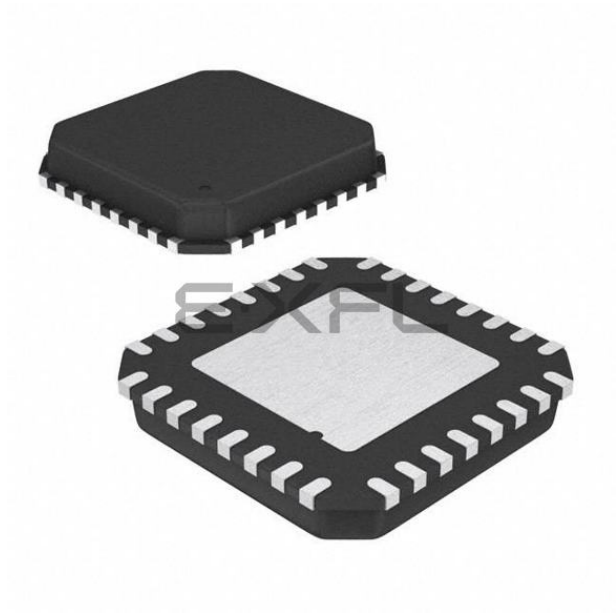


Welcome to E-XFL.COM

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	Active
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	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 11x10b
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/attiny167-mu

1. Description

1.1 Comparison Between ATtiny87 and ATtiny167

ATtiny87 and ATtiny167 are hardware and software compatible. They differ only in memory sizes as shown in Table 1-1.

Table 1-1. Memory Size Summary

Device	Flash	EEPROM	SRAM	Interrupt Vector size
ATtiny167	16K Bytes	512 Bytes	512 Bytes	2-instruction-words / vector
ATtiny87	8K Bytes	512 Bytes	512 Bytes	2-instruction-words / vector

1.2 Part Description

The ATtiny87/167 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATtiny87/167 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATtiny87/167 provides the following features: 8K/16K byte of In-System Programmable Flash, 512 bytes EEPROM, 512 bytes SRAM, 16 general purpose I/O lines, 32 general purpose working registers, one 8-bit Timer/Counter with compare modes, one 8-bit high speed Timer/Counter, Universal Serial Interface, a LIN controller, Internal and External Interrupts, a 11-channel, 10-bit ADC, a programmable Watchdog Timer with internal Oscillator, and three software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counter, ADC, Analog Comparator, and Interrupt system to continue functioning. The Power-down mode saves the register contents, disabling all chip functions until the next Interrupt or Hardware Reset. The ADC Noise Reduction mode stops the CPU and all I/O modules except ADC, to minimize switching noise during ADC conversions.

The device is manufactured using Atmel's high density non-volatile memory technology. The On-chip ISP Flash allows the Program memory to be re-programmed In-System through an SPI serial interface, by a conventional non-volatile memory programmer or by an On-chip boot code running on the AVR core. The Boot program can use any interface to download the application program in the Flash memory. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATtiny87/167 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATtiny87/167 AVR is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

1.6 Resources

A comprehensive set of development tools, application notes and datasheets are available for download on http://www.atmel.com/avr.

1.7 About Code Examples

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

1.8 Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

1.9 Disclaimer

Typical values contained in this datasheet 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 has been characterized.

3.3.1 EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space.

The write access times for the EEPROM are given in Table 3-2. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on Power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See “Preventing EEPROM Corruption” on page 20 for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to “Atomic Byte Programming” on page 18 and “Split Byte Programming” on page 18 for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

3.3.2 Atomic Byte Programming

Using Atomic Byte Programming is the simplest mode. When writing a byte to the EEPROM, the user must write the address into the EEARL Register and data into EEDR Register. If the EEPMn bits are zero, writing EEPE (within four cycles after EEMPE is written) will trigger the erase/write operation. Both the erase and write cycle are done in one operation and the total programming time is given in Table 1. The EEPE bit remains set until the erase and write operations are completed. While the device is busy with programming, it is not possible to do any other EEPROM operations.

3.3.3 Split Byte Programming

It is possible to split the erase and write cycle in two different operations. This may be useful if the system requires short access time for some limited period of time (typically if the power supply voltage falls). In order to take advantage of this method, it is required that the locations to be written have been erased before the write operation. But since the erase and write operations are split, it is possible to do the erase operations when the system allows doing time-critical operations (typically after Power-up).

3.3.4 Erase

To erase a byte, the address must be written to EEAR. If the EEPMn bits are 0b01, writing the EEPE (within four cycles after EEMPE is written) will trigger the erase operation only (programming time is given in Table 1). The EEPE bit remains set until the erase operation completes. While the device is busy programming, it is not possible to do any other EEPROM operations.

3.3.5 Write

To write a location, the user must write the address into EEAR and the data into EEDR. If the EEPMn bits are 0b10, writing the EEPE (within four cycles after EEMPE is written) will trigger the write operation only (programming time is given in Table 1). The EEPE bit remains set until the write operation completes. If the location to be written has not been erased before write, the data that is stored must be considered as lost. While the device is busy with programming, it is not possible to do any other EEPROM operations.

The calibrated Oscillator is used to time the EEPROM accesses. Make sure the Oscillator frequency is within the requirements described in “OSCCAL – Oscillator Calibration Register” on page 37.

The following code examples show one assembly and one C function for erase, write, or atomic write of the EEPROM. The examples assume that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts will occur during execution of these functions.

Assembly Code Example

```
EEPROM_write:
  ; Wait for completion of previous write
  sbic   EECR,EEPE
  rjmp   EEPROM_write
  ; Set Programming mode
  ldi    r16, (0<<EEPM1)|(0<<EEPM0)
  out    EECR, r16
  ; Set up address (r18:r17) in address register
  out    EEARH, r18
  out    EEARL, r17
  ; Write data (r16) to data register
  out    EEDR, r16
  ; Write logical one to EEMPE
  sbi    EECR,EEMPE
  ; Start eeprom write by setting EEPE
  sbi    EECR,EEPE
  ret
```

C Code Example

```
void EEPROM_write(unsigned char ucAddress, unsigned char ucData)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEPE))
    ;
  /* Set Programming mode */
  EECR = (0<<EEPM1)|(0<<EEPM0);
  /* Set up address and data registers */
  EEAR = ucAddress;
  EEDR = ucData;
  /* Write logical one to EEMPE */
  EECR |= (1<<EEMPE);
  /* Start eeprom write by setting EEPE */
  EECR |= (1<<EEPE);
}
```

4. System Clock and Clock Options

The ATtiny87/167 provides a large number of clock sources. They can be divided into two categories: internal and external. Some external clock sources can be shared with the asynchronous timer. After reset, the clock source is determined by the CKSEL Fuses. Once the device is running, software clock switching is possible to any other clock sources.

Hardware controls are provided for clock switching management but some specific procedures must be observed. Clock switching should be performed with caution as some settings could result in the device having an incorrect configuration.

4.1 Clock Systems and their Distribution

Figure 4-1 presents the principal clock systems in the AVR and their distribution. All of the clocks may not need to be active at any given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes or by using features of the dynamic clock switch circuit (See “Power Management and Sleep Modes” on page 42 and “Dynamic Clock Switch” on page 31). The clock systems are detailed below.

Figure 4-1. Clock Distribution

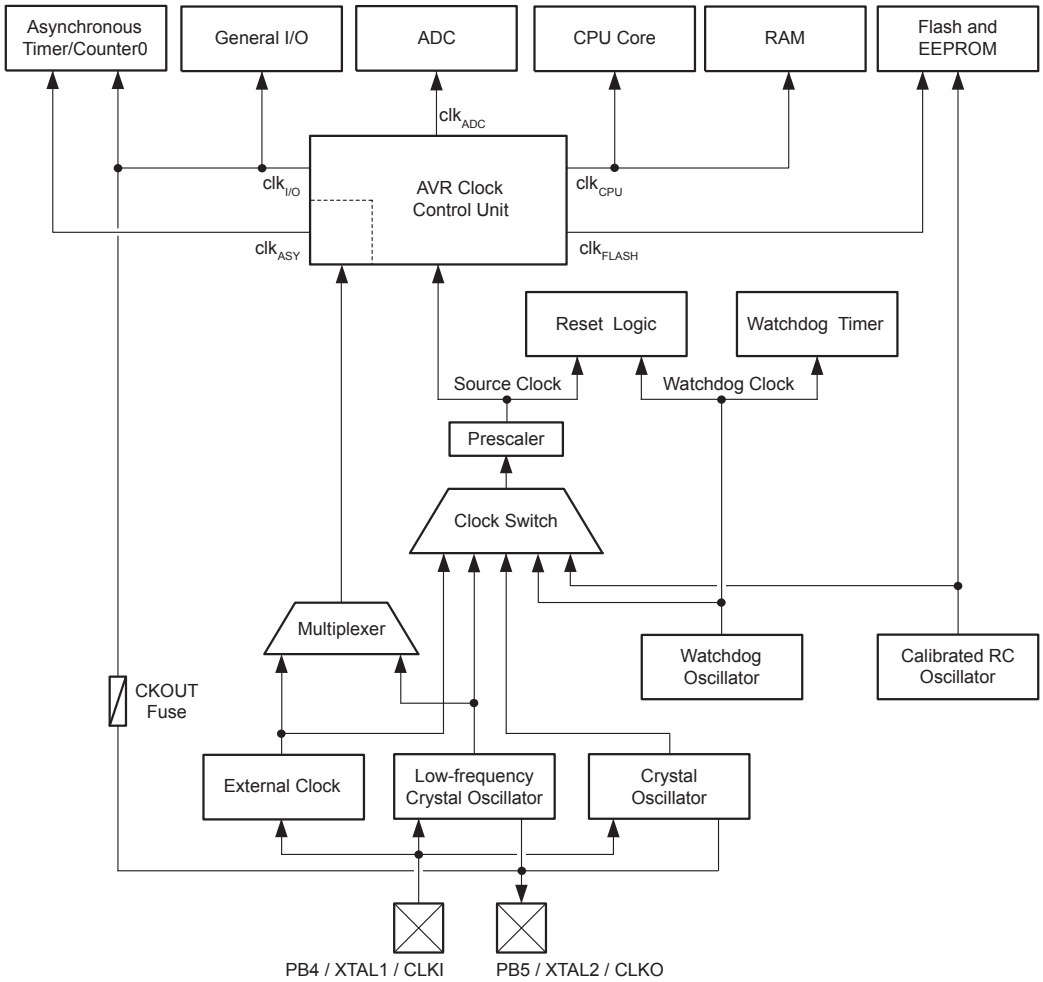
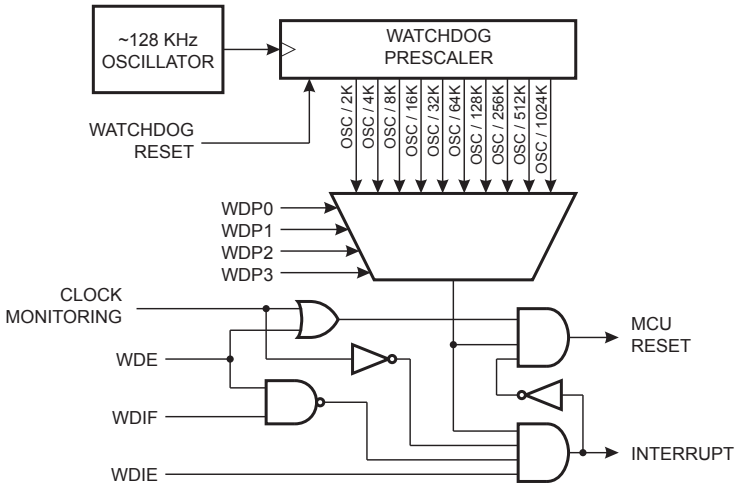


Figure 6-7. Watchdog Timer



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.

The following code example shows one assembly and one C function for changing the time-out value of the Watchdog Timer.

Assembly Code Example[1]

```
WDT_Prescaler_Change:
  ; Turn off global interrupt
  cli
  ; Reset Watchdog Timer
  wdr
  ; Start timed sequence
  lds r16, WDTCR
  ori   r16, (1<<WDCE) | (1<<WDE)
  sts WDTCR, r16
  ; -- Got four cycles to set the new values from here -
  ; Set new prescaler(time-out) value = 64K cycles (~0.5 s)
  ldi   r16, (1<<WDE) | (1<<WDP2) | (1<<WDP0)
  sts WDTCR, r16
  ; -- Finished setting new values, used 2 cycles -
  ; Turn on global interrupt
  sei
  ret
```

C Code Example[1]

```
void WDT_Prescaler_Change(void)
{
  __disable_interrupt();
  __watchdog_reset();
  /* Start timed sequence */
  WDTCR |= (1<<WDCE) | (1<<WDE);
  /* Set new prescaler(time-out) value = 64K cycles (~0.5 s) */
  WDTCR  = (1<<WDE) | (1<<WDP2) | (1<<WDP0);
  __enable_interrupt();
}
```

Notes: 1. See "About Code Examples" on page 6.

2. The Watchdog Timer should be reset before any change of the WDP bits, since a change in the WDP bits can result in a time-out when switching to a shorter time-out period.

6.3.2 Clock monitoring

The Watchdog Timer can be used to detect a loss of system clock. This configuration is driven by the dynamic clock switch circuit. Please refer to Section 4.3.8 "Clock Monitoring" on page 34 for more information.

The alternate pin configuration is as follows:

- **Port A, Bit 7 – PCINT7/ADC7/AIN1/XREF/AREF**
  - PCINT7: Pin Change Interrupt, source 7.
  - ADC7: Analog to Digital Converter, channel 7.
  - AIN1: Analog Comparator Positive Input. This pin is directly connected to the positive input of the Analog Comparator.
  - XREF: Internal Voltage Reference Output. The internal voltage reference 2.56V or 1.1V is output when XREFEN is set and if either 2.56V or 1.1V is used as reference for ADC conversion. When XREF output is enabled, the pin port pull-up and digital output driver are turned off.
  - AREF: External Voltage Reference Input for ADC. The pin port pull-up and digital output driver are disabled when the pin is used as an external voltage reference input for ADC or as when the pin is only used to connect a bypass capacitor for the voltage reference of the ADC.

- **Port A, Bit 6 – PCINT6/ADC6/AIN0/$\overline{SS}$**
  - PCINT6: Pin Change Interrupt, source 6.
  - ADC6: Analog to Digital Converter, channel 6.
  - AIN0: Analog Comparator Negative Input. This pin is directly connected to the negative input of the Analog Comparator.
  - $\overline{SS}$: SPI Slave Select Input. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDA6. As a slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a master, the data direction of this pin is controlled by DDA6. When the pin is forced to be an input, the pull-up can still be controlled by the PORTA6 bit.

- **Port A, Bit 5 – PCINT5/ADC5/T1/USCK/SCL/SCK**
  - PCINT5: Pin Change Interrupt, source 5.
  - ADC5: Analog to Digital Converter, channel 5.
  - T1: Timer/Counter1 Clock Input.
  - USCK: Three-wire Mode USI Clock Input.
  - SCL: Two-wire Mode USI Clock Input.
  - SCK: SPI Master Clock output, Slave Clock input pin. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDA5. When the SPI is enabled as a master, the data direction of this pin is controlled by DDA5. When the pin is forced to be an input, the pull-up can still be controlled by the PORTA5 bit.

- **Port A, Bit 4 – PCINT4/ADC4/ICP1/DI/SDA/MOSI**
  - PCINT4: Pin Change Interrupt, source 4.
  - ADC4: Analog to Digital Converter, channel 4.
  - ICP1: Timer/Counter1 Input Capture Trigger. The PA3 pin can act as an Input Capture pin for Timer/Counter1.
  - 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.

- **Bits 1:0 – WGM0[1:0]: Waveform Generation Mode**

These bits control the counting sequence of the counter, the source for the maximum (TOP) counter value, and what type of waveform generation to be used, see Table 10-4. Modes of operation supported by the Timer/Counter unit are: Normal mode (Counter), Clear Timer on Compare match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes (See "Modes of Operation" on page 91.).

Table 10-4. Waveform Generation Mode Bit Description

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0A at	TOV0 Flag Set on(1)(2)
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0A	Immediate	MAX
3	1	1	Fast PWM	0xFF	TOP	MAX

Notes: 1. MAX = 0xFF,
2. BOTTOM = 0x00.

10.11.2 TCCR0B – Timer/Counter0 Control Register B

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	FOC0A	–	–	–	–	CS02	CS01	CS00	TCCR0B
Read/Write	W	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC0A: Force Output Compare A**

The FOC0A bit is only active when the WGM bits specify a non-PWM mode.

However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0B is written when operating in PWM mode. When writing a logical one to the FOC0A bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC0A output is changed according to its COM0A[1:0] bits setting. Note that the FOC0A bit is implemented as a strobe. Therefore it is the value present in the COM0A[1:0] bits that determines the effect of the forced compare.

A FOC0A strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0A as TOP.

The FOC0A bit is always read as zero.

- **Bits 6:3 – Res: Reserved Bits**

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

- **Bits 2:0 – CS0[2:0]: Clock Select**

Register. When writing the ICR1 Register the high byte must be written to the ICR1H I/O location before the low byte is written to ICR1L.

For more information on how to access the 16-bit registers refer to “Accessing 16-bit Registers” on page 111.

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 106). 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.

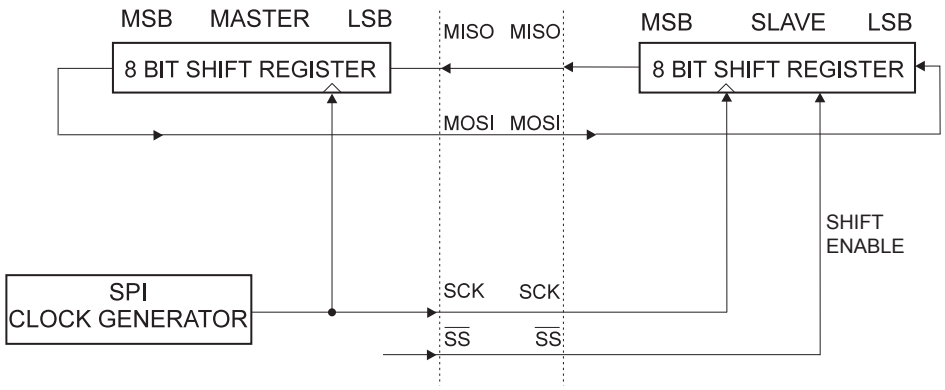
Measurement of an external signal’s duty cycle requires that the trigger edge is changed after each capture. Changing the edge sensing must be done as early as possible after the ICR1 Register has been read. After a change of the edge, the Input Capture Flag (ICF1) must be cleared by software (writing 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).

The interconnection between Master and Slave CPUs with SPI is shown in Figure 13-2. The system consists of two shift Registers, and a Master clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select $\overline{SS}$ pin of the desired Slave. Master and Slave prepare the data to be sent in their respective shift Registers, and the Master generates the required clock pulses on the SCK line to interchange data. Data is always shifted from Master to Slave on the Master Out – Slave In, MOSI, line, and from Slave to Master on the Master In – Slave Out, MISO, line. After each data packet, the Master will synchronize the Slave by pulling high the Slave Select, $\overline{SS}$, line.

When configured as a Master, the SPI interface has no automatic control of the $\overline{SS}$ line. This must be handled by user software before communication can start. When this is done, writing a byte to the SPI Data Register starts the SPI clock generator, and the hardware shifts the eight bits into the Slave. After shifting one byte, the SPI clock generator stops, setting the end of transmission flag (SPIF). If the SPI Interrupt Enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Master may continue to shift the next byte by writing it into SPDR, or signal the end of packet by pulling high the Slave Select, $\overline{SS}$ line. The last incoming byte will be kept in the Buffer Register for later use.

When configured as a Slave, the SPI interface will remain sleeping with MISO tri-stated as long as the $\overline{SS}$ pin is driven high. In this state, software may update the contents of the SPI Data Register, SPDR, but the data will not be shifted out by incoming clock pulses on the SCK pin until the $\overline{SS}$ pin is driven low. As one byte has been completely shifted, the end of transmission flag, SPIF is set. If the SPI Interrupt Enable bit, SPIE, in the SPCR Register is set, an interrupt is requested. The Slave may continue to place new data to be sent into SPDR before reading the incoming data. The last incoming byte will be kept in the Buffer Register for later use.

Figure 13-2. SPI Master-slave Interconnection



The system is single buffered in the transmit direction and double buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first byte is lost.

In SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the frequency of the SPI clock should never exceed $f_{clkio}/4$.

Table 14-2. Relations between the USICS[1:0] and USICLK Setting

USICS1	USICS0	USICLK	USI Data Register Clock Source	4-bit Counter Clock Source
0	0	0	No Clock	No Clock
0	0	1	Software clock strobe (USICLK)	Software clock strobe (USICLK)
0	1	X	Timer/Counter0 Compare Match	Timer/Counter0 Compare Match
1	0	0	External, positive edge	External, both edges
1	1	0	External, negative edge	External, both edges
1	0	1	External, positive edge	Software clock strobe (USITC)
1	1	1	External, negative edge	Software clock strobe (USITC)

- **Bit 1 – USICLK: Clock Strobe**

Writing a one to this bit location strobes the USI Data Register to shift one step and the counter to increment by one, provided that the USICS[1:0] bits are set to zero and by doing so the software clock strobe option is selected. The output will change immediately when the clock strobe is executed, i.e., in the same instruction cycle. The value shifted into the USI Data Register is sampled the previous instruction cycle. The bit will be read as zero.

When an external clock source is selected (USICS1 = 1), the USICLK function is changed from a clock strobe to a Clock Select Register. Setting the USICLK bit in this case will select the USITC strobe bit as clock source for the 4-bit counter (see Table 14-2).

- **Bit 0 – USITC: Toggle Clock Port Pin**

Writing a one to this bit location toggles the USCK/SCL value either from 0 to 1, or from 1 to 0. The toggling is independent of the setting in the Data Direction Register, but if the PORT value is to be shown on the pin the DDB2 must be set as output (to one). This feature allows easy clock generation when implementing master devices. The bit will be read as zero.

When an external clock source is selected (USICS1 = 1) and the USICLK bit is set to one, writing to the USITC strobe bit will directly clock the 4-bit counter. This allows an early detection of when the transfer is done when operating as a master device.

Table 15-1. LIN/UART Command List

LENA	LCMD[2]	LCMD[1]	LCMD[0]	Command	Comment
0	x	x	x	Disable peripheral	
1	0	0	0	Rx Header - LIN Abort	LIN Withdrawal
			1	Tx Header	LCMD[2:0]=000 after Tx
		1	0	Rx Response	LCMD[2:0]=000 after Rx
			1	Tx Response	LCMD[2:0]=000 after Tx
	1	0	0	Byte transfer	no CRC, no Time out LTXDL=LRXDL=0 (LINDLR: read only register)
		1	0	Rx Byte	
		0	1	Tx Byte	
		1	1	Full duplex	

15.4.5 Enable / Disable

Setting the LENA bit in LINCR register enables the LIN/UART controller. To disable the LIN/UART controller, LENA bit must be written to 0. No wait states are implemented, so, the disable command is taken into account immediately.

15.4.6 LIN Commands

Clearing the LCMD[2] bit in LINCR register enables LIN commands.

As shown in Table 15-1 on page 166, four functions controlled by the LCMD[1:0] bits of LINCR register are available (c.f. Figure 15-5 on page 165).

15.4.6.1 Rx Header / LIN Abort Function

This function (or state) is mainly the withdrawal mode of the controller.

When the controller has to execute a master task, this state is the start point before enabling a Tx Header command.

When the controller has only to execute slave tasks, LIN header detection/acquisition is enabled as background function. At the end of such an acquisition (Rx Header function), automatically the appropriate flags are set, and in LIN 1.3, the LINDLR register is set with the uncoded length value.

This state is also the start point before enabling the Tx or the Rx Response command.

A running function (i.e. Tx Header, Tx or Rx Response) can be aborted by clearing LCMD[1:0] bits in LINCR register (See ”Break-in-data” on page 176.). In this case, an abort flag - LABORT - in LINERR register will be set to inform the other software tasks. No wait states are implemented, so, the abort command is taken into account immediately.

Rx Header function is responsible for:

- The BREAK field detection,
- The hardware re-synchronization analyzing the SYNCH field,
- The reception of the PROTECTED IDENTIFIER field, the parity control and the update of the LINDLR register in case of LIN 1.3,
- The starting of the Frame_Time_Out,
- The checking of the LIN communication integrity.

- **Bit 3 – $\overline{\text{LAINC}}$: $\overline{\text{Auto Increment}}$ of Data Buffer Index**

  In LIN mode:

  – 0 = Auto incrementation of FIFO data buffer index (default),

  – 1 = No auto incrementation.

  In UART mode this field is unused.

- **Bits 2:0 – LINDX[2:0]: FIFO LIN Data Buffer Index**

  In LIN mode: location (index) of the LIN response data byte into the FIFO data buffer. The FIFO data buffer is accessed through LINDAT.

  In UART mode this field is unused.

15.6.10 LINDAT – LIN Data Register

Bit	7	6	5	4	3	2	1	0	
(0xD2)	LDATA7	LDATA6	LDATA5	LDATA4	LDATA3	LDATA2	LDATA1	LDATA0	LINDAT
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:0 – LDATA[7:0]: LIN Data In / Data out**

  In LIN mode: FIFO data buffer port.

  In UART mode: data register (no data buffer - no FIFO).

  – In Write access, data out.

  – In Read access, data in.

Figure 17-4. ADC Timing Diagram, First Conversion (Single Conversion Mode)

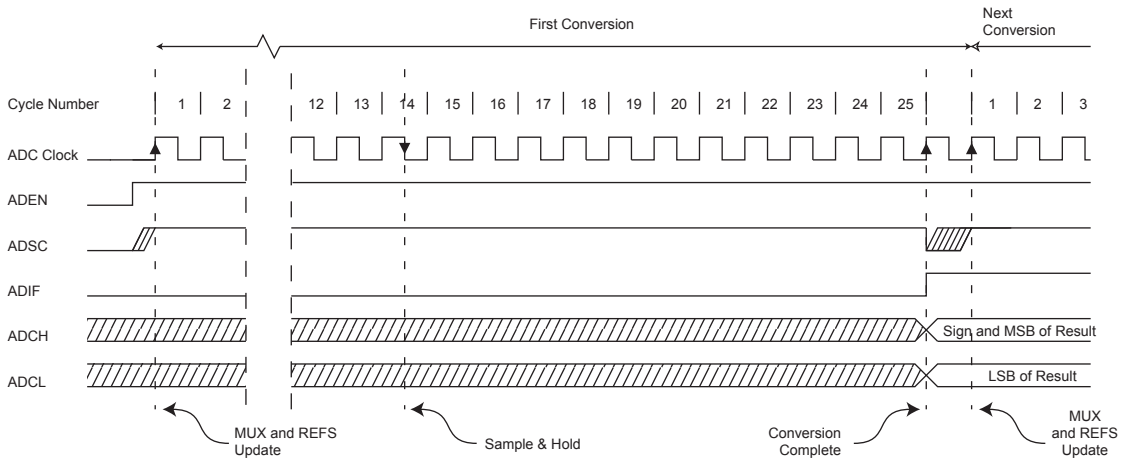


Figure 17-5. ADC Timing Diagram, Single Conversion

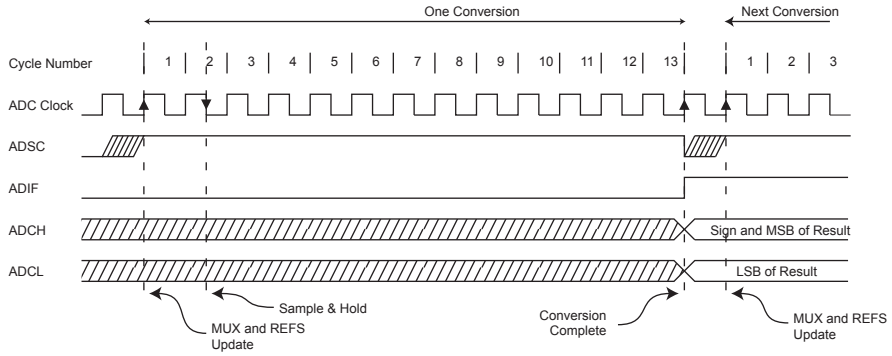


Figure 17-6. ADC Timing Diagram, Auto Triggered Conversion

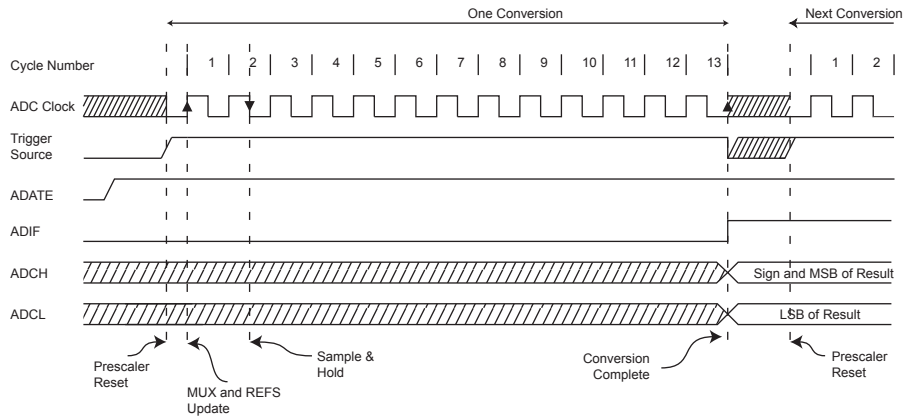


Table 18-1. Analog Comparator Positive Input

ACME	ADEN	MUX[4:0]	Analog Comparator Positive Input - Comment	
0	x	x xxxx$_b$	AIN1	ADC Switched On
x	1	x xxxx$_b$	AIN1	
1	0	0 0000$_b$	ADC0	ADC Switched Off.
1	0	0 0001$_b$	ADC1	
1	0	0 0010$_b$	ADC2	
1	0	0 0011$_b$	ADC3 / ISRC	
1	0	0 0100$_b$	ADC4	
1	0	0 0101$_b$	ADC5	
1	0	0 0110$_b$	ADC6	
1	0	0 0111$_b$	ADC7	
1	0	0 1000$_b$	ADC8	
1	0	0 1001$_b$	ADC9	
1	0	0 1010$_b$	ADC10	
1	0	Other	This doesn't make sense - Don't use.	

18.1.2 Analog Compare Negative Input

It is possible to select an internal voltage reference to replace the negative input to the Analog Comparator. The output of a 2-bit DAC using the Internal Voltage Reference of the DAC is available when ACIRS bit of ACSR register is set. The voltage reference division factor is done by ACIR[1:0] of ADCSRB register.
If ACIRS is cleared, AIN0 pin is applied to the negative input to the Analog Comparator.

Table 18-2. Analog Comparator Negative Input

ACIRS	ACIR[1:0]	REFS[1:0]	Analog Comparator Negative Input - Comment
0	x	x	AIN0
1	x	0 0$_b$ 0 1$_b$ 1 0$_b$	Reserved
1	0 0$_b$	1 1$_b$	2.56 V - using Internal 2.56V Voltage Reference
1	0 1$_b$	1 1$_b$	1.28 V ($^1/_2$ of 2.56 V) - using Internal 2.56V Voltage Reference
1	1 0$_b$	1 1$_b$	0.64 V ($^1/_4$ of 2.56 V - using Internal 2.56V Voltage Reference
1	1 1$_b$	1 1$_b$	0.32 V ($^1/_8$ of 2.56 V) - using Internal 2.56V Voltage Reference

Table 22-11. Parallel Programming Characteristics, V_{CC} = 5V ± 10%

Symbol	Parameter	Min	Typ	Max	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I_{PP}	Programming Enable Current			250	µA
t_{DVXH}	Data and Control Valid before XTAL1 High	67			ns
t_{XLXH}	XTAL1 Low to XTAL1 High	200			ns
t_{XHXL}	XTAL1 Pulse Width High	150			ns
t_{XLDX}	Data and Control Hold after XTAL1 Low	67			ns
t_{XLWL}	XTAL1 Low to $\overline{WR}$ Low	0			ns
t_{BVPH}	BS1 Valid before PAGEL High	67			ns
t_{PLBX}	BS1 Hold after PAGEL Low	67			ns
t_{WLBX}	BS2/1 Hold after $\overline{WR}$ Low	67			ns
t_{PLWL}	PAGEL Low to $\overline{WR}$ Low	67			ns
t_{BVWL}	BS1 Valid to $\overline{WR}$ Low	67			ns
t_{WLWH}	$\overline{WR}$ Pulse Width Low	150			ns
t_{WLRL}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ Low	0		1	µs
t_{WLRH}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High(1)	3.7		4.5	ms
t_{WLRH_CE}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Chip Erase(2)	7.5		9	ms
t_{XLOL}	XTAL1 Low to $\overline{OE}$ Low	0			ns
t_{BVDV}	BS1 Valid to DATA valid	0		250	ns
t_{OLDV}	$\overline{OE}$ Low to DATA Valid			250	ns
t_{OHDZ}	$\overline{OE}$ High to DATA Tri-stated			250	ns

Notes: 1. t_{WLRH} is valid for the Write Flash, Write EEPROM, Write Fuse bits and Write Lock bits commands.

2. t_{WLRH_CE} is valid for the Chip Erase command.

Figure 23-2. Active Supply Current vs. Frequency (1 - 16 MHz)

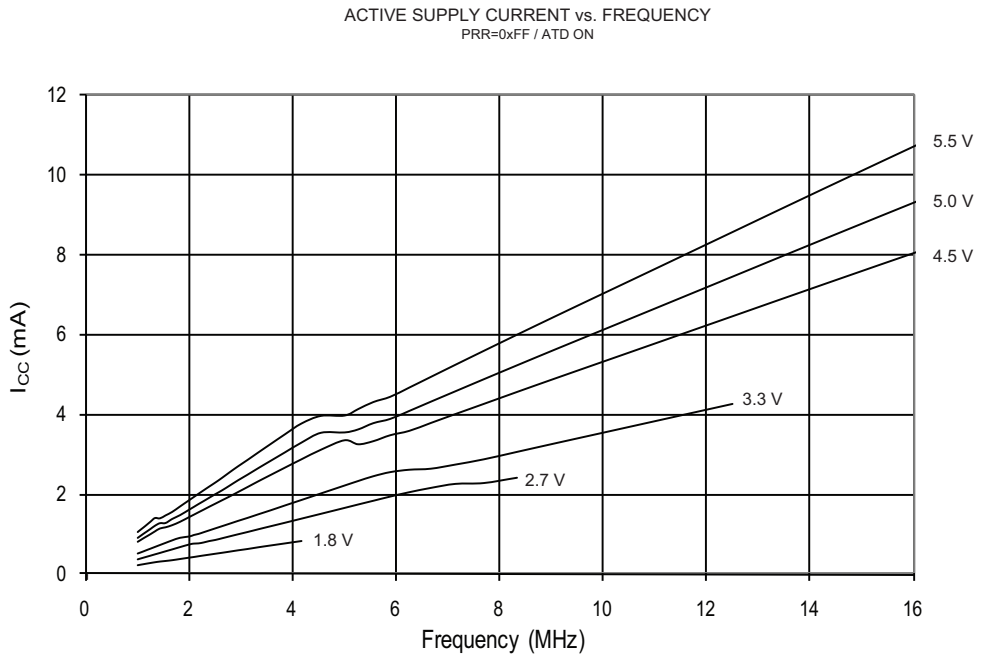
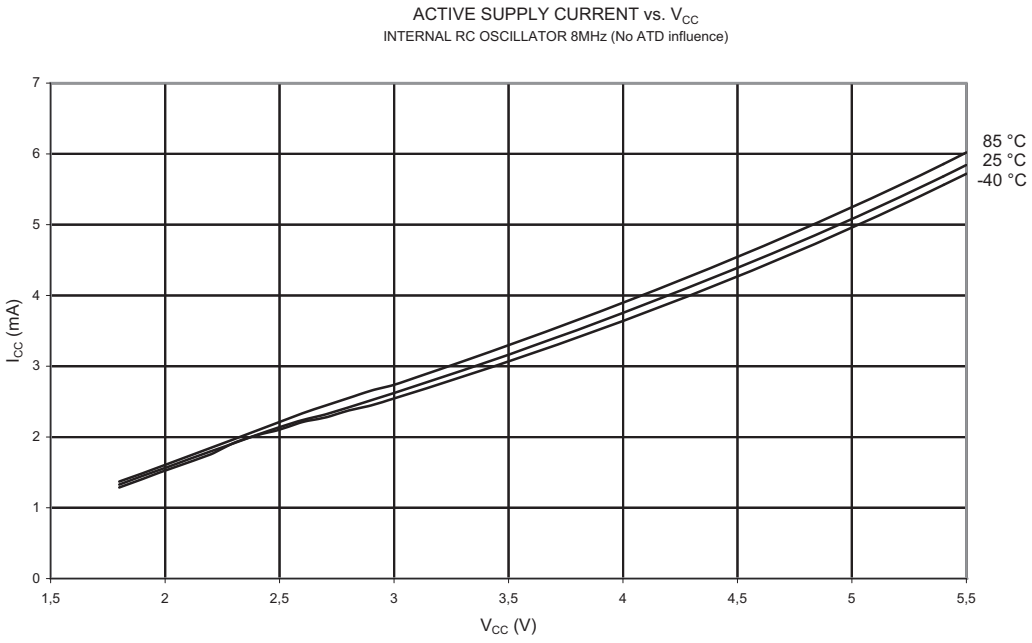


Figure 23-3. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)



```
  ; Select watchdog clock ( 128KHz, fast rising power)
     ldi    temp3,((0x03<<CSEL0)|(0x02<<CSUT0))
     sts    CLKSELR, temp3        ; (*)
; (*) !!! Loose gain control of crystal oscillator !!!
; ==> WORKAROUND ...
     sts    CLKSELR, temp1
; ...
```

3. **'*Disable Clock Source*' command remains enabled.**
In the Dynamic Clock Switch module, the '*Disable Clock Source*' command remains running after disabling the targeted clock source (the clock source is set in the CLKSELR register).

Problem fix / workaround.
After a '*Disable Clock Source*' command, reset the CLKCSR register writing 0x80.

Code example:

```
  ; Select crystal oscillator
     ldi    temp1,(0x0F<<CSEL0)
     sts    CLKSELR, temp1
  ; Disable clock source (crystal oscillator)
     ldi    temp2,(1<<CLKCCE)
     ldi    temp3,(0x01<<CLKC0)   ; CSEL = "0001"
     sts    CLKCSR,temp2          ; Enable CLKCSR register access
     sts    CLKCSR,temp3          ; (*) Disable crystal oscillator clock
; (*) !!! At this moment, if any other clock source is selected by CLKSELR,
;         this clock source will also stop !!!
; ==> WORKAROUND ...
     sts    CLKCSR,temp2
```

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**