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?

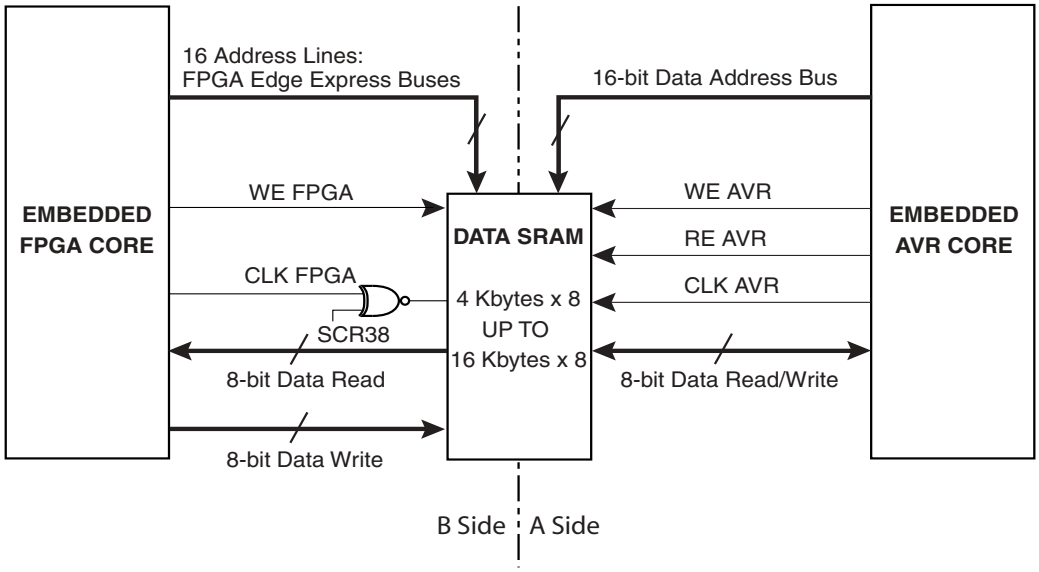
At their core, **FPGAs** are semiconductor devices that can

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	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at94k10al-25bqc

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).

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

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

B Side

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

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

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

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

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

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

Table 7. Summary Table for AVR and FPGA SRAM Addressing

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

System Control

Configuration Modes

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

Table 10. Configuration Modes

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

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

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

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

System Control Register – FPGA/AVR

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

Table 11. FPSLIC System Control Register

Bit	Description
SCR0 - SCR1	Reserved
SCR2	0 = Enable Cascading 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
SCR6	0 = OTS Disabled 1 = OTS Enabled Setting SCR6 makes the OTS (output tri-state) pin an input which controls the global tri-state control for all user I/O. This junction allows the user at any time to tristate all user I/O and isolate the chip.
SCR7 - SCR12	Reserved
SCR13	0 = CCLK Normal Operation 1 = CCLK Continues After Configuration. Setting bit SCR13 allows the CCLK pin to continue to run after configuration download is completed. This bit is valid for Master mode, mode 0 only. The CCLK is not available internally on the device. If it is required in the design, it must be connected to another device I/O.
SCR14 - SCR15	Reserved
SCR16 - SCR23	0 = GCK 0:7 Always Enabled 1 = GCK 0:7 Disabled During Internal and External Configuration Download. Setting SCR16:SCR23 allows the user to disable the input buffers driving the global clocks. The clock buffers are enabled and disabled synchronously with the rising edge of the respective GCK signal, and stop in a High “1” state. Setting one of these bits disables the appropriate GCK input buffer only and has no effect on the connection from the input buffer to the FPGA array.
SCR24 - SCR25	0 = FCK 0:1 Always Enabled 1 = FCK 0:1 Disabled During Internal and External Configuration Download. Setting SCR24:SCR25 allows the user to disable the input buffers driving the fast clocks. The clock buffers are enabled and disabled synchronously with the rising edge of the respective FCK signal, and stop in a High “1” state. Setting one of these bits disables the appropriate FCK input buffer only and has no effect on the connection from the input buffer to the FPGA array.
SCR26	0 = Disable On-chip Debugger 1 = Enable On-chip Debugger. JTAG Enable, SCR27, must also be set (one) and the configuration memory lockout, SCR4, must be clear (zero) for the user to have access to internal scan chains.
SCR27	0 = Disable TAP at user FPGA I/O Ports 1 = Enable TAP at user FPGA I/O Ports. Device ID scan chain and AVR I/O boundary scan chain are available. The user must set (one) the On-chip Debug Enable, SCR26, and must keep the configuration memory lockout, SCR4, clear (zero) for the user to have access to internal scan chains.
SCR28 - SCR29	Reserved
SCR30	0 = Global Set/Reset Normal 1 = Global Set/Reset Active (Low) During Internal and External Configuration Download. SCR30 allows the Global set/reset to hold the core DFFs in reset during any configuration download. The Global set/reset net is released at the end of configuration download on the rising edge of CON, if set.
SCR31	0 = Disable I/O Tri-state 1 = I/O Tri-state During (Internal and External) Configuration Download. SCR31 forces all user defined I/O pins to go tri-state during configuration download. Tri-state is released at the end of configuration download on the rising edge of CON, if set.

Instruction Set Nomenclature (Summary)

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

Status Register (SREG)

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

Registers and Operands

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

I/O Registers

Stack

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

Flags

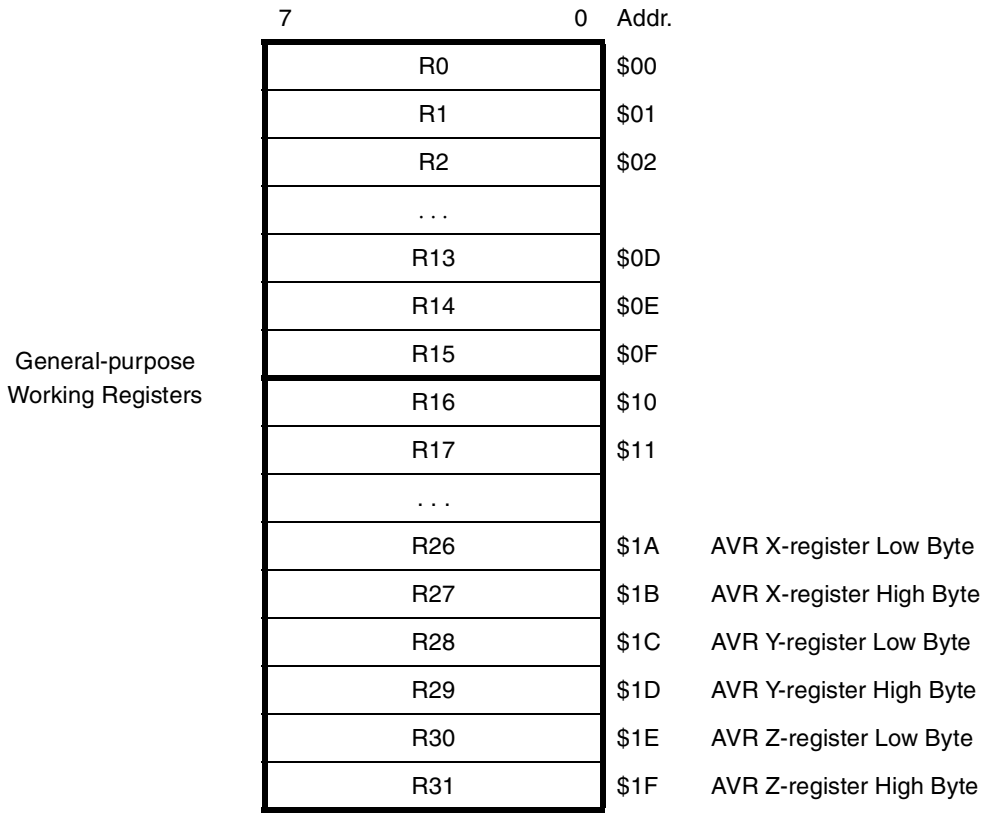
⇔:	Flag affected by instruction
0:	Flag cleared by instruction
1:	Flag set by instruction
-:	Flag not affected by instruction

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

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 29. The Parallel Instruction Fetches and Instruction Executions

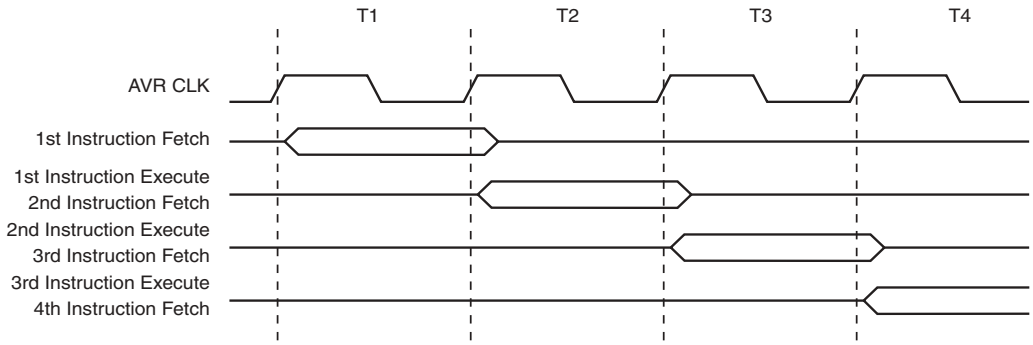
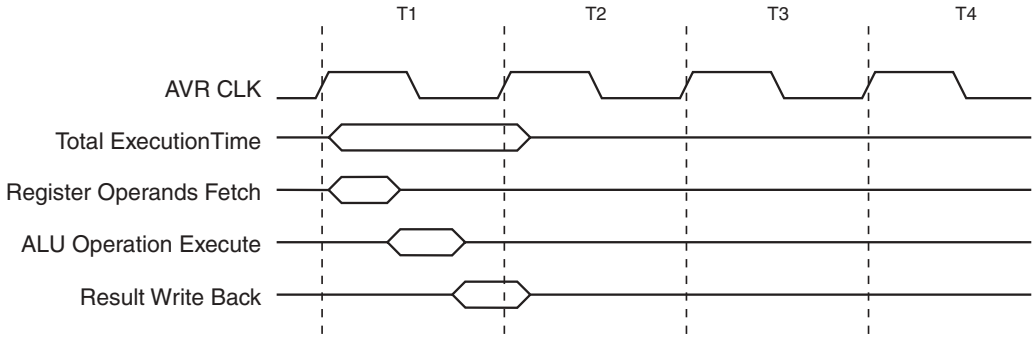


Figure 30 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 30. Single Cycle ALU Operation



The internal data SRAM access is performed in two system clock cycles as described in Figure 31.

Figure 31. On-chip Data SRAM Access Cycles

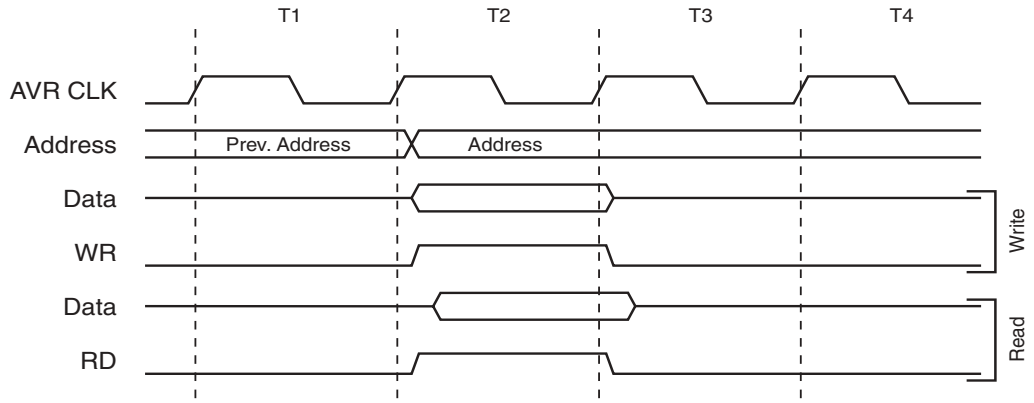


Table 16. Reset Characteristics (V_{CC} = 3.3V)

Symbol	Parameter	Minimum	Typical	Maximum	Units
$V_{POT(1)}$	Power-on Reset Threshold (Rising)	1.0	1.4	1.8	V
	Power-on Reset Threshold (Falling)	0.4	0.6	0.8	V
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage		$V_{CC}/2$		V
T_{TOUT}	Reset Delay Time-out Period		5		CPU cycles
		0.4 3.2 12.8	0.5 4.0 16.0	0.6 4.8 19.2	ms

Note: 1. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling).

Power-on Reset

A Power-on Reset (POR) circuit ensures that the device is reset from power-on. As shown in Figure 35, an internal timer clocked from the Watchdog Timer oscillator prevents the MCU from starting until after a certain period after V_{CC} has reached the Power-on Threshold voltage – V_{POT}, regardless of the V_{CC} rise time (see Figure 36 and Figure 37).

Figure 36. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}

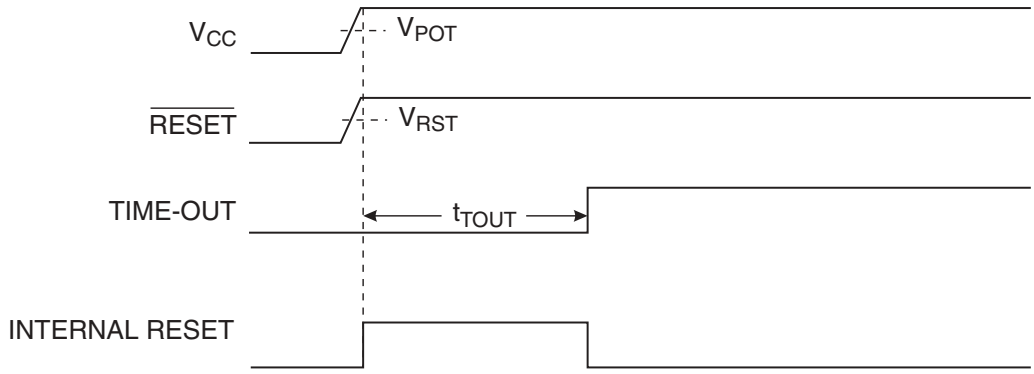
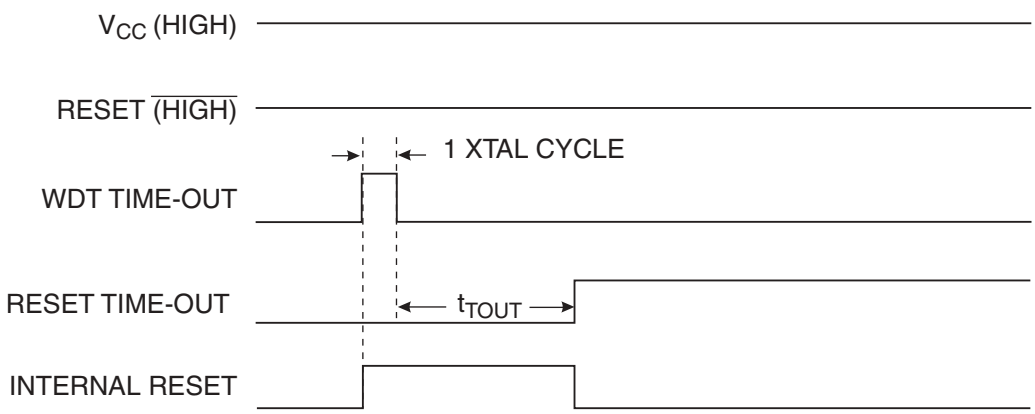


Figure 37. Watchdog Reset during Operation



JTAG Interface and On-chip Debug System

Features

- **JTAG (IEEE std. 1149.1 Compliant) Interface**
- **AVR I/O Boundary-scan Capabilities According to the JTAG Standard**
- **Debugger Access to:**
  - **All Internal Peripheral Units**
  - **AVR Program and Data SRAM**
  - **The Internal Register File**
  - **Program Counter/Instruction**
  - **FPGA/AVR Interface**
- **Extensive On-chip Debug Support for Break Conditions, Including**
  - **Break on Change of Program Memory Flow**
  - **Single Step Break**
  - **Program Memory Breakpoints on Single Address or Address Range**
  - **Data Memory Breakpoints on Single Address or Address Range**
  - **FPGA Hardware Break**
  - **Frame Memory Breakpoint on Single Address**
- **On-chip Debugging Supported by AVR Studio version 4 or above**

Overview

The AVR IEEE std. 1149.1 compliant JTAG interface is used for on-chip debugging.

The On-Chip Debug support is considered being private JTAG instructions, and distributed within ATMEL and to selected third-party vendors only.

Figure 39 shows a block diagram of the JTAG interface and the On-Chip Debug system. The TAP Controller is a state machine controlled by the TCK and TMS signals. The TAP Controller selects either the JTAG Instruction Register or one of several Data Registers as the scan chain (shift register) between the TDI - input and TDO - output. The Instruction Register holds JTAG instructions controlling the behavior of a Data Register.

Of the Data Registers, the ID-Register, Bypass Register, and the AVR I/O Boundary-Scan Chain are used for board-level testing. The Internal Scan Chain and Break-Point Scan Chain are used for On-Chip debugging only.

The Test Access Port – TAP

The JTAG interface is accessed through four of the AVR's pins. In JTAG terminology, these pins constitute the Test Access Port - TAP. These pins are:

- TMS: Test Mode Select. This pin is used for navigating through the TAP-controller state machine.
- TCK: Test Clock. JTAG operation is synchronous to TCK
- TDI: Test Data In. Serial input data to be shifted in to the Instruction Register or Data Register (Scan Chains)
- TDO: Test Data Out. Serial output data from Instruction register or Data Register

The IEEE std. 1149.1 also specifies an optional TAP signal; TRST - Test ReSeT - which is not provided.

When the JTAGEN bit is unprogrammed, these four TAP pins revert to normal operation. When programmed, the input TAP signals are internally pulled High and the JTAG is enabled for Boundary-Scan. System Designer sets this bit by default.

For the On-Chip Debug system, in addition the $\overline{\text{RESET}}$ pin is monitored by the debugger to be able to detect external reset sources. The debugger can also pull the $\overline{\text{RESET}}$ pin Low to reset the whole system, assuming only open collectors on reset line are used in the application.

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 50. Timer/Counter0 Block Diagram

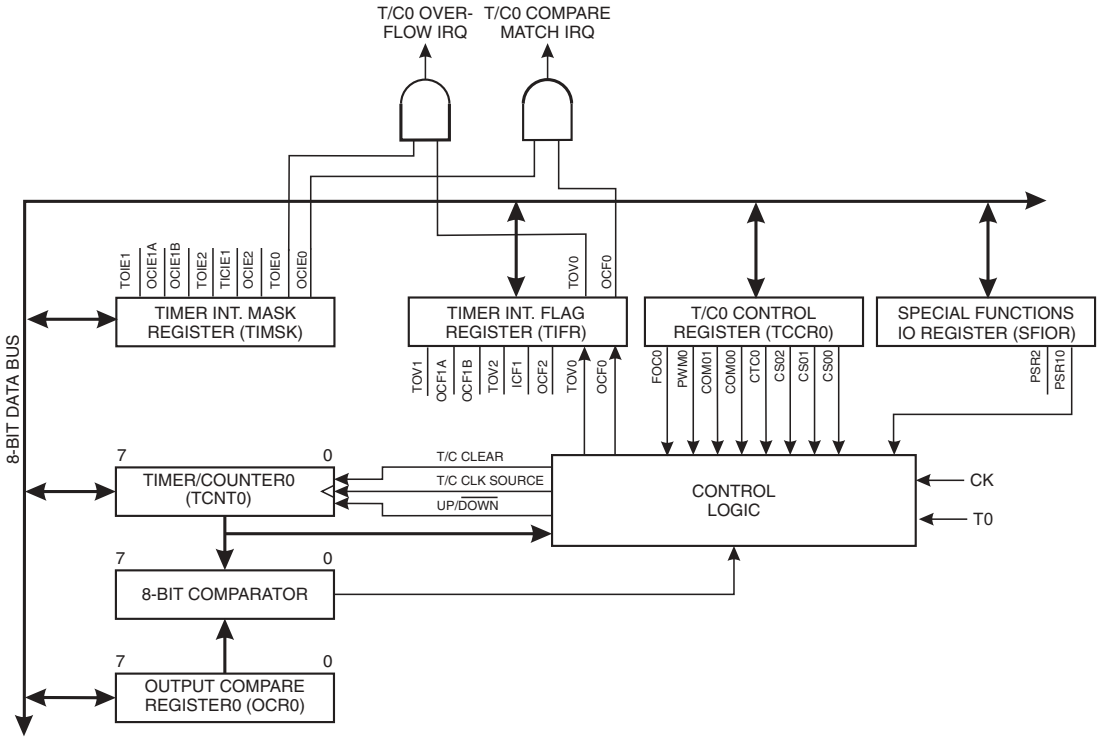
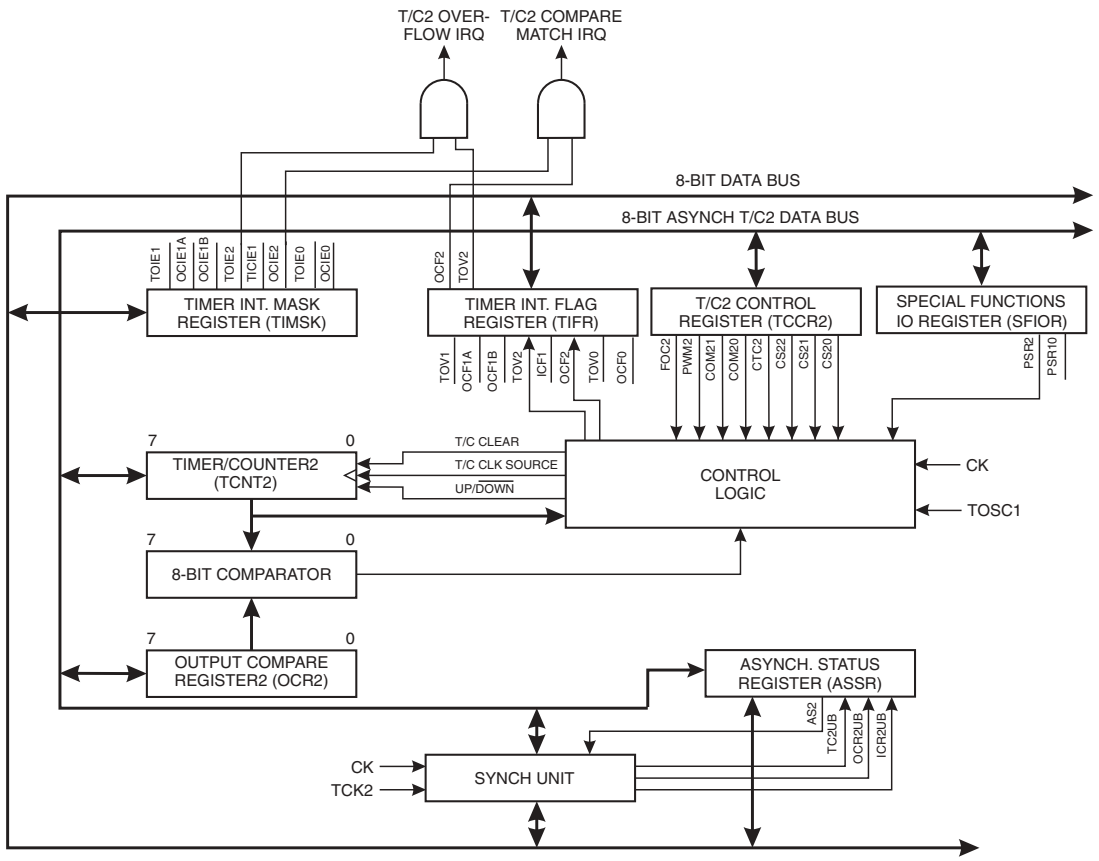


Figure 51. Timer/Counter2 Block Diagram



8-bit Multiplication

Doing an 8-bit multiply using the hardware multiplier is simple: just load the operands into two registers (or only one for square multiply) and execute one of the multiply instructions. The result will be placed in register pair R1:R0. However, note that only the MUL instruction does not have register usage restrictions. Figure 59 shows the valid (operand) register usage for each of the multiply instructions.

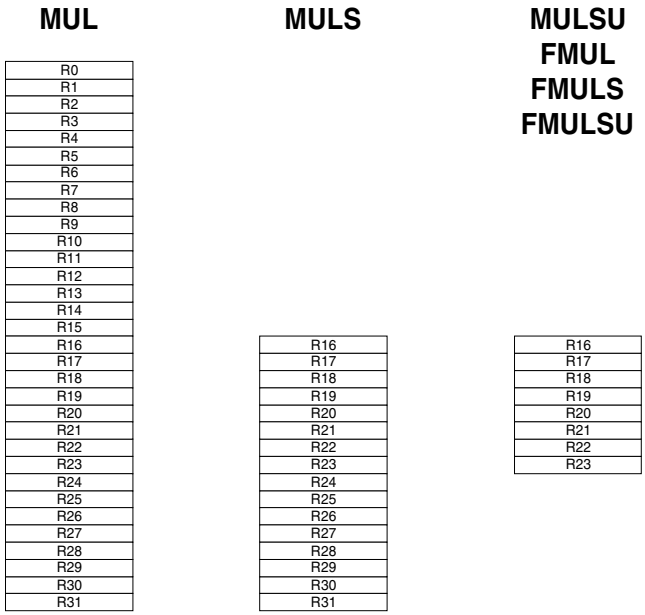
Example 1 – Basic Usage

The first example shows an assembly code that reads the port B input value and multiplies this value with a constant (5) before storing the result in register pair R17:R16.

```
in    r16,PINB      ; Read pin values
ldi   r17,5         ; Load 5 into r17
mul   r16,r17       ; r1:r0 = r17 * r16
movw  r17:r16,r1:r0; Move the result to the r17:r16
                    ; register pair
```

Note the use of the MOVW instruction. This example is valid for all of the multiply instructions.

Figure 59. Valid Register Usage



Example 2 – Special Cases

This example shows some special cases of the MUL instruction that are valid.

```
lds   r0,variableA; Load r0 with SRAM variable A
lds   r1,variableB; Load r1 with SRAM variable B
mul   r1,r0       ; r1:r0 = variable A * variable B

lds   r0,variableA; Load r0 with SRAM variable A
mul   r0,r0       ; r0:r0 = square(variable A)
```

Even though the operand is put in the result register pair R1:R0, the operation gives the correct result since R1 and R0 are fetched in the first clock cycle and the result is stored back in the second clock cycle.

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.

UART0 Control and Status Registers – UCSR0A

Bit	7	6	5	4	3	2	1	0	
$0B ($2B)	RXC0	TXC0	UDRE0	FE0	OR0	-	U2X0	MPCM0	UCSR0A
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

UART1 Control and Status Registers – UCSR1A

Bit	7	6	5	4	3	2	1	0	
$02 ($22)	RXC1	TXC1	UDRE1	FE1	OR1	-	U2X1	MPCM1	UCSR1A
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 - RXC0/RXC1: UART Receive Complete**

This bit is set (one) when a received character is transferred from the Receiver Shift register to UDRn. The bit is set regardless of any detected framing errors. When the RXCIEn bit in UCSRnB is set, the UART Receive Complete interrupt will be executed when RXCn is set (one). RXCn is cleared by reading UDRn. When interrupt-driven data reception is used, the UART Receive Complete Interrupt routine must read UDRn in order to clear RXCn, otherwise a new interrupt will occur once the interrupt routine terminates.

- **Bit 6 - TXC0/TXC1: UART Transmit Complete**

This bit is set (one) when the entire character (including the stop bit) in the Transmit Shift register has been shifted out and no new data has been written to UDRn. This flag is especially useful in half-duplex communications interfaces, where a transmitting application must enter receive mode and free the communications bus immediately after completing the transmission.

When the TXCIEn bit in UCSRnB is set, setting of TXCn causes the UART Transmit Complete interrupt to be executed. TXCn is cleared by the hardware when executing the corresponding interrupt handling vector. Alternatively, the TXCn bit is cleared (zero) by writing a logic 1 to the bit.

- **Bit 5 - UDRE0/UDRE1: UART Data Register Empty**

This bit is set (one) when a character written to UDRn is transferred to the Transmit shift register. Setting of this bit indicates that the transmitter is ready to receive a new character for transmission.

When the UDRIEn bit in UCSRnB is set, the UART Transmit Complete interrupt will be executed as long as UDREn is set and the global interrupt enable bit in SREG is set. UDREn is cleared by writing UDRn. When interrupt-driven data transmittal is used, the UART Data Register Empty Interrupt routine must write UDRn in order to clear UDREn, otherwise a new interrupt will occur once the interrupt routine terminates.

UDREn is set (one) during reset to indicate that the transmitter is ready.

- **Bit 4 - FE0/FE1: Framing Error**

This bit is set if a Framing Error condition is detected, i.e., when the stop bit of an incoming character is zero.

The FEn bit is cleared when the stop bit of received data is one.

- **Bit 3 - OR0/OR1: OverRun**

This bit is set if an Overrun condition is detected, i.e., when a character already present in the UDRn register is not read before the next character has been shifted into the Receiver Shift register. The ORn bit is buffered, which means that it will be set once the valid data still in UDRn is read.

The ORn bit is cleared (zero) when data is received and transferred to UDRn.

- **Bit 2 - Res: Reserved Bit**

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

- **Bits 1 - U2X0/U2X1: Double the UART Transmission Speed**

When this bit is set (one) the UART speed will be doubled. This means that a bit will be transmitted/received in eight CPU clock periods instead of 16 CPU clock periods. For a detailed description, see “Double Speed Transmission” on page 128”.

- **Bit 0 - MPCM0/MPCM1: Multi-processor Communication Mode**

This bit is used to enter Multi-processor Communication Mode. The bit is set when the Slave MCU waits for an address byte to be received. When the MCU has been addressed, the MCU switches off the MPCMn bit, and starts data reception.

For a detailed description, see “Multi-processor Communication Mode” on page 123.

UART0 Control and Status Registers – UCSR0B

Bit	7	6	5	4	3	2	1	0	
$0A ($2A)	**RXCIE0**	**TXCIE0**	**UDRIE0**	**RXEN0**	**TXEN0**	**CHR90**	**RXB80**	**TXB80**	**UCSR0B**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	1	0	

UART1 Control and Status Registers – UCSR1B

Bit	7	6	5	4	3	2	1	0	
$01 ($21)	**RXCIE1**	**TXCIE1**	**UDRIE1**	**RXEN1**	**TXEN1**	**CHR91**	**RXB81**	**TXB81**	**UCSR1B**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	1	0	

- **Bit 7 - RXCIE0/RXCIE1: RX Complete Interrupt Enable**

When this bit is set (one), a setting of the RXCn bit in UCSRnA will cause the Receive Complete interrupt routine to be executed provided that global interrupts are enabled.

- **Bit 6 - TXCIE0/TXCIE1: TX Complete Interrupt Enable**

When this bit is set (one), a setting of the TXCn bit in UCSRnA will cause the Transmit Complete interrupt routine to be executed provided that global interrupts are enabled.

- **Bit 5 - UDRIE0/UDREI1: UART Data Register Empty Interrupt Enable**

When this bit is set (one), a setting of the UDREn bit in UCSRnA will cause the UART Data Register Empty interrupt routine to be executed provided that global interrupts are enabled.

- **Bit 4 - RXEN0/RXEN1: Receiver Enable**

This bit enables the UART receiver when set (one). When the receiver is disabled, the TXCn, ORn and FEn status flags cannot become set. If these flags are set, turning off RXENn does not cause them to be cleared.

Table 43. Status Codes for Slave Receiver Mode (Continued)

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	
$90	Previously addressed with general call; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
$98	Previously addressed with general call; 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
$A0	A STOP condition or repeated START condition has been received while still addressed as Slave	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 78. PortE Schematic Diagram (Pin PE2)

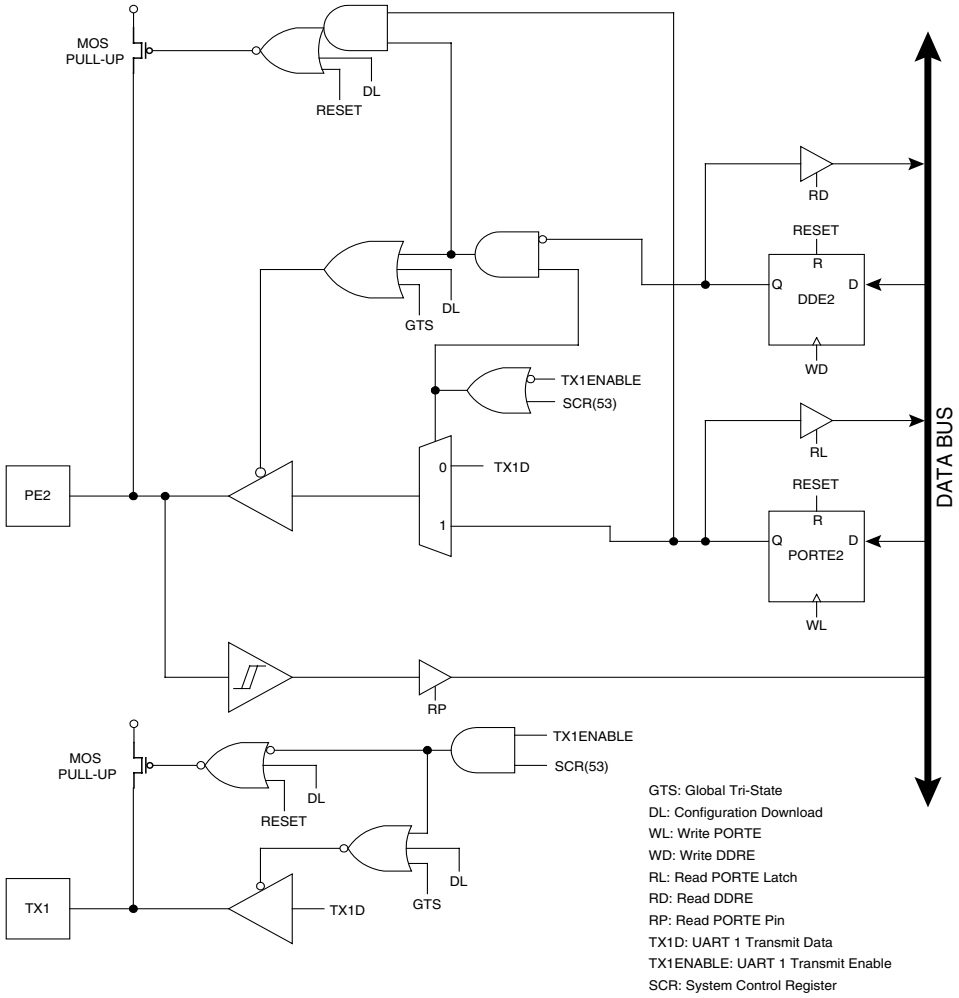


Figure 82. PortE Schematic Diagram (Pin PE7)

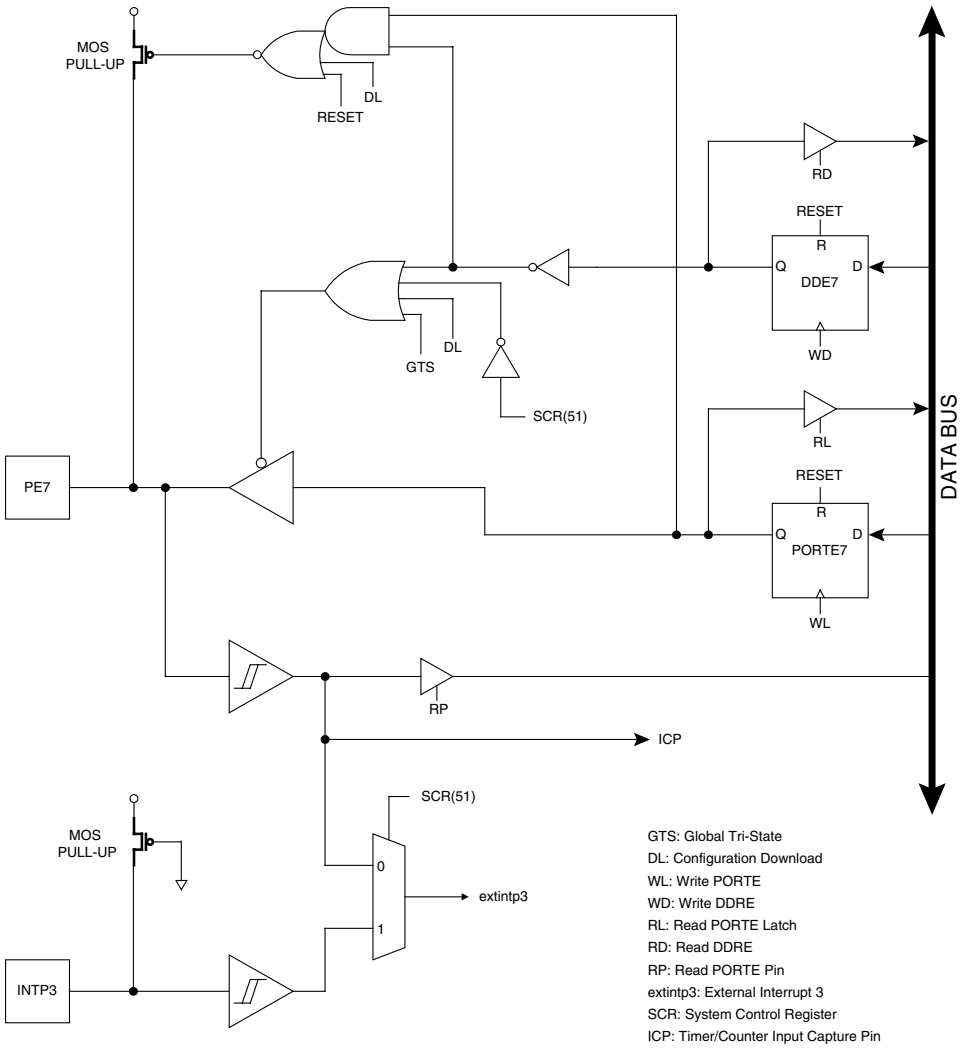


Table 56. AT94K Pin List (Continued)

AT94K05 96 FPGA I/O	AT94K10 192 FPGA I/O	AT94K40 384 FPGA I/O	Packages			
			PC84	TQ100	PQ144	PQ208
		I/O10				
		I/O11				
		I/O12				
		VCC(1)				
		GND				
		I/O13				
		I/O14				
I/O7	I/O7	I/O15				10
I/O8	I/O8	I/O16				11
	I/O9	I/O17				12
	I/O10	I/O18				13
		GND				
		I/O19				
		I/O20				
	I/O11	I/O21				
	I/O12	I/O22				
		I/O23				
		I/O24				
GND	GND	GND			8	14
I/O9, FCK1	I/O13, FCK1	I/O25, FCK1			9	15
I/O10	I/O14	I/O26			10	16
I/O11 (A20)	I/O15 (A20)	I/O27 (A20)	17	6	11	17
I/O12 (A21)	I/O16 (A21)	I/O28 (A21)	18	7	12	18
	VCC(1)	VCC(1)				
	I/O17	I/O29				
	I/O18	I/O30				
		GND				
		I/O31				
		I/O32				
		I/O33				
		I/O34				
		I/O35				
		I/O36				
		GND				

Notes:
1. VCC is I/O high voltage. Please refer to the “Designing in Split Power Supply Support for AT94KAL and AT94SAL Devices” application note.
2. VDD is core high voltage. Please refer to the “Designing in Split Power Supply Support for AT94KAL and AT94SAL Devices” application note.
3. Unbonded pins are No Connects.

Table 56. AT94K Pin List (Continued)

AT94K05 96 FPGA I/O	AT94K10 192 FPGA I/O	AT94K40 384 FPGA I/O	Packages			
			PC84	TQ100	PQ144	PQ208
I/O55	I/O83	I/O167			62	88
I/O56	I/O84	I/O168			63	89
GND	GND	GND			64	90
		I/O169				
		I/O170				
	I/O85	I/O171				
	I/O86	I/O172				
		I/O173				
		I/O174				
		GND				
		I/O175				
		I/O176				
	I/O87	I/O177				91
	I/O88	I/O178				92
I/O57	I/O89	I/O179				93
I/O58	I/O90	I/O180				94
		GND				
		VCC(1)				
		I/O181				
		I/O182				
I/O59 (TD2)	I/O91 (TD2)	I/O183 (TD2)	48	45	65	95
I/O60 (TD1)	I/O92 (TD1)	I/O184 (TD1)	49	46	66	96
		I/O185				
		I/O186				
		GND				
		I/O187				
		I/O188				
I/O61	I/O93	I/O189			67	97
I/O62	I/O94	I/O190			68	98
I/O63 (TD0)	I/O95 (TD0)	I/O191 (TD0)	50	47	69	99
I/O64, GCK4	I/O96, GCK4	I/O192, GCK4	51	48	70	100
GND	GND	GND	52	49	71	101
$\overline{CON}$	$\overline{CON}$	$\overline{CON}$	53	50	72	103
East Side						

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.