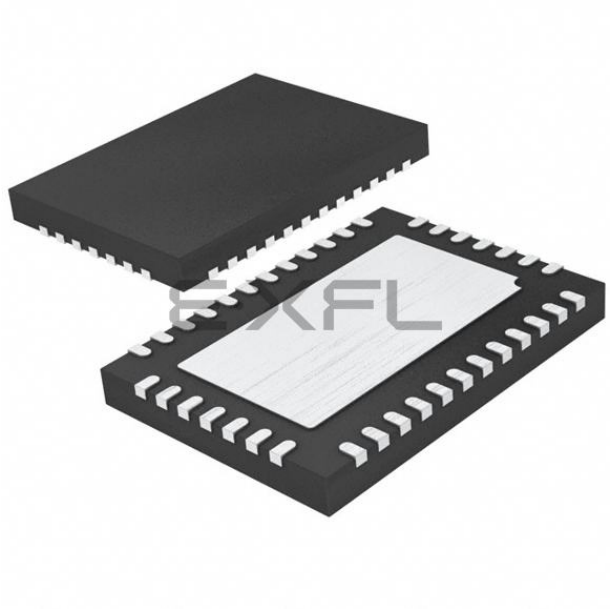


Welcome to E-XFL.COM

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	16KB (16K x 8)
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 (TC)
Mounting Type	Surface Mount
Package / Case	38-VFQFN Exposed Pad
Supplier Device Package	38-VQFN (5x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/ata6617c-p3qw

Table 2-2. Maximum Ratings of the LIN-SBC (Continued)

Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Parameters	Symbol	Min.	Typ.	Max.	Unit
LIN - DC voltage		 –27		 +40	 V
Logic pins (RXD, TXD, EN, NRES, NTRIG, WD_OSC, MODE, TM)		–0.3		+5.5	V
Output current NRES	I_{NRES}			+2	mA
PVCC DC voltage VCC DC voltage		–0.3 –0.3		+5.5 +6.5	V V

Table 2-3. Maximum Ratings of the Microcontroller

Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Parameters	Symbol	Min.	Typ.	Max.	Unit
Voltage on any pin except $\overline{\text{RESET}}$ with respect to Ground		–0.5		MCUVCC + 0.5	V
Voltage on $\overline{\text{RESET}}$ with respect to GND		–0.5		13.0	V
Voltage on MCUVCC with respect to GND		–0.5		6.0	V
DC current per I/O pin				40.0	mA
DC current MCUVCC and GND pins				200.0	mA
Injection current at MCUVCC = 0V to 5V(2)				±5.0	mA

Notes: 1. Maximum current per port = ±30mA
2. Functional corruption may occur

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;
}
```

4.4.3.6 Preventing EEPROM Corruption

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

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

EEPROM data corruption can easily be avoided by following this design recommendation: Keep the AVR® RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal brown-out detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low V_{CC} reset protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

4.6.3 Idle Mode

When the SM1..0 bits are written to 00, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing the SPI, analog comparator, ADC, USI start condition, asynchronous Timer/Counter, watchdog, and the interrupt system to continue operating. This sleep mode basically halts clk_{CPU} and clk_{FLASH}, while allowing the other clocks to run.

Idle mode enables the MCU to wake up from external triggered interrupts as well as internal ones like the SPI interrupts. If wake-up from the analog comparator interrupt is not required, the analog comparator can be powered down by setting the ACD bit in the analog comparator control and status register – ACSR. This will reduce power consumption in Idle mode. If the ADC is enabled, a conversion starts automatically when this mode is entered.

4.6.4 ADC Noise Reduction Mode

When the SM1..0 bits are written to 01, the SLEEP instruction makes the MCU enter ADC noise reduction mode, stopping the CPU but allowing the ADC, the external interrupts, the USI start condition, the asynchronous Timer/Counter and the watchdog to continue operating (if enabled). This sleep mode basically halts $clk_{I/O}$, clk_{CPU}, and clk_{FLASH}, while allowing the other clocks to run.

This improves the noise environment for the ADC, enabling higher resolution measurements. If the ADC is enabled, a conversion starts automatically when this mode is entered. Apart from the ADC conversion complete interrupt, only an external reset, a watchdog system reset, a watchdog interrupt, a brown-out reset, a USI start condition interrupt, an asynchronous Timer/Counter interrupt, an SPM/EEPROM ready interrupt, an external level interrupt on INT0 or INT1 or a pin change interrupt can wake up the MCU from ADC noise reduction mode.

4.6.5 Power-down Mode

When the SM1..0 bits are written to 10, the SLEEP instruction makes the MCU enter power-down mode. In this mode, the external oscillator is stopped, while the external interrupts, the USI start condition, and the Watchdog continue operating (if enabled). Only an external reset, a watchdog system reset, a watchdog interrupt, a brown-out reset, the USI start condition interrupt, an external level interrupt on INT0 or INT1, or a pin change interrupt can wake up the MCU. This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

Note that if a level triggered interrupt is used for wake-up from power-down mode, the changed level must be held for some time to wake up the MCU. Refer to Section 4.9 “External Interrupts” on page 79 for details.

When waking up from power-down mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is defined by the same CKSEL fuses that define the reset time-out period, as described in Section 4.5.2 “Clock Sources” on page 47.

4.6.6 Power-save Mode

When the SM1..0 bits are written to 11, the SLEEP instruction makes the MCU enter power-save mode. This mode is identical to power-down, with one exception:

If Timer/Counter0 is clocked asynchronously, i.e., the AS0 bit in ASSR is set, Timer/Counter0 will run during sleep. The device can wake up from either timer overflow or output compare event from Timer/Counter0 if the corresponding Timer/Counter0 interrupt enable bits are set in TIMSK0, and the global interrupt enable bit in SREG is set.

If the asynchronous timer is **NOT** clocked asynchronously, power-down mode is recommended instead of power-save mode because the contents of the registers in the asynchronous timer should be considered undefined after wake-up in power-save mode if AS0 is 0.

This sleep mode basically halts all clocks except clk_{ASY}, allowing operation only of asynchronous modules, including Timer/Counter0 if clocked asynchronously.

4.6.7 Power Reduction Register

The power reduction register (PRR), see Section 4.6.9.3 “PRR – Power Reduction Register” on page 66, provides a method to stop the clock to individual peripherals to reduce power consumption. The current state of the peripheral is frozen and the I/O registers can not be read or written. Resources used by the peripheral when stopping the clock will remain occupied, hence the peripheral should in most cases be disabled before stopping the clock. Waking up a module, which is done by clearing the bit in PRR, puts the module in the same state as before shutdown.

Module shutdown can be used in Idle mode and Active mode to significantly reduce the overall power consumption. In all other sleep modes, the clock is already stopped.

4.6.8.8 On-chip Debug System

If the On-chip debug system is enabled by the DWEN Fuse and the chip enters sleep mode, the main clock source is enabled and hence always consumes power. In the deeper sleep modes, this will contribute significantly to the total current consumption.

4.6.9 Register Description

4.6.9.1 SMCR – Sleep Mode Control Register

The sleep mode Control Register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	SM1	SM0	SE	SMCR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..3 Res: Reserved Bits**

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

- **Bits 2..1 – SM1..0: Sleep Mode Select Bits 1, and 0**

These bits select between the four available sleep modes as shown in Table 4-17.

Table 4-17. Sleep Mode Select

SM1	SM0	Sleep Mode
0	0	Idle
0	1	ADC noise reduction
1	0	Power-down
1	1	Power-save

- **Bit 0 – SE: Sleep Enable**

The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmer's purpose, it is recommended to write the sleep enable (SE) bit to one just before the execution of the SLEEP instruction and to clear it immediately after waking up.

4.6.9.2 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
	–	BODS	BODSE	PUD	–	–	–	–	MCUCR
Read/Write	R	R/W	R/W	R/W	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6 – BODS: BOD Sleep**

The BODS bit must be written to logic one in order to turn off BOD during sleep, see Table 4-16 on page 62. Writing to the BODS bit is controlled by a timed sequence and an enable bit, BODSE in MCUCR. To disable BOD in relevant sleep modes, both BODS and BODSE must first be set to one. Then, to set the BODS bit, BODS must be set to one and BODSE must be set to zero within four clock cycles.

The BODS bit is active three clock cycles after it is set. A sleep instruction must be executed while BODS is active in order to turn off the BOD for the actual sleep mode. The BODS bit is automatically cleared after three clock cycles.

- **Bit 5 – BODSE: BOD Sleep Enable**

BODSE enables setting of BODS control bit, as explained in BODS bit description. BOD disable is controlled by a timed sequence.

The following code example shows one assembly and one C function for turning off the watchdog timer. The example assumes that interrupts are controlled (e.g. by disabling interrupts globally) so that no interrupts will occur during the execution of these functions.

Assembly Code Example[1]

```
WDT_off:
       ; Turn off global interrupt
       cli
       ; Reset Watchdog Timer
       wdr
       ; Clear WDRF in MCUSR
       in    r16, MCUSR
       andi r16, (0xff & (0<<WDRF))
       out   MCUSR, r16
       ; Write logical one to WDCE and WDE
       ; Keep old prescaler setting to prevent unintentional time-out
       lds r16, WDTCR
       ori   r16, (1<<WDCE) | (1<<WDE)
       sts WDTCR, r16
       ; Turn off WDT
       ldi   r16, (0<<WDE)
       sts WDTCR, r16
       ; Turn on global interrupt
       sei
       ret
```

C Code Example[1]

```
void WDT_off(void)
{
       __disable_interrupt();
       __watchdog_reset();
       /* Clear WDRF in MCUSR */
       MCUSR &= ~(1<<WDRF);
       /* Write logical one to WDCE and WDE */
       /* Keep old prescaler setting to prevent unintentional time-out */
       WDTCR |= (1<<WDCE) | (1<<WDE);
       /* Turn off WDT */
       WDTCR = 0x00;
       __enable_interrupt();
}
```

Notes:
1. See Section 4.2.7 “About Code Examples” on page 29.
2. If the watchdog is accidentally enabled, for example by a runaway pointer or brown-out condition, the device will be reset and the watchdog timer will stay enabled. If the code is not set up to handle the watchdog, this might lead to an eternal loop of time-out resets. To avoid this situation, the application software should always clear the watchdog system reset flag (WDRF) and the WDE control bit in the initialization routine, even if the watchdog is not in use.

4.9.3.5 Pin Change Interrupt Flag Register – PCIFR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	PCIF1	PCIF0	PCIFR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7, 2 – Res: Reserved Bits**

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

- **Bit 1 - PCIF1: Pin Change Interrupt Flag 1**

When a logic change on any PCINT15..8 pin triggers an interrupt request, PCIF1 becomes set (one). If the I-bit in SREG and the PCIE1 bit in PCICR are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bit 0 - PCIF0: Pin Change Interrupt Flag 0**

When a logic change on any PCINT7..0 pin triggers an interrupt request, PCIF0 becomes set (one). If the I-bit in SREG and the PCIE0 bit in PCICR are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

4.9.3.6 Pin Change Mask Register 1 – PCMSK1

Bit	7	6	5	4	3	2	1	0	
	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
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	

- **Bit 7..0 – PCINT15..8: Pin Change Enable Mask 15..8**

Each PCINT15..8-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT15..8 is set and the PCIE1 bit in EIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT15..8 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

4.9.3.7 Pin Change Mask Register 0 – PCMSK0

Bit	7	6	5	4	3	2	1	0	
	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
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	

- **Bit 7..0 – PCINT7..0: Pin Change Enable Mask 7..0**

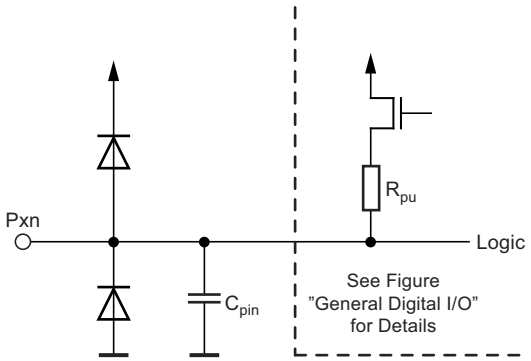
Each PCINT7..0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is set and the PCIE0 bit in EIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

4.10 I/O-Ports

4.10.1 Introduction

All AVR® ports have true read-modify-write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in Figure 4-24. Refer to Section 4.23 “Electrical Characteristics” on page 237 for a complete list of parameters.

Figure 4-24. I/O Pin Equivalent Schematic



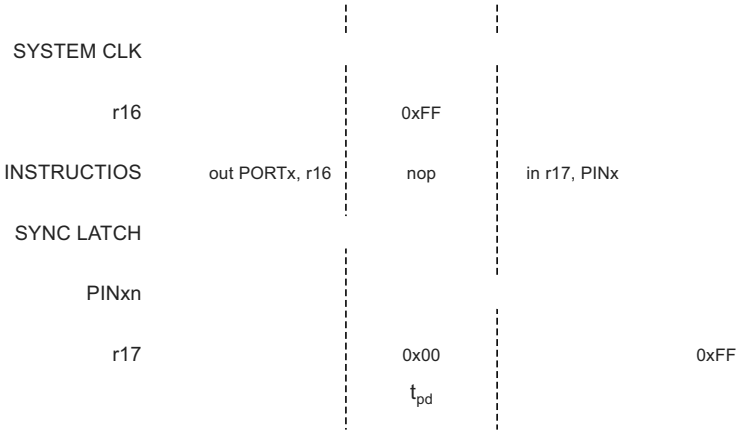
All registers and bit references in this section are written in general form. A lower case “x” represents the numbering letter for the port, and a lower case “n” represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn. The physical I/O registers and bit locations are listed in Section 4.10.4 “Register Description for I/O Ports” on page 101.

Three I/O memory address locations are allocated for each port, one each for the data register – PORTx, data direction register – DDRx, and the port input pins – PINx. The port input pins I/O location is read only, while the data register and the data direction register are read/write. However, writing a logic one to a bit in the PINx register, will result in a toggle in the corresponding bit in the data register. In addition, the pull-up disable – PUD bit in MCUCR or PUDx in PORTCR disables the pull-up function for all pins in all ports when set.

Using the I/O port as general digital I/O is described in Section 4.10.2 “Ports as General Digital I/O” on page 84. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in Section 4.10.3 “Alternate Port Functions” on page 89. Refer to the individual module sections for a full description of the alternate functions.

Note that enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

Figure 4-28. Synchronization when Reading a Software Assigned Pin Value



The following code example shows how to set port B pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 7 as input with pull-ups assigned to port pins 6 and 7. The resulting pin values are read back again, but as previously discussed, a nop instruction is included to be able to read back the value recently assigned to some of the pins.

Assembly Code Example[1]

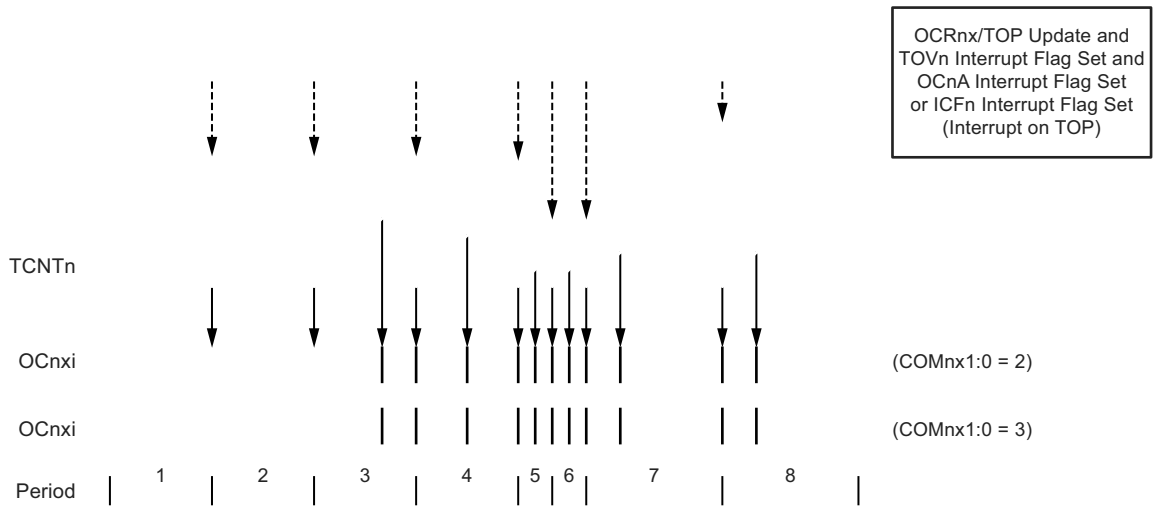
```
...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi    r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
ldi    r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
out    PORTB,r16
out    DDRB,r17
; Insert nop for synchronization
nop
; Read port pins
in     r16,PINB
...
```

C Code Example

```
unsigned char i;
...
/* Define pull-ups and set outputs high */
/* Define directions for port pins */
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
/* Insert nop for synchronization*/
__no_operation();
/* Read port pins */
i = PINB;
...
```

Note: 1. For the assembly program, two temporary registers are used to minimize the time from pull-ups are set on pins 0, 1, 6, and 7, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

Figure 4-51. Fast PWM Mode, Timing Diagram



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

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the compare registers. If the TOP value is lower than any of the compare registers, a compare match will never occur between the TCNT1 and the OCR1A/B. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCR1A/B registers are written.

The procedure for updating ICR1 differs from updating OCR1A when used for defining the TOP value. The ICR1 register is not double buffered. This means that if ICR1 is changed to a low value when the counter is running with none or a low prescaler value, there is a risk that the new ICR1 value written is lower than the current value of TCNT1. The result will then be that the counter will miss the compare match at the TOP value. The counter will then have to count to the MAX value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. The OCR1A register however, is double buffered. This feature allows the OCR1A I/O location to be written anytime. When the OCR1A I/O location is written the value written will be put into the OCR1A buffer register. The OCR1A compare register will then be updated with the value in the Buffer Register at the next timer clock cycle the TCNT1 matches TOP. The update is done at the same timer clock cycle as the TCNT1 is cleared and the TOV1 flag is set.

Using the ICR1 register for defining TOP works well when using fixed TOP values. By using ICR1, the OCR1A register is free to be used for generating a PWM output on OC1A. However, if the base PWM frequency is actively changed (by changing the TOP value), using the OCR1A as TOP is clearly a better choice due to its double buffer feature.

In fast PWM mode, the compare units allow generation of PWM waveforms on the OC1A/B pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1A/B1:0 to three (see Table 4-36 on page 141). The actual OC1A/B value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1A/B) and OC1A/Bi is set. The PWM waveform is generated by setting (or clearing) the OC1A/B Register at the compare match between OCR1A/B and TCNT1, and clearing (or setting) the OC1A/B Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

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

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \times (1 + TOP)}$$

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

The extreme values for the OCR1A/B Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR1A/B is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCR1A/B equal to TOP will result in a constant high or low output (depending on the polarity of the output set by the COM1A/B1:0 bits).

4.13.11 16-bit Timer/Counter Register Description

4.13.11.1 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 4-35 shows the COM1A/B1:0 bit functionality when the WGM13:0 bits are set to a normal or a CTC mode (non-PWM).

Table 4-35. 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 4-36 shows the COM1A/B1:0 bit functionality when the WGM13:0 bits are set to the fast PWM mode.

Table 4-36. 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 4.13.9.3 “Fast PWM Mode” on page 134 for more details.

Table 4-37 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 4-37. 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 4.13.9.4 “Phase Correct PWM Mode” on page 136 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 4-38. 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 4.13.9 “Modes of Operation” on page 133).

Table 4-38. 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

Note: 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.

4.13.11.3 Timer/Counter1 Control Register C – TCCR1C

Bit	7	6	5	4	3	2	1	0	
	FOC1A	FOC1B	–	–	–	–	–	–	TCCR1C
Read/Write	R/W	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC1A: Force Output Compare for Channel A**

- **Bit 6 – FOC1B: Force Output Compare for Channel B**

The FOC1A/FOC1B bits are only active when the WGM13:0 bits specifies a non-PWM mode. However, for ensuring compatibility with future devices, these bits must be set to zero when TCCR1A is written when operating in a PWM mode. When writing a logical one to the FOC1A/FOC1B bit, an immediate compare match is forced on the waveform generation unit. The OC1nx output is changed according to its COM1A/B1:0 and OC1nx bits setting. Note that the FOC1A/FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1A/B1:0 bits that determine the effect of the forced compare.

A FOC1A/FOC1B strobe will not generate any interrupt nor will it clear the timer in clear timer on compare match (CTC) mode using OCR1A as TOP.

The FOC1A/FOC1B bits are always read as zero.

4.13.11.4 Timer/Counter1 Control Register D – TCCR1D

Bit	7	6	5	4	3	2	1	0	
	OC1BX	OC1BW	OC1BV	OC1BU	OC1AX	OC1AW	OC1AV	OC1AU	TCCR1D
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	

- **Bit 7:4 – OC1Bi: Output Compare Pin Enable for Channel B**

The OC1Bi bits enable the output compare pins of channel B as shown in Figure 4-49 on page 132.

- **Bit 3:0 – OC1Ai: Output Compare Pin Enable for Channel A**

The OC1Ai bits enable the output compare pins of channel A as shown in Figure 4-49 on page 132.

4.13.11.5 Timer/Counter1 – TCNT1H and TCNT1L

Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
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	

The two Timer/Counter I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See Section 4.13.3 “Accessing 16-bit Registers” on page 123.

Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a compare match between TCNT1 and one of the OCR1A/B Registers.

Writing to the TCNT1 Register blocks (removes) the compare match on the following timer clock for all compare units.

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; LSB
sts          USICR,r17

lds          r16,USIDR
ret
```

4.15.3.3 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
```

The code is size optimized using only eight instructions (+ ret). The code example assumes that the DO is configured as output and USCK pin is configured as input in the DDR register. The value stored in register r16 prior to the function is called is transferred to the master device, and when the transfer is completed the data received from the master is stored back into the r16 register.

Note that the first two instructions is for initialization only and needs only to be executed once.These instructions sets three-wire mode and positive edge USI data register clock. The loop is repeated until the USI counter overflow flag is set.

4.16.6.3 LIN Enable Interrupt Register - LINENIR

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	LENERR	LENIDOK	LENTXOK	LENRXOK	LINENIR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:4 - Reserved Bits**
  - These bits are reserved for future use. For compatibility with future devices, they must be written to zero when LINENIR is written.

- **Bit 3 - LENERR: Enable Error Interrupt**
  - 0 = Error interrupt masked,
  - 1 = Error interrupt enabled.

- **Bit 2 - LENIDOK: Enable Identifier Interrupt**
  - 0 = Identifier interrupt masked,
  - 1 = Identifier interrupt enabled.

- **Bit 1 - LENTXOK: Enable Transmit Performed Interrupt**
  - 0 = Transmit performed interrupt masked,
  - 1 = Transmit performed interrupt enabled.

- **Bit 0 - LENRXOK: Enable Receive Performed Interrupt**
  - 0 = Receive performed interrupt masked,
  - 1 = Receive performed interrupt enabled.

4.16.6.4 LIN Error Register - LINERR

Bit	7	6	5	4	3	2	1	0	
	LABORT	LTOERR	LOVERR	LFERR	LSERR	LPERR	LCERR	LBERR	LINERR
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - LABORT: Abort Flag**
  - 0 = No warning,
  - 1 = LIN abort command occurred.

  This bit is cleared when LERR bit in LINSIR is cleared.

- **Bit 6 - LTOERR: Frame_Time_Out Error Flag**
  - 0 = No error,
  - 1 = Frame_Time_Out error.

  This bit is cleared when LERR bit in LINSIR is cleared.

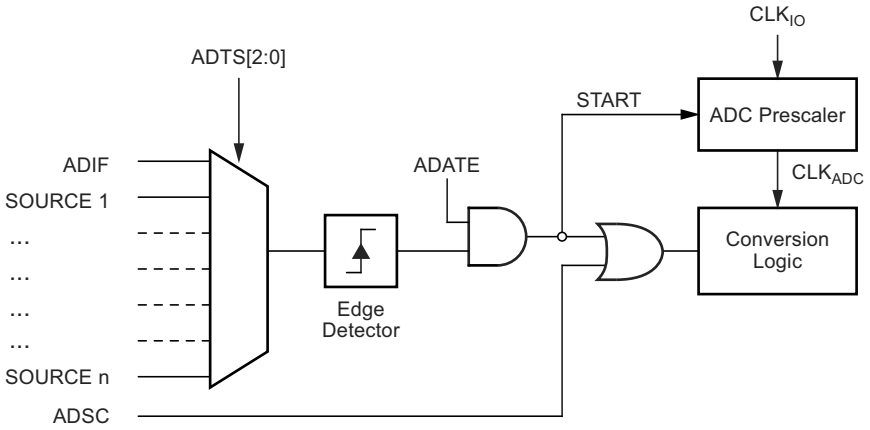
- **Bit 5 - LOVERR: Overrun Error Flag**
  - 0 = No error,
  - 1 = Overrun error.

  This bit is cleared when LERR bit in LINSIR is cleared.

- **Bit 4 - LFERR: Framing Error Flag**
  - 0 = No error,
  - 1 = Framing error.

  This bit is cleared when LERR bit in LINSIR is cleared.

Figure 4-83. ADC Auto Trigger Logic

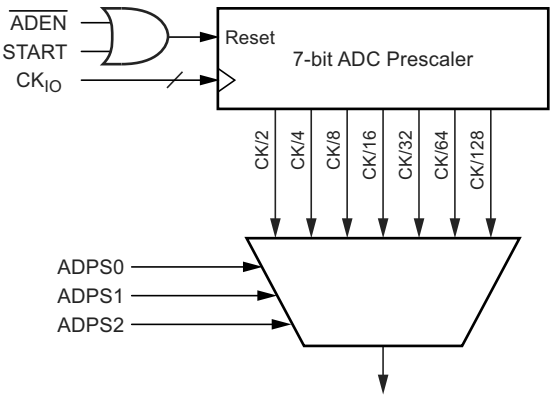


Using the ADC interrupt flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in free running mode, constantly sampling and updating the ADC data register. The first conversion must be started by writing a logical one to the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

If auto triggering is enabled, single conversions can be started by writing ADSC in ADCSRA register to one. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as one during a conversion, independently of how the conversion was started.

4.18.5 Prescaling and Conversion Timing

Figure 4-84. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz to get maximum resolution. If a lower resolution than 10bits is needed, the input clock frequency to the ADC can be higher than 200kHz to get a higher sample rate.

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz. The prescaling is set by the ADPS bits in ADCSRA register. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA register. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

When initiating a single ended conversion by setting the ADSC bit in ADCSRA register, the conversion starts at the following rising edge of the ADC clock cycle.

A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA register is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.

4.21 Flash Programming

The device provides a self-programming mechanism for downloading and uploading program code by the MCU itself. The self-programming can use any available data interface (i.e. LIN, USART, ...) and associated protocol to read code and write (program) that code into the program memory.

The program memory is updated in a page by page fashion. Before programming a page with the data stored in the temporary page buffer, the page must be erased. The temporary page buffer is filled one word at a time using SPM and the buffer can be filled either before the page erase command or between a page erase and a page write operation:

- Alternative 1, fill the buffer before a page erase
  - Fill temporary page buffer
  - Perform a page erase
  - Perform a page write
- Alternative 2, fill the buffer after page erase
  - Perform a page erase
  - Fill temporary page buffer
  - Perform a page write

If only a part of the page needs to be changed, the rest of the page must be stored (for example in the temporary page buffer) before the erase, and then be re-written. When using alternative 1, the boot loader provides an effective read-modify-write feature which allows the user software to first read the page, do the necessary changes, and then write back the modified data. If alternative 2 is used, it is not possible to read the old data while loading since the page is already erased. The temporary page buffer can be accessed in a random sequence. It is essential that the page address used in both the page erase and page write operation is addressing the same page.

4.21.1 Self-programming the Flash

4.21.1.1 Performing Page Erase by SPM

To execute page erase, set up the address in the Z-pointer, write “00000011_b” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE in the Z-register. Other bits in the Z-pointer will be ignored during this operation.

- The CPU is halted during the page erase operation.

4.21.1.2 Filling the Temporary Buffer (Page Loading)

To write an instruction word, set up the address in the Z-pointer and data in R1:R0, write “00000001_b” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The content of PCWORD in the Z-register is used to address the data in the temporary buffer. The temporary buffer will auto-erase after a page write operation or by writing the CTPB bit in SPMCSR. It is also erased after a system reset. Note that it is not possible to write more than one time to each address without erasing the temporary buffer.

If the EEPROM is written in the middle of an SPM Page Load operation, all data loaded will be lost.

4.21.1.3 Performing a Page Write

To execute page write, set up the address in the Z-pointer, write “00000101_b” to SPMCSR and execute SPM within four clock cycles after writing SPMCSR. The data in R1 and R0 is ignored. The page address must be written to PCPAGE. Other bits in the Z-pointer must be written to zero during this operation.

- The CPU is halted during the page write operation.

4.21.2 Addressing the Flash during Self-programming

The Z-pointer is used to address the SPM commands. The Z pointer consists of the Z-registers ZL and ZH in the register file. The number of bits actually used is implementation dependent.

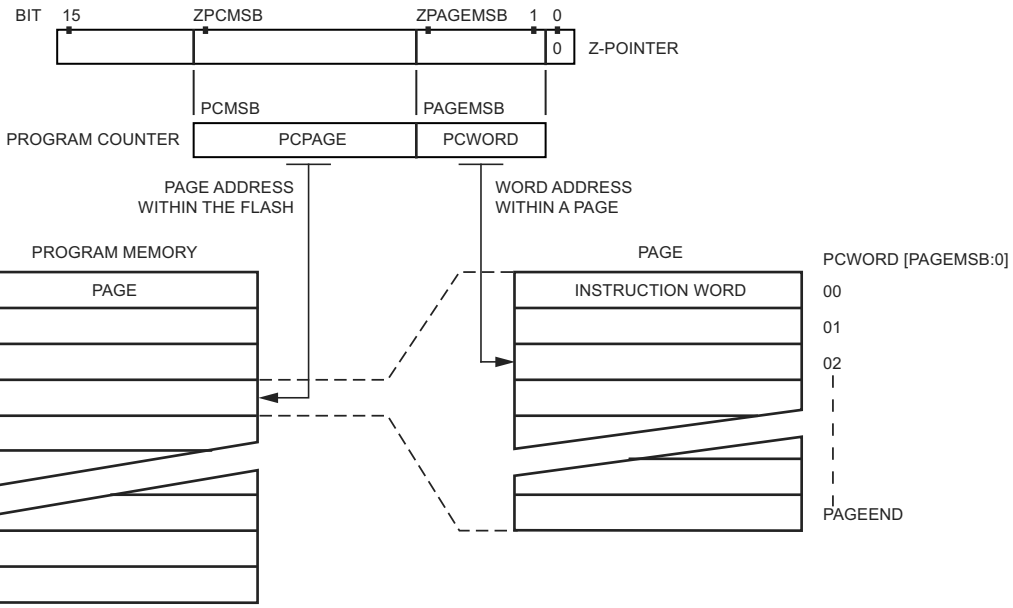
Bit	15	14	13	12	11	10	9	8	
	Z15	**Z14**	**Z13**	**Z12**	**Z11**	**Z10**	**Z9**	**Z8**	**ZH (R31)**
	Z7	**Z6**	**Z5**	**Z4**	**Z3**	**Z2**	**Z1**	**Z0**	**ZL (R30)**
Bit	7	6	5	4	3	2	1	0	

Since the flash is organized in pages (see Table 4-73 on page 225), the program counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in Figure 4-96.

Note that the page erase and page write operations are addressed independently. Therefore it is of major importance that the software addresses the same page in both the page erase and page write operation.

The LPM instruction uses the Z-pointer to store the address. Since this instruction addresses the flash byte-by-byte, also the LSB (bit Z0) of the Z-pointer is used.

Figure 4-96. Addressing the Flash During SPM(1)



Note: 1. The different variables used in Table 4-66 are listed in Table 4-73 on page 225.

4.22.2 Fuse Bit

The Atmel® ATtiny87/167 has three fuse bytes. Table 4-69, Table 4-70 & Table 4-71 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 4-69. 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 4-70. 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. See Section 4.10.3.4 “Alternate Functions of Port B” on page 97 for description of RSTDISBL fuse.
2. The SPIEN fuse is not accessible in serial programming mode.
3. See Section 4.7.3.3 “Watchdog Timer Control Register - WDTCR” on page 74 for details.
4. See Table 4-85 on page 240 for BODLEVEL fuse coding.

4.22.8.2 Serial Programming Instruction set

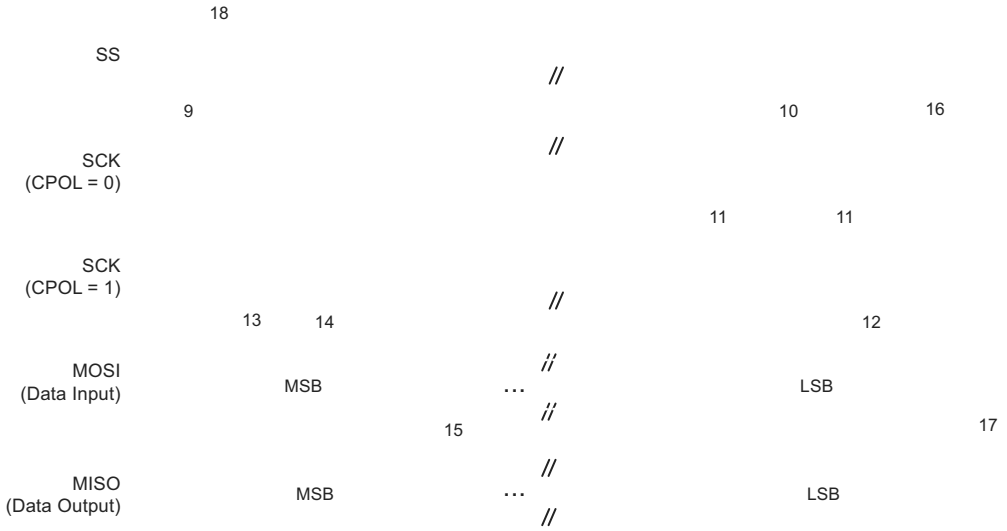
Table 4-80 on page 235 and Figure 4-105 on page 236 describes the Instruction set

Table 4-80. Serial Programming Instruction Set

Instruction/Operation	Instruction Format			
	Byte 1	Byte 2	Byte 3	Byte4
Programming enable	0xAC	0x53	0x00	0x00
Chip erase (program memory/EEPROM)	0xAC	0x80	0x00	0x00
Poll RDY/$\overline{\text{BSY}}$	0xF0	0x00	0x00	data byte out
Load Instructions				
Load extended address byte(1)	0x4D	0x00	Extended add.	0x00
Load program memory page, high byte	0x48	add. MSB	add. LSB	high data byte in
Load program memory page, low byte	0x40	add. MSB	add. LSB	low data byte in
Load EEPROM memory page (page access)	0xC1	0x00	0000 000aa $_b$	data byte in
Read Instructions				
Read program memory, high byte	0x28	add. MSB	add. LSB	high data byte out
Read program memory, low byte	0x20	add. MSB	add. LSB	low data byte out
Read EEPROM memory	0xA0	0x00	00aa aaaa	data byte out
Read lock bits	0x58	0x00	0x00	data byte out
Read signature byte	0x30	0x00	0000 000aa	data byte out
Read fuse bits	0x50	0x00	0x00	data byte out
Read fuse high bits	0x58	0x08	0x00	data byte out
Read extended fuse bits	0x50	0x08	0x00	data byte out
Read calibration byte	0x38	0x00	0x00	data byte out
Write Instructions(6)				
Write program memory page	0x4C	add. MSB	add. LSB	0x00
Write EEPROM memory	0xC0	0x00	00aa aaaa $_b$	data byte in
Write EEPROM memory page (page access)	0xC2	0x00	00aa aa00 $_b$	0x00
Write lock bits	0xAC	0xE0	0x00	data byte in
Write fuse bits	0xAC	0xA0	0x00	data byte in
Write fuse high bits	0xAC	0xA8	0x00	data byte in
Write extended fuse bits	0xAC	0xA4	0x00	data byte in

Notes:
1. Not all instructions are applicable for all parts.
2. a = address
3. Bits are programmed ‘0’, unprogrammed ‘1’.
4. To ensure future compatibility, unused fuses and lock bits should be unprogrammed (‘1’).
5. Refer to the corresponding section for fuse and lock bits, calibration and signature bytes and page size.
6. Instructions accessing program memory use a word address. This address may be random within the page range.
7. See http://www.atmel.com/avr for Application Notes regarding programming and programmers.

Figure 4-113. SPI Interface Timing Requirements (Slave Mode)



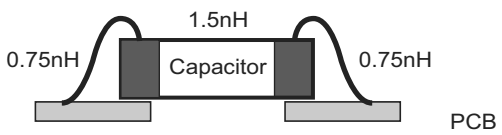
4.24 Decoupling Capacitors

The operating frequency (i.e. system clock) of the processor determines in 95% of cases the value needed for microcontroller decoupling capacitors.

The hypotheses used as first evaluation for decoupling capacitors are:

- The operating frequency (f_{op}) supplies itself the maximum peak levels of noise. The main peaks are located at f_{op} and $2 \times f_{op}$.
- An SMC capacitor connected to 2 micro-vias on a PCB has the following characteristics:
  - 1.5 nH from the connection of the capacitor to the PCB,
  - 1.5 nH from the capacitor intrinsic inductance.

Figure 4-114. Capacitor Description



According to the operating frequency of the product, the decoupling capacitances are chosen considering the frequencies to filter, f_{op} and $2 \times f_{op}$.

The relation between frequencies to cut and decoupling characteristics are defined by:

$$fop = \frac{1}{2\pi\sqrt{LC_1}} \quad \text{and} \quad 2 \times fop = \frac{1}{2\pi\sqrt{LC_2}}$$

where:

- L: the inductance equivalent to the global inductance on the V_{CC}/GND lines.
- C_1 & C_2: decoupling capacitors ($C_1 = 4 \times C_2$).