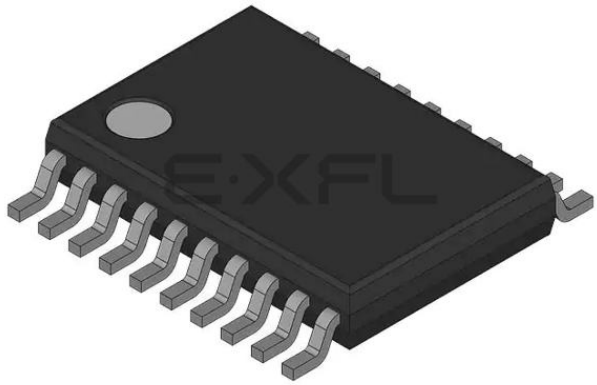


Welcome to E-XFL.COM

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)	4.5V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 150°C (TA)
Mounting Type	Surface Mount
Package / Case	20-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	20-TSSOP
Purchase URL	https://www.e-xfl.com/product-detail/atmel/attiny167-a15xd

1.7 Pin Description

1.7.1 Vcc

Supply voltage.

1.7.2 GND

Ground.

1.7.3 AVcc

Analog supply voltage.

1.7.4 AGND

Analog ground.

1.7.5 Port A (PA7..PA0)

Port A is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port A pins that are externally pulled low will source current if the pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A also serves the functions of various special features of the Atmel® ATtiny87/167 as listed on Section 9.3.3 “Alternate Functions of Port A” on page 73.

1.7.6 Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATtiny87/167 as listed on Section 9.3.4 “Alternate Functions of Port B” on page 78.

1.8 Resources

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

1.9 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.

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.

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.

2.2 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.

2.3 Status Register

The status register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the status register is updated after all ALU operations, as specified in the instruction set reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The status egister is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

Here is a “light” C-code that describes such a sequence of commands.

C Code Example

```
void ClockSwiching (unsigned char clk_number, unsigned char sut) {

#define CLOCK_RECOVER  0x05
#define CLOCK_ENABLE   0x02
#define CLOCK_SWITCH   0x04
#define CLOCK_DISABLE  0x01

unsigned char previous_clk, temp;

  // Disable interrupts
    temp = SREG; asm ("cli");
  // Save the current system clock source
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK_RECOVER;
    previous_clk = CLKSELR & 0x0F;
  // Enable the new clock source
    CLKSELR = ((sut << 4 ) & 0x30) | (clk_number & 0x0F);
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK_ENABLE;
  // Wait for clock validity
    while ((CLKCSR & (1 << CLKRDY)) == 0);
  // Switch clock source
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK_SWITCH;
  // Wait for effective switching
    while (1){
        CLKCSR = 1 << CLKCCE;
        CLKCSR = CLOCK_RECOVER;
        if ((CLKSELR & 0x0F) == (clk_number & 0x0F)) break;
    }
  // Shut down unneeded clock source
    if (previous_clk != (clk_number & 0x0F)) {
        CLKSELR = previous_clk;
        CLKCSR = 1 << CLKCCE;
        CLKCSR = CLOCK_DISABLE;
    }
  // Re-enable interrupts
    SREG = temp;
}
```

Warning: In the Atmel® ATtiny87/167, only one among the three external clock sources can be enabled at a given time. Moreover, the enables of the external clock and of the external low-frequency oscillator are shared with the asynchronous timer.

Here is a “light” C-code of a clock switching function using automatic clock monitoring.

C Code Example

```
void ClockSwiching (unsigned char clk_number, unsigned char sut) {

#define CLOCK_RECOVER   0x05
#define CLOCK_ENABLE    0x02
#define CLOCK_SWITCH    0x04
#define CLOCK_DISABLE   0x01
#define WD_ARL_ENABLE   0x06

#define WD_2048CYCLES   0x07

unsigned char previous_clk, temp;

  // Disable interrupts
    temp = SREG; asm ("cli");
  // Save the current system clock source
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK_RECOVER;
    previous_clk = CLKSELR & 0x0F;
  // Enable the new clock source
    CLKSELR = ((sut << 4 ) & 0x30) | (clk_number & 0x0F);
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK_ENABLE;
  // Wait for clock validity
    while ((CLKCSR & (1 << CLKRDY)) == 0);

  // Enable the watchdog in automatic reload mode
    WDTCSR = (1 << WDCE) | (1 << WDE);
    WDTCSR = (1 << WDE ) | WD_2048CYCLES;
    CLKCSR = 1 << CLKCCE;
    CLKCSR = WD_ARL_ENABLE;
  // Switch clock source
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK_SWITCH;
  // Wait for effective switching
    while (1){
        CLKCSR = 1 << CLKCCE;
        CLKCSR = CLOCK_RECOVER;
        if ((CLKSELR & 0x0F) == (clk_number & 0x0F)) break;
    }
  // Shut down unneeded clock source
    if (previous_clk != (clk_number & 0x0F)) {
    CLKSELR = previous_clk;
    CLKCSR = 1 << CLKCCE;
    CLKCSR = CLOCK_DISABLE;
}
  // Re-enable interrupts
    SREG = temp;
}
```

4.5.4 CLKSELR - Clock Selection Register

Bit	7	6	5	4	3	2	1	0	
	-	COUT	CSUT1	CSUT0	CSEL3	CSEL2	CSEL1	CSEL0	CLKSELR
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	~ (CKOUT) fuse	SUT1..0 fuses			CKSEL3..0 fuses			

- **Bit 7– Res: Reserved Bit**

This bit is reserved bit in the Atmel® ATtiny87/167 and will always read as zero.

- **Bit 6 – COUT: Clock Out**

The COUT bit is initialized with ~(CKOUT) fuse bit.
The COUT bit is only used in case of '*CKOUT*' command. Refer to Section 4.2.7 "Clock Output Buffer" on page 32 for using.
In case of '*recover system clock Source*' command, COUT it is not affected (no recovering of this setting).

- **Bits 5:4 – CSUT1:0: Clock Start-up Time**

CSUT bits are initialized with the values of SUT fuse bits.
In case of '*enable/disable clock source*' command, CSUT field provides the code of the clock start-up time. Refer to subdivisions of Section 4.2 "Clock Sources" on page 26 for code of clock start-up times.
In case of '*recover system clock source*' command, CSUT field is not affected (no recovering of SUT code).

- **Bits 3:0 – CSEL3:0: Clock Source Select**

CSEL bits are initialized with the values of CKSEL fuse bits.
In case of '*enable/disable clock source*', '*request for clock availability*' or '*clock source switch*' command, CSEL field provides the code of the clock source. Refer to Table 4-1 on page 26 and subdivisions of Section 4.2 "Clock Sources" on page 26 for clock source codes.
In case of '*recover system clock source*' command, CSEL field contains the code of the clock source used to drive the clock control unit as described in Figure 4-1 on page 25.

7.2 Program Setup in ATtiny87

The most typical and general program setup for the reset and interrupt vector addresses in Atmel® ATtiny87 is (2-byte step - using “rjmp” instruction):

```
Address(1) Label Code                 Comments
0x0000           rjmp   RESET          ; Reset Handler
0x0001           rjmp   INT0addr       ; IRQ0 Handler
0x0002           rjmp   INT1addr       ; IRQ1 Handler
0x0003           rjmp   PCINT0addr     ; PCINT0 Handler
0x0004           rjmp   PCINT1addr     ; PCINT1 Handler
0x0005           rjmp   WDTaddr        ; Watchdog Timer Handler
0x0006           rjmp   ICP1addr       ; Timer1 Capture Handler
0x0007           rjmp   OC1Aaddr       ; Timer1 Compare A Handler
0x0008           rjmp   OC1Baddr       ; Timer1 Compare B Handler
0x0009           rjmp   OVF1addr       ; Timer1 Overflow Handler
0x000A           rjmp   OC0Aaddr       ; Timer0 Compare A Handler
0x000B           rjmp   OVF0addr       ; Timer0 Overflow Handler
0x000C           rjmp   LINTCaddr      ; LIN Transfer Complete Handler
0x000D           rjmp   LINERRaddr     ; LIN Error Handler
0x000E           rjmp   SPIaddr        ; SPI Transfer Complete Handler
0x000F           rjmp   ADCCaddr       ; ADC Conversion Complete Handler
0x0010           rjmp   ERDYaddr       ; EEPROM Ready Handler
0x0011           rjmp   ACIaddr        ; Analog Comparator Handler
0x0012           rjmp   USISTARTaddr   ; USI Start Condition Handler
0x0013           rjmp   USIOVFaddr     ; USI Overflow Handler

0x0014 RESET:    ldi    r16, high(RAMEND); Main program start
0x0015           out    SPH,r16          ; Set Stack Pointer to top of RAM
0x0016           ldi    r16, low(RAMEND)
0x0017           out    SPL,r16
0x0018           sei                     ; Enable interrupts
0x0019           <instr>  xxx
   ...     ...    ...    ...
```

Note: 1. 16-bit address

9.4 Register Description for I/O Ports

9.4.1 Port A Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
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	

9.4.2 Port A Data Direction Register – DDRA

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
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	

9.4.3 Port A Input Pins Register – PINA

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

9.4.4 Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
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	

9.4.5 Port B Data Direction Register – DDRB

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
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	

9.4.6 Port B Input Pins Register – PINB

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	R/(W)	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

10.8 Timer/Counter Timing Diagrams

The following figures show the timer/counter in synchronous mode, and the timer clock (clk_T0) is therefore shown as a clock enable signal. In asynchronous mode, $clk_{I/O}$ should be replaced by the timer/counter oscillator clock. The figures include information on when interrupt flags are set. Figure 10-8 contains timing data for basic timer/counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 10-8. Timer/Counter Timing Diagram, no Prescaling

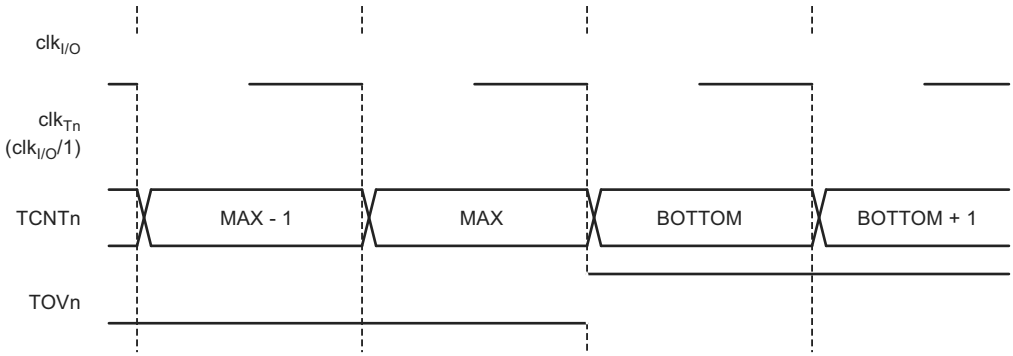


Figure 10-9 shows the same timing data, but with the prescaler enabled.

Figure 10-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}$/8)

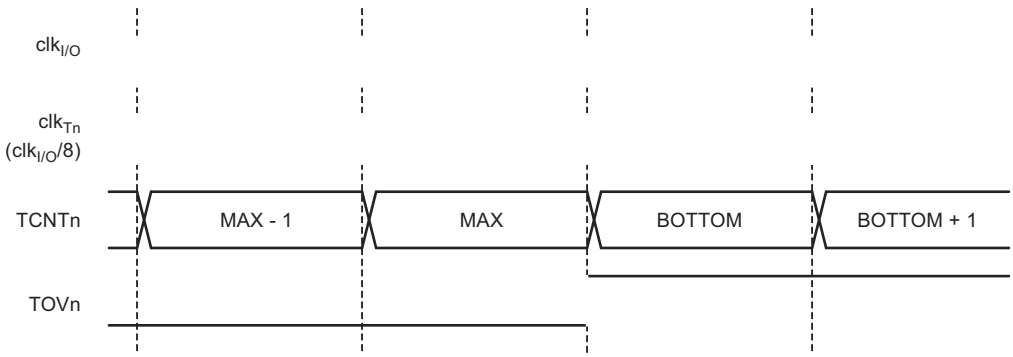
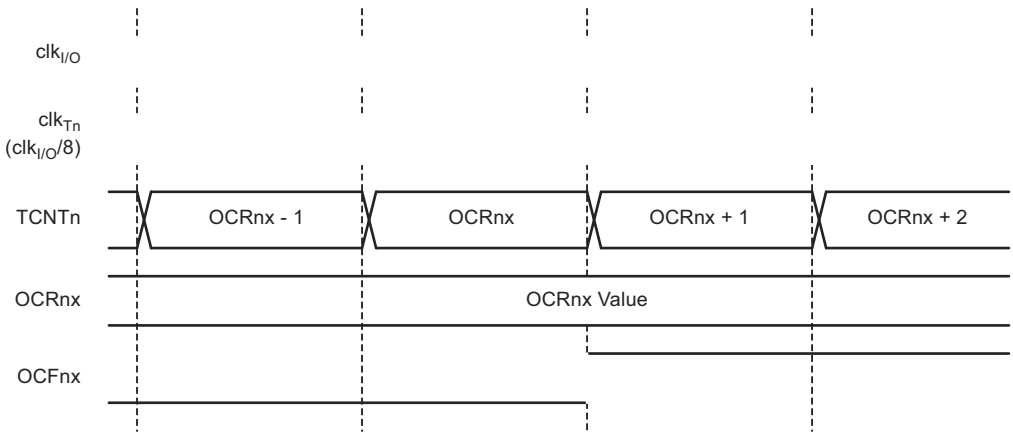


Figure 10-10 shows the setting of OCF0A in all modes except CTC mode.

Figure 10-10.Timer/Counter Timing Diagram, Setting of OCF0A, with Prescaler ($f_{clk\_I/O}$/8)



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.

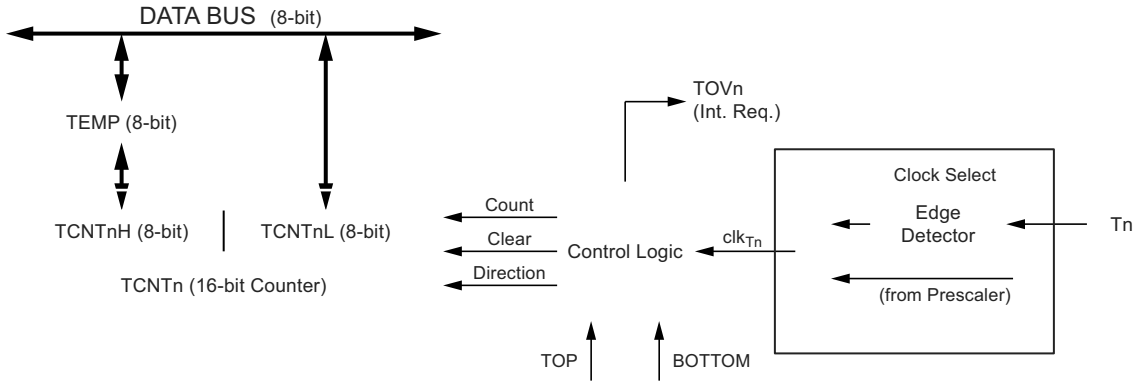
12.4 Timer/Counter Clock Sources

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.

12.5 Counter Unit

The main part of the 16-bit timer/counter is the programmable 16-bit bi-directional counter unit. Figure 12-2 shows a block diagram of the counter and its surroundings.

Figure 12-2. Counter Unit Block Diagram



Signal description (internal signals):

Count	Increment or decrement TCNT1 by 1.
Direction	Select between increment and decrement.
Clear	Clear TCNT1 (set all bits to zero).
clk_T1	Timer/counter clock.
TOP	Signalize that TCNT1 has reached maximum value.
BOTTOM	Signalize that TCNT1 has reached minimum value (zero).

The 16-bit counter is mapped into two 8-bit I/O memory locations: counter High (TCNT1H) containing the upper eight bits of the counter, and counter low (TCNT1L) containing the lower eight bits. The TCNT1H register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNT1H I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNT1H value when the TCNT1L is read, and TCNT1H is updated with the temporary register value when TCNT1L is written. This allows the CPU to read or write the entire 16-bit counter value within one clock cycle via the 8-bit data bus. It is important to notice that there are special cases of writing to the TCNT1 register when the counter is counting that will give unpredictable results. The special cases are described in the sections where they are of importance.

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_T1). The clk_T1 can be generated from an external or internal clock source, selected by the clock select bits (CS12:0). When no clock source is selected (CS12:0 = 0) the timer is stopped. However, the TCNT1 value can be accessed by the CPU, independent of whether clk_T1 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 waveform generation mode bits (WGM13:0) located in the timer/counter control registers A and B (TCCR1A and TCCR1B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the output compare outputs OC1A/B. For more details about advanced counting sequences and waveform generation, see Section 12.9 “Modes of Operation” on page 115.

The timer/counter overflow flag (TOV1) is set according to the mode of operation selected by the WGM13:0 bits. TOV1 can be used for generating a CPU interrupt.

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)

12.11 16-bit Timer/Counter Register Description

12.11.1 Timer/Counter1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
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	

- **Bit 7:6 – COM1A1:0: Compare Output Mode for Channel A**

- **Bit 5:4 – COM1B1:0: Compare Output Mode for Channel B**

The COM1A1:0 and COM1B1:0 control the output compare pins (OC1Ai and OC1Bi respectively) behavior. If one or both of the COM1A1:0 bits are written to one, the OC1Ai output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COM1B1:0 bit are written to one, the OC1Bi output overrides the normal port functionality of the I/O pin it is connected to. However, note that the data direction register (DDR) bit and OC1xi bit (TCCR1D) corresponding to the OC1Ai or OC1Bi pin must be set in order to enable the output driver.

When the OC1Ai or OC1Bi is connected to the pin, the function of the COM1A/B1:0 bits is dependent of the WGM13:0 bits setting. Table 12-1 shows the COM1A/B1:0 bit functionality when the WGM13:0 bits are set to a Normal or a CTC mode (non-PWM).

Table 12-1. Compare Output Mode, non-PWM

OC1Ai OC1Bi	COM1A1 COM1B1	COM1A0 COM1B0	Description
0	x	x	Normal port operation, OC1A/OC1B disconnected.
1	0	0	
	0	1	Toggle OC1A/OC1B on compare match.
	1	0	Clear OC1A/OC1B on compare match (set output to low level).
	1	1	Set OC1A/OC1B on compare match (set output to high level).

Table 12-2 shows the COM1A/B1:0 bit functionality when the WGM13:0 bits are set to the fast PWM mode.

Table 12-2. Compare Output Mode, Fast 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 Set OC1A/OC1B at TOP
1	1	1	Set OC1A/OC1B on compare match Clear OC1A/OC1B at TOP

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. In this case the compare match is ignored, but the set or clear is done at TOP. See Section 12.9.3 “Fast PWM Mode” on page 117 for more details.

Figure 13-3. SPI Transfer Format with CPHA = 0

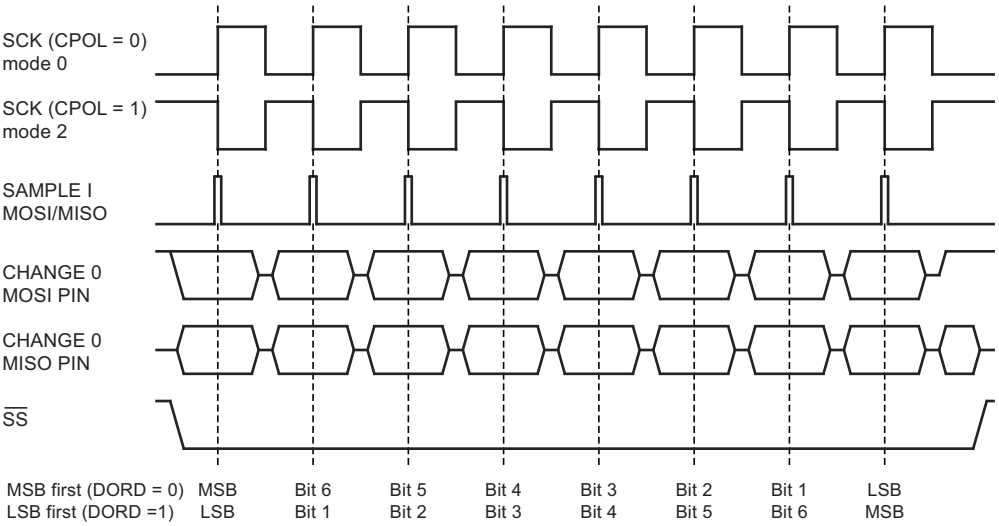
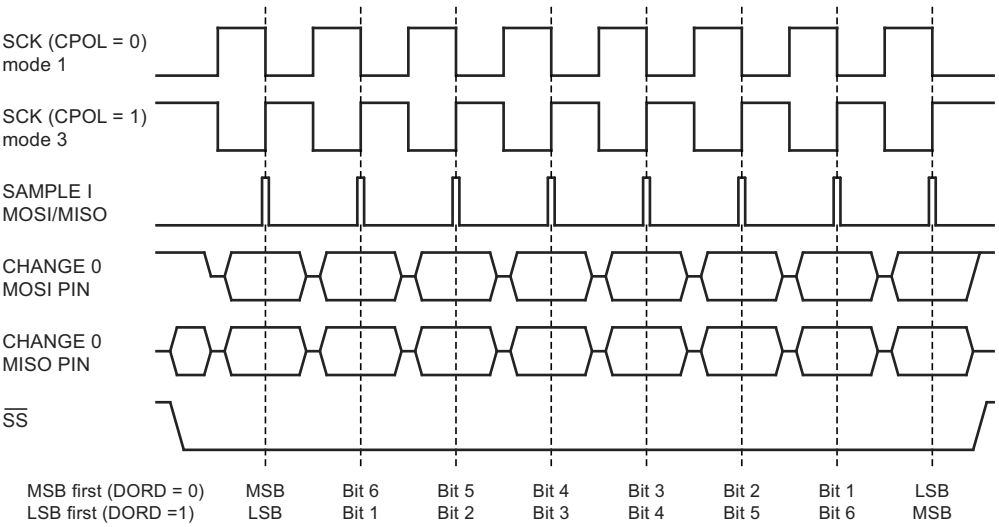


Figure 13-4. SPI Transfer Format with CPHA = 1



- **Bits 3:0 – USICNT3..0: Counter Value**

These bits reflect the current 4-bit counter value. The 4-bit counter value can directly be read or written by the CPU.

The 4-bit counter increments by one for each clock generated either by the external clock edge detector, by a timer/counter0 compare match, or by software using USICLK or USITC strobe bits. The clock source depends of the setting of the USICS1..0 bits. For external clock operation a special feature is added that allows the clock to be generated by writing to the USITC strobe bit. This feature is enabled by write a one to the USICLK bit while setting an external clock source (USICS1 = 1).

Note that even when no wire mode is selected (USIWM1..0 = 0) the external clock input (USCK/SCL) are can still be used by the counter.

14.5.4 USICR – USI Control Register

Bit	7	6	5	4	3	2	1	0	
	USISIE	**USIOIE**	**USIWM1**	**USIWM0**	**USICS1**	**USICS0**	**USICLK**	**USITC**	**USICR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

The control register includes interrupt enable control, wire mode setting, clock select setting, and clock strobe.

- **Bit 7 – USISIE: Start Condition Interrupt Enable**

Setting this bit to one enables the start condition detector interrupt. If there is a pending interrupt when the USISIE and the global interrupt enable flag is set to one, this will immediately be executed. Refer to the USISIF bit description on page 145 for further details.

- **Bit 6 – USIOIE: Counter Overflow Interrupt Enable**

Setting this bit to one enables the counter overflow interrupt. If there is a pending interrupt when the USIOIE and the global interrupt enable flag is set to one, this will immediately be executed. Refer to the USIOIF bit description on page 145 for further details.

- **Bit 5:4 – USIWM1:0: Wire Mode**

These bits set the type of wire mode to be used. Basically only the function of the outputs are affected by these bits. Data and clock inputs are not affected by the mode selected and will always have the same function. The counter and USI data register can therefore be clocked externally, and data input sampled, even when outputs are disabled. The relations between USIWM1:0 and the USI operation is summarized in Table 14-1 on page 147.

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.

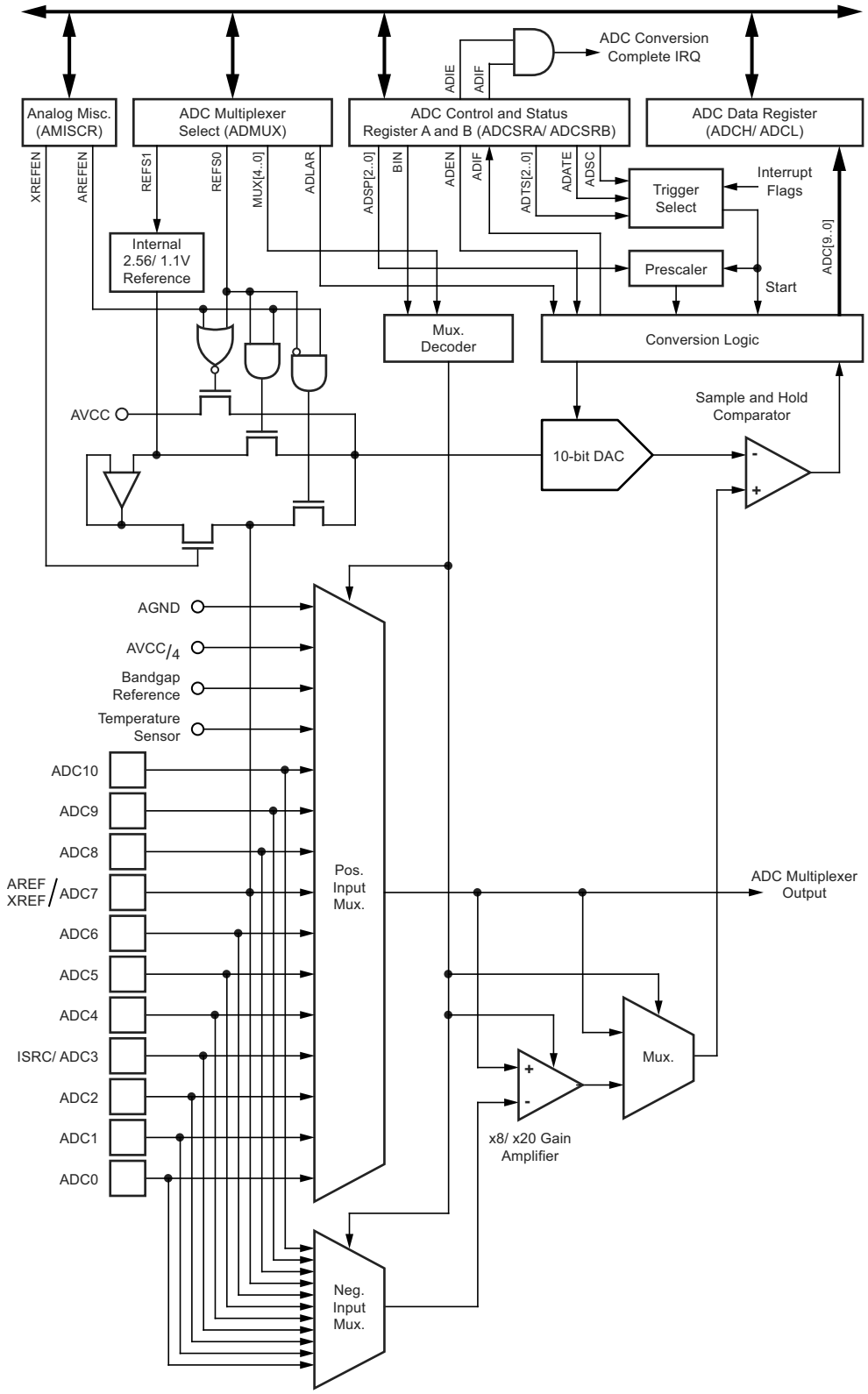
- Byte transfer: the UART is selected but both Rx and Tx services are disabled,
- Rx byte: only the Rx service is enable but Tx service is disabled,
- Tx byte: only the Tx service is enable but Rx service is disabled,
- Full duplex: the UART is selected and both Rx and Tx services are enabled.

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.

Figure 17-1. Analog to Digital Converter Block Schematic



18.1.3 DIDR0 – Digital Input Disable Register 0

Bit	7	6	5	4	3	2	1	0	
	ADC7D / AIN1D	ADC6D / AIN0D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
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 7,6 – AIN1D, AIN0D: AIN1D and AIN0D Digital Input Disable**

When this bit is written logic one, the digital input buffer on the corresponding analog compare pin is disabled. The corresponding PIN register bit will always read as zero when this bit is set. When an analog signal is applied to the AIN0/1 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

18.2 Analog Comparator Inputs

18.2.1 Analog Compare Positive Input

It is possible to select any of the inputs of the ADC positive input multiplexer to replace the positive input to the analog comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the analog comparator multiplexer enable bit (ACME in ADCSRB register) is set and the ADC is switched off (ADEN in ADCSRA register is zero), MUX[4..0] in ADMUX register select the input pin to replace the positive input to the analog comparator, as shown in Table 18-2 on page 196. If ACME is cleared or ADEN is set, AIN1 pin is applied to the positive input to the analog comparator.

Table 18-2. 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.	

21.2 Fuse Bits

The Atmel ATtiny87/167 has three fuse bytes. Table 21-3, Table 21-4 and Table 21-5 describe briefly the functionality of all the fuses and how they are mapped into the fuse bytes.

The SPM instruction is enabled for the whole flash if the SELFPRGEN fuse is programmed (“0”), otherwise it is disabled.

Note that the fuses are read as logical zero, “0”, if they are programmed.

Table 21-3. Extended Fuse Byte

Fuse Extended Byte	Bit No	Description	Default Value
–	7	–	1 (unprogrammed)
–	6	–	1 (unprogrammed)
–	5	–	1 (unprogrammed)
–	4	–	1 (unprogrammed)
–	3	–	1 (unprogrammed)
–	2	–	1 (unprogrammed)
–	1	–	1 (unprogrammed)
SELFPRGEN	0	Self programming enable	1 (unprogrammed)

Table 21-4. Fuse High Byte

Fuse High Byte	Bit No	Description	Default Value
RSTDISBL(1)	7	External reset disable	1 (unprogrammed)
DWEN	6	DebugWIRE enable	1 (unprogrammed)
SPIEN(2)	5	Enable serial program and data downloading	0 (programmed, SPI programming enabled)
WDTON(3)	4	Watchdog timer always on	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the chip erase	1 (unprogrammed, EEPROM not preserved)
BODLEVEL2(4)	2	Brown-out detector trigger level	1 (unprogrammed)
BODLEVEL1(4)	1	Brown-out detector trigger level	1 (unprogrammed)
BODLEVEL0(4)	0	Brown-out detector trigger level	1 (unprogrammed)

Notes:
1. Section 9.3.4 “Alternate Functions of Port B” on page 78 for description of RSTDISBL fuse.
2. The SPIEN fuse is not accessible in serial programming mode.
3. Section 6.3.3 “Watchdog Timer Control Register - WDTCR” on page 55 for details.
4. See Table 22-5 on page 226 for BODLEVEL fuse coding.

3. The serial programming instructions will not work if the communication is out of synchronization. When in sync. the second byte (0x53), will echo back when issuing the third byte of the programming enable instruction. Whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the 0x53 did not echo back, give $\overline{\text{RESET}}$ a positive pulse and issue a new programming enable command.
4. The flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 5 LSB of the address and data together with the load program memory page instruction. To ensure correct loading of the page, the data low byte must be loaded before data high byte is applied for a given address. The program memory page is stored by loading the write program memory age instruction with the 6 MSB of the address. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least t_{WD_FLASH} before issuing the next page.
(See Table 21-14) accessing the serial programming interface before the flash write operation completes can result in incorrect programming.
5. **A:** The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate write instruction. An EEPROM memory location is first automatically erased before new data is written. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte.
(See Table 21-14) in a chip erased device, no 0xFFs in the data file(s) need to be programmed.
B: The EEPROM array is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 2 LSB of the address and data together with the load EEPROM memory page instruction. The EEPROM memory page is stored by loading the write EEPROM memory page instruction with the 6 MSB of the address. When using EEPROM page access only byte locations loaded with the Load EEPROM memory page instruction is altered. The remaining locations remain unchanged. If polling (RDY/$\overline{\text{BSY}}$) is not used, the used must wait at least t_{WD_EEPROM} before issuing the next page (See Table 21-8 on page 210). In a chip erased device, no 0xFF in the data file(s) need to be programmed.
6. Any memory location can be verified by using the read instruction which returns the content at the selected address at serial output MISO.
7. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
8. Power-off sequence (if needed):
Set $\overline{\text{RESET}}$ to “1”.
Turn Vcc power off.

Table 21-14. Minimum Wait Delay Before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t_{WD_FLASH}	4.5ms
t_{WD_EEPROM}	4.0ms
t_{WD_ERASE}	4.0ms
t_{WD_FUSE}	4.5ms

24.3 Supply Current of I/O modules

The table below can be used to calculate the additional current consumption for the different I/O modules idle mode. The enabling or disabling of the I/O modules are controlled by the power reduction register. See Section 5.9.3 “PRR – Power Reduction Register” on page 46 for details.

Table 24-1. Additional Current Consumption for the different I/O modules (absolute values)

Module	Vcc = 5.0V Freq. = 16MHz	Vcc = 5.0V Freq. = 8MHz	Vcc = 3.0V Freq. = 8MHz	Vcc = 3.0V Freq. = 4MHz	Units
LIN/UART	0.77	0.37	0.20	0.10	mA
SPI	0.31	0.14	0.08	0.04	mA
TIMER-1	0.28	0.13	0.08	0.04	mA
TIMER-0	0.41	0.20	0.10	0.05	mA
USI	0.14	0.05	0.04	0.02	mA
ADC	0.48	0.22	0.10	0.05	mA

24.4 Power-down Supply Current

Figure 24-8. Power-down Supply Current versus Vcc (Watchdog Timer Disabled)

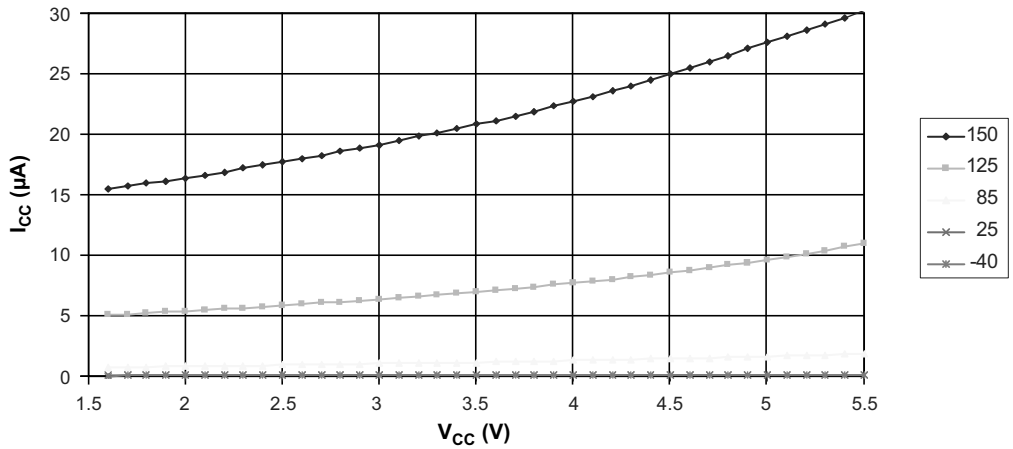


Figure 24-9. Power-down Supply Current versus Vcc (Watchdog Timer Enabled)