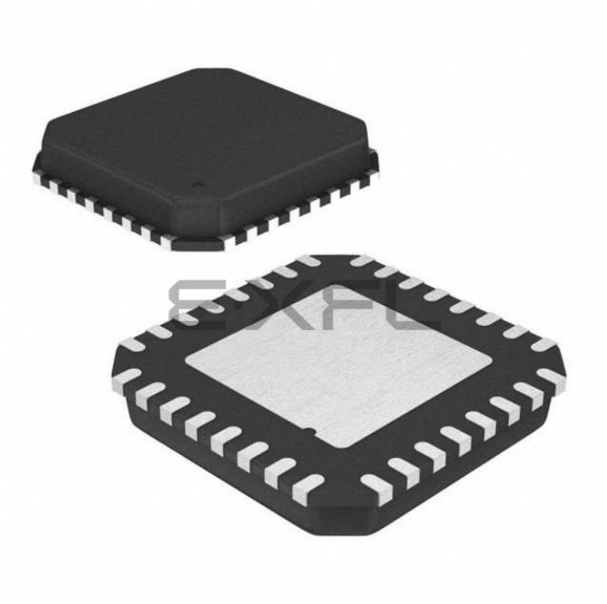
# Microchip Technology - ATMEGA88V-10MI Datasheet



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 "Embedded - Microcontrollers"

| **Details** | |
|---|---|
| Product Status | Obsolete |
| Core Processor | AVR |
| Core Size | 8-Bit |
| Speed | 10MHz |
| Connectivity | I²C, 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) | 1.8V ~ 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/atmega88v-10mi |

The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

Program Flash memory space is divided in two sections, the Boot Program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset routine (before subroutines or interrupts are executed). The Stack Pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, 0x20 - 0x5F. In addition, the ATmega48/88/168 has Extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

## 4.3 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, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the "Instruction Set" section for a detailed description.

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;
}
```

### 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.

**Table 6-13.** Start-up Times for the External Clock Selection

| Power Conditions | Start-up Time from Power-down and Power-save | Additional Delay from Reset ($V_{CC}$ = 5.0V) | SUT1..0 |
|---|---|---|---|
| BOD enabled | 6 CK | 14CK | 00 |
| Fast rising power | 6 CK | 14CK + 4.1 ms | 01 |
| Slowly rising power | 6 CK | 14CK + 65 ms | 10 |
| Reserved | | | 11 |

When applying an external clock, it is required to avoid sudden changes in the applied clock frequency to ensure stable operation of the MCU. A variation in frequency of more than 2% from one clock cycle to the next can lead to unpredictable behavior. If changes of more than 2% is required, ensure that the MCU is kept in Reset during the changes.

Note that the System Clock Prescaler can be used to implement run-time changes of the internal clock frequency while still ensuring stable operation. Refer to ”System Clock Prescaler” on page 34 for details.

## 6.9 Clock Output Buffer

The device can output the system clock on the CLKO pin. To enable the output, the CKOUT Fuse has to be programmed. This mode is suitable when the chip clock is used to drive other circuits on the system. The clock also will be output during reset, and the normal operation of I/O pin will be overridden when the fuse is programmed. Any clock source, including the internal RC Oscillator, can be selected when the clock is output on CLKO. If the System Clock Prescaler is used, it is the divided system clock that is output.

## 6.10 Timer/Counter Oscillator

The device can operate its Timer/Counter2 from an external 32.768 kHz watch crystal or a external clock source. The Timer/Counter Oscillator Pins (TOSC1 and TOSC2) are shared with XTAL1 and XTAL2. This means that the Timer/Counter Oscillator can only be used when an internal RC Oscillator is selected as system clock source. See Figure 6-2 on page 28 for crystal connection.

Applying an external clock source to TOSC1 requires EXTCLK in the ASSR Register written to logic one. See ”Asynchronous operation of the Timer/Counter” on page 155 for further description on selecting external clock as input instead of a 32 kHz crystal.

## 6.11 System Clock Prescaler

The ATmega48/88/168 has a system clock prescaler, and the system clock can be divided by setting the ”Clock Prescale Register – CLKPR” on page 357. This feature can be used to decrease the system clock frequency and the power consumption when the requirement for processing power is low. This can be used with all clock source options, and it will affect the clock frequency of the CPU and all synchronous peripherals. $clk_{I/O}$, $clk_{ADC}$, $clk_{CPU}$, and $clk_{FLASH}$ are divided by a factor as shown in Table 8-1 on page 44.

When switching between prescaler settings, the System Clock Prescaler ensures that no glitches occurs in the clock system. It also ensures that no intermediate frequency is higher than neither the clock frequency corresponding to the previous setting, nor the clock frequency corresponding to the new setting. The ripple counter that implements the prescaler runs at the

## 7.7 Minimizing Power Consumption

There are several possibilities to consider when trying to minimize the power consumption in an AVR controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device's functions are operating. All functions not needed should be disabled. In particular, the following modules may need special consideration when trying to achieve the lowest possible power consumption.

### 7.7.1 Analog to Digital Converter

If enabled, the ADC will be enabled in all sleep modes. To save power, the ADC should be disabled before entering any sleep mode. When the ADC is turned off and on again, the next conversion will be an extended conversion. Refer to "Analog-to-Digital Converter" on page 239 for details on ADC operation.

### 7.7.2 Analog Comparator

When entering Idle mode, the Analog Comparator should be disabled if not used. When entering ADC Noise Reduction mode, the Analog Comparator should be disabled. In other sleep modes, the Analog Comparator is automatically disabled. However, if the Analog Comparator is set up to use the Internal Voltage Reference as input, the Analog Comparator should be disabled in all sleep modes. Otherwise, the Internal Voltage Reference will be enabled, independent of sleep mode. Refer to "Analog Comparator" on page 235 for details on how to configure the Analog Comparator.

### 7.7.3 Brown-out Detector

If the Brown-out Detector is not needed by the application, this module should be turned off. If the Brown-out Detector is enabled by the BODLEVEL Fuses, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to "Brown-out Detection" on page 46 for details on how to configure the Brown-out Detector.

### 7.7.4 Internal Voltage Reference

The Internal Voltage Reference will be enabled when needed by the Brown-out Detection, the Analog Comparator or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be used immediately. Refer to "Internal Voltage Reference" on page 48 for details on the start-up time.

### 7.7.5 Watchdog Timer

If the Watchdog Timer is not needed in the application, the module should be turned off. If the Watchdog Timer is enabled, it will be enabled in all sleep modes and hence always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to "Watchdog Timer" on page 49 for details on how to configure the Watchdog Timer.

### 7.7.6 Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important is then to ensure that no pins drive resistive loads. In sleep modes where both the I/O clock ($clk_{I/O}$) and the ADC clock ($clk_{ADC}$) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In

- **Bit 1 – EXTRF: External Reset Flag**

This bit is set if an External Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 0 – PORF: Power-on Reset Flag**

This bit is set if a Power-on Reset occurs. The bit is reset only by writing a logic zero to the flag.

To make use of the Reset Flags to identify a reset condition, the user should read and then Reset the MCUSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags.

## 8.1 Internal Voltage Reference

ATmega48/88/168 features an internal bandgap reference. This reference is used for Brown-out Detection, and it can be used as an input to the Analog Comparator or the ADC.

### 8.1.1 Voltage Reference Enable Signals and Start-up Time

The voltage reference has a start-up time that may influence the way it should be used. The start-up time is given in Table 8-4. To save power, the reference is not always turned on. The reference is on during the following situations:

1. When the BOD is enabled (by programming the BODLEVEL [2..0] Fuses).
2. When the bandgap reference is connected to the Analog Comparator (by setting the ACBG bit in ACSR).
3. When the ADC is enabled.

Thus, when the BOD is not enabled, after setting the ACBG bit or enabling the ADC, the user must always allow the reference to start up before the output from the Analog Comparator or ADC is used. To reduce power consumption in Power-down mode, the user can avoid the three conditions above to ensure that the reference is turned off before entering Power-down mode.

**Table 8-4.** Internal Voltage Reference Characteristics

| Symbol | Parameter | Condition | Min(1) | Typ(1) | Max(1) | Units |
|---|---|---|---|---|---|---|
| $V_{BG}$ | Bandgap reference voltage | $V_{CC}$=2.7<br>$T_A$=25°C | 1.0 | 1.1 | 1.2 | V |
| $t_{BG}$ | Bandgap reference start-up time | $V_{CC}$=2.7<br>$T_A$=25°C | | 40 | 70 | µs |
| $I_{BG}$ | Bandgap reference current consumption | $V_{CC}$=2.7<br>$T_A$=25°C | | 10 | | µA |

Note: 1. Values are guidelines only. Actual values are TBD.

# 9. Interrupts

This section describes the specifics of the interrupt handling as performed in ATmega48/88/168. For a general explanation of the AVR interrupt handling, refer to ”Reset and Interrupt Handling” on page 12.

The interrupt vectors in ATmega48, ATmega88 and ATmega168 are generally the same, with the following differences:

- Each Interrupt Vector occupies two instruction words in ATmega168, and one instruction word in ATmega48 and ATmega88.
- ATmega48 does not have a separate Boot Loader Section. In ATmega88 and ATmega168, the Reset Vector is affected by the BOOTRST fuse, and the Interrupt Vector start address is affected by the IVSEL bit in MCUCR.

## 9.1 Interrupt Vectors in ATmega48

**Table 9-1.** Reset and Interrupt Vectors in ATmega48

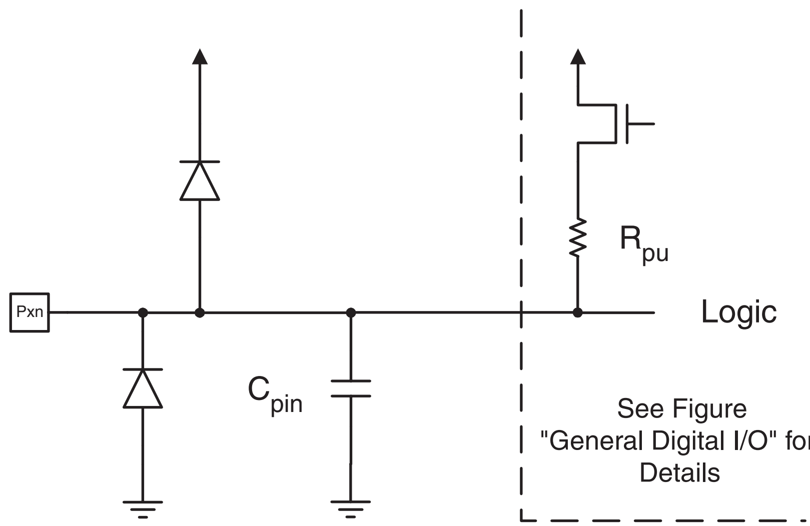
| Vector No. | Program Address | Source | Interrupt Definition |
|---|---|---|---|
| 1 | 0x000 | RESET | External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset |
| 2 | 0x001 | INT0 | External Interrupt Request 0 |
| 3 | 0x002 | INT1 | External Interrupt Request 1 |
| 4 | 0x003 | PCINT0 | Pin Change Interrupt Request 0 |
| 5 | 0x004 | PCINT1 | Pin Change Interrupt Request 1 |
| 6 | 0x005 | PCINT2 | Pin Change Interrupt Request 2 |
| 7 | 0x006 | WDT | Watchdog Time-out Interrupt |
| 8 | 0x007 | TIMER2 COMPA | Timer/Counter2 Compare Match A |
| 9 | 0x008 | TIMER2 COMPB | Timer/Counter2 Compare Match B |
| 10 | 0x009 | TIMER2 OVF | Timer/Counter2 Overflow |
| 11 | 0x00A | TIMER1 CAPT | Timer/Counter1 Capture Event |
| 12 | 0x00B | TIMER1 COMPA | Timer/Counter1 Compare Match A |
| 13 | 0x00C | TIMER1 COMPB | Timer/Coutner1 Compare Match B |
| 14 | 0x00D | TIMER1 OVF | Timer/Counter1 Overflow |
| 15 | 0x00E | TIMER0 COMPA | Timer/Counter0 Compare Match A |
| 16 | 0x00F | TIMER0 COMPB | Timer/Counter0 Compare Match B |
| 17 | 0x010 | TIMER0 OVF | Timer/Counter0 Overflow |
| 18 | 0x011 | SPI, STC | SPI Serial Transfer Complete |
| 19 | 0x012 | USART, RX | USART Rx Complete |
| 20 | 0x013 | USART, UDRE | USART, Data Register Empty |
| 21 | 0x014 | USART, TX | USART, Tx Complete |
| 22 | 0x015 | ADC | ADC Conversion Complete |
| 23 | 0x016 | EE READY | EEPROM Ready |

# 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.

**Figure 10-1.** I/O Pin Equivalent Schematic




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.

- **T1/OC0B/PCINT21 – Port D, Bit 5**

T1, Timer/Counter1 counter source.

OC0B, Output Compare Match output: The PD5 pin can serve as an external output for the Timer/Counter0 Compare Match B. The PD5 pin has to be configured as an output (DDD5 set (one)) to serve this function. The OC0B pin is also the output pin for the PWM mode timer function.

PCINT21: Pin Change Interrupt source 21. The PD5 pin can serve as an external interrupt source.

- **XCK/T0/PCINT20 – Port D, Bit 4**

XCK, USART external clock.

T0, Timer/Counter0 counter source.

PCINT20: Pin Change Interrupt source 20. The PD4 pin can serve as an external interrupt source.

- **INT1/OC2B/PCINT19 – Port D, Bit 3**

INT1, External Interrupt source 1: The PD3 pin can serve as an external interrupt source.

OC2B, Output Compare Match output: The PD3 pin can serve as an external output for the Timer/Counter0 Compare Match B. The PD3 pin has to be configured as an output (DDD3 set (one)) to serve this function. The OC2B pin is also the output pin for the PWM mode timer function.

PCINT19: Pin Change Interrupt source 19. The PD3 pin can serve as an external interrupt source.

- **INT0/PCINT18 – Port D, Bit 2**

INT0, External Interrupt source 0: The PD2 pin can serve as an external interrupt source.

PCINT18: Pin Change Interrupt source 18. The PD2 pin can serve as an external interrupt source.

- **TXD/PCINT17 – Port D, Bit 1**

TXD, Transmit Data (Data output pin for the USART). When the USART Transmitter is enabled, this pin is configured as an output regardless of the value of DDD1.

PCINT17: Pin Change Interrupt source 17. The PD1 pin can serve as an external interrupt source.

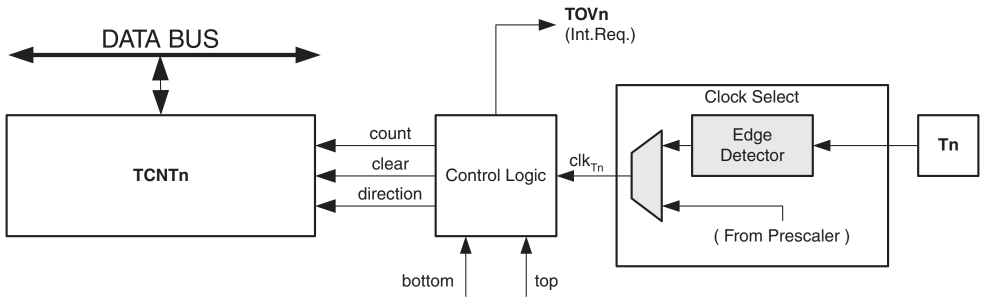
- **RXD/PCINT16 – Port D, Bit 0**

RXD, Receive Data (Data input pin for the USART). When the USART Receiver is enabled this pin is configured as an input regardless of the value of DDD0. When the USART forces this pin to be an input, the pull-up can still be controlled by the PORTD0 bit.

PCINT16: Pin Change Interrupt source 16. The PD0 pin can serve as an external interrupt source.

Table 10-10 and Table 10-11 relate the alternate functions of Port D to the overriding signals shown in Figure 10-5 on page 69.

**Figure 12-2.** Counter Unit Block Diagram




Signal description (internal signals):

| | |
|---|---|
| **count** | Increment or decrement TCNT0 by 1. |
| **direction** | Select between increment and decrement. |
| **clear** | Clear TCNT0 (set all bits to zero). |
| **$clk_{Tn}$** | Timer/Counter clock, referred to as $clk_{T0}$ in the following. |
| **top** | Signalize that TCNT0 has reached maximum value. |
| **bottom** | Signalize that TCNT0 has reached minimum value (zero). |

Depending of the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock ($clk_{T0}$). $clk_{T0}$ can be generated from an external or internal clock source, selected by the Clock Select bits (CS02:0). When no clock source is selected (CS02:0 = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether $clk_{T0}$ 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 WGM01 and WGM00 bits located in the Timer/Counter Control Register (TCCR0A) and the WGM02 bit located in the Timer/Counter Control Register B (TCCR0B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC0A and OC0B. For more details about advanced counting sequences and waveform generation, see ”Modes of Operation” on page 93.

The Timer/Counter Overflow Flag (TOV0) is set according to the mode of operation selected by the WGM02:0 bits. TOV0 can be used for generating a CPU interrupt.

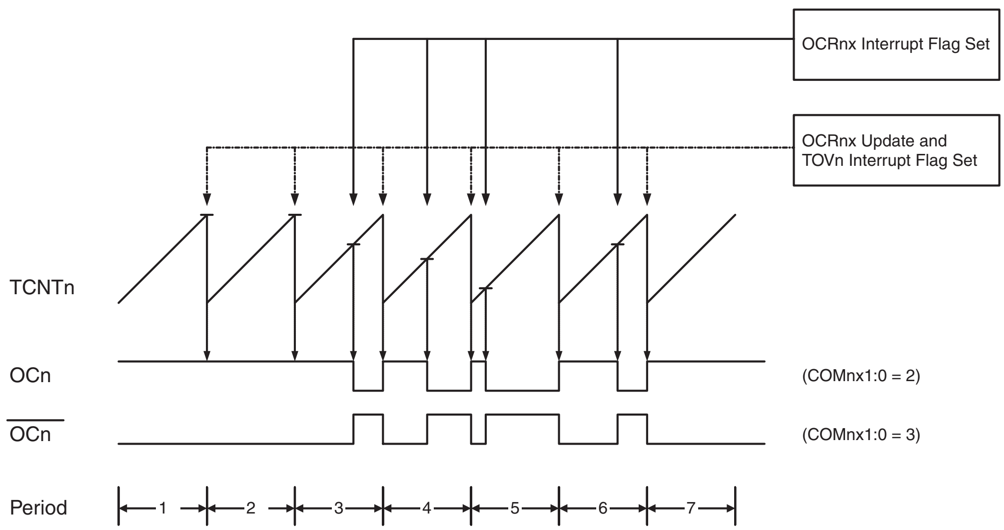
## 12.4 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the Output Compare Registers (OCR0A and OCR0B). Whenever TCNT0 equals OCR0A or OCR0B, the comparator signals a match. A match will set the Output Compare Flag (OCF0A or OCF0B) 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 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 WGM02:0 bits and Compare Output mode (COM0x1:0) 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 93).

Figure 12-3 shows a block diagram of the Output Compare unit.

PWM mode is shown in Figure 12-6. The TCNT0 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 TCNT0 slopes represent compare matches between OCR0x and TCNT0.

**Figure 12-6.** Fast PWM Mode, Timing Diagram




The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches TOP. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC0x pins. Setting the COM0x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM0x1:0 to three: Setting the COM0A1:0 bits to one allows the OC0A pin to toggle on Compare Matches if the WGM02 bit is set. This option is not available for the OC0B pin (see Table 12-6 on page 100). The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC0x Register at the compare match between OCR0x and TCNT0, and clearing (or setting) the OC0x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{\text{clk\_I/O}}}{N \cdot 256}$$

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

The extreme values for the OCR0A Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR0A is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR0A equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM0A1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC0x to toggle its logical level on each compare match (COM0x1:0 = 1). The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0A is set to zero. This

## 12.8 8-bit Timer/Counter Register Description

### 12.8.1 Timer/Counter Control Register A – TCCR0A

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | COM0A1 | COM0A0 | COM0B1 | COM0B0 | – | – | WGM01 | WGM00 | TCCR0A |
| Read/Write | R/W | R/W | R/W | R/W | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bits 7:6 – COM0A1:0: Compare Match Output A Mode**

These bits control the Output Compare pin (OC0A) behavior. If one or both of the COM0A1:0 bits are set, the OC0A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC0A pin must be set in order to enable the output driver.

When OC0A is connected to the pin, the function of the COM0A1:0 bits depends on the WGM02:0 bit setting. Table 12-2 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

**Table 12-2.** Compare Output Mode, non-PWM Mode

| COM0A1 | COM0A0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC0A disconnected. |
| 0 | 1 | Toggle OC0A on Compare Match |
| 1 | 0 | Clear OC0A on Compare Match |
| 1 | 1 | Set OC0A on Compare Match |

Table 12-3 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

**Table 12-3.** Compare Output Mode, Fast PWM Mode[(1)]

| COM0A1 | COM0A0 | Description |
|---|---|---|
| 0 | 0 | Normal port operation, OC0A disconnected. |
| 0 | 1 | WGM02 = 0: Normal Port Operation, OC0A Disconnected.<br>WGM02 = 1: Toggle OC0A on Compare Match. |
| 1 | 0 | Clear OC0A on Compare Match, set OC0A at TOP |
| 1 | 1 | Set OC0A on Compare Match, clear OC0A at TOP |

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See ”Fast PWM Mode” on page 94 for more details.

# 13. 16-bit Timer/Counter1 with PWM

The 16-bit Timer/Counter unit allows accurate program execution timing (event management), wave generation, and signal timing measurement. The main features are:

- **True 16-bit Design (i.e., Allows 16-bit PWM)**
- **Two independent Output Compare Units**
- **Double Buffered Output Compare Registers**
- **One Input Capture Unit**
- **Input Capture Noise Canceler**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Variable PWM Period**
- **Frequency Generator**
- **External Event Counter**
- **Four independent interrupt Sources (TOV1, OCF1A, OCF1B, and ICF1)**

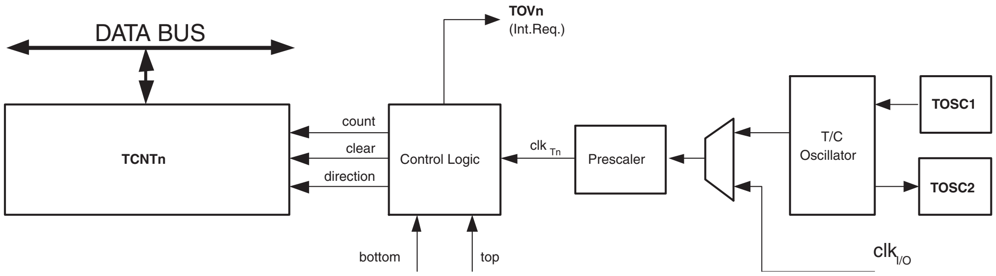
## 13.1 Overview

Most register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, and a lower case “x” replaces the Output Compare unit channel. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT1 for accessing Timer/Counter1 counter value and so on.

A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 13-1. For the actual placement of I/O pins, refer to ”Pinout ATmega48/88/168” on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the ”16-bit Timer/Counter Register Description” on page 128.

The PRTIM1 bit in ”Power Reduction Register - PRR” on page 40 must be written to zero to enable Timer/Counter1 module.

**Figure 15-2.** Counter Unit Block Diagram




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.

- **Bits 5:4 – UPMn1:0: Parity Mode**

These bits enable and set type of parity generation and check. If enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPMn setting. If a mismatch is detected, the UPEn Flag in UCSRnA will be set.

**Table 17-5.** UPMn Bits Settings

| UPMn1 | UPMn0 | Parity Mode |
|---|---|---|
| 0 | 0 | Disabled |
| 0 | 1 | Reserved |
| 1 | 0 | Enabled, Even Parity |
| 1 | 1 | Enabled, Odd Parity |

- **Bit 3 – USBSn: Stop Bit Select**

This bit selects the number of stop bits to be inserted by the Transmitter. The Receiver ignores this setting.

**Table 17-6.** USBS Bit Settings

| USBSn | Stop Bit(s) |
|---|---|
| 0 | 1-bit |
| 1 | 2-bit |

- **Bit 2:1 – UCSZn1:0: Character Size**

The UCSZn1:0 bits combined with the UCSZn2 bit in UCSRnB sets the number of data bits (Character SiZe) in a frame the Receiver and Transmitter use.

**Table 17-7.** UCSZn Bits Settings

| UCSZn2 | UCSZn1 | UCSZn0 | Character Size |
|---|---|---|---|
| 0 | 0 | 0 | 5-bit |
| 0 | 0 | 1 | 6-bit |
| 0 | 1 | 0 | 7-bit |
| 0 | 1 | 1 | 8-bit |
| 1 | 0 | 0 | Reserved |
| 1 | 0 | 1 | Reserved |
| 1 | 1 | 0 | Reserved |
| 1 | 1 | 1 | 9-bit |

- **Bit 0 – UCPOLn: Clock Polarity**

This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOLn bit sets the relationship between data output change and data input sample, and the synchronous clock (XCKn).

Writing this bit to one enables the USART Receiver in MSPIM mode. The Receiver will override normal port operation for the RxDn pin when enabled. Disabling the Receiver will flush the receive buffer. Only enabling the receiver in MSPI mode (i.e. setting RXENn=1 and TXENn=0) has no meaning since it is the transmitter that controls the transfer clock and since only master mode is supported.

- **Bit 3 - TXENn: Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxDn pin when enabled. The disabling of the Transmitter (writing TXENn 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 port.

- **Bit 2:0 - Reserved Bits in MSPI mode**

When in MSPI mode, these bits are reserved for future use. For compatibility with future devices, these bits must be written to zero when UCSRnB is written.

### 18.6.4 USART MSPIM Control and Status Register n C - UCSRnC

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | UMSELn1 | UMSELn0 | - | - | - | UDORDn | UCPHAn | UCPOLn | UCSRnC |
| Read/Write | R/W | R/W | R | R | R | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | |

- **Bit 7:6 - UMSELn1:0: USART Mode Select**

These bits select the mode of operation of the USART as shown in Table 18-3. See "USART Control and Status Register n C – UCSRnC" on page 189 for full description of the normal USART operation. The MSPIM is enabled when both UMSELn bits are set to one. The UDORDn, UCPHAn, and UCPOLn can be set in the same write operation where the MSPIM is enabled.

**Table 18-3.** UMSELn Bits Settings

| UMSELn1 | UMSELn0 | Mode |
|---|---|---|
| 0 | 0 | Asynchronous USART |
| 0 | 1 | Synchronous USART |
| 1 | 0 | (Reserved) |
| 1 | 1 | Master SPI (MSPIM) |

- **Bit 5:3 - Reserved Bits in MSPI mode**

When in MSPI mode, these bits are reserved for future use. For compatibility with future devices, these bits must be written to zero when UCSRnC is written.

- **Bit 2 - UDORDn: Data Order**

When set to one the LSB of the data word is transmitted first. When set to zero the MSB of the data word is transmitted first. Refer to the Frame Formats section page 4 for details.

- **Bit 1 - UCPHAn: Clock Phase**

The UCPHAn bit setting determine if data is sampled on the leasing edge (first) or tailing (last) edge of XCKn. Refer to the SPI Data Modes and Timing section page 4 for details.

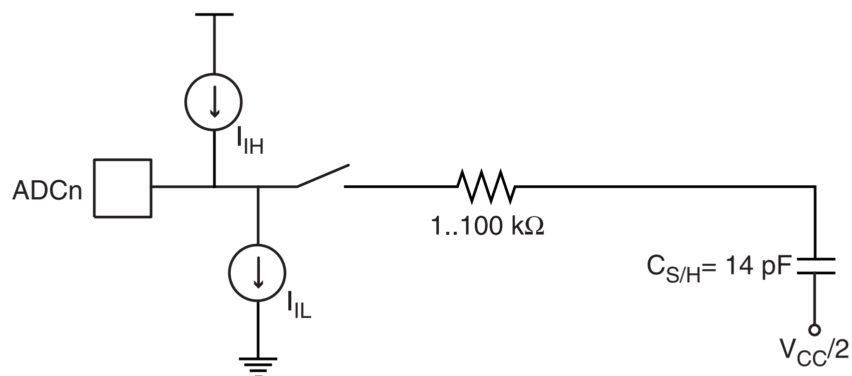
### 21.5.1 Analog Input Circuitry

The analog input circuitry for single ended channels is illustrated in Figure 21-8. An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

The ADC is optimized for analog signals with an output impedance of approximately 10 kΩ or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long time the source needs to charge the S/H capacitor, with can vary widely. The user is recommended to only use low impedant sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor.

Signal components higher than the Nyquist frequency ($f_{ADC}/2$) should not be present for either kind of channels, to avoid distortion from unpredictable signal convolution. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

**Figure 21-8.** Analog Input Circuitry




### 21.5.2 Analog Noise Canceling Techniques

Digital circuitry inside and outside the device generates EMI which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

a. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.

b. The $AV_{CC}$ pin on the device should be connected to the digital $V_{CC}$ supply voltage via an LC network as shown in Figure 21-9.

c. Use the ADC noise canceler function to reduce induced noise from the CPU.

d. If any ADC [3..0] port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress. However, using the 2-wire Interface (ADC4

# 24. Boot Loader Support – Read-While-Write Self-Programming, ATmega88 and ATmega168

In ATmega88 and ATmega168, the Boot Loader Support provides a real Read-While-Write Self-Programming mechanism for downloading and uploading program code by the MCU itself. This feature allows flexible application software updates controlled by the MCU using a Flash-resident Boot Loader program. The Boot Loader program can use any available data interface and associated protocol to read code and write (program) that code into the Flash memory, or read the code from the program memory. The program code within the Boot Loader section has the capability to write into the entire Flash, including the Boot Loader memory. The Boot Loader can thus even modify itself, and it can also erase itself from the code if the feature is not needed anymore. The size of the Boot Loader memory is configurable with fuses and the Boot Loader has two separate sets of Boot Lock bits which can be set independently. This gives the user a unique flexibility to select different levels of protection.

## 24.1 Boot Loader Features

- **Read-While-Write Self-Programming**
- **Flexible Boot Memory Size**
- **High Security (Separate Boot Lock Bits for a Flexible Protection)**
- **Separate Fuse to Select Reset Vector**
- **Optimized Page[(1)] Size**
- **Code Efficient Algorithm**
- **Efficient Read-Modify-Write Support**

Note: 1. A page is a section in the Flash consisting of several bytes (see Table 25-8 on page 284) used during programming. The page organization does not affect normal operation.

## 24.2 Application and Boot Loader Flash Sections

The Flash memory is organized in two main sections, the Application section and the Boot Loader section (see Figure 24-2). The size of the different sections is configured by the BOOTSZ Fuses as shown in Table 24-6 on page 276 and Figure 24-2. These two sections can have different level of protection since they have different sets of Lock bits.

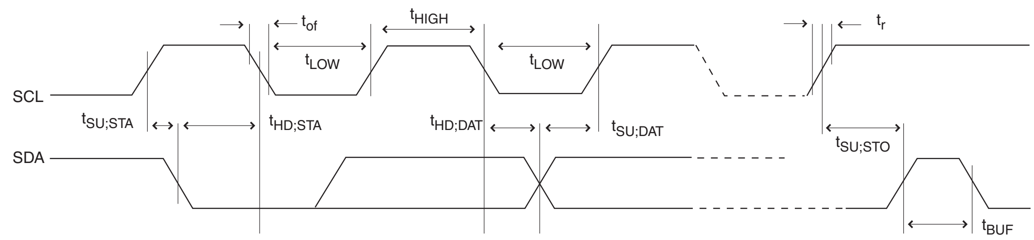
### 24.2.1 Application Section

The Application section is the section of the Flash that is used for storing the application code. The protection level for the Application section can be selected by the application Boot Lock bits (Boot Lock bits 0), see Table 24-2 on page 268. The Application section can never store any Boot Loader code since the SPM instruction is disabled when executed from the Application section.

### 24.2.2 BLS – Boot Loader Section

While the Application section is used for storing the application code, the The Boot Loader software must be located in the BLS since the SPM instruction can initiate a programming when executing from the BLS only. The SPM instruction can access the entire Flash, including the BLS itself. The protection level for the Boot Loader section can be selected by the Boot Loader Lock bits (Boot Lock bits 1), see Table 24-3 on page 268.

**Figure 26-4.** 2-wire Serial Bus Timing




## 26.7 SPI Timing Characteristics

See Figure 26-5 and Figure 26-6 for details.

**Table 26-3.** SPI Timing Parameters

| | Description | Mode | Min | Typ | Max | |
|---|---|---|---|---|---|---|
| 1 | SCK period | Master | | See Table 16-4 | | ns |
| 2 | SCK high/low | Master | | 50% duty cycle | | |
| 3 | Rise/Fall time | Master | | 3.6 | | |
| 4 | Setup | Master | | 10 | | |
| 5 | Hold | Master | | 10 | | |
| 6 | Out to SCK | Master | | $0.5 \bullet t_{sck}$ | | |
| 7 | SCK to out | Master | | 10 | | |
| 8 | SCK to out high | Master | | 10 | | |
| 9 | $\overline{SS}$ low to out | Slave | | 15 | | |
| 10 | SCK period | Slave | $4 \bullet t_{ck}$ | | | |
| 11 | SCK high/low[1] | Slave | $2 \bullet t_{ck}$ | | | |
| 12 | Rise/Fall time | Slave | | | 1600 | |
| 13 | Setup | Slave | 10 | | | |
| 14 | Hold | Slave | $t_{ck}$ | | | |
| 15 | SCK to out | Slave | | 15 | | |
| 16 | SCK to $\overline{SS}$ high | Slave | 20 | | | |
| 17 | $\overline{SS}$ high to tri-state | Slave | | 10 | | |
| 18 | $\overline{SS}$ low to SCK | Slave | 20 | | | |

Note: 1. In SPI Programming mode the minimum SCK high/low period is:
- 2 $t_{CLCL}$ for $f_{CK}$ < 12 MHz
- 3 $t_{CLCL}$ for $f_{CK}$ > 12 MHz

2. All DC Characteristics contained in this datasheet are based on simulation and characterization of other AVR microcontrollers manufactured in the same process technology. These values are preliminary values representing design targets, and will be updated after characterization of actual silicon.

# 28. Register Summary

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Page |
|---|---|---|---|---|---|---|---|---|---|---|
| (0xFF) | Reserved | – | – | – | – | – | – | – | – | |
| (0xFE) | Reserved | – | – | – | – | – | – | – | – | |
| (0xFD) | Reserved | – | – | – | – | – | – | – | – | |
| (0xFC) | Reserved | – | – | – | – | – | – | – | – | |
| (0xFB) | Reserved | – | – | – | – | – | – | – | – | |
| (0xFA) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF9) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF8) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF7) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF6) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF5) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF4) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF3) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF2) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF1) | Reserved | – | – | – | – | – | – | – | – | |
| (0xF0) | Reserved | – | – | – | – | – | – | – | – | |
| (0xEF) | Reserved | – | – | – | – | – | – | – | – | |
| (0xEE) | Reserved | – | – | – | – | – | – | – | – | |
| (0xED) | Reserved | – | – | – | – | – | – | – | – | |
| (0xEC) | Reserved | – | – | – | – | – | – | – | – | |
| (0xEB) | Reserved | – | – | – | – | – | – | – | – | |
| (0xEA) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE9) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE8) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE7) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE6) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE5) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE4) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE3) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE2) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE1) | Reserved | – | – | – | – | – | – | – | – | |
| (0xE0) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDF) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDE) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDD) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDC) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDB) | Reserved | – | – | – | – | – | – | – | – | |
| (0xDA) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD9) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD8) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD7) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD6) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD5) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD4) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD3) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD2) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD1) | Reserved | – | – | – | – | – | – | – | – | |
| (0xD0) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCF) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCE) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCD) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCC) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCB) | Reserved | – | – | – | – | – | – | – | – | |
| (0xCA) | Reserved | – | – | – | – | – | – | – | – | |
| (0xC9) | Reserved | – | – | – | – | – | – | – | – | |
| (0xC8) | Reserved | – | – | – | – | – | – | – | – | |
| (0xC7) | Reserved | – | – | – | – | – | – | – | – | |
| (0xC6) | UDR0 | USART I/O Data Register | | | | | | | | 187 |
| (0xC5) | UBRR0H | | | | | USART Baud Rate Register High | | | | 191 |
| (0xC4) | UBRR0L | USART Baud Rate Register Low | | | | | | | | 191 |
| (0xC3) | Reserved | – | – | – | – | – | – | – | – | |
| (0xC2) | UCSR0C | UMSEL01 | UMSEL00 | UPM01 | UPM00 | USBS0 | UCSZ01 /UDORD0 | UCSZ00 / UCPHA0 | UCPOL0 | 189/203 |
| (0xC1) | UCSR0B | RXCIE0 | TXCIE0 | UDRIE0 | RXEN0 | TXEN0 | UCSZ02 | RXB80 | TXB80 | 188 |
| (0xC0) | UCSR0A | RXC0 | TXC0 | UDRE0 | FE0 | DOR0 | UPE0 | U2X0 | MPCM0 | 187 |