



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	Active
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	16KB (8K 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-TQFP
Supplier Device Package	32-TQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega168-20aur

Email: info@E-XFL.COM

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

5.2 SRAM Data Memory

Figure 5-3 shows how the ATmega48/88/168 SRAM Memory is organized.

The ATmega48/88/168 is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in the Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

The lower 768/1280/1280 data memory locations address both the Register File, the I/O memory, Extended I/O memory, and the internal data SRAM. The first 32 locations address the Register File, the next 64 location the standard I/O memory, then 160 locations of Extended I/O memory, and the next 512/1024/1024 locations address the internal data SRAM.

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 direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations 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 or incremented.

The 32 general purpose working registers, 64 I/O Registers, 160 Extended I/O Registers, and the 512/1024/1024 bytes of internal data SRAM in the ATmega48/88/168 are all accessible through all these addressing modes. The Register File is described in "General Purpose Register File" on page 10.

Figure 5-3. Data Memory Map

Data Memory

32 Registers	0x0000 - 0x001F
64 I/O Registers	0x0020 - 0x005F
160 Ext I/O Reg.	0x0060 - 0x00FF
	0x0100
Internal SRAM	
(512/1024/1024 x 8)	
· · · ·	0x02FF/0x04FF/0x04FF

5.2.1 Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in Figure 5-4.



ATmega48/88/168

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

Assembly Code Example					
EEPROM_read:					
; Wait for completion of previous write					
sbic EECR, EEPE					
rjmp EEPROM_read					
; Set up address (r18:r17) in address register					
out EEARH, r18					
out EEARL, r17					
; Start eeprom read by writing EERE					
sbi EECR, EERE					
; Read data from Data Register					
in r16,EEDR					
ret					

C Code Example

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
    /* Wait for completion of previous write */
    while(EECR & (1<<EEPE))
    ;
    /* Set up address register */
    EEAR = uiAddress;
    /* Start eeprom read by writing EERE */
    EECR |= (1<<EERE);
    /* Return data from Data Register */
    return EEDR;
}</pre>
```

5.3.5 Preventing EEPROM Corruption

During periods of low V_{CC} , the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage is too low.

EEPROM data corruption can easily be avoided by following this design recommendation:

Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low V_{CC} reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

frequency of the undivided clock, which may be faster than the CPU's clock frequency. Hence, it is not possible to determine the state of the prescaler - even if it were readable, and the exact time it takes to switch from one clock division to the other cannot be exactly predicted. From the time the CLKPS values are written, it takes between T1 + T2 and T1 + 2 * T2 before the new clock frequency is active. In this interval, 2 active clock edges are produced. Here, T1 is the previous clock period, and T2 is the period corresponding to the new prescaler setting.

To avoid unintentional changes of clock frequency, a special write procedure must befollowed to change the CLKPS bits:

- 1. Write the Clock Prescaler Change Enable (CLKPCE) bit to one and all other bitsin CLKPR to zero.
- 2. Within four cycles, write the desired value to CLKPS while writing a zero to CLKPCE.

Interrupts must be disabled when changing prescaler setting to make sure the write procedure is not interrupted.

6.11.1 Clock Prescale Register – CLKPR



Bit 7 – CLKPCE: Clock Prescaler Change Enable

The CLKPCE bit must be written to logic one to enable change of the CLKPS bits. The CLKPCE bit is only updated when the other bits in CLKPR are simultaneously written to zero. CLKPCE is cleared by hardware four cycles after it is written or when CLKPS bits are written. Rewriting the CLKPCE bit within this time-out period does neither extend the time-out period, nor clear the CLKPCE bit.

• Bits 3..0 – CLKPS3..0: Clock Prescaler Select Bits 3 - 0

These bits define the division factor between the selected clock source and the internal system clock. These bits can be written run-time to vary the clock frequency to suit the application requirements. As the divider divides the master clock input to the MCU, the speed of all synchronous peripherals is reduced when a division factor is used. The division factors are given in Table 6-14.

The CKDIV8 Fuse determines the initial value of the CLKPS bits. If CKDIV8 is unprogrammed, the CLKPS bits will be reset to "0000". If CKDIV8 is programmed, CLKPS bits are reset to "0011", giving a division factor of 8 at start up. This feature should be used if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. Note that any value can be written to the CLKPS bits regardless of the CKDIV8 Fuse setting. The Application software must ensure that a sufficient division factor is chosen if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. The device is shipped with the CKDIV8 Fuse programmed.



8. System Control and Reset

8.0.1 Resetting the AVR

During reset, all I/O Registers are set to their initial values, and the program starts execution from the Reset Vector. For the ATmega168, the instruction placed at the Reset Vector must be a JMP – Absolute Jump – instruction to the reset handling routine. For the ATmega48 and ATmega88, the instruction placed at the Reset Vector must be an RJMP – Relative 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. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa (ATmega88/168 only). The circuit diagram in Figure 8-1 shows the reset logic. Table 8-1 defines the electrical parameters of the reset circuitry.

The I/O ports of the AVR are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running.

After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the SUT and CKSEL Fuses. The different selections for the delay period are presented in "Clock Sources" on page 26.

8.0.2 Reset Sources

The ATmega48/88/168 has four sources of reset:

- Power-on Reset. The MCU is reset when the supply voltage is below the Power-on Reset threshold (V_{POT}).
- External Reset. The MCU is reset when a low level is present on the RESET pin for longer than the minimum pulse length.
- Watchdog System Reset. The MCU is reset when the Watchdog Timer period expires and the Watchdog System Reset mode is enabled.
- Brown-out Reset. The MCU is reset when the supply voltage V_{CC} is below the Brown-out Reset threshold (V_{BOT}) and the Brown-out Detector is enabled.



10.3.3 Alternate Functions of Port C

The Port C pins with alternate functions are shown in Table 10-6.

Port Pin	Alternate Function				
PC6	RESET (Reset pin) PCINT14 (Pin Change Interrupt 14)				
PC5	ADC5 (ADC Input Channel 5) SCL (2-wire Serial Bus Clock Line) PCINT13 (Pin Change Interrupt 13)				
PC4	ADC4 (ADC Input Channel 4) SDA (2-wire Serial Bus Data Input/Output Line) PCINT12 (Pin Change Interrupt 12)				
PC3	ADC3 (ADC Input Channel 3) PCINT11 (Pin Change Interrupt 11)				
PC2	ADC2 (ADC Input Channel 2) PCINT10 (Pin Change Interrupt 10)				
PC1	PC1 ADC1 (ADC Input Channel 1) PCINT9 (Pin Change Interrupt 9)				
PC0	ADC0 (ADC Input Channel 0) PCINT8 (Pin Change Interrupt 8)				

 Table 10-6.
 Port C Pins Alternate Functions

The alternate pin configuration is as follows:

• RESET/PCINT14 – Port C, Bit 6

RESET, Reset pin: When the RSTDISBL Fuse is programmed, this pin functions as a normal I/O pin, and the part will have to rely on Power-on Reset and Brown-out Reset as its reset sources. When the RSTDISBL Fuse is unprogrammed, the reset circuitry is connected to the pin, and the pin can not be used as an I/O pin.

If PC6 is used as a reset pin, DDC6, PORTC6 and PINC6 will all read 0.

PCINT14: Pin Change Interrupt source 14. The PC6 pin can serve as an external interrupt source.

• SCL/ADC5/PCINT13 - Port C, Bit 5

SCL, 2-wire Serial Interface Clock: When the TWEN bit in TWCR is set (one) to enable the 2wire Serial Interface, pin PC5 is disconnected from the port and becomes the Serial Clock I/O pin for the 2-wire Serial Interface. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50 ns on the input signal, and the pin is driven by an open drain driver with slew-rate limitation.

PC5 can also be used as ADC input Channel 5. Note that ADC input channel 5 uses digital power.

PCINT13: Pin Change Interrupt source 13. The PC5 pin can serve as an external interrupt source.

• SDA/ADC4/PCINT12 – Port C, Bit 4

SDA, 2-wire Serial Interface Data: When the TWEN bit in TWCR is set (one) to enable the 2-wire Serial Interface, pin PC4 is disconnected from the port and becomes the Serial Data I/O pin for



ATmega48/88/168





An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCF1A or ICF1 Flag according to the register used to define the TOP value. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR1A or ICR1 is lower than the current value of TCNT1, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. In many cases this feature is not desirable. An alternative will then be to use the fast PWM mode using OCR1A for defining TOP (WGM13:0 = 15) since the OCR1A then will be double buffered.

For generating a waveform output in CTC mode, the OC1A output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM1A1:0 = 1). The OC1A value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OC1A = 1). The waveform generated will have a maximum frequency of $f_{OC1A} = f_{clk_l/O}/2$ when OCR1A is set to zero (0x0000). The waveform frequency is defined by the following equation:

$$f_{OCnA} = \frac{f_{clk_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

The N variable represents the prescaler factor (1, 8, 64, 256, or 1024).

As for the Normal mode of operation, the TOV1 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

13.8.3 Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGM13:0 = 5, 6, 7, 14, or 15) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is set on the compare match between TCNT1 and OCR1x, and cleared at TOP. In inverting Compare Output mode output is cleared on compare match and set at TOP. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct and phase and frequency correct PWM modes that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), hence reduces total system cost.





The PWM resolution for fast PWM can be fixed to 8-, 9-, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated by using the following equation:

$$R_{FPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In fast PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM13:0 = 5, 6, or 7), the value in ICR1 (WGM13:0 = 14), or the value in OCR1A (WGM13:0 = 15). The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 13-7. The figure shows fast PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x Interrupt Flag will be set when a compare match occurs.

Figure 13-7. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV1) is set each time the counter reaches TOP. In addition the OC1A or ICF1 Flag is set at the same timer clock cycle as TOV1 is set when either OCR1A or ICR1 is used for defining the TOP value. If one of the interrupts are enabled, the interrupt handler routine can be used for updating the TOP and compare values.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNT1 and the OCR1x. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCR1x Registers are written.

The procedure for updating ICR1 differs from updating OCR1A when used for defining the TOP value. The ICR1 Register is not double buffered. This means that if ICR1 is changed to a low value when the counter is running with none or a low prescaler value, there is a risk that the new ICR1 value written is lower than the current value of TCNT1. The result will then be that the counter will miss the compare match at the TOP value. The counter will then have to count to the MAX value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. The OCR1A Register however, is double buffered. This feature allows the OCR1A I/O location



the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated using the following equation:

$$R_{PFCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In phase and frequency correct PWM mode the counter is incremented until the counter value matches either the value in ICR1 (WGM13:0 = 8), or the value in OCR1A (WGM13:0 = 9). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct and frequency correct PWM mode is shown on Figure 13-9. The figure shows phase and frequency correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x Interrupt Flag will be set when a compare match occurs.

Figure 13-9. Phase and Frequency Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV1) is set at the same timer clock cycle as the OCR1x Registers are updated with the double buffer value (at BOTTOM). When either OCR1A or ICR1 is used for defining the TOP value, the OC1A or ICF1 Flag set when TCNT1 has reached TOP. The Interrupt Flags can then be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a compare match will never occur between the TCNT1 and the OCR1x.

As Figure 13-9 shows the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR1x Registers are updated at BOTTOM, the length of the rising and the falling slopes will always be equal. This gives symmetrical output pulses and is therefore frequency correct. When the ICR1 is used as TOP value (see description of the WGM13:0 bits located in the TCCR1A and the TCCR1B Register), the ICP1 is disconnected and consequently the Input Capture function is disabled.

• Bit 5 – Reserved Bit

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to zero when TCCR1B is written.

• Bit 4:3 – WGM13:2: Waveform Generation Mode

See TCCR1A Register description.

• Bit 2:0 - CS12:0: Clock Select

The three Clock Select bits select the clock source to be used by the Timer/Counter, see Figure 13-10 and Figure 13-11.

CS12	CS11	CS10	Description			
0	0	0	No clock source (Timer/Counter stopped).			
0	0	1	clk _{I/O} /1 (No prescaling)			
0	1	0	clk _{I/O} /8 (From prescaler)			
0	1	1	clk _{I/O} /64 (From prescaler)			
1	0	0	clk _{I/O} /256 (From prescaler)			
1	0	1	clk _{I/O} /1024 (From prescaler)			
1	1	0	External clock source on T1 pin. Clock on falling edge.			
1	1	1	External clock source on T1 pin. Clock on rising edge.			

Table 13-5. Clock Select Bit Description

If external pin modes are used for the Timer/Counter1, transitions on the T1 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

13.10.3 Timer/Counter1 Control Register C – TCCR1C



• Bit 7 – FOC1A: Force Output Compare for Channel A

• Bit 6 – FOC1B: Force Output Compare for Channel B

The FOC1A/FOC1B bits are only active when the WGM13:0 bits specifies a non-PWM mode. However, for ensuring compatibility with future devices, these bits must be set to zero when TCCR1A is written when operating in a PWM mode. When writing a logical one to the FOC1A/FOC1B bit, an immediate compare match is forced on the Waveform Generation unit. The OC1A/OC1B output is changed according to its COM1x1:0 bits setting. Note that the FOC1A/FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1x1:0 bits that determine the effect of the forced compare.





Figure 15-3. Output Compare Unit, Block Diagram

The OCR2x Register is double buffered when using any of the Pulse Width Modulation (PWM) modes. For the Normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR2x Compare Register to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR2x Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR2x Buffer Register, and if double buffering is disabled the CPU will access the OCR2x directly.

15.4.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC2x) bit. Forcing compare match will not set the OCF2x Flag or reload/clear the timer, but the OC2x pin will be updated as if a real compare match had occurred (the COM2x1:0 bits settings define whether the OC2x pin is set, cleared or toggled).

15.4.2 Compare Match Blocking by TCNT2 Write

All CPU write operations to the TCNT2 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR2x to be initialized to the same value as TCNT2 without triggering an interrupt when the Timer/Counter clock is enabled.

15.4.3 Using the Output Compare Unit

Since writing TCNT2 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT2 when using the Output Compare channel, independently of whether the Timer/Counter is running or not. If the value written to TCNT2 equals the OCR2x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT2 value equal to BOTTOM when the counter is downcounting.





The setup of the OC2x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC2x value is to use the Force Output Compare (FOC2x) strobe bit in Normal mode. The OC2x Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COM2x1:0 bits are not double buffered together with the compare value. Changing the COM2x1:0 bits will take effect immediately.

15.5 Compare Match Output Unit

The Compare Output mode (COM2x1:0) bits have two functions. The Waveform Generator uses the COM2x1:0 bits for defining the Output Compare (OC2x) state at the next compare match. Also, the COM2x1:0 bits control the OC2x pin output source. Figure 15-4 shows a simplified schematic of the logic affected by the COM2x1:0 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 COM2x1:0 bits are shown. When referring to the OC2x state, the reference is for the internal OC2x Register, not the OC2x pin.





The general I/O port function is overridden by the Output Compare (OC2x) from the Waveform Generator if either of the COM2x1:0 bits are set. However, the OC2x 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 OC2x pin (DDR_OC2x) must be set as output before the OC2x 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 OC2x state before the output is enabled. Note that some COM2x1:0 bit settings are reserved for certain modes of operation. See Section "15.8" on page 149.



17. USART0

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device. The main features are:

- Full Duplex Operation (Independent Serial Receive and Transmit Registers)
- Asynchronous or Synchronous Operation
- Master or Slave Clocked Synchronous Operation
- High Resolution Baud Rate Generator
- Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits
- Odd or Even Parity Generation and Parity Check Supported by Hardware
- Data OverRun Detection
- Framing Error Detection
- Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter
- Three Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete
- Multi-processor Communication Mode
- Double Speed Asynchronous Communication Mode

The USART can also be used in Master SPI mode, see "USART in SPI Mode" on page 196. The Power Reduction USART bit, PRUSARTO, in "Power Reduction Register - PRR" on page 40 must be disabled by writing a logical zero to it.

17.1 Overview

A simplified block diagram of the USART Transmitter is shown in Figure 17-1. CPU accessible I/O Registers and I/O pins are shown in bold.

The dashed boxes in the block diagram separate the three main parts of the USART (listed from the top): Clock Generator, Transmitter and Receiver. Control Registers are shared by all units. The Clock Generation logic consists of synchronization logic for external clock input used by synchronous slave operation, and the baud rate generator. The XCKn (Transfer Clock) pin is only used by synchronous transfer mode. The Transmitter consists of a single write buffer, a serial Shift Register, Parity Generator and Control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without any delay between frames. The Receiver is the most complex part of the USART module due to its clock and data recovery units. The recovery units are used for asynchronous data reception. In addition to the recovery units, the Receiver includes a Parity Checker, Control logic, a Shift Register and a two level receive buffer (UDRn). The Receiver supports the same frame formats as the Transmitter, and can detect Frame Error, Data OverRun and Parity Errors.



contains data to be transmitted that has not yet been moved into the Shift Register. For compatibility with future devices, always write this bit to zero when writing the UCSRnA Register.

When the Data Register Empty Interrupt Enable (UDRIEn) bit in UCSRnB is written to one, the USART Data Register Empty Interrupt will be executed as long as UDREn is set (provided that global interrupts are enabled). UDREn is cleared by writing UDRn. When interrupt-driven data transmission is used, the Data Register Empty interrupt routine must either write new data to UDRn in order to clear UDREn or disable the Data Register Empty interrupt, otherwise a new interrupt will occur once the interrupt routine terminates.

The Transmit Complete (TXCn) Flag bit is set one when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer. The TXCn Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXCn Flag is useful in half-duplex communication interfaces (like the RS-485 standard), where a transmitting application must enter receive mode and free the communication bus immediately after completing the transmission.

When the Transmit Compete Interrupt Enable (TXCIEn) bit in UCSRnB is set, the USART Transmit Complete Interrupt will be executed when the TXCn Flag becomes set (provided that global interrupts are enabled). When the transmit complete interrupt is used, the interrupt handling routine does not have to clear the TXCn Flag, this is done automatically when the interrupt is executed.

17.5.4 Parity Generator

The Parity Generator calculates the parity bit for the serial frame data. When parity bit is enabled (UPMn1 = 1), the transmitter control logic inserts the parity bit between the last data bit and the first stop bit of the frame that is sent.

17.5.5 Disabling the Transmitter

The disabling of the Transmitter (setting the TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxDn pin.

17.6 Data Reception – The USART Receiver

The USART Receiver is enabled by writing the Receive Enable (RXENn) bit in the UCSRnB Register to one. When the Receiver is enabled, the normal pin operation of the RxDn pin is overridden by the USART and given the function as the Receiver's serial input. The baud rate, mode of operation and frame format must be set up once before any serial reception can be done. If synchronous operation is used, the clock on the XCKn pin will be used as transfer clock.

17.6.1 Receiving Frames with 5 to 8 Data Bits

The Receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate or XCKn clock, and shifted into the Receive Shift Register until the first stop bit of a frame is received. A second stop bit will be ignored by the Receiver. When the first stop bit is received, i.e., a complete serial frame is present in the Receive Shift Register, the contents of the Shift Register will be moved into the receive buffer. The receive buffer can then be read by reading the UDRn I/O location.

19. 2-wire Serial Interface

19.1 **Features**

- Simple Yet Powerful and Flexible Communication Interface, only two Bus Lines Needed
- Both Master and Slave Operation Supported
- Device can Operate as Transmitter or Receiver
- 7-bit Address Space Allows up to 128 Different Slave Addresses
- Multi-master Arbitration Support
- Up to 400 kHz Data Transfer Speed
- Slew-rate Limited Output Drivers
- · Noise Suppression Circuitry Rejects Spikes on Bus Lines
- Fully Programmable Slave Address with General Call Support
- Address Recognition Causes Wake-up When AVR is in Sleep Mode

2-wire Serial Interface Bus Definition 19.2

The 2-wire Serial Interface (TWI) is ideally suited for typical microcontroller applications. The TWI protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for data (SDA). The only external hardware needed to implement the bus is a single pull-up resistor for each of the TWI bus lines. All devices connected to the bus have individual addresses, and mechanisms for resolving bus contention are inherent in the TWI protocol.





19.2.1 **TWI Terminology**

The following definitions are frequently encountered in this section.

Table 19-1. I WI Terminology					
Term	Description				
Master	The device that initiates and terminates a transmission. The Master also generates the SCL clock.				
Slave	The device addressed by a Master.				
Transmitter	The device placing data on the bus.				
Receiver	The device reading data from the bus.				







the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control over the bus.

 Table 19-4.
 Status codes for Master Receiver Mode

Status Code		Application Software Response					
(TWSR)	Status of the 2-wire Serial Bus		To TWCR				
are 0	Hardware	To/from TWDR	STA	STO	TWIN T	TWE A	Next Action Taken by TWI Hardware
0x08	A START condition has been transmitted	Load SLA+R	0	0	1	х	SLA+R will be transmitted ACK or NOT ACK will be received
0x10	A repeated START condition	Load SLA+R or	0	0	1	х	SLA+R will be transmitted
has been transmitted	has been transmitted	Load SLA+W	0	0	1	х	ACK or NOT ACK will be received SLA+W will be transmitted Logic will switch to Master Transmitter mode
0x38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or	0	0	1	х	2-wire Serial Bus will be released and not addressed Slave mode will be entered
		No TWDR action	1	0	1	х	A START condition will be transmitted when the bus becomes free
0x40 SLA+R has been	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
0x48	SLA+R has been transmitted;	No TWDR action or	1	0	1	Х	Repeated START will be transmitted
NOT	NOT ACK has been received	No TWDR action or	0	1	1	Х	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	х	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x50 Data byte has been re ACK has been return	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
0x58	Data byte has been received; NOT ACK has been returned	Read data byte or	1	0	1	Х	Repeated START will be transmitted
		Read data byte or	0	1	1	Х	STOP condition will be transmitted and TWSTO Flag will be reset
		Read data byte	1	1	1	Х	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset

read, neither register is updated and the result from the conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL Registers is re-enabled.

The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the Data Registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

21.2 Starting a Conversion

A single conversion is started by disabling the Power Reduction ADC bit, PRADC, in "Power Reduction Register - PRR" on page 40 by writing a logical zero to it and writing a logical one to the ADC Start Conversion bit, ADSC. This bit stays high as long as the conversion is in progress and will be cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.

Alternatively, a conversion can be triggered automatically by various sources. Auto Triggering is enabled by setting the ADC Auto Trigger Enable bit, ADATE in ADCSRA. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB (See description of the ADTS bits for a list of the trigger sources). When a positive edge occurs on the selected trigger signal, the ADC prescaler is reset and a conversion is started. This provides a method of starting conversions at fixed intervals. If the trigger signal still is set when the conversion completes, a new conversion will not be started. If another positive edge occurs on the trigger signal during conversion, the edge will be ignored. Note that an Interrupt Flag will be set even if the specific interrupt is disabled or the Global Interrupt Enable bit in SREG is cleared. A conversion can thus be triggered without causing an interrupt. However, the Interrupt Flag must be cleared in order to trigger a new conversion at the next interrupt event.





Using the ADC Interrupt Flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC Data Register. The first conversion must be started by writing a logical one to the ADSC bit in ADCSRA. In this mode the ADC will perform successive conversions independently of whether the ADC Interrupt Flag, ADIF is cleared or not.





When designing a system where debugWIRE will be used, the following observations must be made for correct operation:

- Pull-up resistors on the dW/(RESET) line must not be smaller than 10kΩ. The pull-up resistor is not required for debugWIRE functionality.
- Connecting the RESET pin directly to V_{CC} will not work.
- Capacitors connected to the RESET pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

22.4 Software Break Points

debugWIRE supports Program memory Break Points by the AVR Break instruction. Setting a Break Point in AVR Studio[®] will insert a BREAK instruction in the Program memory. The instruction replaced by the BREAK instruction will be stored. When program execution is continued, the stored instruction will be executed before continuing from the Program memory. A break can be inserted manually by putting the BREAK instruction in the program.

The Flash must be re-programmed each time a Break Point is changed. This is automatically handled by AVR Studio through the debugWIRE interface. The use of Break Points will therefore reduce the Flash Data retention. Devices used for debugging purposes should not be shipped to end customers.

22.5 Limitations of debugWIRE

The debugWIRE communication pin (dW) is physically located on the same pin as External Reset (RESET). An External Reset source is therefore not supported when the debugWIRE is enabled.

The debugWIRE system accurately emulates all I/O functions when running at full speed, i.e., when the program in the CPU is running. When the CPU is stopped, care must be taken while accessing some of the I/O Registers via the debugger (AVR Studio).

A programmed DWEN Fuse enables some parts of the clock system to be running in all sleep modes. This will increase the power consumption while in sleep. Thus, the DWEN Fuse should be disabled when debugWire is not used.

22.6 debugWIRE Related Register in I/O Memory

The following section describes the registers used with the debugWire.

22.6.1 debugWire Data Register – DWDR



The DWDR Register provides a communication channel from the running program in the MCU to the debugger. This register is only accessible by the debugWIRE and can therefore not be used as a general purpose register in the normal operations.

23. Self-Programming the Flash, ATmega48

In ATmega48, there is no Read-While-Write support, and no separate Boot Loader Section. The SPM instruction can be executed from the entire Flash.

²⁵⁶ **ATmega48/88/168**



Figure 27-8. Idle Supply Current vs. Frequency (1 - 24 MHz)









PRR bit	Additional Current consumption compared to Active with external clock (see Figure 27-1 and Figure 27-2)	Additional Current consumption compared to Idle with external clock (see Figure 27-7 and Figure 27-8)
PRUSART0	3.3%	18%
PRTWI	4.8%	26%
PRTIM2	4.7%	25%
PRTIM1	2.0%	11%
PRTIM0	1.6%	8.5%
PRSPI	6.1%	33%
PRADC	4.9%	26%

Table 27-2. Additional Current Consumption (percentage) in Active and Idle mode

It is possible to calculate the typical current consumption based on the numbers from Table 2 for other V_{CC} and frequency settings than listed in Table 1.

27.3.0.1 Example 1

Calculate the expected current consumption in idle mode with USART0, TIMER1, and TWI enabled at V_{CC} = 3.0V and F = 1MHz. From Table 2, third column, we see that we need to add 18% for the USART0, 26% for the TWI, and 11% for the TIMER1 module. Reading from Figure 3, we find that the idle current consumption is ~0,075mA at V_{CC} = 3.0V and F = 1MHz. The total current consumption in idle mode with USART0, TIMER1, and TWI enabled, gives:

 $ICC_{total} \approx 0.075 mA \bullet (1 + 0.18 + 0.26 + 0.11) \approx 0.116 mA$

27.3.0.2 Example 2

Same conditions as in example 1, but in active mode instead. From Table 2, second column we see that we need to add 3.3% for the USARTO, 4.8% for the TWI, and 2.0% for the TIMER1 module. Reading from Figure 1, we find that the active current consumption is ~0,42mA at V_{CC} = 3.0V and F = 1MHz. The total current consumption in idle mode with USARTO, TIMER1, and TWI enabled, gives:

 $ICC_{total} \approx 0.42 mA \bullet (1 + 0.033 + 0.048 + 0.02) \approx 0.46 mA$

27.3.0.3 Example 3

All I/O modules should be enabled. Calculate the expected current consumption in active mode at $V_{CC} = 3.6V$ and F = 10MHz. We find the active current consumption without the I/O modules to be ~ 4.0mA (from Figure 2). Then, by using the numbers from Table 2 - second column, we find the total current consumption:

 $ICC_{total} \approx 4.0 mA \bullet (1 + 0.033 + 0.048 + 0.047 + 0.02 + 0.016 + 0.061 + 0.049) \approx 5.1 mA$



The Asynchronous oscillator does not stop when entering power down mode. This leads to higher power consumption than expected.

Problem fix / Workaround

Manually disable the asynchronous timer before entering power down.

32.2 Errata ATmega88

The revision letter in this section refers to the revision of the ATmega88 device.

32.2.1 Rev. A

Writing to EEPROM does not work at low Operating Voltages

Part may hang in reset

1. Writing to EEPROM does not work at low operating voltages

Writing to the EEPROM does not work at low voltages.

Problem Fix/Workaround

Do not write the EEPROM at voltages below 4.5 Volts. This will be corrected in rev. B.

2. Part may hang in reset

Some parts may get stuck in a reset state when a reset signal is applied when the internal reset state-machine is in a specific state. The internal reset state-machine is in this state for approximately 10 ns immediately before the part wakes up after a reset, and in a 10 ns window when altering the system clock prescaler. The problem is most often seen during In-System Programming of the device. There are theoretical possibilities of this happening also in run-mode. The following three cases can trigger the device to get stuck in a reset-state:

- Two succeeding resets are applied where the second reset occurs in the 10ns window before the device is out of the reset-state caused by the first reset.

- A reset is applied in a 10 ns window while the system clock prescaler value is updated by software.

- Leaving SPI-programming mode generates an internal reset signal that can trigger this case.

The two first cases can occur during normal operating mode, while the last case occurs only during programming of the device.

Problem Fix/Workaround

The first case can be avoided during run-mode by ensuring that only one reset source is active. If an external reset push button is used, the reset start-up time should be selected such that the reset line is fully debounced during the start-up time.

The second case can be avoided by not using the system clock prescaler.

The third case occurs during In-System programming only. It is most frequently seen when using the internal RC at maximum frequency.

If the device gets stuck in the reset-state, turn power off, then on again to get the device out of this state.

32.2.2 Rev. D

No errata.

348 ATmega48/88/168