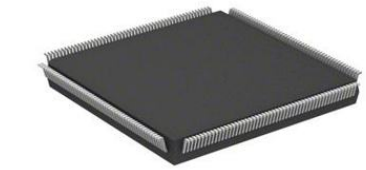


Welcome to E-XFL.COM

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

Embedded - FPGAs (Field Programmable Gate Arrays) with Microcontrollers represent a cutting-edge category of electronic components that combine the flexibility of FPGA technology with the processing power of integrated microcontrollers. This hybrid approach offers a versatile solution for designing and implementing complex digital systems that require both programmable logic and embedded processing capabilities.

What Are Embedded - FPGAs with Microcontrollers?

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

Details	
Product Status	Obsolete
Core Type	8-Bit AVR
Speed	25 MHz
Interface	I²C, UART
Program SRAM Bytes	4K-16K
FPGA SRAM	2kb
EEPROM Size	-
Data SRAM Bytes	4K ~ 16K
FPGA Core Cells	256
FPGA Gates	5K
FPGA Registers	436
Voltage - Supply	3V ~ 3.6V
Mounting Type	Surface Mount
Operating Temperature	-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/at94k05al-25dqi

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

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

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

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

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

Table 3. Clock Activity in Various Modes

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

Figure 17. Corner I/Os

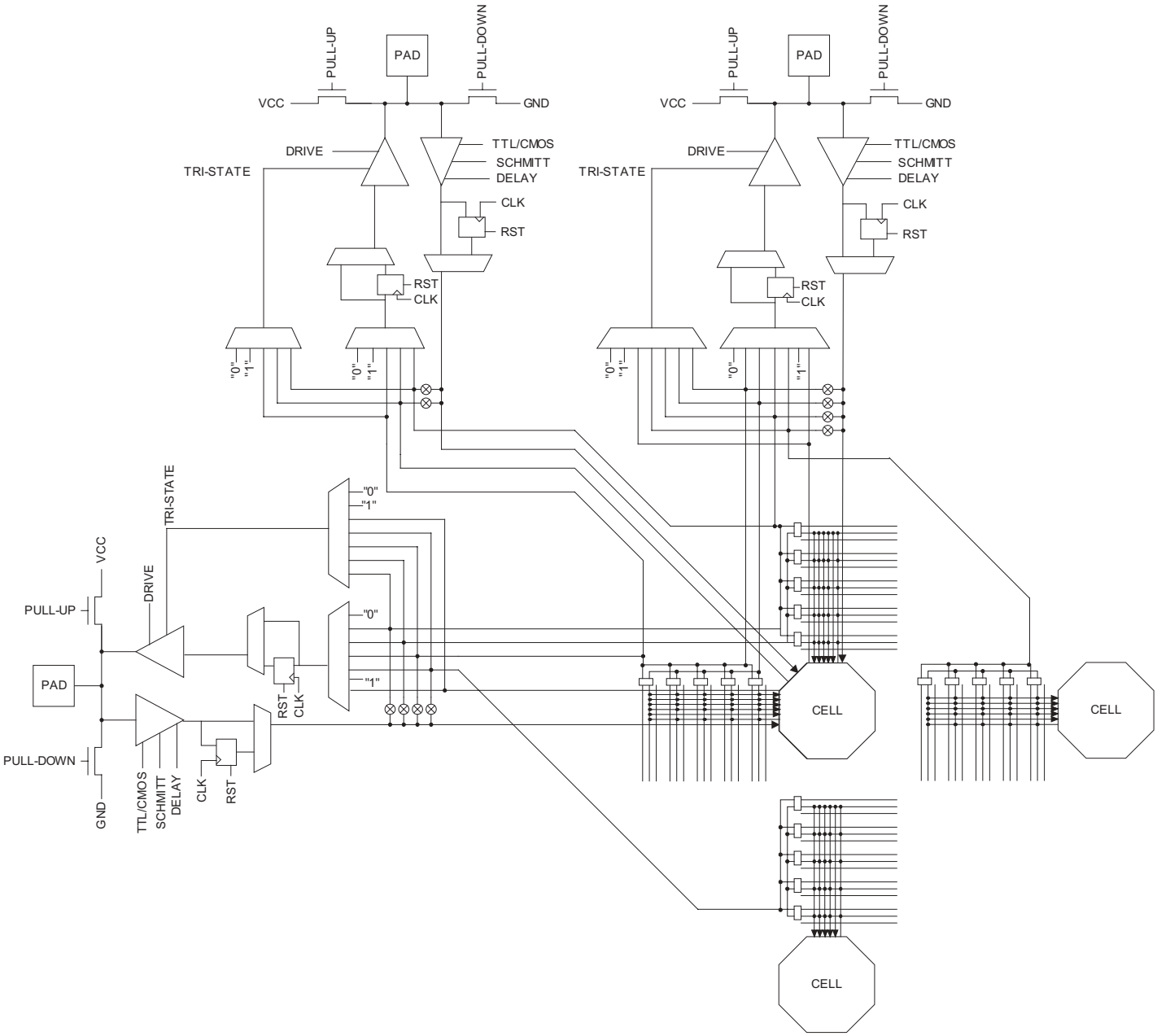


Figure 21. AVR SRAM Data Memory Write Using “ST” Instruction

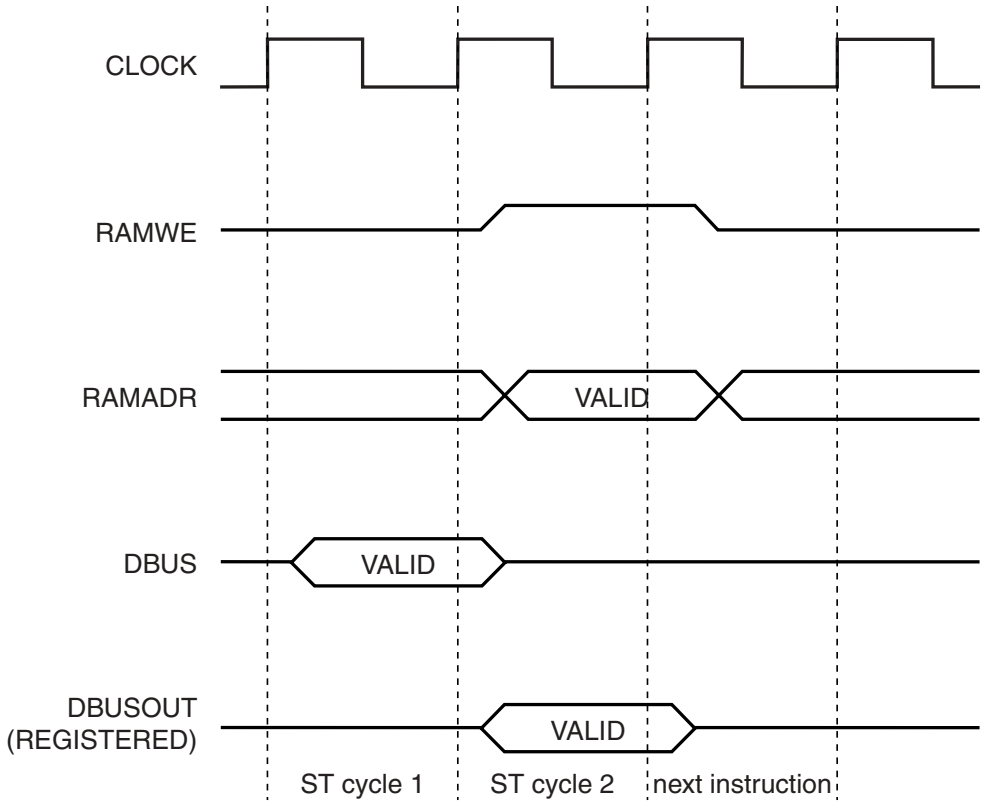


Figure 22. AVR SRAM Data Memory Read Using “LD” Instruction

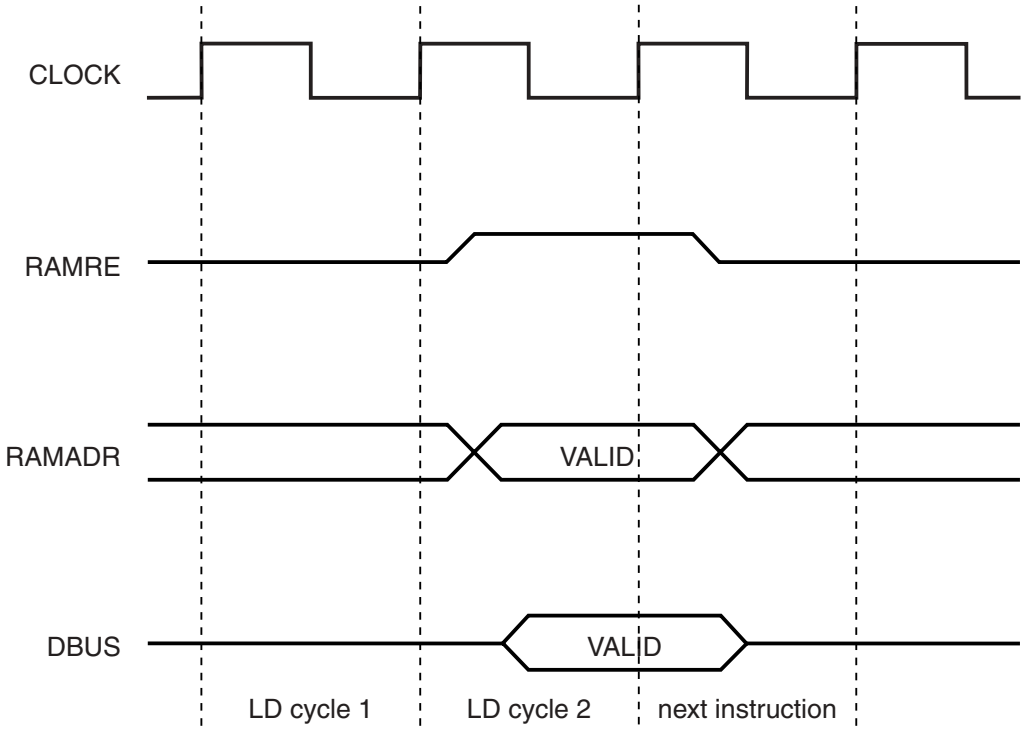


Table 11. FPSLIC System Control Register

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.

X-register, Y-register and Z-register

Registers R26..R31 have some added functions to their general-purpose usage. These registers are address pointers for indirect addressing of the SRAM. The three indirect address registers X, Y and Z have functions as fixed displacement, automatic increment and decrement (see the descriptions for the different instructions).

ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general-purpose working registers. Within a single clock cycle, ALU operations between registers in the register file are executed. The ALU operations are divided into three main categories – arithmetic, logical and bit-functions.

Multiplier Unit

The high-performance AVR Multiplier operates in direct connection with all the 32 general-purpose working registers. This unit performs 8 x 8 multipliers every two clock cycles. See multiplier details on page 106.

SRAM Data Memory

External data SRAM (or program) cannot be used with the FPSLIC AT94K family.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement and Indirect with Post-increment. In the register file, registers R26 to R31 feature the indirect addressing pointer registers.

The Indirect with Displacement mode features a 63 address locations reach from the base address given by the Y- or Z-register.

When using register indirect addressing modes with automatic Pre-decrement and Post-increment, the address registers X, Y and Z are decremented and incremented.

The entire data address space including the 32 general-purpose working registers and the 64 I/O registers are all accessible through all these addressing modes. See the next section for a detailed description of the different addressing modes.

Program and Data Addressing Modes

The embedded AVR core supports powerful and efficient addressing modes for access to the program memory (SRAM) and data memory (SRAM, Register File and I/O Memory). This section describes the different addressing modes supported by the AVR architecture.

Register Direct, Single-register Rd

The operand is contained in register d (Rd).

Register Direct, Two Registers Rd and Rr

Operands are contained in register r (Rr) and d (Rd). The result is stored in register d (Rd).

I/O Direct

Operand address is contained in 6 bits of the instruction word. *n* is the destination or source register address.

Data Direct

A 16-bit data address is contained in the 16 LSBs of a two-word instruction. Rd/Rr specify the destination or source register.

Data Indirect with Displacement

Operand address is the result of the Y- or Z-register contents added to the address contained in 6 bits of the instruction word.

AT94K Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reference Page
$16 ($36)	FISUC	FPGA I/O Select, Interrupt Mask/Flag Register C (Reserved on AT94K05)								54, 56
$15 ($35)	FISUB	FPGA I/O Select, Interrupt Mask/Flag Register B								54, 56
$14 ($34)	FISUA	FPGA I/O Select, Interrupt Mask/Flag Register A								54, 56
$13 ($33)	FISCR	FIADR						XFIS1	XFIS0	53
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	124
$11 ($31)	DDRD	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0	124
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	124
$0F ($2F)	Reserved									
$0E ($2E)	Reserved									
$0D ($2D)	Reserved									
$0C ($2C)	UDR0	UART0 I/O Data Register								101
$0B ($2B)	UCSR0A	RXC0	TXC0	UDRE0	FE0	OR0		U2X0	MPCM0	101
$0A ($2A)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	CHR90	RXB80	TXB80	103
$09 ($29)	UBRR0	UART0 Baud-rate Register								105
$08 ($28)	OCDR (Reserved)	IDRD								Reserved[1]
$07 ($27)	PORTE	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0	126
$06 ($26)	DDRE	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0	126
$05 ($25)	PINE	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0	126
$04 ($24)	Reserved									
$03 ($23)	UDR1	UART1 I/O Data Register								101
$02 ($22)	UCSR1A	RXC1	TXC1	UDRE1	FE1	OR1		U2X1	MPCM1	101
$01 ($21)	UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	CHR91	RXB81	TXB81	103
$00 ($20)	UBRR1	UART1 Baud-rate Register								105

Note: 1. The On-chip Debug Register (OCDR) is detailed on the “FPSLIC On-chip Debug System” distributed within Atmel and select third-party vendors only under Non-Disclosure Agreement (NDA). Contact fpslic@atmel.com for a copy of this document.

The embedded AVR core I/Os and peripherals, and all the virtual FPGA peripherals are placed in the I/O space. The different I/O locations are directly accessed by the IN and OUT instructions transferring data between the 32 x 8 general-purpose working registers and the I/O space. I/O registers within the address range $00 – $1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. When using the I/O specific instructions IN, OUT, the I/O register address $00 – $3F are used, see Figure 32. When addressing I/O registers as SRAM, $20 must be added to this address. All I/O register addresses throughout this document are shown with the SRAM address in parentheses.

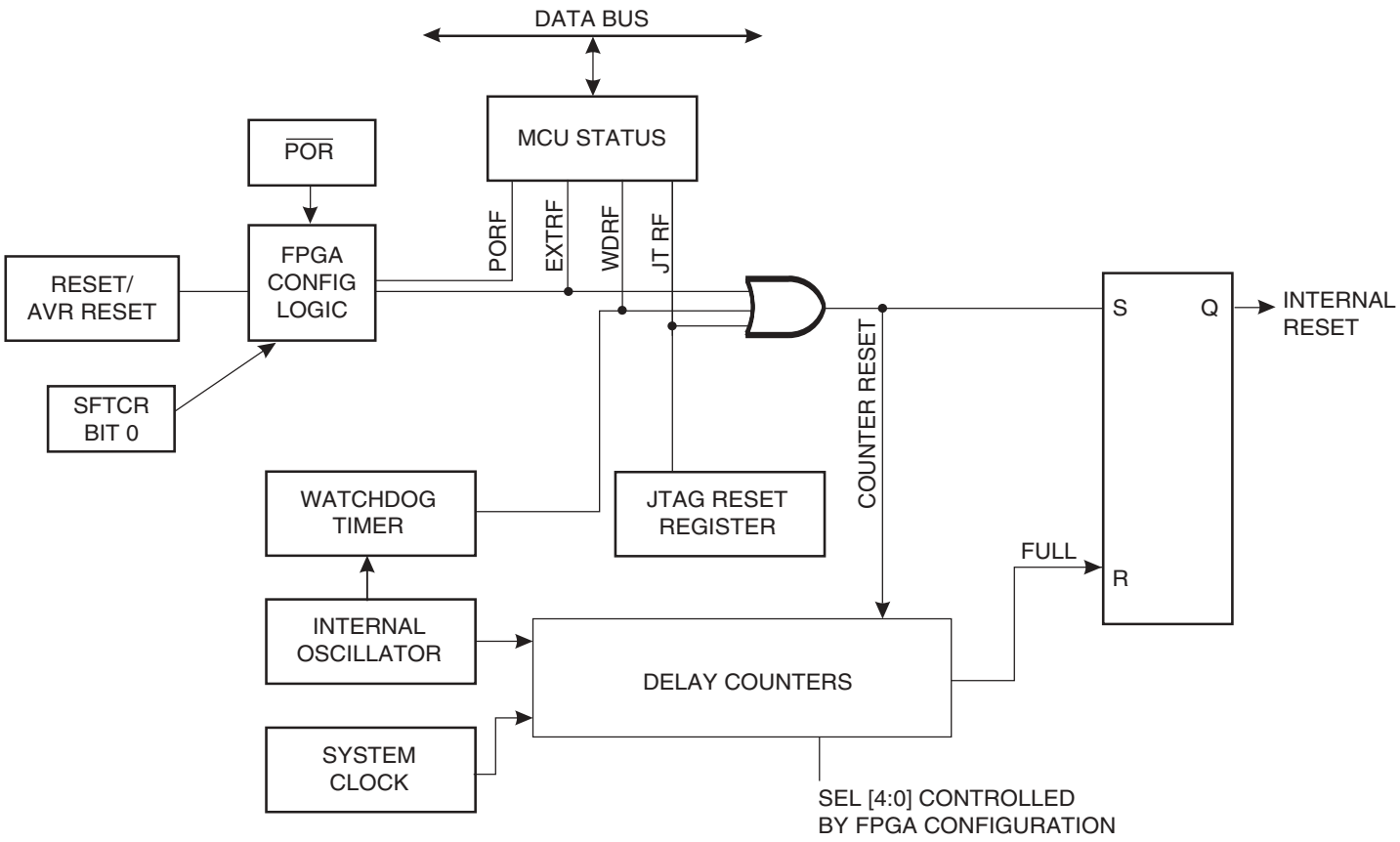
Reset Sources

The embedded AVR core has five sources of reset:

- External Reset. The MCU is reset immediately when a low-level is present on the RESET or AVR RESET pin.
- Power-on Reset. The MCU is reset upon chip power-up and remains in reset until the FPGA configuration has entered Idle mode.
- Watchdog Reset. The MCU is reset when the Watchdog Timer period expires and the watchdog is enabled.
- Software Reset. The MCU is reset when the SRST bit in the Software Control register is set (one).
- JTAG AVR Reset. The MCU is reset as long as there is a logic one in the Reset Register, one of the scan chains of the JTAG system. See “IEEE 1149.1 (JTAG) Boundary-scan” on page 73.

During reset, all I/O registers except the MCU Status register are then set to their Initial Values, and the program starts execution from address $0000. The instruction placed in address $0000 must be a JMP – absolute jump instruction to the reset handling routine. If the program never enables an interrupt source, the interrupt vectors are not used, and regular program code can be placed at these locations. The circuit diagram in Figure 35 shows the reset logic. Table 16 defines the timing and electrical parameters of the reset circuitry.

Figure 35. Reset Logic



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

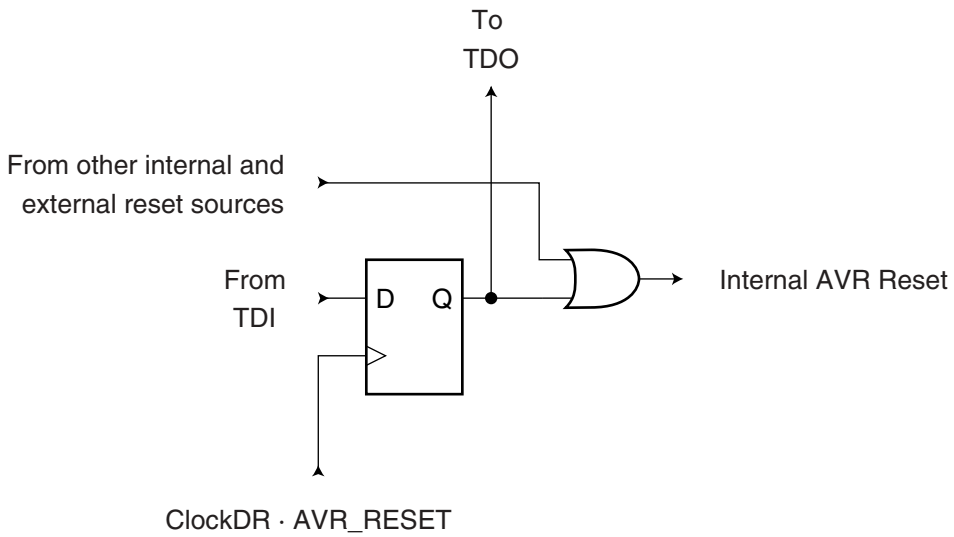
Manufacturer ID

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.

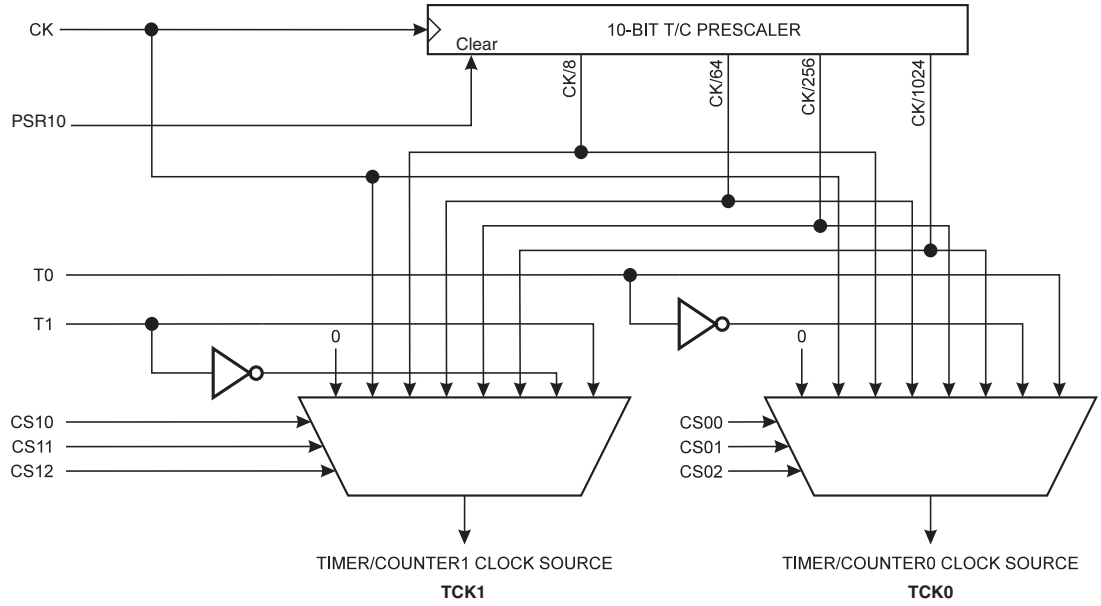
Timer/Counters

The FPSLIC provides three general-purpose Timer/Counters: two 8-bit T/Cs and one 16-bit T/C. Timer/Counter2 can optionally be asynchronously clocked from an external oscillator. This oscillator is optimized for use with a 32.768 kHz watch crystal, enabling use of Timer/Counter2 as a Real-time Clock (RTC). Timer/Counters 0 and 1 have individual prescaling selection from the same 10-bit prescaling timer. Timer/Counter2 has its own prescaler. Both these prescalers can be reset by setting the corresponding control bits in the Special Functions I/O Register (SFIOR). See “Special Function I/O Register – SFIOR” on page 86 for a detailed description. These Timer/Counters can either be used as a timer with an internal clock time-base or as a counter with an external pin connection which triggers the counting.

Timer/Counter Prescalers

For Timer/Counters 0 and 1, see Figure 48, the four prescaled selections are: CK/8, CK/64, CK/256 and CK/1024, where CK is the oscillator clock. For the two Timer/Counters 0 and 1, CK, external source, and stop, can also be selected as clock sources. Setting the PSR10 bit in SFIOR resets the prescaler. This allows the user to operate with a predictable prescaler. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a prescaler reset will affect both Timer/Counters.

Figure 48. Prescaler for Timer/Counter0 and 1



The clock source for Timer/Counter2 prescaler, see Figure 49, is named PCK2. PCK2 is by default connected to the main system clock CK. By setting the AS2 bit in ASSR, Timer/Counter2 is asynchronously clocked from the TOSC1 pin. This enables use of Timer/Counter2 as a Real-time Clock (RTC). When AS2 is set, pins TOSC1 and TOSC2 are disconnected from Port D. A crystal can then be connected between the TOSC1 and TOSC2 pins to serve as an independent clock source for Timer/Counter2. The oscillator is optimized for use with a 32.768 kHz crystal. Alternatively, an external clock signal can be applied to TOSC1. The frequency of this clock must be lower than one fourth of the CPU clock and not higher than 1 MHz. Setting the PSR2 bit in SFIOR resets the prescaler. This allows the user to operate with a predictable prescaler.

The 16-bit Timer/Counter1 can select the clock source from CK, prescaled CK, or an external pin. In addition it can be stopped as described in section “Timer/Counter1 Control Register B – TCCR1B” on page 98. The different status flags (overflow, compare match and capture event) are found in the Timer/Counter Interrupt Flag Register – TIFR. Control signals are found in the Timer/Counter1 Control Registers – TCCR1A and TCCR1B. The interrupt enable/disable settings for Timer/Counter1 are found in the Timer/Counter Interrupt Mask Register – TIMSK.

When Timer/Counter1 is externally clocked, the external signal is synchronized with the oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

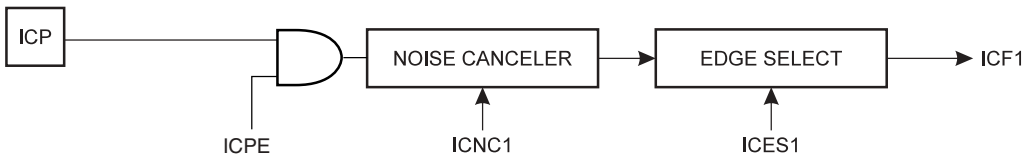
The 16-bit Timer/Counter1 features both a high-resolution and a high-accuracy usage with the lower prescaling opportunities. Similarly, the high-prescaling opportunities makes the Timer/Counter1 useful for lower speed functions or exact-timing functions with infrequent actions.

The Timer/Counter1 supports two Output Compare functions using the Output Compare Register 1 A and B – OCR1A and OCR1B as the data sources to be compared to the Timer/Counter1 contents. The Output Compare functions include optional clearing of the counter on compareA match, and actions on the Output Compare pins on both compare matches.

Timer/Counter1 can also be used as a 8-, 9- or 10-bit Pulse Width Modulator. In this mode, the counter and the OCR1A/OCR1B registers serve as a dual-glitch-free stand-alone PWM with centered pulses. Alternatively, the Timer/Counter1 can be configured to operate at twice the speed in PWM mode, but without centered pulses. Refer to page 101 for a detailed description on this function.

The Input Capture function of Timer/Counter1 provides a capture of the Timer/Counter1 contents to the Input Capture Register – ICR1, triggered by an external event on the Input Capture Pin – PE7(ICP). The actual capture event settings are defined by the Timer/Counter1 Control Register – TCCR1B.

Figure 55. ICP Pin Schematic Diagram



ICPE: Input Capture Pin Enable

If the noise canceler function is enabled, the actual trigger condition for the capture event is monitored over four samples, and all four must be equal to activate the capture flag.

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)

Example 3 – Multiply-accumulate Operation

The final example of 8-bit multiplication shows a multiply-accumulate operation. The general formula can be written as:

$$c(n) = a(n) \times b + c(n-1)$$

```
                ; r17:r16 = r18 * r19 + r17:r16

in    r18,PINB  ; Get the current pin value on port B
ldi   r19,b     ; Load constant b into r19
muls  r19,r18   ; r1:r0 = variable A * variable B
add   r16,r0    ; r17:r16 += r1:r0
adc   r17,r1
```

Typical applications for the multiply-accumulate operation are FIR (Finite Impulse Response) and IIR (Infinite Impulse Response) filters, PID regulators and FFT (Fast Fourier Transform). For these applications the FMULS instruction is particularly useful. The main advantage of using the FMULS instruction instead of the MULS instruction is that the 16-bit result of the FMULS operation always may be approximated to a (well-defined) 8-bit format, see “Using Fractional Numbers” on page 111.

16-bit Multiplication

The new multiply instructions are specifically designed to improve 16-bit multiplication. This section presents solutions for using the hardware multiplier to do multiplication with 16-bit operands.

Figure 60 schematically illustrates the general algorithm for multiplying two 16-bit numbers with a 32-bit result (C = A • B). AH denotes the high byte and AL the low byte of the A operand. CMH denotes the middle high byte and CML the middle low byte of the result C. Equal notations are used for the remaining bytes.

The algorithm is basic for all multiplication. All of the partial 16-bit results are shifted and added together. The sign extension is necessary for signed numbers only, but note that the carry propagation must still be done for unsigned numbers.

Figure 60. 16-bit Multiplication, General Algorithm

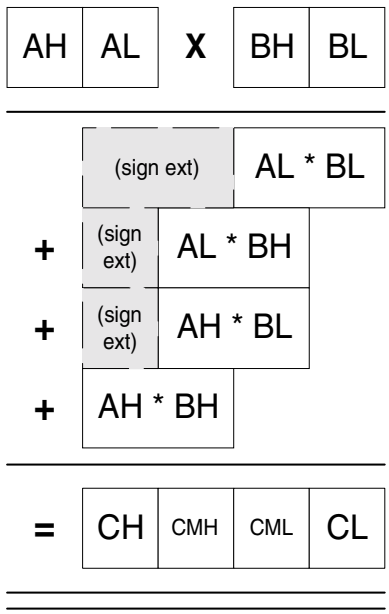


Table 42. Status Codes for Master Receiver Mode

Status Code (TWSR)	Status of the 2-wire Serial Bus and 2-wire Serial Hardware	Application Software Response					Next Action Taken by 2-wire Serial Hardware
		To/From TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$08	A START condition has been transmitted	Load SLA+R	X	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
$10	A repeated START condition has been transmitted	Load SLA+R or	X	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
		Load SLA+W	X	0	1	X	SLA+W will be transmitted Logic will switch to Master Transmitter mode
$38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or	0	0	1	X	2-wire serial bus will be released and not addressed Slave mode will be entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free
$40	SLA+R has been transmitted; ACK has been received	No TWDR action or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	0	0	1	1	Data byte will be received and ACK will be returned
$48	SLA+R has been transmitted; NOT ACK has been received	No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset
$50	Data byte has been received; ACK has been returned	Read data byte or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	0	0	1	1	Data byte will be received and ACK will be returned
$58	Data byte has been received; NOT ACK has been returned	Read data byte or	1	0	1	X	Repeated START will be transmitted
		Read data byte or	0	1	1	X	STOP condition will be transmitted and TWSTO flag will be reset
		Read data byte	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO flag will be reset

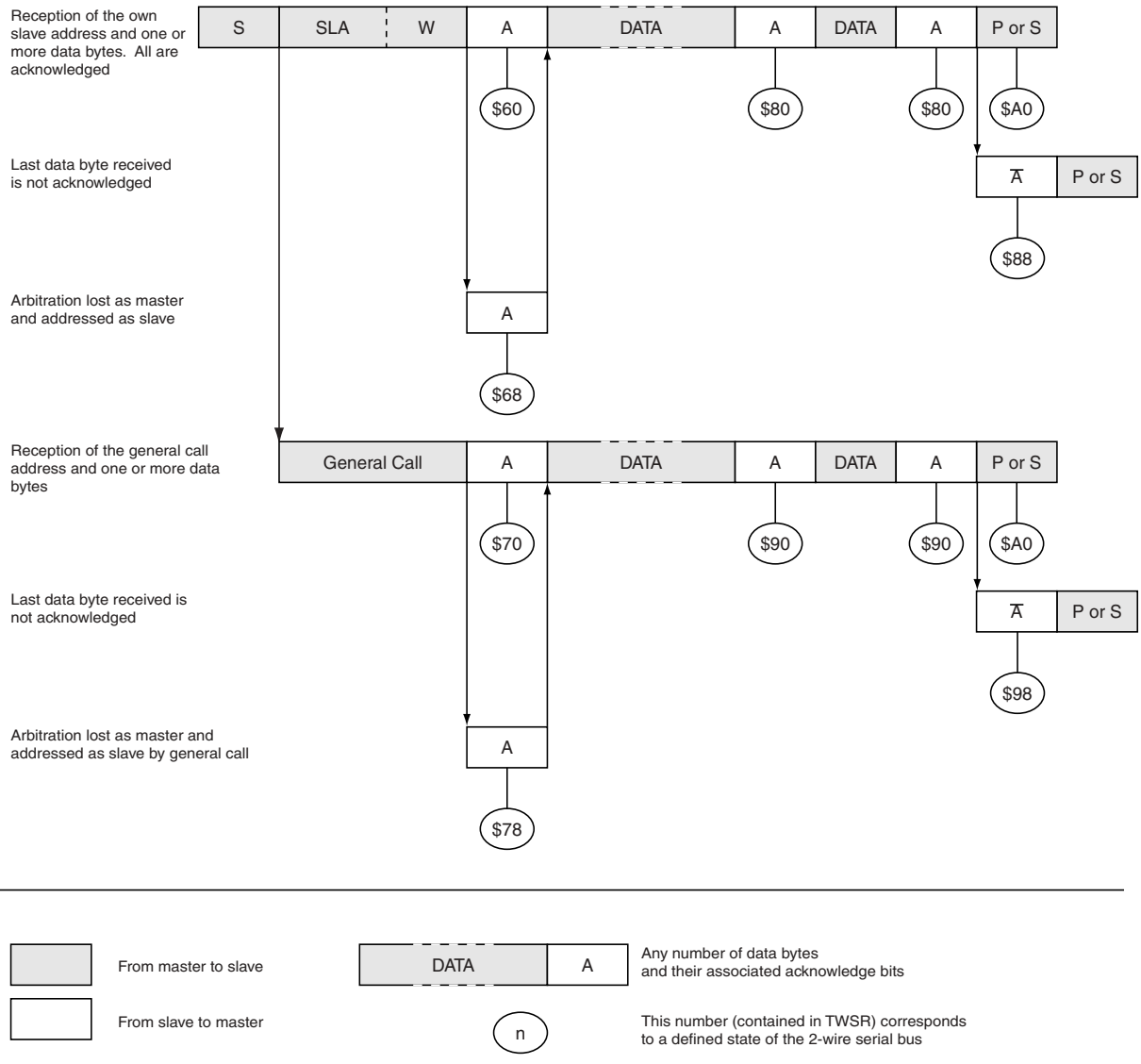
Table 43. Status Codes for Slave Receiver Mode

Status Code (TWSR)	Status of the 2-wire Serial Bus and 2-wire Serial Hardware	Application Software Response					Next Action Taken by 2-wire Serial Hardware
		To/From TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
$60	Own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$68	Arbitration lost in SLA+R/W as Master; own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$70	General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$78	Arbitration lost in SLA+R/W as Master; General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$80	Previously addressed with own SLA+W; data has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
$88	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if GC = “1”
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if GC = “1”; a START condition will be transmitted when the bus becomes free

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 73. Formats and States in the Slave Receiver Mode



I/O Ports

All AVR ports have true read-modify-write functionality when used as general I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies for changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input).

PortD

PortD is an 8-bit bi-directional I/O port with internal pull-up resistors.

Three I/O memory address locations are allocated for the PortD, one each for the Data Register – PORTD, $12($32), Data Direction Register – DDRD, $11($31) and the Port D Input Pins – PIND, $10($30). The Port D Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

The PortD output buffers can sink 20 mA. As inputs, PortD pins that are externally pulled Low will source current if the pull-up resistors are activated.

PortD Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
$12	**PORTD7**	**PORTD6**	**PORTD5**	**PORTD4**	**PORTD3**	**PORTD2**	**PORTD1**	**PORTD0**	**PORTD**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

PortD Data Direction Register – DDRD

Bit	7	6	5	4	3	2	1	0	
$11	**DDD7**	**DDD6**	**DDD5**	**DDD4**	**DDD3**	**DDD2**	**DDD1**	**DDD0**	**DDRD**
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	

PortD Input Pins Address – PIND

Bit	7	6	5	4	3	2	1	0	
$10	**PIND7**	**PIND6**	**PIND5**	**PIND4**	**PIND3**	**PIND2**	**PIND1**	**PIND0**	**PIND**
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	Pull1	Pull1	Pull1	Pull1	Pull1	Pull1	Pull1	Pull1	

The PortD Input Pins address – PIND – is not a register, and this address enables access to the physical value on each PortD pin. When reading PORTD, the PortD Data Latch is read, and when reading PIND, the logical values present on the pins are read.

PortD as General Digital I/O

PDn, General I/O pin: The DDDn bit in the DDRD register selects the direction of this pin. If DDDn is set (one), PDn is configured as an output pin. If DDDn is cleared (zero), PDn is configured as an input pin. If PDn is set (one) when configured as an input pin the MOS pull-up resistor is activated. To switch the pull-up resistor off the PDn has to be cleared (zero) or the pin has to be configured as an output pin. The port pins are input with pull-up when a reset condition becomes active, even if the clock is not running, see Table 46.

Table 47. PortE Pins Alternate Functions Controlled by SCR and AVR I/O Registers

Port Pin	Alternate Function	Input	Output
PE0	TX0 (UART0 transmit pin)	External Timer0 clock	-
PE1	RX0 (UART0 receive pin)	-	Output compare Timer0/PWM0
PE2	TX1 (UART1 transmit pin)	-	-
PE3	RX1 (UART1 receive pin)	-	Output compare Timer2/PWM2
PE4	INT0 (external Interrupt0 input)	External Timer1 clock	-
PE5	INT1 (external Interrupt0 input)	-	Output compare Timer1B/PWM1B
PE6	INT2 (external Interrupt0 input)	-	Output compare Timer1A/PWM1A
PE7	INT3 (external Interrupt0 input)	Input capture Counter1	

When the pins are used for the alternate function the DDRE and PORTE register has to be set according to the alternate function description.

PortE Data Register – PORTE

Bit	7	6	5	4	3	2	1	0	
$07 ($27)	**PORTE7**	**PORTE6**	**PORTE5**	**PORTE4**	**PORTE3**	**PORTE2**	**PORTE1**	**PORTE0**	**PORTE**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

PortE Data Direction Register – DDRE

Bit	7	6	5	4	3	2	1	0	
$06 ($26)	**DDE7**	**DDE6**	**DDE5**	**DDE4**	**DDE3**	**DDE2**	**DDE1**	**DDE0**	**DDRE**
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	

PortE Input Pins Address – PINE

Bit	7	6	5	4	3	2	1	0	
$05 ($25)	**PINE7**	**PINE6**	**PINE5**	**PINE4**	**PINE3**	**PINE2**	**PINE1**	**PINE0**	**PINE**
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	Pull1	Pull1	Pull1	Pull1	Pull1	Pull1	Pull1	Pull1	

The PortE Input Pins address – PINE – is not a register, and this address enables access to the physical value on each PortE pin. When reading PORTE, the PortE Data Latch is read, and when reading PINE, the logical values present on the pins are read.

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

Cell Function	Parameter	Path	-25	Units	Notes
Async RAM					
Write	t_{WECYC} (Minimum)	cycle time	12.0	ns	–
Write	t_{WEL} (Minimum)	we	5.0	ns	Pulse Width Low
Write	t_{WEH} (Minimum)	we	5.0	ns	Pulse Width High
Write	t_{setup} (Minimum)	wr addr setup-> we	5.3	ns	
Write	t_{hold} (Minimum)	wr addr hold -> we	0.0	ns	–
Write	t_{setup} (Minimum)	din setup -> we	5.0	ns	
Write	t_{hold} (Minimum)	din hold -> we	0.0	ns	–
Write	t_{hold} (Minimum)	oe hold -> we	0.0	ns	
Write/Read	t_{PD} (Maximum)	din -> dout	8.7	ns	rd addr = wr addr
Read	t_{PD} (Maximum)	rd addr -> dout	6.3	ns	
Read	t_{PZX} (Maximum)	oe -> dout	2.9	ns	–
Read	t_{PXZ} (Maximum)	oe -> dout	3.5	ns	
Sync RAM					
Write	t_{CYC} (Minimum)	cycle time	12.0	ns	
Write	t_{CLKL} (Minimum)	clk	5.0	ns	–
Write	t_{CLKH} (Minimum)	clk	5.0	ns	Pulse Width High
Write	t_{setup} (Minimum)	we setup-> clk	3.2	ns	
Write	t_{hold} (Minimum)	we hold -> clk	0.0	ns	–
Write	t_{setup} (Minimum)	wr addr setup-> clk	5.0	ns	
Write	t_{hold} (Minimum)	wr addr hold -> clk	0.0	ns	–
Write	t_{setup} (Minimum)	wr data setup-> clk	3.9	ns	
Write	t_{hold} (Minimum)	wr data hold -> clk	0.0	ns	–
Write/Read	t_{PD} (Maximum)	din -> dout	8.7	ns	rd addr = wr addr
Write/Read	t_{PD} (Maximum)	clk -> dout	5.8	ns	rd addr = wr addr
Read	t_{PD} (Maximum)	rd addr -> dout	6.3	ns	
Read	t_{PZX} (Maximum)	oe -> dout	2.9	ns	–
Read	t_{PXZ} (Maximum)	oe -> dout	3.5	ns	

CMOS buffer delays are measured from a V_{IH} of 1/2 V_{CC} at the pad to the internal V_{IH} at A. The input buffer load is constant. Buffer delay is to a pad voltage of 1.5V with one output switching. Parameter based on characterization and simulation; not tested in production. An FPGA power calculation is available in Atmel's System Designer software (see also page 160).