E·XFL



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "<u>Embedded -</u> <u>Microcontrollers</u>"

Details

Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	20MHz
Connectivity	I ² C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	23
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VFQFN Exposed Pad
Supplier Device Package	32-VQFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega88-20mj

Email: info@E-XFL.COM

Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong



resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

The various special features of Port D are elaborated in "Alternate Functions of Port D" on page 78.

2.3.7 AV_{cc}

2.3.9

 AV_{CC} is the supply voltage pin for the A/D Converter, PC3..0, and ADC7..6. It should be externally connected to V_{CC} , even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter. Note that PC6..4 use digital supply voltage, V_{CC} .

2.3.8 AREF AREF is the analog reference pin for the A/D Converter.

ADC7..6 (TQFP and QFN/MLF Package Only)

In the TQFP and QFN/MLF package, ADC7..6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

3. About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C compiler documentation for more details.

For I/O Registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBRS", "SBRC", "SBR", and "CBR".

When the write access time has elapsed, the EEPE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEPE has been set, the CPU is halted for two cycles before the next instruction is executed.

• Bit 0 – EERE: EEPROM Read Enable

The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be written to a logic one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed.

The user should poll the EEPE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR Register.

The calibrated Oscillator is used to time the EEPROM accesses. Table 5-2 lists the typical programming time for EEPROM access from the CPU.

 Table 5-2.
 EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles	Typ Programming Time
EEPROM write (from CPU)	26,368	3.3 ms

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (e.g. by disabling interrupts globally) so that no interrupts will occur during execution of these functions. The examples also assume that no Flash Boot Loader is present in the software. If such code is present, the EEPROM write function must also wait for any ongoing SPM command to finish.





10. I/O-Ports

10.1 Introduction

All AVR ports have true Read-Modify-Write functionality when used as general digital 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 when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in Figure 10-1. Refer to "Electrical Characteristics" on page 299 for a complete list of parameters.





All registers and bit references in this section are written in general form. A lower case "x" represents the numbering letter for the port, and a lower case "n" represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn. The physical I/O Registers and bit locations are listed in "Register Description for I/O Ports" on page 81.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. However, writing a logic one to a bit in the PINx Register, will result in a toggle in the corresponding bit in the Data Register. In addition, the Pull-up Disable – PUD bit in MCUCR disables the pull-up function for all pins in all ports when set.

Using the I/O port as General Digital I/O is described in "Ports as General Digital I/O" on page 65. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in "Alternate Port Functions" on page 69. Refer to the individual module sections for a full description of the alternate functions.



The setup of the OC0x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC0x value is to use the Force Output Compare (FOC0x) strobe bits in Normal mode. The OC0x Registers keep their values even when changing between Waveform Generation modes.

Be aware that the COM0x1:0 bits are not double buffered together with the compare value. Changing the COM0x1:0 bits will take effect immediately.

12.5 Compare Match Output Unit

The Compare Output mode (COM0x1:0) bits have two functions. The Waveform Generator uses the COM0x1:0 bits for defining the Output Compare (OC0x) state at the next compare match. Also, the COM0x1:0 bits control the OC0x pin output source. Figure 12-4 shows a simplified schematic of the logic affected by the COM0x1:0 bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers (DDR and PORT) that are affected by the COM0x1:0 bits are shown. When referring to the OC0x state, the reference is for the internal OC0x Register, not the OC0x pin. If a system reset occur, the OC0x Register is reset to "0".



Figure 12-4. Compare Match Output Unit, Schematic

The general I/O port function is overridden by the Output Compare (OC0x) from the Waveform Generator if either of the COM0x1:0 bits are set. However, the OC0x pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OC0x pin (DDR_OC0x) must be set as output before the OC0x value is visible on the pin. The port override function is independent of the Waveform Generation mode.

The design of the Output Compare pin logic allows initialization of the OC0x state before the output is enabled. Note that some COM0x1:0 bit settings are reserved for certain modes of operation. See Section "12.8" on page 99.

12.5.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM0x1:0 bits differently in Normal, CTC, and PWM modes. For all modes, setting the COM0x1:0 = 0 tells the Waveform Generator that no action on the OC0x Register is to be performed on the next compare match. For compare output actions in the







Signal description (internal signals):

count	Increment or decrement TCNT2 by 1.
direction	Selects between increment and decrement.
clear	Clear TCNT2 (set all bits to zero).
clk _{Tn}	Timer/Counter clock, referred to as clk_{T2} in the following.
top	Signalizes that TCNT2 has reached maximum value.
bottom	Signalizes that TCNT2 has reached minimum value (zero).

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T2}). clk_{T2} can be generated from an external or internal clock source, selected by the Clock Select bits (CS22:0). When no clock source is selected (CS22:0 = 0) the timer is stopped. However, the TCNT2 value can be accessed by the CPU, regardless of whether clk_{T2} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM21 and WGM20 bits located in the Timer/Counter Control Register (TCCR2A) and the WGM22 located in the Timer/Counter Control Register B (TCCR2B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC2A and OC2B. For more details about advanced counting sequences and waveform generation, see "Modes of Operation" on page 143.

The Timer/Counter Overflow Flag (TOV2) is set according to the mode of operation selected by the WGM22:0 bits. TOV2 can be used for generating a CPU interrupt.

15.4 Output Compare Unit

The 8-bit comparator continuously compares TCNT2 with the Output Compare Register (OCR2A and OCR2B). Whenever TCNT2 equals OCR2A or OCR2B, the comparator signals a match. A match will set the Output Compare Flag (OCF2A or OCF2B) at the next timer clock cycle. If the corresponding interrupt is enabled, the Output Compare Flag generates an Output Compare interrupt. The Output Compare Flag is automatically cleared when the interrupt is executed. Alternatively, the Output Compare Flag can be cleared by software by writing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating mode set by the WGM22:0 bits and Compare Output mode (COM2x1:0) bits. The max and bottom signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation ("Modes of Operation" on page 143).

Figure 15-3 shows a block diagram of the Output Compare unit.



16.1.4 SPI Status Register – SPSR

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	-	-	-	-	-	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 – SPIF: SPI Interrupt Flag

When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If \overline{SS} is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

Bit 6 – WCOL: Write COLlision Flag

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

• Bit 5..1 - Res: Reserved Bits

These bits are reserved bits in the ATmega48/88/168 and will always read as zero.

Bit 0 – SPI2X: Double SPI Speed Bit

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 16-4). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at $f_{osc}/4$ or lower.

The SPI interface on the ATmega48/88/168 is also used for program memory and EEPROM downloading or uploading. See page 295 for serial programming and verification.

16.1.5 SPI Data Register – SPDR



The SPI Data Register is a read/write register used for data transfer between the Register File and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

16.2 Data Modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 16-3 and Figure 16-4. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. This is clearly seen by summarizing Table 16-2 and Table 16-3, as done below.

	f _{osc} = 16.0000 MHz			f _{osc} = 18.4320 MHz				f _{osc} = 20.0000 MHz				
Baud	U2X	n = 0	U2X	n = 1	U2X	U2Xn = 0 U2Xn = 1			U2X	n = 0	U2Xn = 1	
Rate (bps)	UBRR n	Error	UBRR n	Error	UBRR n	Error	UBRR n	Error	UBRR n	Error	UBRR n	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	-	_	4	-7.8%	-	-	4	0.0%
1M	0	0.0%	1	0.0%	-	-	-	-	-	-	-	-
Max. (1)	1 N	1bps	2 N	lbps	1.152	Mbps	2.304	Mbps	1.25	Mbps	2.5	Vbps

Table 17-12.	Examples of UBRRn	Settings for Common	v Used Oscillator Frequencies	(Continued)

1. UBRRn = 0, Error = 0.0%



19.5 Overview of the TWI Module

The TWI module is comprised of several submodules, as shown in Figure 19-9. All registers drawn in a thick line are accessible through the AVR data bus.





19.5.1 SCL and SDA Pins

These pins interface the AVR TWI with the rest of the MCU system. The output drivers contain a slew-rate limiter in order to conform to the TWI specification. The input stages contain a spike suppression unit removing spikes shorter than 50 ns. Note that the internal pull-ups in the AVR pads can be enabled by setting the PORT bits corresponding to the SCL and SDA pins, as explained in the I/O Port section. The internal pull-ups can in some systems eliminate the need for external ones.

19.5.2 Bit Rate Generator Unit

This unit controls the period of SCL when operating in a Master mode. The SCL period is controlled by settings in the TWI Bit Rate Register (TWBR) and the Prescaler bits in the TWI Status Register (TWSR). Slave operation does not depend on Bit Rate or Prescaler settings, but the CPU clock frequency in the Slave must be at least 16 times higher than the SCL frequency. Note



not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the data packet.

- 6. When the data packet has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the data packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
- 7. The application software should now examine the value of TWSR, to make sure that the data packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must write a specific value to TWCR, instructing the TWI hardware to transmit a STOP condition. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the STOP condition. Note that TWINT is NOT set after a STOP condition has been sent.

Even though this example is simple, it shows the principles involved in all TWI transmissions. These can be summarized as follows:

- When the TWI has finished an operation and expects application response, the TWINT Flag is set. The SCL line is pulled low until TWINT is cleared.
- When the TWINT Flag is set, the user must update all TWI Registers with the value relevant for the next TWI bus cycle. As an example, TWDR must be loaded with the value to be transmitted in the next bus cycle.
- After all TWI Register updates and other pending application software tasks have been completed, TWCR is written. When writing TWCR, the TWINT bit should be set. Writing a one to TWINT clears the flag. The TWI will then commence executing whatever operation was specified by the TWCR setting.

In the following an assembly and C implementation of the example is given. Note that the code below assumes that several definitions have been made, for example by using include-files.





Note that data is transmitted both from Master to Slave and vice versa. The Master must instruct the Slave what location it wants to read, requiring the use of the MT mode. Subsequently, data must be read from the Slave, implying the use of the MR mode. Thus, the transfer direction must be changed. The Master must keep control of the bus during all these steps, and the steps should be carried out as an atomical operation. If this principle is violated in a multi master system, another Master can alter the data pointer in the EEPROM between steps 2 and 3, and the Master will read the wrong data location. Such a change in transfer direction is accomplished by transmitting a REPEATED START between the transmission of the address byte and reception of the data. After a REPEATED START, the Master keeps ownership of the bus. The following figure shows the flow in this transfer.





19.9 Multi-master Systems and Arbitration

If multiple masters are connected to the same bus, transmissions may be initiated simultaneously by one or more of them. The TWI standard ensures that such situations are handled in such a way that one of the masters will be allowed to proceed with the transfer, and that no data will be lost in the process. An example of an arbitration situation is depicted below, where two masters are trying to transmit data to a Slave Receiver.

Figure 19-21. An Arbitration Example



Several different scenarios may arise during arbitration, as described below:

- Two or more masters are performing identical communication with the same Slave. In this case, neither the Slave nor any of the masters will know about the bus contention.
- Two or more masters are accessing the same Slave with different data or direction bit. In this case, arbitration will occur, either in the READ/WRITE bit or in the data bits. The masters trying to output a one on SDA while another Master outputs a zero will lose the arbitration. Losing masters will switch to not addressed Slave mode or wait until the bus is free and transmit a new START condition, depending on application software action.



23.0.3 Performing a Page Write

To execute Page Write, set up the address in the Z-pointer, write "00000101" to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE. Other bits in the Z-pointer must be written to zero during this operation.

• The CPU is halted during the Page Write operation.

23.1 Addressing the Flash During Self-Programming

The Z-pointer is used to address the SPM commands.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	Z15	Z14	Z13	Z12	Z11	Z10	Z9	Z8
ZL (R30)	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0
	7	6	5	4	3	2	1	0

Since the Flash is organized in pages (see Table 25-8 on page 284), the Program Counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in Figure 24-3. Note that the Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the software addresses the same page in both the Page Erase and Page Write operation.

The LPM instruction uses the Z-pointer to store the address. Since this instruction addresses the Flash byte-by-byte, also the LSB (bit Z0) of the Z-pointer is used.



Figure 23-1. Addressing the Flash During SPM⁽¹⁾

Note: 1. The different variables used in Figure 24-3 are listed in Table 25-8 on page 284.

24.3 Read-While-Write and No Read-While-Write Flash Sections

Whether the CPU supports Read-While-Write or if the CPU is halted during a Boot Loader software update is dependent on which address that is being programmed. In addition to the two sections that are configurable by the BOOTSZ Fuses as described above, the Flash is also divided into two fixed sections, the Read-While-Write (RWW) section and the No Read-While-Write (NRWW) section. The limit between the RWW- and NRWW sections is given in Table 24-7 on page 277 and Figure 24-2 on page 267. The main difference between the two sections is:

- When erasing or writing a page located inside the RWW section, the NRWW section can be read during the operation.
- When erasing or writing a page located inside the NRWW section, the CPU is halted during the entire operation.

Note that the user software can never read any code that is located inside the RWW section during a Boot Loader software operation. The syntax "Read-While-Write section" refers to which section that is being programmed (erased or written), not which section that actually is being read during a Boot Loader software update.

24.3.1 RWW – Read-While-Write Section

If a Boot Loader software update is programming a page inside the RWW section, it is possible to read code from the Flash, but only code that is located in the NRWW section. During an ongoing programming, the software must ensure that the RWW section never is being read. If the user software is trying to read code that is located inside the RWW section (i.e., by a call/jmp/lpm or an interrupt) during programming, the software might end up in an unknown state. To avoid this, the interrupts should either be disabled or moved to the Boot Loader section. The Boot Loader section is always located in the NRWW section. The RWW Section Busy bit (RWWSB) in the Store Program Memory Control and Status Register (SPMCSR) will be read as logical one as long as the RWW section is blocked for reading. After a programming is completed, the RWWSB must be cleared by software before reading code located in the RWW section. See Section "24.5.1" on page 269. for details on how to clear RWWSB.

24.3.2 NRWW – No Read-While-Write Section

The code located in the NRWW section can be read when the Boot Loader software is updating a page in the RWW section. When the Boot Loader code updates the NRWW section, the CPU is halted during the entire Page Erase or Page Write operation.

Which Section does the Z- pointer Address during the Programming?	Which Section can be read during Programming?	CPU Halted?	Read-While-Write Supported?
RWW Section	NRWW Section	No	Yes
NRWW Section	None	Yes	No

Table 24-1.Read-While-Write Features





25.4 Calibration Byte

The ATmega48/88/168 has a byte calibration value for the internal RC Oscillator. This byte resides in the high byte of address 0x000 in the signature address space. During reset, this byte is automatically written into the OSCCAL Register to ensure correct frequency of the calibrated RC Oscillator.

25.5 Page Size

Device	Flash Size	Page Size	PCWORD	No. of Pages	PCPAGE	PCMSB
ATmega48	2K words (4K bytes)	32 words	PC[4:0]	64	PC[10:5]	10
ATmega88	4K words (8K bytes)	32 words	PC[4:0]	128	PC[11:5]	11
ATmega168	8K words (16K bytes)	64 words	PC[5:0]	128	PC[12:6]	12

 Table 25-8.
 No. of Words in a Page and No. of Pages in the Flash

Device	EEPROM Size	Page Size	PCWORD	No. of Pages	PCPAGE	EEAMSB
ATmega48	256 bytes	4 bytes	EEA[1:0]	64	EEA[7:2]	7
ATmega88	512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8
ATmega168	512 bytes	4 bytes	EEA[1:0]	128	EEA[8:2]	8

25.6 Parallel Programming Parameters, Pin Mapping, and Commands

This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory, Memory Lock bits, and Fuse bits in the ATmega48/88/168. Pulses are assumed to be at least 250 ns unless otherwise noted.

25.6.1 Signal Names

In this section, some pins of the ATmega48/88/168 are referenced by signal names describing their functionality during parallel programming, see Figure 25-1 and Table 25-10. Pins not described in the following table are referenced by pin names.

The XA1/XA0 pins determine the action executed when the XTAL1 pin is given a positive pulse. The bit coding is shown in Table 25-12.

When pulsing \overline{WR} or \overline{OE} , the command loaded determines the action executed. The different Commands are shown in Table 25-13.

- 1. A: Load Command "0010 0000".
- C: Load Data Low Byte. Bit n = "0" programs the Lock bit. If LB mode 3 is programmed (LB1 and LB2 is programmed), it is not possible to program the Boot Lock bits by any External Programming mode.
- 3. Give \overline{WR} a negative pulse and wait for RDY/ \overline{BSY} to go high.

The Lock bits can only be cleared by executing Chip Erase.

25.7.12 Reading the Fuse and Lock Bits

The algorithm for reading the Fuse and Lock bits is as follows (refer to "Programming the Flash" on page 287 for details on Command loading):

- 1. A: Load Command "0000 0100".
- 2. Set $\overline{\text{OE}}$ to "0", BS2 to "0" and BS1 to "0". The status of the Fuse Low bits can now be read at DATA ("0" means programmed).
- 3. Set $\overline{\text{OE}}$ to "0", BS2 to "1" and BS1 to "1". The status of the Fuse High bits can now be read at DATA ("0" means programmed).
- 4. Set OE to "0", BS2 to "1", and BS1 to "0". The status of the Extended Fuse bits can now be read at DATA ("0" means programmed).
- 5. Set $\overline{\text{OE}}$ to "0", BS2 to "0" and BS1 to "1". The status of the Lock bits can now be read at DATA ("0" means programmed).
- 6. Set OE to "1".



Figure 25-6. Mapping Between BS1, BS2 and the Fuse and Lock Bits During Read

25.7.13 Reading the Signature Bytes

The algorithm for reading the Signature bytes is as follows (refer to "Programming the Flash" on page 287 for details on Command and Address loading):

- 1. A: Load Command "0000 1000".
- 2. B: Load Address Low Byte (0x00 0x02).
- 3. Set \overline{OE} to "0", and BS1 to "0". The selected Signature byte can now be read at DATA.
- 4. Set OE to "1".



27.5 Power-Save Supply Current



Figure 27-15. Power-Save Supply Current vs. V_{CC} (Watchdog Timer Disabled)

27.6 Standby Supply Current





STANDBY SUPPLY CURRENT vs. V_{CC} Low Power Crystal Oscillator



Figure 27-29. I/O Pin Input Threshold Voltage vs. V_{CC} (VIL, I/O Pin Read As '0')











Figure 27-36. Bandgap Voltage vs. V_{CC}







27.12 Current Consumption of Peripheral Units





Figure 27-44. ADC Current vs. V_{CC} (ADC at 50 kHz)



ATmega48/88/168

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xBF)	Reserved	-	-	-	-	-	-	-	-	
(0xBE)	Reserved	-	-	-	-	-	-	-	-	
(0xBD)	TWAMR	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	-	216
(0xBC)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	213
(0xBB)	TWDR		1	1	2-wire Serial Inter	face Data Regist	er	1		215
(0xBA)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	216
(0xB9)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0	215
(0xB8)	IWBR				2-wire Serial Interfa	ice Bit Rate Regis	ster			213
(UXB7) (0xB6)	ASSE	_	EXCLK	-	- TCN2UB			TCP2ALIB	TCP2BUB	156
(0xB5)	Beserved		-	-	-	-	-	-	-	150
(0xB4)	OCB2B			ı Tir	ner/Counter2 Outp	It Compare Regis	ster B			153
(0xB3)	OCR2A			Tir	mer/Counter2 Outp	ut Compare Regi	ster A			153
(0xB2)	TCNT2				Timer/Cou	nter2 (8-bit)				153
(0xB1)	TCCR2B	FOC2A	FOC2B	-	-	WGM22	CS22	CS21	CS20	152
(0xB0)	TCCR2A	COM2A1	COM2A0	COM2B1	COM2B0	-	-	WGM21	WGM20	149
(0xAF)	Reserved	-	-	-	-	-	-	-	-	
(0xAE)	Reserved	-	-	-	-	-	-	-	-	
(0xAD)	Reserved	-	-	-	-	-	-	-	-	
(0xAC)	Reserved	-	-	-	-	-	-	-	-	
(0xAB)	Reserved	-	_	-	-	-	-	-	_	
(UXAA)	Reserved	-	_	-	-	-	-	-		
(0xA9) (0xA8)	Beserved	_			_					
(0xA0) (0xA7)	Reserved	_	_	_	_	_	_	_		
(0xA6)	Reserved	_	_	_	_	_	_	_	_	
(0xA5)	Reserved	-	-	-	-	-	-	-	-	
(0xA4)	Reserved	-	_	_	-	_	_	_	-	
(0xA3)	Reserved	-	-	-	-	-	-	-	-	
(0xA2)	Reserved	-	-	-	-	-	-	-	-	
(0xA1)	Reserved	-	-	-	-	-	-	-	-	
(0xA0)	Reserved	-	-	-	-	-	-	-	-	
(0x9F)	Reserved	-	-	-	-	-	-	-	_	
(0x9E)	Reserved	-	-	-	-	-	-	-	-	
(0x9D)	Reserved	-	-	-	-	-	-	-		
(0x9C) (0x9B)	Reserved	_			_					
(0x9A)	Reserved	_	_	_	_	_	_	_	_	
(0x99)	Reserved	_	_	_	_	_	_	_	_	
(0x98)	Reserved	-	-	-	-	-	-	-	-	
(0x97)	Reserved	-	-	-	-	-	-	-	-	
(0x96)	Reserved	-	-	-	-	-	-	-	-	
(0x95)	Reserved	-	-	-	-	-	-	-	-	
(0x94)	Reserved	-	-	-	-	-	-	-	-	
(0x93)	Reserved	-	-	-	-	-	-	-	-	
(0x92)	Reserved	-	-	-	-	-	-	-		
(0x91)	Reserved	-	-	_	_	-	-	_	_	<u> </u>
(0x8F)	Reserved	_	_	_	_	_	_	_	_	
(0x8E)	Reserved	_	_	_	_	_	_	_	_	
(0x8D)	Reserved	-	-	-	-	-	_	-	-	[
(0x8C)	Reserved	-	-	-	-	-	-	-	-	
(0x8B)	OCR1BH		•	Timer/Co	ounter1 - Output Co	mpare Register E	3 High Byte			132
(0x8A)	OCR1BL			Timer/C	ounter1 - Output Co	ompare Register I	B Low Byte			132
(0x89)	OCR1AH			Timer/Co	ounter1 - Output Co	ompare Register A	A High Byte			132
(0x88)	OCR1AL			Timer/C	ounter1 - Output Co	ompare Register	A Low Byte			132
(0x87)	ICR1H			Timer	/Counter1 - Input C	apture Register H	ligh Byte			133
(0x86)	ICR1L			Timer	Counter1 - Input C	apture Register L	low Byte			133
(0x85)	ICNT1H			Tin	ner/Counter1 - Cou	nter Hegister High	n Byte			132
(UX84)	IUNI1L Recorved		_	l in	ner/Counter1 - Cou	mer Register Low	и Буте		_	132
(0x82)	TCCR1C	FOC14	FOC1B		_			_		131
(0x81)	TCCR1B	ICNC1	ICES1	_	WGM13	WGM12	CS12	CS11	CS10	130
(0x80)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10	128
(0x7F)	DIDR1	-	-	-	-	-	-	AIN1D	AIN0D	238
(0x7E)	DIDR0	_	_	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	254





Atmel Corporation

2325 Orchard Parkway San Jose, CA 95131, USA Tel: 1(408) 441-0311 Fax: 1(408) 487-2600

Regional Headquarters

Europe

Atmel Sarl Route des Arsenaux 41 Case Postale 80 CH-1705 Fribourg Switzerland Tel: (41) 26-426-5555 Fax: (41) 26-426-5500

Asia

Room 1219 Chinachem Golden Plaza 77 Mody Road Tsimshatsui East Kowloon Hong Kong Tel: (852) 2721-9778 Fax: (852) 2722-1369

Japan

9F, Tonetsu Shinkawa Bldg. 1-24-8 Shinkawa Chuo-ku, Tokyo 104-0033 Japan Tel: (81) 3-3523-3551 Fax: (81) 3-3523-7581

Atmel Operations

Memory 2325 Orchard Parkway San Jose, CA 95131, USA Tel: 1(408) 441-0311 Fax: 1(408) 436-4314

Microcontrollers

2325 Orchard Parkway San Jose, CA 95131, USA Tel: 1(408) 441-0311 Fax: 1(408) 436-4314

La Chantrerie BP 70602 44306 Nantes Cedex 3, France Tel: (33) 2-40-18-18-18 Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards

Zone Industrielle 13106 Rousset Cedex, France Tel: (33) 4-42-53-60-00 Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd. Colorado Springs, CO 80906, USA Tel: 1(719) 576-3300 Fax: 1(719) 540-1759

Scottish Enterprise Technology Park Maxwell Building East Kilbride G75 0QR, Scotland Tel: (44) 1355-803-000 Fax: (44) 1355-242-743

RF/Automotive

Theresienstrasse 2 Postfach 3535 74025 Heilbronn, Germany Tel: (49) 71-31-67-0 Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd. Colorado Springs, CO 80906, USA Tel: 1(719) 576-3300 Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/

High Speed Converters/RF Datacom Avenue de Rochepleine BP 123 38521 Saint-Egreve Cedex, France Tel: (33) 4-76-58-30-00 Fax: (33) 4-76-58-34-80

Literature Requests www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDI-TIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDEN-TAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© Atmel Corporation 2005. All rights reserved. Atmel[®], logo and combinations thereof, AVR[®], and AVR Studio[®] are registered trademarks, and Everywhere You AreSM are the trademarks of Atmel Corporation or its subsidiaries. Microsoft[®], Windows[®], Windows NT[®], and Windows XP[®] are the registered trademarks of Microsoft Corporation. Other terms and product names may be trademarks of others.

