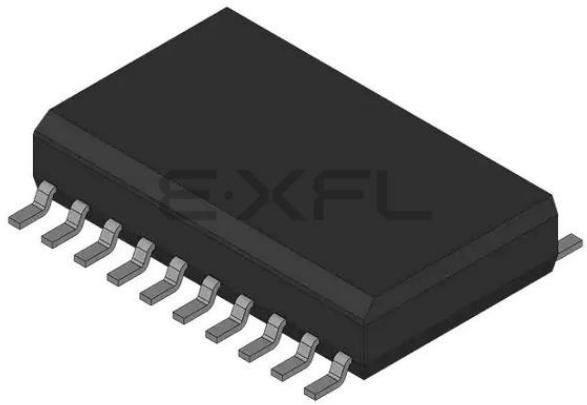


Welcome to E-XFL.COM

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	16MHz
Connectivity	I²C, LINbus, SPI, UART/USART, USI
Peripherals	Brown-out Detect/Reset, POR, PWM, Temp Sensor, WDT
Number of I/O	16
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SOIC (0.295", 7.50mm Width)
Supplier Device Package	20-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/atmel/attiny87-15sz

1.3 Automotive Quality Grade

The Atmel® ATtiny87/167 have been developed and manufactured according to the most stringent requirements of the international standard ISO-TS-16949. This data sheet contains limit values extracted from the results of extensive characterization (temperature and voltage). The quality and reliability of the Atmel ATtiny87/167 have been verified during regular product qualification as per AEC-Q100 grade 1.

As indicated in the ordering information paragraph, this document refers only to grade 1 products, for grade 0 products refer to appendix A.

Table 1-2. Temperature Grade Identification for Automotive Products

Temperature	Temperature Identifier	Comments
–40°C/+125°C	Z	Grade 1
–40°C/+150°C	D	Grade 0

1.4 Disclaimer

Typical values contained in this data sheet are based on simulations and characterization of other AVR® microcontrollers manufactured on the same process technology. Min. and Max values will be available after the device is characterized.

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

3.3.6 Preventing EEPROM Corruption

During periods of low Vcc, 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:

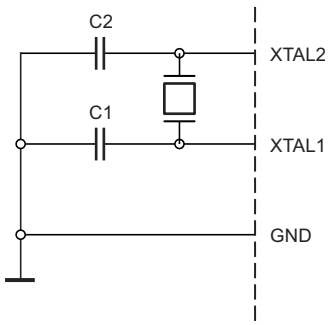
Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low Vcc reset protection circuit can be used. If a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

4.2.4 Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 4-2. Either a quartz crystal or a ceramic resonator may be used.

C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for choosing capacitors for use with crystals are given in Table 4-6. For ceramic resonators, the capacitor values given by the manufacturer should be used.

Figure 4-2. Crystal Oscillator Connections



The oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by CKSEL3..1 fuses or by CSEL3..1 field as shown in Table 4-6.

Table 4-6. Crystal Oscillator Operating Modes

CKSEL3..1[1] CSEL3..1[2]	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
100[3]	0.4 - 0.9	–
101	0.9 - 3.0	12 - 22
110	3.0 - 8.0	12 - 22
111	8.0 - 16.0	12 - 22

Notes:
1. Flash fuse bits.
2. CLKSELR register bits.
3. This option should not be used with crystals, only with ceramic resonators.

4.5.3 CLKCSR – Clock Control & Status Register

Bit	7	6	5	4	3	2	1	0	
	CLKCCE	–	–	CLKRDY	CLKC3	CLKC2	CLKC1	CLKC0	CLKCSR
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – CLKCCE: Clock Control Change Enable**

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

- **Bits 6:5 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny87/167 and will always read as zero.

- **Bits 4 – CLKRDY: Clock Ready Flag**

This flag is the output of the '*clock availability* ' logic.
This flag is cleared by the '*request for clock availability*' command or '*enable clock source*' command being entered.
It is set when '*clock availability*' logic confirms that the (selected) clock is running and is stable. The delay from the request and the flag setting is not fixed, it depends on the clock start-up time, the clock frequency and, of course, if the clock is alive. The user's code has to differentiate between '***no_clock_signal***' and '***clock_signal_not_yet_available***' condition.

- **Bits 3:0 – CLKC3:0: Clock Control Bits 3 - 0**

These bits define the command to provide to the '*Clock Switch'* module. The special write procedure must be followed to change the CLKC3..0 bits (See "Bit 7 – CLKCCE: Clock Control Change Enable" on page 40.).

1. Write the clock control change enable (CLKCCE) bit to one and all other bits in CLKCSR to zero.
2. Within 4 cycles, write the desired value to CLKCSR register while clearing CLKCCE bit.

Interrupts should be disabled when setting CLKCSR register in order not to disturb the procedure.

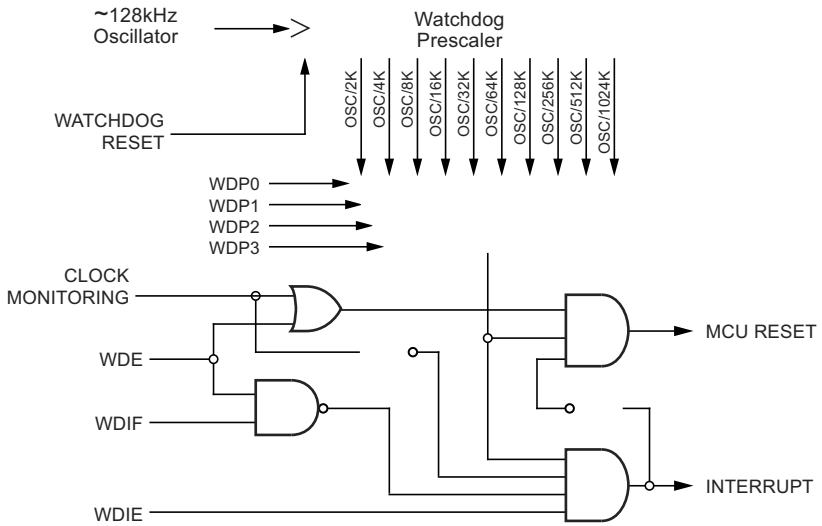
Table 4-11. Clock Command List

Clock Command	CLKC3..0
No command	0000_b
Disable clock source	0001_b
Enable clock source	0010_b
Request for clock availability	0011_b
Clock source switch	0100_b
Recover system clock source code	0101_b
Enable watchdog in automatic reload mode	0110_b
CKOUT command	0111_b
No command	$1xxx_b$

6.3.1 Watchdog Timer Behavior

The watchdog timer (WDT) is a timer counting cycles of a separate on-chip 128KHz oscillator.

Figure 6-7. Watchdog Timer



The WDT gives an interrupt or a system reset when the counter reaches a given time-out value. In normal operation mode, it is required that the system uses the WDR - watchdog timer reset - instruction to restart the counter before the time-out value is reached. If the system doesn't restart the counter, an interrupt or system reset will be issued.

In interrupt mode, the WDT gives an interrupt when the timer expires. This interrupt can be used to wake the device from sleep-modes, and also as a general system timer. One example is to limit the maximum time allowed for certain operations, giving an interrupt when the operation has run longer than expected. In system reset mode, the WDT gives a reset when the timer expires. This is typically used to prevent system hang-up in case of runaway code. The third mode, interrupt and system reset mode, combines the other two modes by first giving an interrupt and then switch to system reset mode. This mode will for instance allow a safe shutdown by saving critical parameters before a system reset.

The watchdog always on (WDTON) fuse, if programmed, will force the watchdog timer to system reset mode. With the fuse programmed the system reset mode bit (WDE) and interrupt mode bit (WDIE) are locked to 1 and 0 respectively. To further ensure program security, alterations to the watchdog set-up must follow timed sequences. The sequence for clearing WDE and changing time-out configuration is as follows:

1. In the same operation, write a logic one to the watchdog change enable bit (WDCE) and WDE. A logic one must be written to WDE regardless of the previous value of the WDE bit.
2. Within the next four clock cycles, write the WDE and watchdog prescaler bits (WDP) as desired, but with the WDCE bit cleared. This must be done in one operation.

- **PCINT10/OC1AV/USCK/SCL – Port B, Bit 2**

PCINT10: pin change interrupt, source 10.

OC1AV: output compare and PWM Output A-V for timer/counter1. The PB2 pin has to be configured as an output (DDB2 set (one)) to serve this function. The OC1AV pin is also the output pin for the PWM mode timer function (c.f. OC1AV bit of TCCR1D register).

USCK: three-wire mode USI clock input.

SCL: two-wire mode USI clock input.

- **PCINT9/OC1BU/DO – Port B, Bit 1**

PCINT9: pin change interrupt, source 9.

OC1BU: output compare and PWM output B-U for timer/counter1. The PB1 pin has to be configured as an output (DDB1 set (one)) to serve this function. The OC1BU pin is also the output pin for the PWM mode timer function (c.f. OC1BU bit of TCCR1D register).

DO: three-wire mode USI data output. Three-wire mode data output overrides PORTB1 and it is driven to the port when the data direction bit DDB1 is set. PORTB1 still enables the pull-up, if the direction is input and PORTB1 is set (one).

- **PCINT8/OC1AU/DI/SDA – Port B, Bit 0**

IPCINT8: pin change interrupt, source 8.

OC1AU: output compare and PWM output A-U for timer/counter1. The PB0 pin has to be configured as an output (DDB0 set (one)) to serve this function. The OC1AU pin is also the output pin for the PWM mode timer function (c.f. OC1AU bit of TCCR1D register).

DI: three-wire mode USI data input. USI three-wire mode does not override normal port functions, so pin must be configure as an input for DI function.

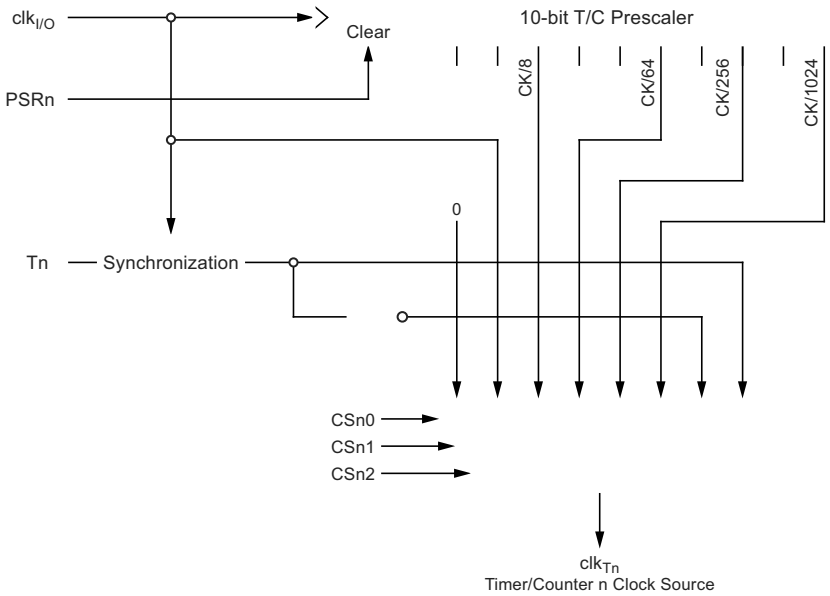
SDA: two-wire mode serial interface (USI) data input / output.

Table 9-7 and Table 9-8 on page 81 relate the alternate functions of Port B to the overriding signals shown in Figure 9-6 on page 70.

Table 9-7. Overriding Signals for Alternate Functions in PB7..PB4

Signal Name	PB7/PCINT15/ADC10/ OC1BX/RESET/dW	PB6/PCINT14/ADC9/ OC1AX/INT0	PB5/PCINT13/ADC8/ OC1BW/XTAL2/CLKO	PB4/PCINT12/ OC1AW/XTAL1/CLKI
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC1B_ENABLE **&** OC1BX	OC1A_ENABLE **&** OC1AX	OC1B_ENABLE **&** OC1BW	OC1A_ENABLE **&** OC1AW
PVOV	OC1B	OC1A	OC1B	OC1A
PTOE	0	0	0	0
DIEOE	ADC10D **|** (PCIE1 **&** PCMSK15)	ADC9D **|** INT0_ENABLE **|** (PCIE1 **&** PCMSK14)	ADC8D **|** (PCIE1 **&** PCMSK13)	(PCIE1 **&** PCMSK13)
DIEOV	PCIE1 **&** PCMSK15	INT0_ENABLE **|** (PCIE1 **&** PCMSK14)	PCIE1 **&** PCMSK13	1
DI	PCINT15	PCINT14 **-/-** INT1	PCINT13	PCINT12
AIO	RESET **-/-** ADC10 **-/-**	ADC9 **-/-** ISRC	ADC8 **-/-** XTAL2	XTAL1 **-/-** CLKI

Figure 11-2. Prescaler for Timer/Counter(1)



Note: 1. The synchronization logic on the input pin (T1) is shown in Figure 11-1 on page 101.

11.2 Timer/Counter1 Prescalers Register Description

11.2.1 General Timer/Counter Control Register – GTCCR

Bit	7	6	5	4	3	2	1	0	
	TSM	**–**	**–**	**–**	**–**	**–**	**PSR0**	**PSR1**	**GTCCR**
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TSM: Timer/Counter Synchronization Mode**

Writing the TSM bit to one activates the timer/counter synchronization mode. In this mode, the value that is written to the PSR0 and PSR1 bits is kept, hence keeping the corresponding prescaler reset signals asserted. This ensures that the corresponding timer/counters are halted and can be configured to the same value without the risk of one of them advancing during configuration. When the TSM bit is written to zero, the PSR0 and PSR1 bits are cleared by hardware, and the timer/counters start counting simultaneously.

- **Bit 0 – PSR1: Prescaler Reset Timer/Counter1**

When this bit is one, timer/counter1 prescaler will be reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set.

The following code examples show how to do an atomic write of the TCNT1 register contents. Writing any of the OCR1A/B or ICR1 registers can be done by using the same principle.

Assembly Code Example[1]

```
TIM16_WriteTCNT1:
  ; Save global interrupt flag
  in     r18,SREG
  ; Disable interrupts
  cli
  ; Set TCNT1 to r17:r16
  sts    TCNT1H,r17
  sts    TCNT1L,r16
  ; Restore global interrupt flag
  out    SREG,r18
  ret
```

C Code Example[1]

```
void TIM16_WriteTCNT1(unsigned int i)
{
  unsigned char sreg;
  unsigned int i;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  _CLI();
  /* Set TCNT1 to i */
  TCNT1 = i;
  /* Restore global interrupt flag */
  SREG = sreg;
}
```

Note: 1. The example code assumes that the part specific header file is included.

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNT1.

12.3.2 Reusing the Temporary High Byte Register

If writing to more than one 16-bit register where the high byte is the same for all registers written, then the high byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

12.4 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the Clock Select (CS12:0) bits located in the Timer/Counter control Register B (TCCR1B). For details on clock sources and prescaler, see Section 11. “Timer/Counter1 Prescaler” on page 101.

12.6.1 Input Capture Trigger Source

The main trigger source for the input capture unit is the input capture pin (ICP1). Only timer/counter1 can alternatively use the analog comparator output as trigger source for the input capture unit. The analog comparator is selected as trigger source by setting the Analog Comparator Input Capture (ACIC) bit in the analog comparator control and status register (ACSR). Be aware that changing trigger source can trigger a capture. The input capture flag must therefore be cleared after the change.

Both the input capture pin (ICP1) and the analog comparator output (ACO) inputs are sampled using the same technique as for the T1 pin (Figure 11-1 on page 101). The edge detector is also identical. However, when the noise canceler is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. Note that the input of the noise canceler and edge detector is always enabled unless the timer/counter is set in a waveform generation mode that uses ICR1 to define TOP.

An input capture can be triggered by software by controlling the port of the ICP1 pin.

12.6.2 Noise Canceler

The noise canceler improves noise immunity by using a simple digital filtering scheme. The noise canceler input is monitored over four samples, and all four must be equal for changing the output that in turn is used by the edge detector.

The noise canceler is enabled by setting the input capture noise canceler (ICNC1) bit in timer/counter control register B (TCCR1B). When enabled the noise canceler introduces additional four system clock cycles of delay from a change applied to the input, to the update of the ICR1 register. The noise canceler uses the system clock and is therefore not affected by the prescaler.

12.6.3 Using the Input Capture Unit

The main challenge when using the input capture unit is to assign enough processor capacity for handling the incoming events. The time between two events is critical. If the processor has not read the captured value in the ICR1 register before the next event occurs, the ICR1 will be overwritten with a new value. In this case the result of the capture will be incorrect.

When using the input capture interrupt, the ICR1 register should be read as early in the interrupt handler routine as possible. Even though the input capture interrupt has relatively high priority, the maximum interrupt response time is dependent on the maximum number of clock cycles it takes to handle any of the other interrupt requests.

Using the input capture unit in any mode of operation when the TOP value (resolution) is actively changed during operation, is not recommended.

Measurement of an external signal’s duty cycle requires that the trigger edge is changed after each capture. Changing the edge sensing must be done as early as possible after the ICR1 register has been read. After a change of the edge, the input capture flag (ICF1) must be cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the clearing of the ICF1 flag is not required (if an interrupt handler is used).

12.7 Output Compare Units

The 16-bit comparator continuously compares TCNT1 with the output compare register (OCR1A/B). If TCNT equals OCR1A/B the comparator signals a match. A match will set the output compare flag (OCF1A/B) at the next timer clock cycle. If enabled (OCIE1A/B = 1), the output compare flag generates an output compare interrupt. The OCF1A/B flag is automatically cleared when the interrupt is executed. Alternatively the OCF1A/B flag can be cleared by software by writing a logical one to its I/O bit locations. The waveform generator uses the match signal to generate an output according to operating mode set by the waveform generation mode (WGM13:0) bits and compare output mode (COM1A/B1:0) bits. The TOP and BOTTOM signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation (see Section 12.9 “Modes of Operation” on page 115)

Table 12-3 shows the COM1A/B1:0 bit functionality when the WGM13:0 bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 12-3. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM(1)

OC1Ai OC1Bi	COM1A1 COM1B1	COM1A0 COM1B0	Description
0	x	x	Normal port operation, OC1A/OC1B disconnected.
1	0	0	
1	0	1	WGM13=0: Normal port operation, OC1A/OC1B disconnected. WGM13=1: Toggle OC1A on compare match, OC1B reserved.
1	1	0	Clear OC1A/OC1B on compare match when up-counting. Set OC1A/OC1B on compare match when downcounting.
1	1	1	Set OC1A/OC1B on compare match when up-counting. Clear OC1A/OC1B on compare match when downcounting.

Note: 1. A special case occurs when OC1A/OC1B equals TOP and COM1A1/COM1B1 is set. See Section 12.9.4 “Phase Correct PWM Mode” on page 118 for more details.

- **Bit 3:2 – Reserved Bits**

These bits are reserved for future use.

- **Bit 1:0 – WGM11:0: Waveform Generation Mode**

Combined with the WGM13:2 bits found in the TCCR1B register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 12-4. Modes of operation supported by the timer/counter unit are: normal mode (counter), Clear timer on compare match (CTC) mode, and three types of pulse width modulation (PWM) modes (see Section 12.9 “Modes of Operation” on page 115).

Table 12-4. Waveform Generation Mode Bit Description (1)

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1A/B at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, phase correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, phase correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, phase correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, phase and frequency correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, phase and frequency correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, phase correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, phase correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

Notes: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

14. USI – Universal Serial Interface

14.1 Features

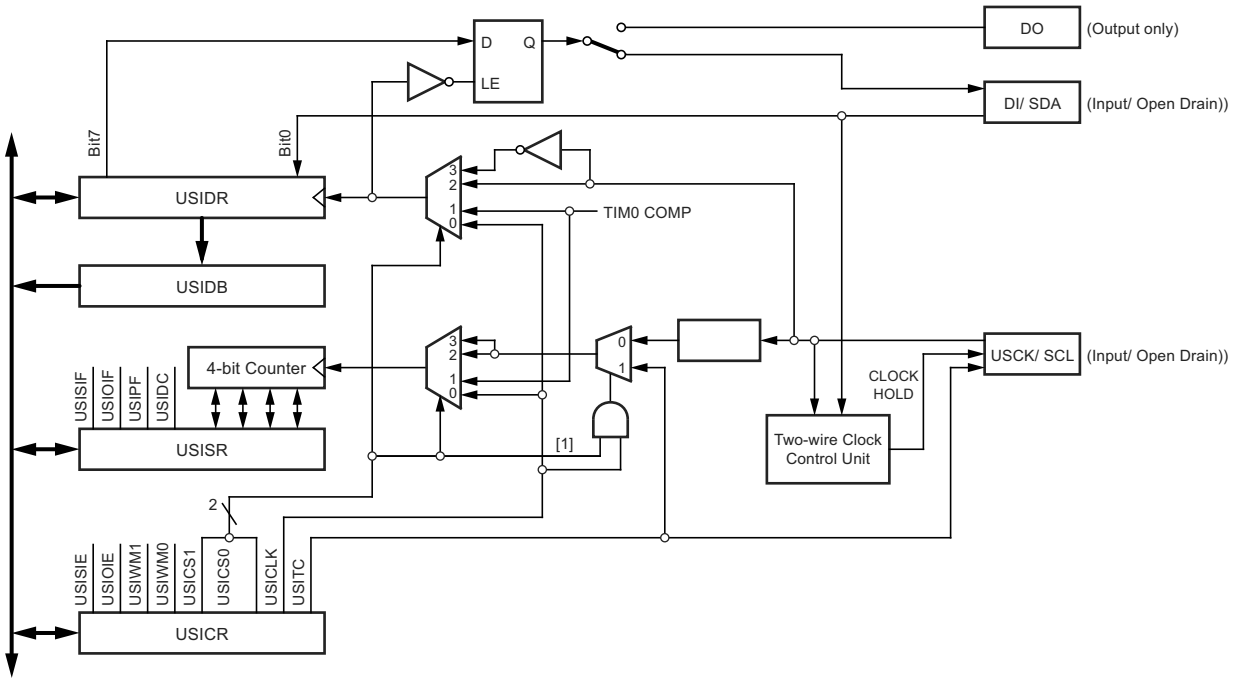
- Two-wire synchronous data transfer (master or slave)
- Three-wire synchronous data transfer (master or slave)
- Data received interrupt
- Wake up from idle mode
- In two-wire mode: Wake-up from all sleep modes, including power-down mode
- Two-wire start condition detector with interrupt capability

14.2 Overview

The universal serial interface, or USI, provides the basic hardware resources needed for serial communication. Combined with a minimum of control software, the USI allows significantly higher transfer rates and uses less code space than solutions based on software only. Interrupts are included to minimize the processor load.

A simplified block diagram of the USI is shown on Figure 14-1 for the actual placement of I/O pins, refer to Section 1.6 “Pin Configuration” on page 6. 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 Section 14.5 “Register Descriptions” on page 144.

Figure 14-1. Universal Serial Interface, Block Diagram



The 8-bit USI data register is directly accessible via the data bus and contains the incoming and outgoing data. The register has no buffering so the data must be read as quickly as possible to ensure that no data is lost. The USI data register is a serial shift register and the most significant bit that is the output of the serial shift register is connected to one of two output pins depending of the wire mode configuration.

A transparent latch is inserted between the USI data register output and output pin, which delays the change of data output to the opposite clock edge of the data input sampling. The serial input is always sampled from the data input (DI) pin independent of the configuration.

The code is size optimized using only eight instructions (+ ret). The code example assumes that the DO and USCK pins are enabled as output in the DDRA or DDRB register. The value stored in register r16 prior to the function is called is transferred to the slave device, and when the transfer is completed the data received from the slave is stored back into the r16 register.

The second and third instructions clears the USI counter overflow flag and the USI counter value. The fourth and fifth instruction set three-wire mode, positive edge shift register clock, count at USITC strobe, and toggle USCK. The loop is repeated 16 times.

The following code demonstrates how to use the USI module as a SPI master with maximum speed (fsck = fck/4):

```
SPITransfer_Fast:

    sts    USIDR,r16
    ldi    r16,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)
    ldi    r17,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)|(1<<USICLK)
    sts    USICR,r16 ; MSB
    sts    USICR,r17
    sts    USICR,r16
    sts    USICR,r17
    sts    USICR,r16
    sts    USICR,r17
    sts    USICR,r16
    sts    USICR,r17
    sts    USICR,r16
    sts    USICR,r17
    sts    USICR,r16
    sts    USICR,r17
    sts    USICR,r16
    sts    USICR,r17
    sts    USICR,r16
    sts    USICR,r17
    sts    USICR,r16 ; LSB
    sts    USICR,r17
    lds    r16,USIDR
    ret
```

14.3.3 SPI Slave Operation Example

The following code demonstrates how to use the USI module as a SPI slave:

```
init:
    ldi    r16,(1<<USIWM0)|(1<<USICS1)
    sts    USICR,r16
...
SlaveSPITransfer:
    sts    USIDR,r16
    ldi    r16,(1<<USIOIF)
    sts    USISR,r16
SlaveSPITransfer_loop:
    lds    r16, USISR
    sbrs   r16, USIOIF
    rjmp   SlaveSPITransfer_loop
    lds    r16,USIDR
    ret
```

15.4.6.3 Rx and TX Response Functions

These functions are initiated by the slave task of a LIN node. They must be used after sending an header (master task) or after receiving an header (considered as belonging to the slave task). When the TX response order is sent, the transmission begins. A Rx response order can be sent up to the reception of the last serial bit of the first byte (before the stop-bit).

In LIN 1.3, the header slot configures the LINDLR register. In LIN 2.1, the user must configure the LINDLR register, either LRXDL[3..0] for *Rx response* either LTXDL[3..0] for *Tx response*.

When the command starts, the controller checks the LIN13 bit of the LINCR register to apply the right rule for computing the checksum. Checksum calculation over the DATA bytes and the PROTECTED IDENTIFIER byte is called enhanced checksum and it is used for communication with LIN 2.1 slaves. Checksum calculation over the DATA bytes only is called classic checksum and it is used for communication with LIN 1.3 slaves. Note that identifiers 60 (0x3C) to 63 (0x3F) shall always use classic checksum.

At the end of this reception or transmission, the controller automatically returns to *Rx header / LIN abort* state (i.e. LCMD[1..0] = 00) after setting the appropriate flags.

If an LIN error occurs, the reception or the transmission is stopped, the appropriate flags are set and the LIN bus is left to recessive state.

During these functions, the controller is responsible for:

- The initialization of the checksum operator,
- The transmission or the reception of ‘n’ data with the update of the checksum calculation,
- The transmission or the checking of the CHECKSUM field,
- The checking of the frame_time_out,
- The checking of the LIN communication integrity.

While the controller is sending or receiving a response, BREAK and SYNCH fields can be detected and the identifier of this new header will be recorded. Of course, specific errors on the previous response will be maintained with this identifier reception.

15.4.6.4 Handling Data of LIN response

A FIFO data buffer is used for data of the LIN response. After setting all parameters in the LINSEL register, repeated accesses to the LINDAT register perform data read or data write (c.f. Section 15.5.15 “Data Management” on page 164).

Note that LRXDL[3..0] and LTXDL[3..0] are not linked to the data access.

15.4.7 UART Commands

Setting the LCMD[2] bit in LINENR register enables UART commands.

Tx byte and Rx byte services are independent as shown in Table 15-1 on page 153.

- Byte transfer: the UART is selected but both Rx and Tx services are disabled,
- Rx byte: only the Rx service is enable but Tx service is disabled,
- Tx byte: only the Tx service is enable but Rx service is disabled,
- Full duplex: the UART is selected and both Rx and Tx services are enabled.

This combination of services is controlled by the LCMD[1..0] bits of LINENR register (c.f. Figure 15-5 on page 153).

15.4.7.1 Data Handling

The FIFO used for LIN communication is disabled during UART accesses. LRXDL[3..0] and LTXDL[3..0] values of LINDLR register are then irrelevant. LINDAT register is then used as data register and LINSEL register is not relevant.

15.6.2 LIN Status and Interrupt Register - LINSIR

Bit	7	6	5	4	3	2	1	0	
	LIDST2	LIDST1	LIDST0	LBUSY	LERR	LIDOK	LTXOK	LRXOK	LINSIR
Read/Write	R	R	R	R	R/Wone	R/Wone	R/Wone	R/Wone	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:5 - LIDST[2:0]: Identifier Status**
  - 0xx = no specific identifier,
  - 100 = identifier 60 (0x3C),
  - 101 = identifier 61 (0x3D),
  - 110 = identifier 62 (0x3E),
  - 111 = identifier 63 (0x3F).

- **Bit 4 - LBUSY: Busy Signal**
  - 0 = not busy,
  - 1 = busy (receiving or transmitting).

- **Bit 3 - LERR: Error Interrupt**

It is a logical OR of LINERR register bits. This bit generates an interrupt if its respective enable bit - LENERR - is set in LINENIR.

- 0 = no error,
- 1 = an error has occurred.

The user clears this bit by writing 1 in order to reset this interrupt. Resetting LERR also resets all LINERR bits. In UART mode, this bit is also cleared by reading LINDAT.

- **Bit 2 - LIDOK: Identifier Interrupt**

This bit generates an interrupt if its respective enable bit - LENIDOK - is set in LINENIR.

- 0 = no identifier,
- 1 = slave task: Identifier present, master task: Tx header complete.

  The user clears this bit by writing 1, in order to reset this interrupt.

- **Bit 1 - LTXOK: Transmit Performed Interrupt**

This bit generates an interrupt if its respective enable bit - LENTXOK - is set in LINENIR.

- 0 = no Tx,
- 1 = Tx response complete.

The user clears this bit by writing 1, in order to reset this interrupt.
In UART mode, this bit is also cleared by writing LINDAT.

- **Bit 0 - LRXOK: Receive Performed Interrupt**

This bit generates an interrupt if its respective enable bit - LENRXOK - is set in LINENIR.

- 0 = no Rx
- 1 = Rx response complete.

The user clears this bit by writing 1, in order to reset this interrupt.
In UART mode, this bit is also cleared by reading LINDAT.

Figure 17-6. ADC Timing Diagram, Auto Triggered Conversion

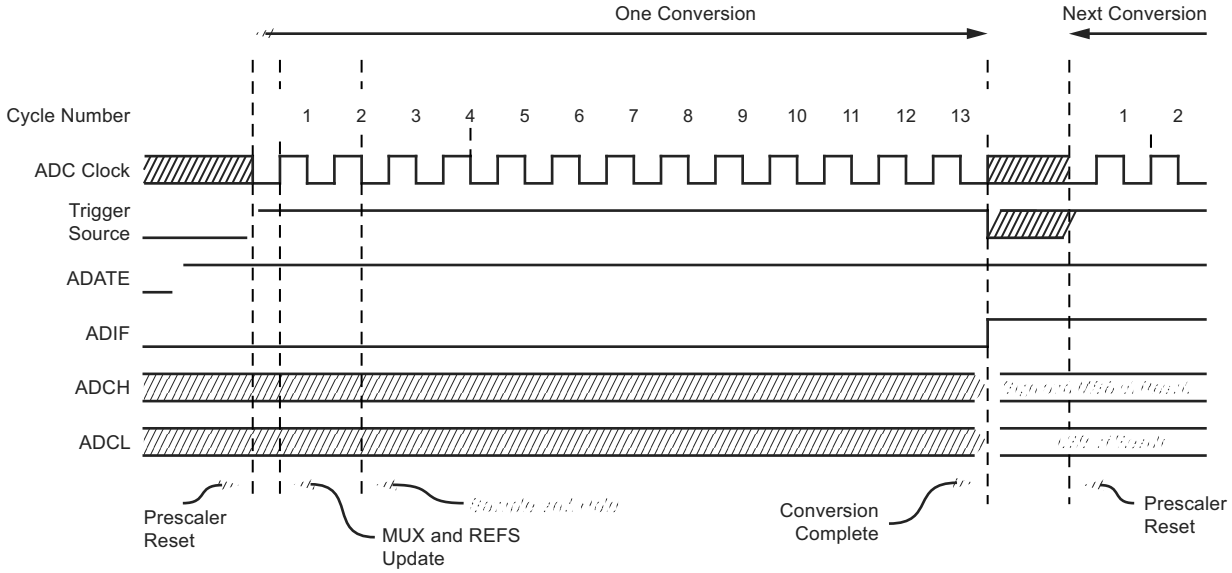


Figure 17-7. ADC Timing Diagram, Free Running Conversion

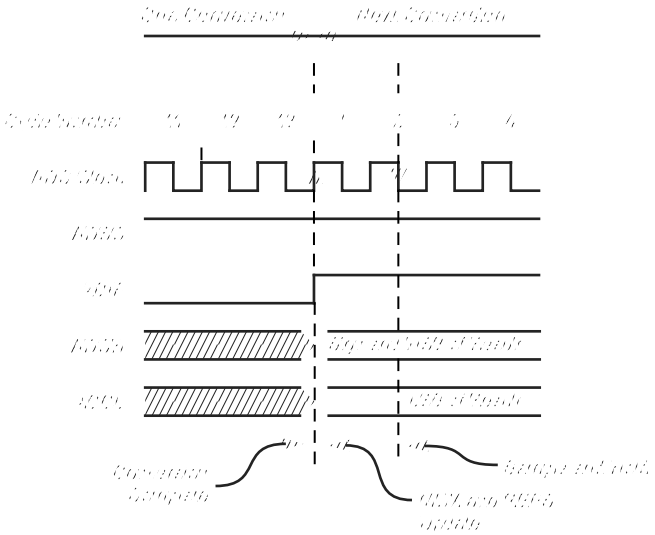


Table 17-1. ADC Conversion Time

Condition	Sample and Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5 cycles	25 cycles
Normal conversions	1.5 cycles	13 cycles
Auto Triggered conversions	2 cycles	13.5 cycles

17.11.3 ADCL and ADCH – The ADC Data Register

17.11.3.1 ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

17.11.3.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers.

When ADCL is read, the ADC data register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

The ADLAR bit in ADMUX, and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

- **ADC9:0: ADC Conversion Result**

These bits represent the result from the conversion, as detailed in Section 17.8 “ADC Conversion Result” on page 186.

17.11.4 ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
	BIN	ACME	ACIR1	ACIR0	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7– BIN: Bipolar Input Mode**

The gain stage is working in the unipolar mode as default, but the bipolar mode can be selected by writing the BIN bit in the ADCSRB register. In the unipolar mode only one-sided conversions are supported and the voltage on the positive input must always be larger than the voltage on the negative input. Otherwise the result is saturated to the voltage reference. In the bipolar mode two-sided conversions are supported and the result is represented in the two’s complement form. In the unipolar mode the resolution is 10 bits and the bipolar mode the resolution is 9 bits + 1 sign bit.

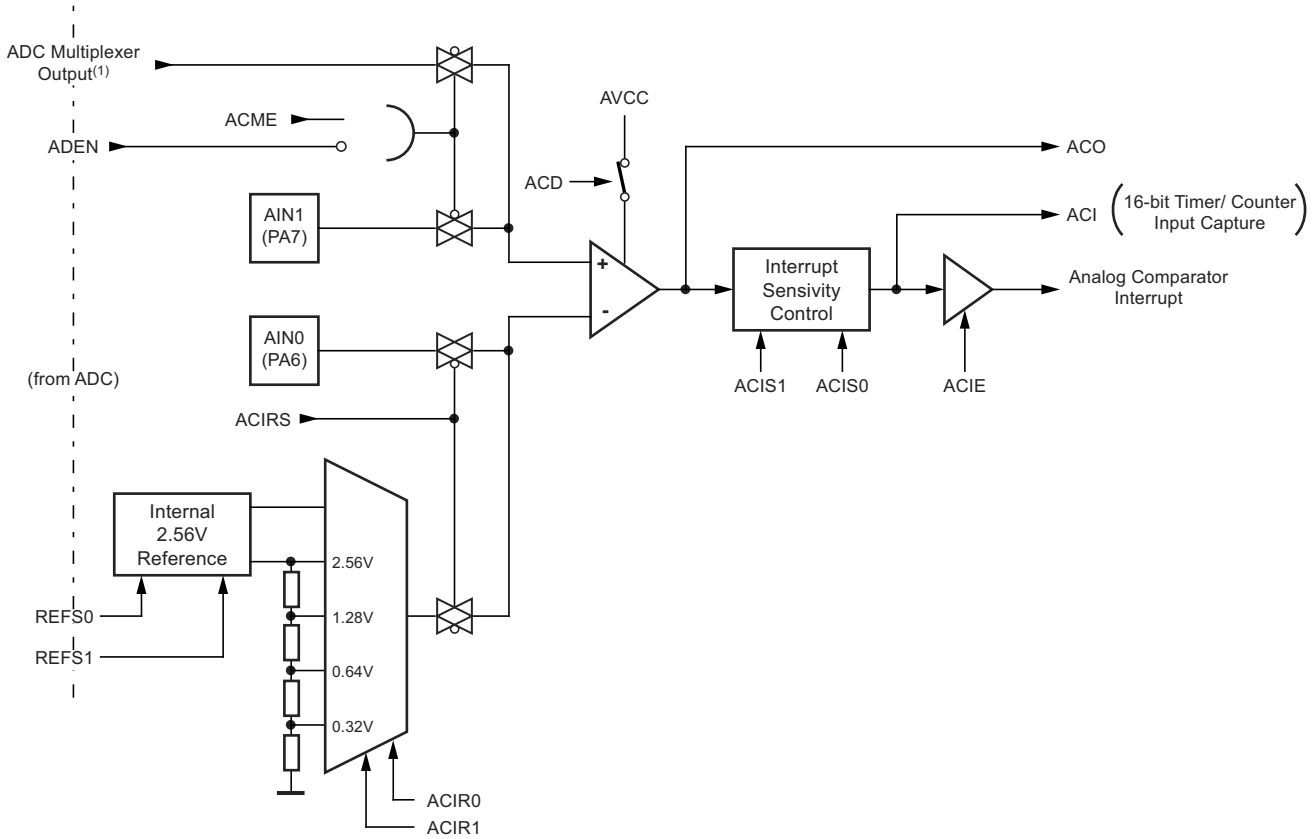
- **Bit 3 – Res: Reserved Bit**

This bit is reserved for future use. For compatibility with future devices it must be written to zero when ADCSRB register is written.

18. AnaComp - Analog Comparator

The analog comparator compares the input values on the positive pin (AIN1) and negative pin (AIN0). When the voltage on the positive pin is higher than the voltage on the negative pin, the analog comparator output, ACO, is set. The comparator can trigger a separate interrupt, exclusive to the analog comparator. The user can select interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 18-1.

Figure 18-1. Analog Comparator Block Diagram(1)(2)



Notes: 1. See Table 18-2 on page 196 and Table 18-3 on page 197
2. Refer to Figure 1-2 on page 6 and Table 9-3 on page 73 for analog comparator pin placement.

18.1 Register Description

18.1.1 ADC Control and Status Register B – ADCSRB

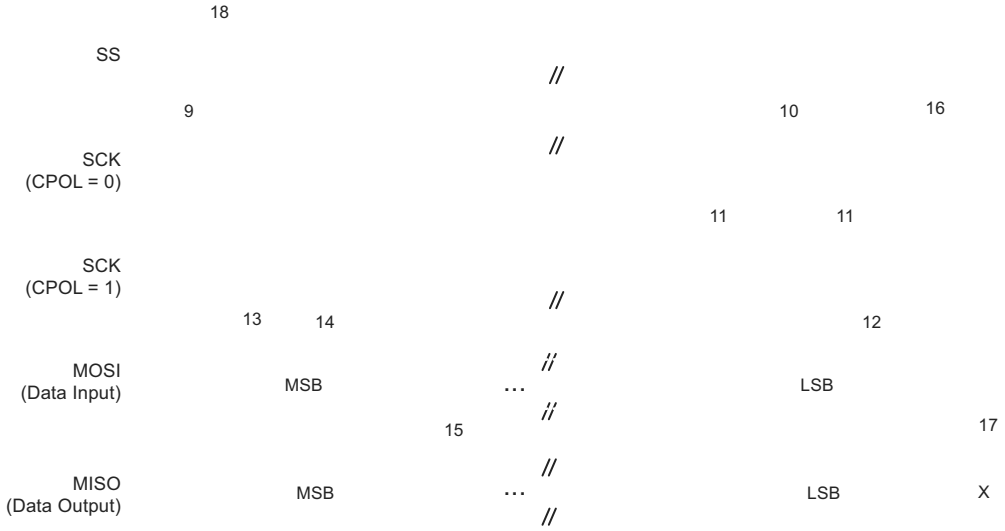
Bit	7	6	5	4	3	2	1	0	
	BIN	**ACME**	**ACIR1**	**ACIR0**	**—**	**ADTS2**	**ADTS1**	**ADTS0**	**ADCSRB**
Read/Write	R	R/W	R/W	R/W	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6 – ACME: Analog Comparator Multiplexer Enable**

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer selects the positive input to the analog comparator. When this bit is written logic zero, AIN1 is applied to the positive input of the analog comparator.

When the analog to digital converter (ADC) is configured as single ended input channel, it is possible to select any of the ADC[10..0] pins to replace the positive input to the analog comparator. The ADC multiplexer (MUX[4..0]) is used to select this input, and consequently, the ADC must be switched off to utilize this feature.

Figure 22-7. SPI Interface Timing Requirements (Slave Mode)



25. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x93)	Reserved									
(0x92)	Reserved									
(0x91)	Reserved									
(0x90)	Reserved									
(0x8F)	Reserved									
(0x8E)	Reserved									
(0x8D)	Reserved									
(0x8C)	Reserved									
(0x8B)	OCR1BH	OCR1B15	OCR1B14	OCR1B13	OCR1B12	OCR1B11	OCR1B10	OCR1B9	OCR1B8	128
(0x8A)	OCR1BL	OCR1B7	OCR1B6	OCR1B5	OCR1B4	OCR1B3	OCR1B2	OCR1B1	OCR1B0	128
(0x89)	OCR1AH	OCR1A15	OCR1A14	OCR1A13	OCR1A12	OCR1A11	OCR1A10	OCR1A9	OCR1A8	128
(0x88)	OCR1AL	OCR1A7	OCR1A6	OCR1A5	OCR1A4	OCR1A3	OCR1A2	OCR1A1	OCR1A0	128
(0x87)	ICR1H	ICR115	ICR114	ICR113	ICR112	ICR111	ICR110	ICR19	ICR18	128
(0x86)	ICR1L	ICR17	ICR16	ICR15	ICR14	ICR13	ICR12	ICR11	ICR10	128
(0x85)	TCNT1H	TCNT115	TCNT114	TCNT113	TCNT112	TCNT111	TCNT110	TCNT19	TCNT18	127
(0x84)	TCNT1L	TCNT17	TCNT16	TCNT15	TCNT14	TCNT13	TCNT12	TCNT11	TCNT10	127
(0x83)	TCCR1D	OC1BX	OC1BW	OC1BV	OC1BU	OC1AX	OC1AW	OC1AV	OC1AU	127
(0x82)	TCCR1C	FOC1A	FOC1B	–	–	–	–	–	–	127
(0x81)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	126
(0x80)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	124
(0x7F)	DIDR1	–	ADC10D	ADC9D	ADC8D	–	–	–	–	192
(0x7E)	DIDR0	ADC7D/AIN1D	ADC6D/AIN0D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	192, 196
(0x7D)	Reserved									
(0x7C)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	188
(0x7B)	ADCSRB	BIN	ACME	ACIR1	ACIR0	–	ADTS2	ADTS1	ADTS0	191, 194
(0x7A)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	190
(0x79)	ADCH	- / ADC9	- / ADC8	- / ADC7	- / ADC6	- / ADC5	- / ADC4	ADC9 / ADC3	ADC8 / ADC2	191
(0x78)	ADCL	ADC7 / ADC1	ADC6 / ADC0	ADC5 / -	ADC4 / -	ADC3 / -	ADC2 / -	ADC1 / -	ADC0 /	191
(0x77)	AMISCR	–	–	–	–	–	AREFEN	XREFEN	ISRCEN	175, 175
(0x76)	Reserved									
(0x75)	Reserved									
(0x74)	Reserved									
(0x73)	Reserved									
(0x72)	Reserved									

Notes:
1. Address bits exceeding EEAMSB (Table 21-8 on page 210) are don't care.
2. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
3. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
4. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
5. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The Atmel® ATtiny87/167 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in 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.

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**