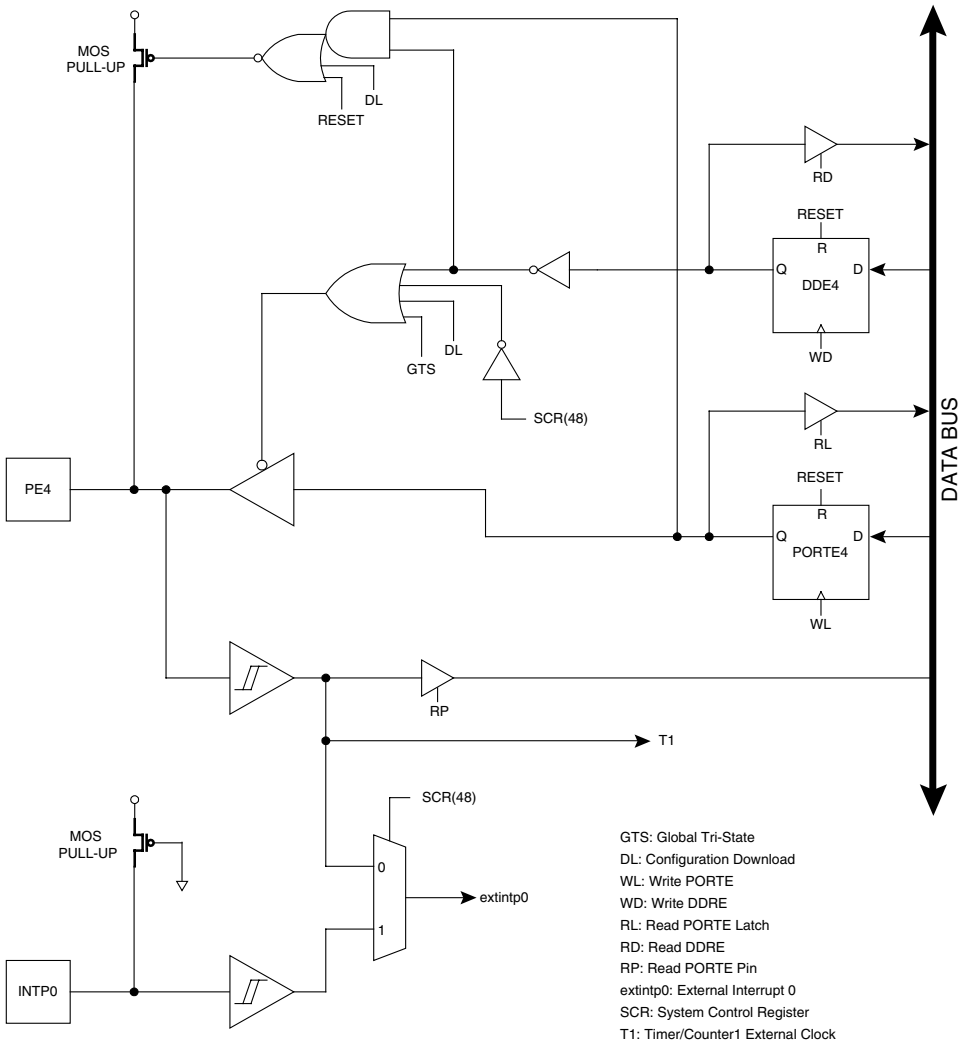
The 2-wire Serial Interface provides a serial interface that meets the 2-wire Serial Bus specification and supports Master/Slave and Transmitter/Receiver operation at up to 400 kHz bus clock rate. The 2-wire Serial Interface has hardware support for the 7-bit addressing, but is easily extended to 10-bit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

Figure 69. 2-wire Serial Bus Timing Diagram



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

Figure 79. PortE Schematic Diagram (Pin PE4)



Power-On Power Supply Requirements

Atmel FPGAs require a minimum rated power supply current capacity to insure proper initialization, and the power supply ramp-up time does affect the current required. A fast ramp-up time requires more current than a slow ramp-up time.

Table 49. Power-On Power Supply Requirements[1]

Device	Description	Maximum Current[2][3]
AT94K05AL AT94K10AL	Maximum Current Supply	50 mA
AT94K40AL	Maximum Current Supply	100 mA

Notes:
1. This specification applies to Commercial and Industrial grade products only.
2. Devices are guaranteed to initialize properly at 50% of the minimum current listed above. A larger capacity power supply may result in a larger initialization current.
3. Ramp-up time is measured from 0 V DC to 3.6 V DC. Peak current required lasts less than 2 ms, and occurs near the internal power on reset threshold voltage.

FPSLIC Dual-port SRAM Characteristics

The Dual-port SRAM operates in single-edge clock controlled mode during read operations, and a double-edge controlled mode during write operations. Addresses are clocked internally on the rising edge of the clock signal (ME). Any change of address without a rising edge of ME is not considered.

In read mode, the rising clock edge triggers data read without any significant constraint on the length of the clock pulse. The WE signal must be changed and held Low before the rising edge of ME to signify a read cycle. The WE signal should then remain Low until the falling edge of the clock.

In write mode, data applied to the inputs is latched on either the falling edge of WE or the falling edge of the clock, whichever comes earlier, and written to memory. Also, WE must be High before the rising edge of ME to signify a write cycle. If data inputs change during a write cycle, only the value present at the write end is considered and written to the address clocked at the ME rise. A write cycle ending on WE fall does not turn into a read cycle – the next cycle will be a read cycle if WE remains Low during rising edge of ME.

Figure 83. SRAM Read Cycle Timing Diagram

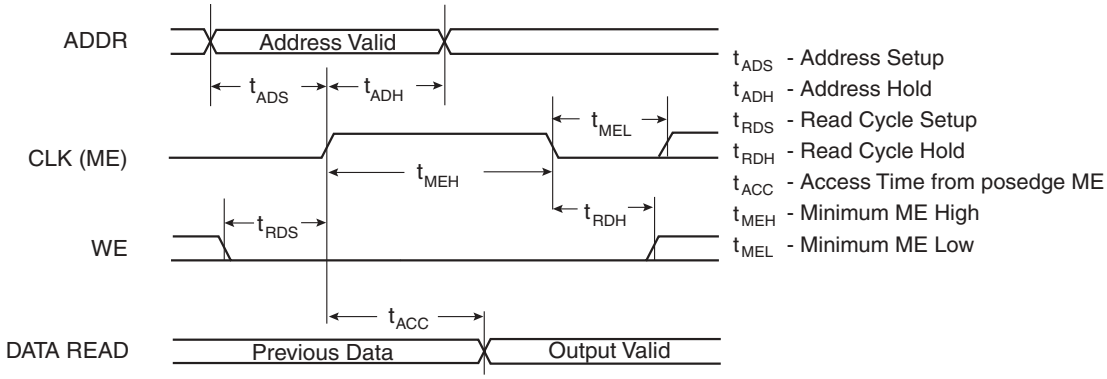
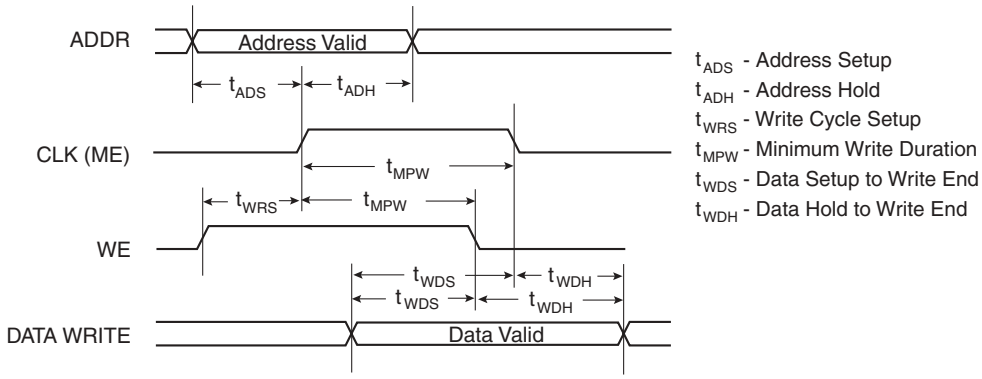


Figure 84. SRAM Write Cycle Timing Diagram



Frame Interface

The FPGA Frame Clock phase is selectable (see “System Control Register – FPGA/AVR” on page 30). This document refers to the clock at the FPGA/Dual-port SRAM interface as ME (the relation of ME to data, address and write enable does not change). By default, FrameClock is inverted (ME = ~FrameClock). Selecting the non-inverted phase assigns ME = FrameClock. Recall, the Dual-port SRAM operates in single-edge clock controlled mode during read operations, and double-edge clock controlled mode during writes. Addresses are clocked internally on the rising edge of the clock signal (ME). Any change of address without a rising edge of ME is not considered.

AC Timing Characteristics – 3.3V Operation

Delays are based on fixed loads and are described in the notes.
Maximum times based on worst case: V_{CC} = 3.0V, temperature = 70°C
Minimum times based on best case: V_{CC} = 3.6V, temperature = 0°C
Maximum delays are the average of t_{PDLH} and t_{PDHL}.

Clocks and Reset Input buffers are measured from a V_{IH} of 1.5V at the input pad to the internal V_{IH} of 50% of V_{CC}.
Maximum times for clock input buffers and internal drivers are measured for rising edge delays only.

Cell Function	Parameter	Path	Device	-25	Units	Notes
Global Clocks and Set/Reset						
GCK Input Buffer	t_{PD} (Maximum)	pad -> clock pad -> clock	AT94K05 AT94K10 AT94K40	1.2 1.5 1.9	ns ns	Rising Edge Clock
FCK Input Buffer	t_{PD} (Maximum)	pad -> clock pad -> clock	AT94K05 AT94K10 AT94K40	0.7 0.8 0.9	ns ns	Rising Edge Clock
Clock Column Driver	t_{PD} (Maximum)	clock -> colclk clock -> colclk	AT94K05 AT94K10 AT94K40	1.3 1.8 2.5	ns ns	Rising Edge Clock
Clock Sector Driver	t_{PD} (Maximum)	colclk -> secclk colclk -> secclk	AT94K05 AT94K10 AT94K40	1.0 1.0 1.0	ns ns	Rising Edge Clock
GSRN Input Buffer	t_{PD} (Maximum)	colclk -> secclk colclk -> secclk	AT94K05 AT94K10 AT94K40	5.4 8.2	ns ns	–
Global Clock to Output	t_{PD} (Maximum)	clock pad -> out clock pad -> out	AT94K05 AT94K10 AT94K40	12.6 13.4 14.5	ns ns	Rising Edge Clock Fully Loaded Clock Tree Rising Edge DFF 20 mA Output Buffer 50 pf Pin Load
Fast Clock to Output	t_{PD} (Maximum)	clock pad -> out clock pad -> out	AT94K05 AT94K10 AT94K40	12.1 12.7 13.5	ns ns	Rising Edge Clock Fully Loaded Clock Tree Rising Edge DFF 20 mA Output Buffer 50 pf Pin Load