Microchip Technology - AT94K40AL-25DQI Datasheet

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	18 MHz
Interface	I²C, UART
Program SRAM Bytes	20K-32K
FPGA SRAM	18kb
EEPROM Size	-
Data SRAM Bytes	4K ~ 16K
FPGA Core Cells	2304
FPGA Gates	40K
FPGA Registers	2862
Voltage - Supply	3V ~ 3.6V
Mounting Type	Surface Mount
Operating Temperature	-40°C ~ 85°C
Package / Case	208-BFQFP
Supplier Device Package	208-PQFP (28x28)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at94k40al-25dqi

The Busing Network

Figure 3. Busing Network

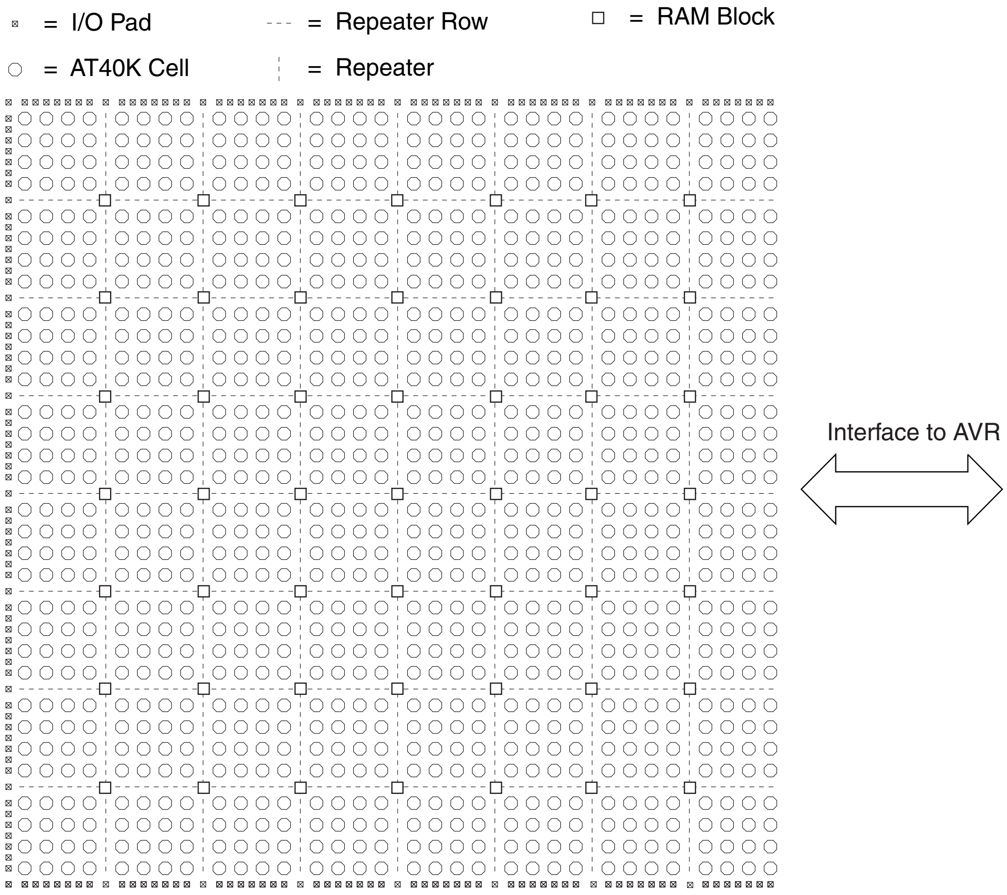
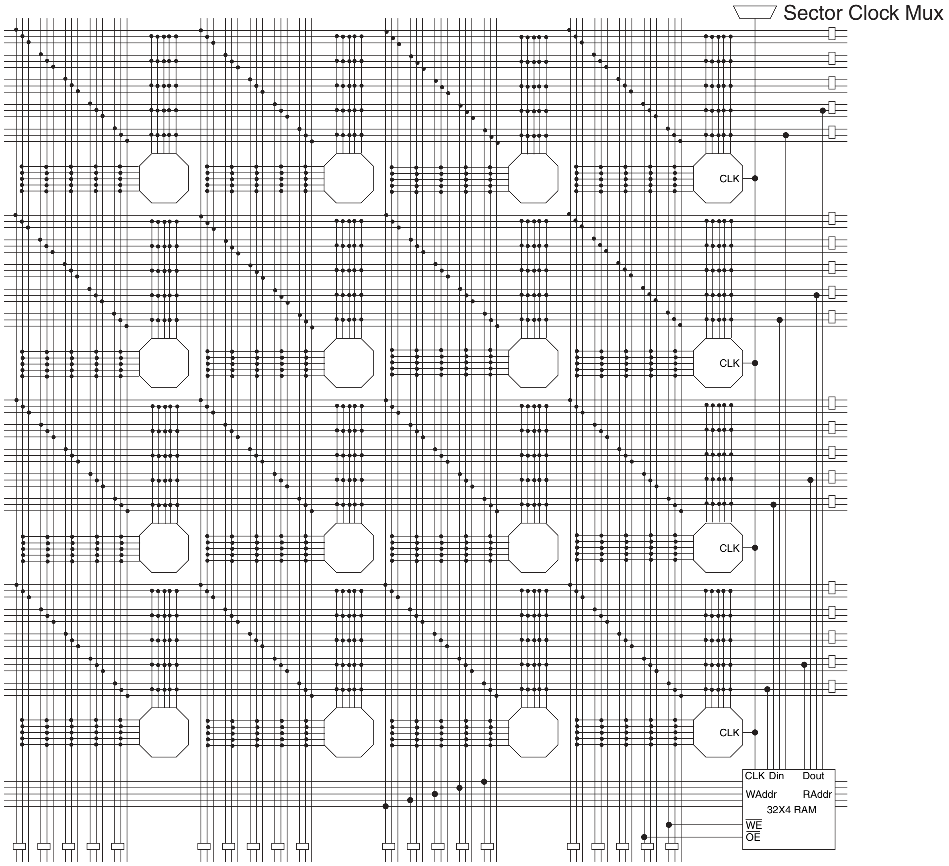


Figure 4 depicts one of five identical FPGA busing planes. Each plane has three bus resources: a local-bus resource (the middle bus) and two express-bus resources. Bus resources are connected via repeaters. Each repeater has connections to two adjacent local-bus segments and two express-bus segments.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

most RAM blocks, RAddr is on the left and WAddr is on the right. For the right-most RAM blocks, WAddr is on the left and RAddr is tied off. For single-ported RAM, WAddr is the READ/WRITE address port and Din is the (bi-directional) data port. The right-most RAM blocks can be used only for single-ported memories. $\overline{\text{WE}}$ and $\overline{\text{OE}}$ connect to the vertical express buses in the same column on Plane V_1 and V_2, respectively. WAddr, RAddr, $\overline{\text{WE}}$ and $\overline{\text{OE}}$ connect to express buses that are full length at array edge.

Reading and writing the 32 x 4 dual-port RAM are independent of each other. Reading the 32 x 4 dual-port RAM is completely asynchronous. Latches are transparent; when Load is logic 1, data flows through; when Load is logic 0, data is latched. Each bit in the 32 x 4 dual-port RAM is also a transparent latch. The front-end latch and the memory latch together and form an edge-triggered flip-flop. When a bit nibble is (Write) addressed and LOAD is logic 1 and $\overline{\text{WE}}$ is logic 0, DATA flows through the bit. When a nibble is not (Write) addressed or LOAD is logic 0 or $\overline{\text{WE}}$ is logic 1, DATA is latched in the nibble. The two CLOCK muxes are controlled together; they both select CLOCK or they both select “1”. CLOCK is obtained from the clock for the sector-column immediately to the left and immediately above the RAM block. Writing any value to the RAM Clear Byte during configuration clears the RAM, see Figure 5 and Figure 6.

Figure 8. FPGA RAM Connections (One RAM Block)



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

Table 11. FPSLIC System Control Register

Bit	Description
SCR32 - SCR34	Reserved
SCR35	0 = AVR Reset Pin Disabled 1 = AVR Reset Pin Enabled (active Low Reset) SCR35 allows the AVR Reset pin to reset the AVR only.
SCR36	0 = Protect AVR Program SRAM 1 = Allow Writes to AVR Program SRAM (Excluding Boot Block) SCR36 protects AVR program code from writes by the FPGA.
SCR37	0 = AVR Program SRAM Boot Block Protect 1 = AVR Program SRAM Boot Block Allows Overwrite
SCR38	0 = (default) Frame Clock Inverted to AVR Data/Program SRAM 1 = Non-inverting Clock Into AVR Data/Program SRAM
SCR39	Reserved

SCR40 - SCR41	SCR41 = 0, SCR40 = 0 16 Kbytes x 16 Program/4 Kbytes x 8 Data
	SCR41 = 0, SCR40 = 1 14 Kbytes x 16 Program/8 Kbytes x 8 Data
	SCR41 = 1, SCR40 = 0 12 Kbytes x 16 Program/12 Kbytes x 8 Data
	SCR41 = 1, SCR40 = 1 10 Kbytes x 16 Program/16 Kbytes x 8 Data
	SCR40 : SCR41 AVR program/data SRAM partitioning (set by using the AT94K Device Options in System Designer).

SCR 42 - SCR47	Reserved
SCR48	0 = EXT-INT0 Driven By Port E<4> 1 = EXT-INT0 Driven By INTP0 pad SCR48 : SCR53 Defaults dependent on package selected.
SCR49	0 = EXT-INT1 Driven By Port E<5> 1 = EXT-INT1 Driven By INTP1 pad SCR48 : SCR53 Defaults dependent on package selected.
SCR50	0 = EXT-INT2 Driven By Port E<6> 1 = EXT-INT2 Driven By INTP2 pad SCR48 : SCR53 Defaults dependent on package selected.
SCR51	0 = EXT-INT3 Driven By Port E<7> 1 = EXT-INT3 Driven By INTP3 pad SCR48 : SCR53 Defaults dependent on package selected.
SCR52	0 = UART0 Pins Assigned to Port E<1:0> 1 = UART0 Pins Assigned to UART0 pads SCR48 : SCR53 Defaults dependent on package selected.
SCR53	0 = UART1 Pins Assigned to Port E<3:2> 1 = UART1 Pins Assigned to UART1 pads SCR48 : SCR53 Defaults dependent on package selected. On packages less than 144-pins, there is reduced access to AVR ports. Port D is not available externally in the smallest package and Port E becomes dual-purpose I/O to maintain access to the UARTs and external interrupt pins. The Pin List (East Side) on page 177 shows exactly which pins are available in each package.
SCR54	0 = AVR Port D I/O With 6 mA Drive 1 = AVR Port D I/O With 20 mA Drive
SCR55	0 = AVR Port E I/O With 6 mA Drive 1 = AVR Port E I/O With 20 mA Drive

AVR Core and Peripherals

- AVR Core
- Watchdog Timer/On-chip Oscillator
- Oscillator-to-Internal Clock Circuit
- Oscillator-to-Timer/Counter for Real-time Clock
- 16-bit Timer/Counter and Two 8-bit Timer/Counters
- Interrupt Unit
- Multiplier
- UART (0)
- UART (1)
- I/O Port D (full 8 bits available on 144-pin or higher devices)
- I/O Port E

The embedded AVR core is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. The embedded AVR core achieves throughputs approaching 1 MIPS per MHz by executing powerful instructions in a single-clock-cycle, and allows the system architect 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 x 8 general-purpose working registers. All the 32 x 8 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent register bytes 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.

The embedded AVR core provides the following features: 16 general-purpose I/O lines, 32 x 8 general-purpose working registers, Real-time Counter (RTC), 3 flexible timer/counters with compare modes and PWM, 2 UARTs, programmable Watchdog Timer with internal oscillator, 2-wire serial port, and three software-selectable Power-saving modes. The Idle mode stops the CPU while allowing the SRAM, timer/counters, two-wire serial port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Power-save mode, the timer oscillator continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping.

The embedded AVR core is supported with a full suite of program and system development tools, including C compilers, macro assemblers, program debugger/simulators and evaluation kits.

Figure 29. The Parallel Instruction Fetches and Instruction Executions

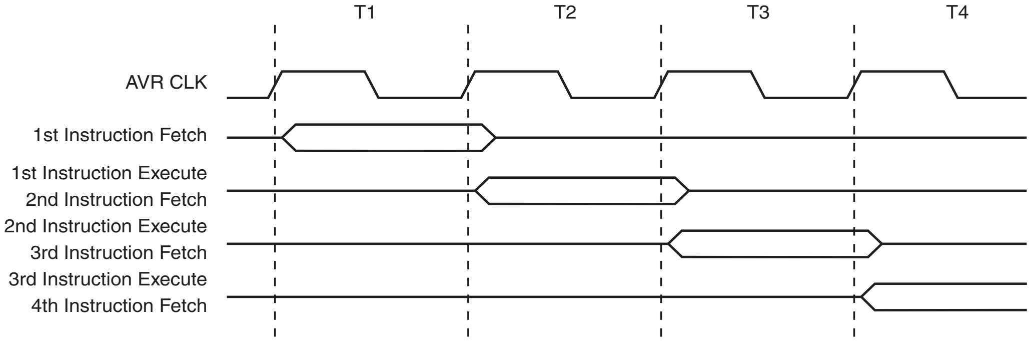
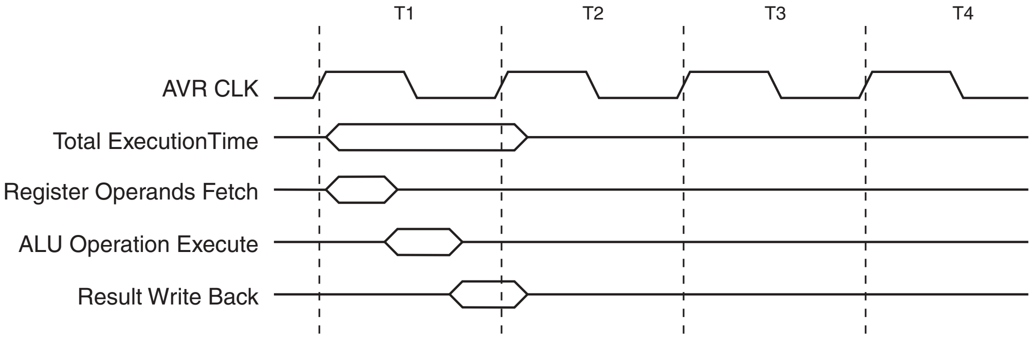


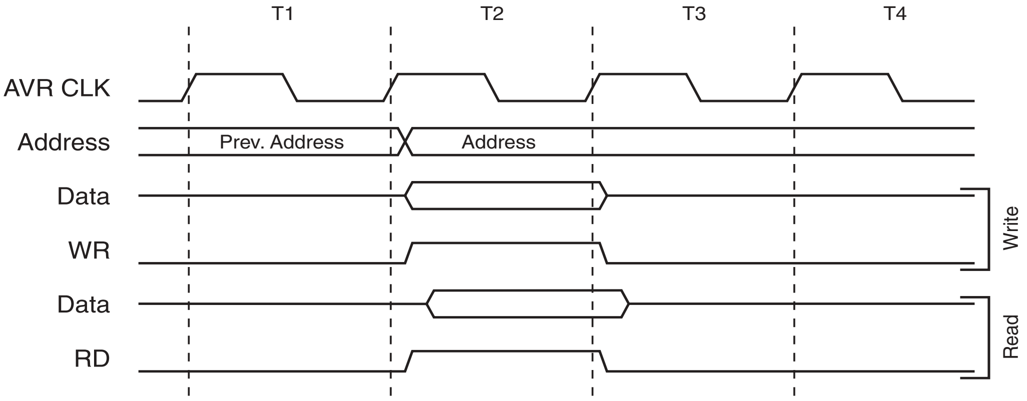
Figure 30 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

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



FPGA I/O Interrupt Control by AVR

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

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

Interrupt Control Registers – FISUA..D

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- **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.

Assuming Run-Test/Idle is the present state, a typical scenario for using the JTAG interface is

- At the TMS input, apply the sequence 1, 1, 0, 0 at the rising edges of TCK to enter the Shift Instruction Register - Shift-IR state. While TMS is Low, shift the 4 bit JTAG instructions into the JTAG instruction register from the TDI input at the rising edge of TCK, while the captured IR-state 0x01 is shifts out on the TDO pin. The JTAG Instruction selects a particular Data Register as path between TDI and TDO and controls the circuitry surrounding the selected Data Register.
- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. The instruction is latched onto the parallel output from the shift register path in the Update-IR state. The Exit-IR, Pause-IR, and Exit2-IR states are only used for navigating the state machine.
- At the TMS input, apply the sequence 1, 0, 0 at the rising edges of TCK to enter the Shift Data Register - Shift-DR state. While TMS is Low, upload the selected Data Register (selected by the present JTAG instruction in the JTAG Instruction Register) from the TDI input at the rising edge of TCK. At the same time, the parallel inputs to the Data Register captured in the Capture-DR state shifts out on the TDO pin.
- Apply the TMS sequence 1, 1, 0 to re-enter the Run-Test/Idle state. If the selected Data Register has a latched parallel-output, the latching takes place in the Update-DR state. The Exit-DR, Pause-DR, and Exit2-DR states are only used for navigating the state machine.

As shown in Figure 40 on page 70, the Run-Test/Idle[1] state need not be entered between selecting JTAG instruction and using Data Registers, and some JTAG instructions may select certain functions to be performed in the Run-Test/Idle, making it unsuitable as an Idle state.

Note: 1. Independent of the initial state of the TAP Controller, the Test-Logic-Reset state can always be entered by holding TMS High for 5 TCK clock periods.

Using the Boundary-scan Chain

A complete description of the Boundary-Scan capabilities are given in the section “IEEE 1149.1 (JTAG) Boundary-scan” on page 73.

Using the On-chip Debug System

As shown in Figure 39, the hardware support for On-Chip Debugging consists mainly of

- A scan chain on the interface between the internal AVR CPU and the internal peripheral units
- A breakpoint unit
- A communication interface between the CPU and JTAG system
- A scan chain on the interface between the internal AVR CPU and the FPGA
- A scan chain on the interface between the internal Program/Data SRAM and the FPGA

All read or modify/write operations needed for implementing the Debugger are done by applying AVR instructions via the internal AVR CPU Scan Chain. The CPU sends the result to an I/O memory mapped location which is part of the communication interface between the CPU and the JTAG system.

Part Number

The part number is a 16 bit code identifying the component. The JTAG Part Number for AVR devices is listed in Table 19.

Table 19. JTAG Part Number

Device	Part Number (Hex)
AT94K05	0xdd77
AT94K10	0xdd73
AT94K40	0xdd76

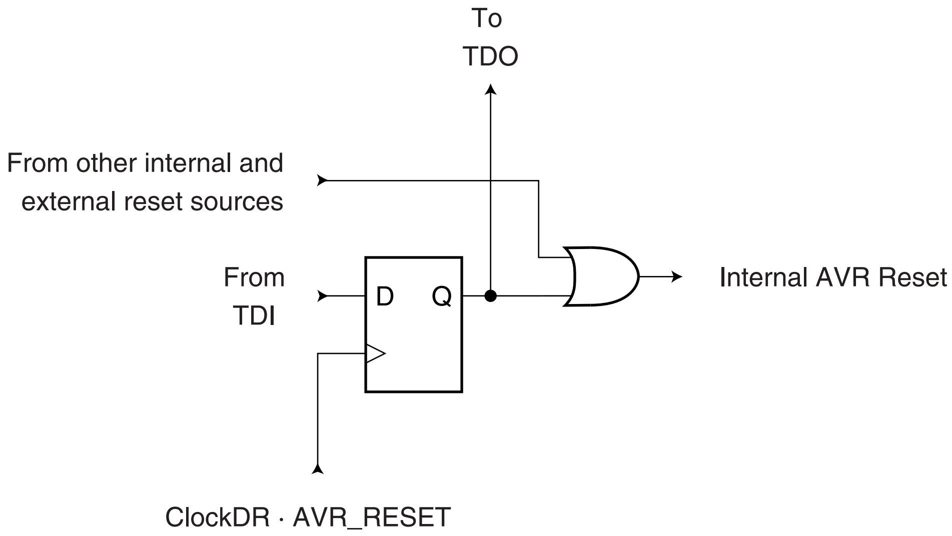
Manufacturer ID

The manufacturer ID for ATMEL is 0x01F (11 bits).

AVR Reset Register

The AVR Reset Register is a Test Data Register used to reset the AVR. A high value in the Reset Register corresponds to pulling the external AVRResetn Low. The AVR is reset as long as there is a high value present in the AVR Reset Register. Depending on the Bit settings for the clock options, the CPU will remain reset for a Reset Time-Out Period after releasing the AVR Reset Register. The output from this Data Register is not latched, so the reset will take place immediately, see Figure 42.

Figure 42. Reset Register



Boundary-scan Chain

The Boundary-scan Chain has the capability of driving and observing the logic levels on the AVR's digital I/O pins.

See "Boundary-scan Chain" on page 76 for a complete description.

Boundary-scan Specific JTAG Instructions

The instruction register is 4-bit wide, supporting up to 16 instructions. Listed below are the JTAG instructions useful for Boundary-Scan operation. Note that the optional HIGHZ instruction is not implemented.

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

The OPCODE for each instruction is shown behind the instruction name in hex format. The text describes which data register is selected as path between TDI and TDO for each instruction.

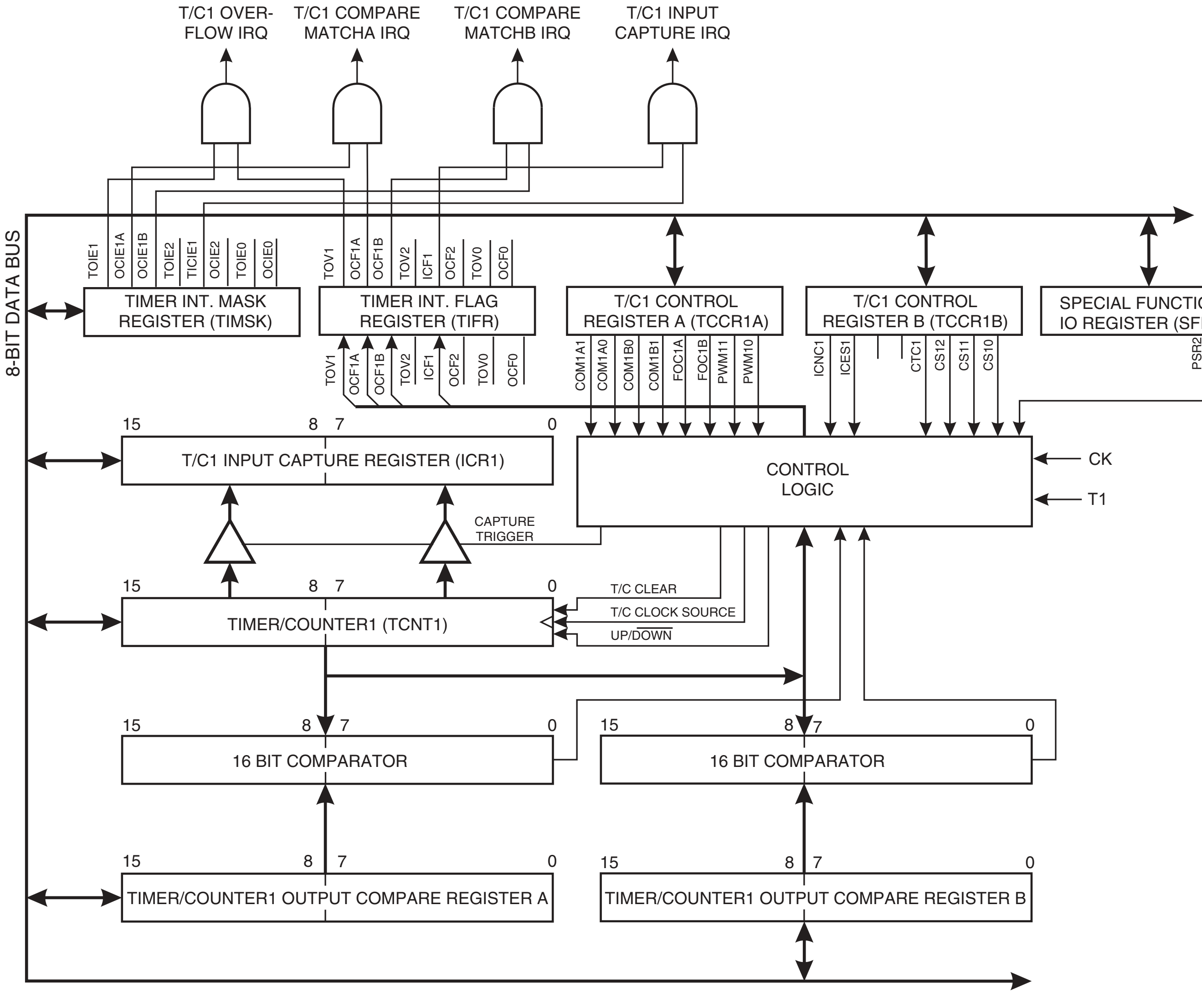
of the timer clock, that is, the timer is always advanced by at least one before the processor can read the counter value. The interrupt flags are updated three processor cycles after the processor clock has started. During these cycles, the processor executes instructions, but the interrupt condition 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



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)

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.

- **Bits 7..0 - 2-wire Serial Bit-rate Register**

TWBR selects the division factor for the bit-rate generator. The bit-rate generator is a frequency divider which generates the SCL clock frequency in the Master modes according to the following equation:

$$\text{Bit-rate} = \frac{f_{\text{CK}}}{16 + 2(\text{TWBR})}$$

- Bit-rate = SCL frequency
- f_{CK} = CPU Clock frequency
- TWBR = Contents of the 2-wire Serial Bit Rate Register

Both the receiver and the transmitter can stretch the Low period of the SCL line when waiting for user response, thereby reducing the average bit rate.

The 2-wire Serial Control Register – TWCR

Bit	7	6	5	4	3	2	1	0	
$36 ($56)	**TWINT**	**TWEA**	**TWSTA**	**TWSTO**	**TWWC**	**TWEN**	**-**	**TWIE**	**TWCR**
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - TWINT: 2-wire Serial Interrupt Flag**

This bit is set by the hardware when the 2-wire Serial Interface has finished its current job and expects application software response. If the I-bit in the SREG and TWIE in the TWCR register are set (one), the MCU will jump to the interrupt vector at address $0046. While the TWINT flag is set, the bus SCL clock line Low period is stretched. The TWINT flag must be cleared by software by writing a logic 1 to it. Note that this flag is not automatically cleared by the hardware when executing the interrupt routine. Also note that clearing this flag starts the operation of the 2-wire Serial Interface, so all accesses to the 2-wire Serial Address Register – TWAR, 2-wire Serial Status Register – TWSR, and 2-wire Serial Data Register – TWDR must be complete before clearing this flag.

- **Bit 6 - TWEA: 2-wire Serial Enable Acknowledge Flag**

TWEA flag controls the generation of the acknowledge pulse. If the TWEA bit is set, the ACK pulse is generated on the 2-wire Serial Bus if the following conditions are met:

- The device's own Slave address has been detected
- A general call has been received, while the TWGCE bit in the TWAR is set
- A data byte has been received in Master Receiver or Slave Receiver mode

By setting the TWEA bit Low the device can be virtually disconnected from the 2-wire Serial Bus temporarily. Address recognition can then be resumed by setting the TWEA bit again.

- **Bit 5 - TWSTA: 2-wire Serial Bus START Condition Flag**

The TWSTA flag is set by the CPU when it desires to become a Master on the 2-wire Serial Bus. The 2-wire serial hardware checks if the bus is available, and generates a Start condition on the bus if the bus is free. However, if the bus is not free, the 2-wire Serial Interface waits until a STOP condition is detected, and then generates a new Start condition to claim the bus Master status.

Figure 71. Formats and States in the Master Transmitter Mode

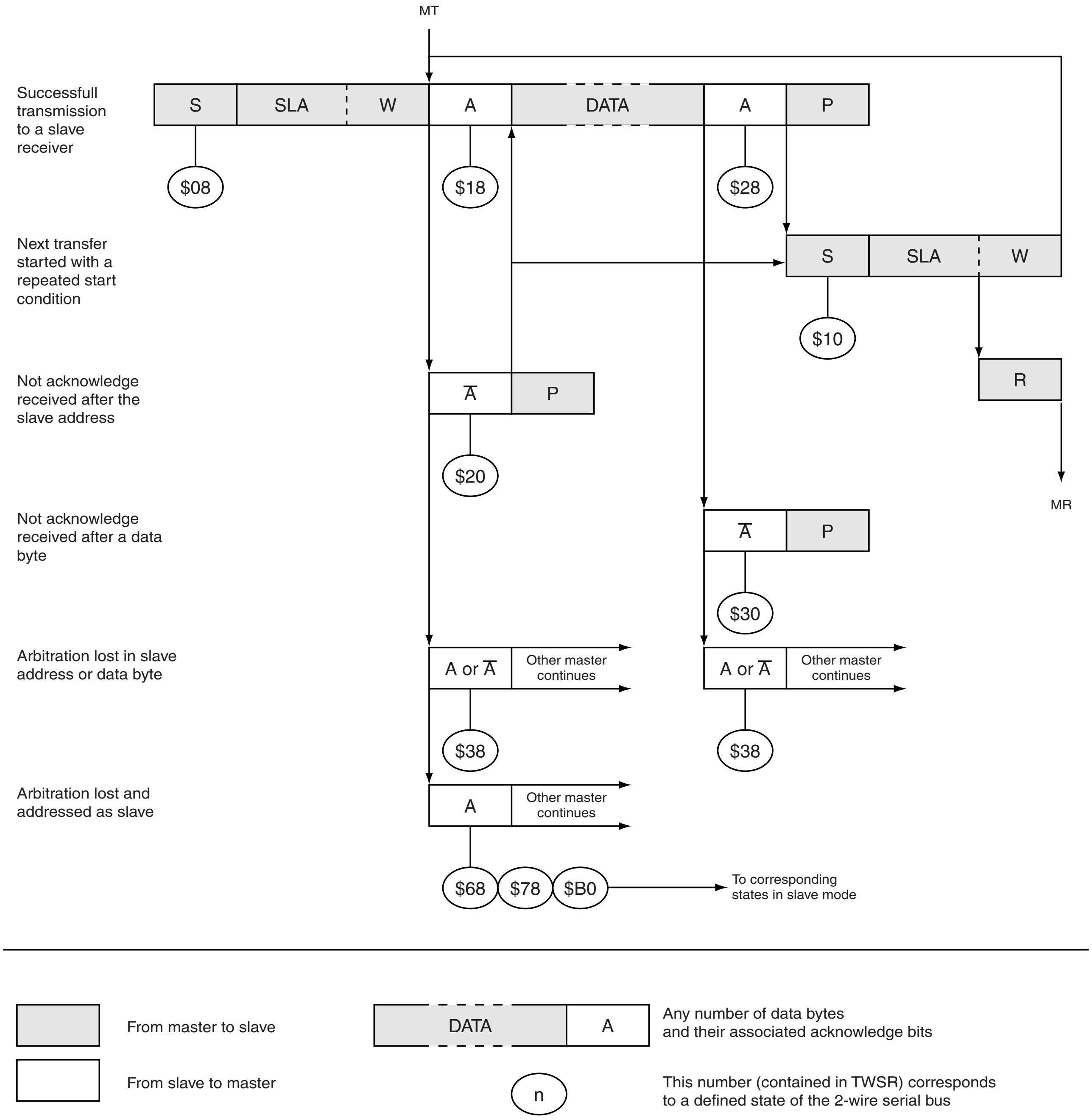

MT
Successfull transmission to a slave receiver
S
SLA
W
A
DATA
A
P
$08
$18
$28
Next transfer started with a repeated start condition
S
SLA
W
$10
R
MR
Not acknowledge received after the slave address
$\overline{A}$
P
$20
Not acknowledge received after a data byte
$\overline{A}$
P
$30
Arbitration lost in slave address or data byte
A or $\overline{A}$
Other master continues
$38
A or $\overline{A}$
Other master continues
$38
Arbitration lost and addressed as slave
A
Other master continues
$68
$78
$B0
To corresponding states in slave mode
From master to slave
From slave to master
DATA
A
Any number of data bytes and their associated acknowledge bits
n
This number (contained in TWSR) corresponds to a defined state of the 2-wire serial bus

Figure 74. Formats and States in the Slave Transmitter Mode

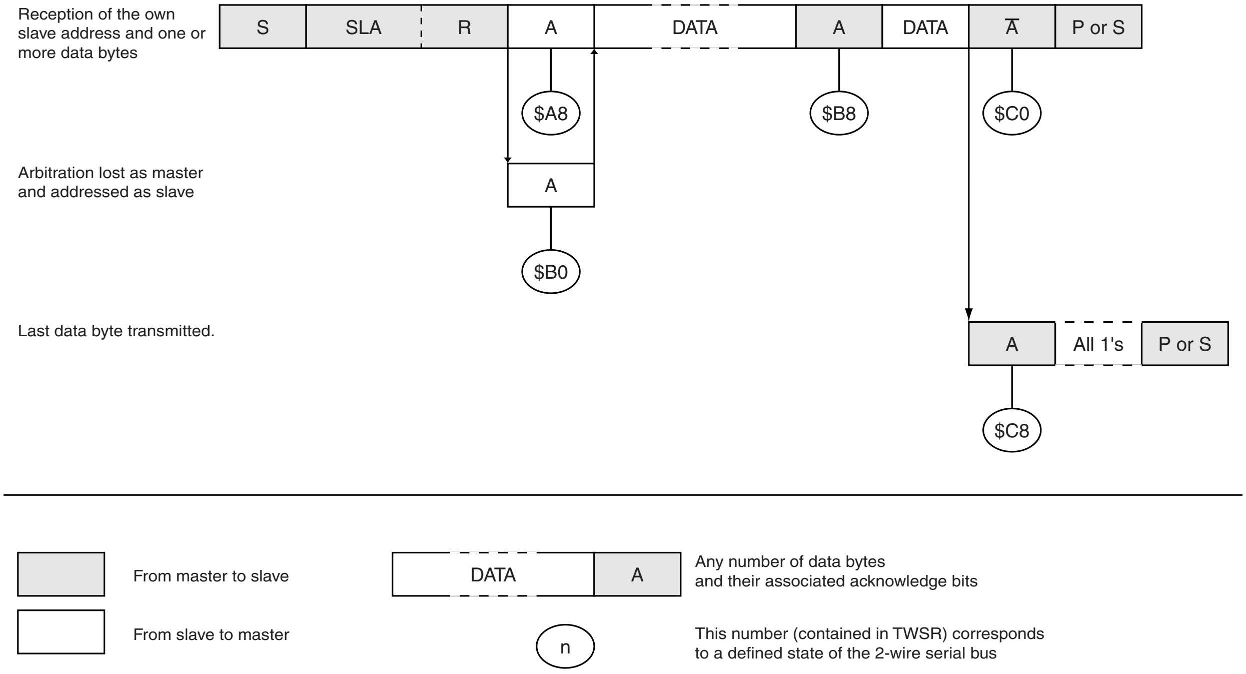
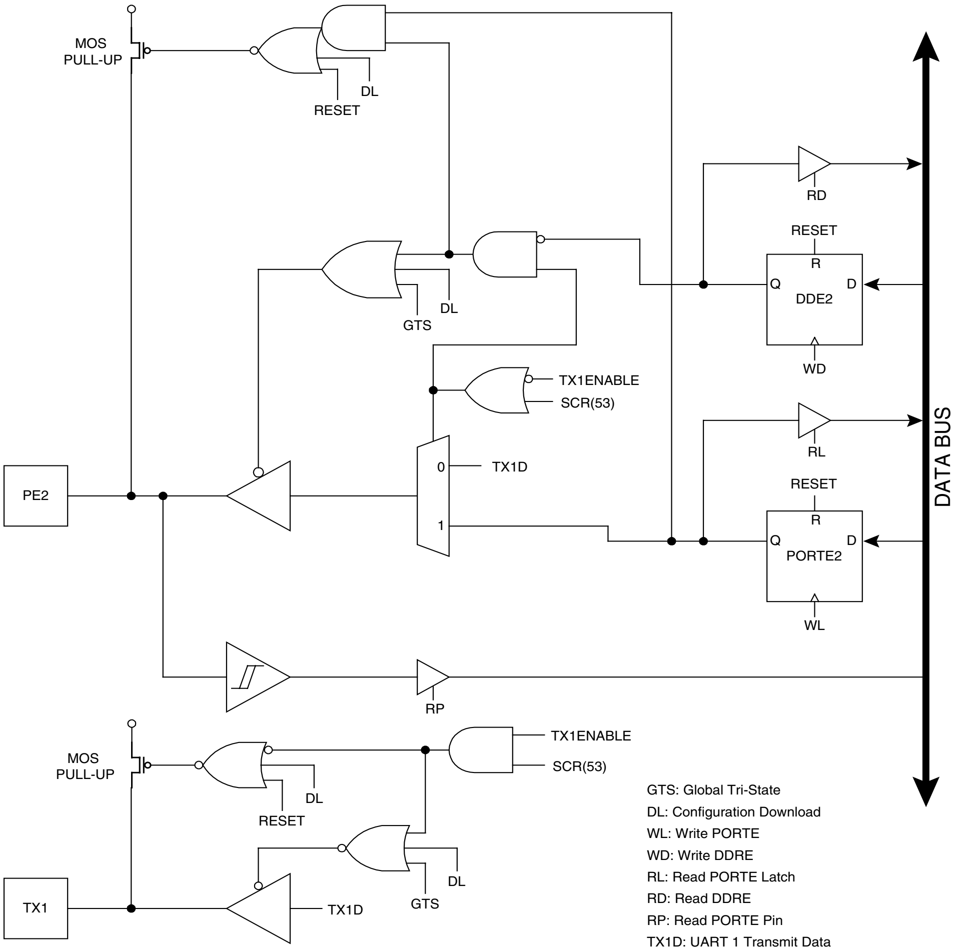

Reception of the own slave address and one or more data bytes
S
SLA
R
A
DATA
A
DATA
A̅
P or S
$A8
$B8
$C0
Arbitration lost as master and addressed as slave
A
$B0
Last data byte transmitted.
A
All 1's
P or S
$C8
From master to slave
From slave to master
DATA
A
Any number of data bytes and their associated acknowledge bits
n
This number (contained in TWSR) corresponds to a defined state of the 2-wire serial bus

Table 45. Status Codes for Miscellaneous States

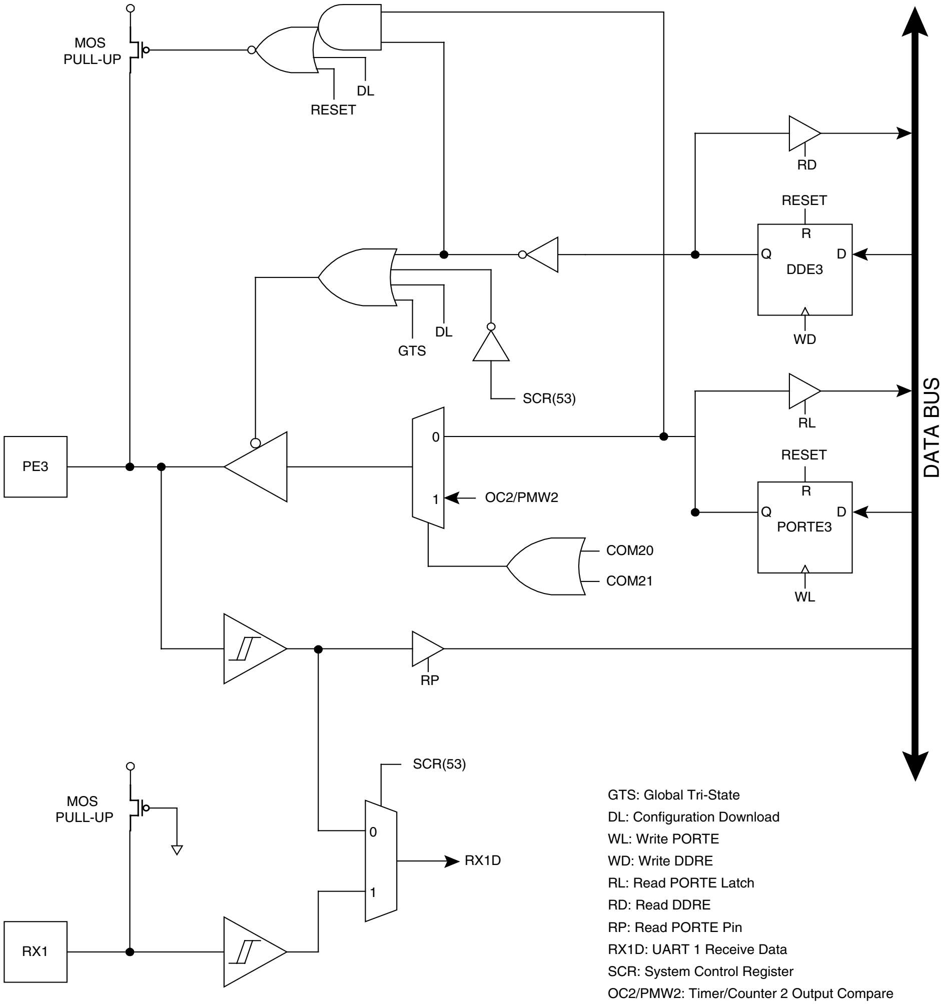
Status Code (TWSR)	Status of the 2-wire Serial Bus and 2-wire Serial Hardware	Application Software Response					Next Action Taken by 2-wire Serial Hardware
			To TWCR				
		To/From TWDR	STA	STO	TWINT	TWEA	
$F8	No relevant state information available; TWINT = “0”	No TWDR action	No TWCR action				Wait or proceed current transfer
$00	Bus error due to an illegal START or STOP condition	No TWDR action	0	1	1	X	Only the internal hardware is affected; no STOP condition is sent on the bus. In all cases, the bus is released and TWSTO is cleared.

Figure 78. PortE Schematic Diagram (Pin PE2)



GTS: Global Tri-State
DL: Configuration Download
WL: Write PORTE
WD: Write DDRE
RL: Read PORTE Latch
RD: Read DDRE
RP: Read PORTE Pin
TX1D: UART 1 Transmit Data
TX1ENABLE: UART 1 Transmit Enable
SCR: System Control Register

PortE Schematic Diagram (Pin PE3)



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

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 of Contents