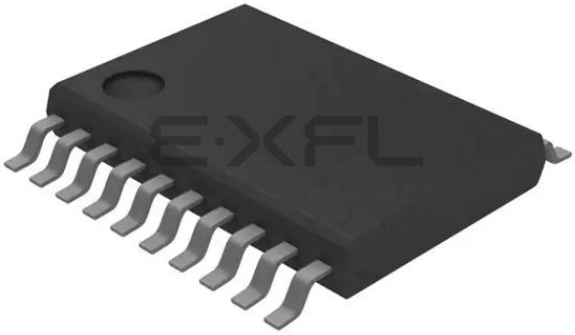


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	I²C, LINbus, SPI, UART/USART, USI
Peripherals	Brown-out Detect/Reset, POR, PWM, Temp Sensor, WDT
Number of I/O	16
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	20-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	20-TSSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny87-15xz

2.7.1 Interrupt Behavior

When an interrupt occurs, the global interrupt enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a return from interrupt instruction – RETI – is executed.

There are basically two types of interrupts. The first type is triggered by an event that sets the interrupt flag. For these interrupts, the program counter is vectored to the actual interrupt vector in order to execute the interrupt handling routine, and hardware clears the corresponding interrupt flag. Interrupt flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the interrupt flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the global interrupt enable bit is cleared, the corresponding interrupt flag(s) will be set and remembered until the global interrupt enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have interrupt flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.

When the AVR® exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the status register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

Assembly Code Example

```
in   r16, SREG     ; store SREG value
cli    ; disable interrupts during timed sequence
sbi  EECR, EEMPE   ; start EEPROM write
sbi  EECR, EEPE
out  SREG, r16     ; restore SREG value (I-bit)
```

C Code Example

```
char cSREG;
cSREG = SREG; /* store SREG value */
/* disable interrupts during timed sequence */
_CLI();
EECR |= (1<<EEMPE); /* start EEPROM write */
EECR |= (1<<EEPE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

Assembly Code Example

```
sei      ; set Global Interrupt Enable
sleep    ; enter sleep, waiting for interrupt
; note: will enter sleep before any pending
; interrupt(s)
```

C Code Example

```
_SEI();   /* set Global Interrupt Enable */
_SLEEP(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

3.4 I/O Memory

The I/O space definition of the Atmel® ATtiny87/167 is shown in Section "" on page 243.

All ATtiny87/167 I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The Atmel ATtiny87/167 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in opcode for the IN and OUT instructions. For the extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

The I/O and peripherals control registers are explained in later sections.

3.4.1 General Purpose I/O Registers

The Atmel ATtiny87/167 contains three general purpose I/O registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and status flags.

The general purpose I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

3.5 Register Description

3.5.1 EEARH and EEARL – EEPROM Address Register

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	-	-	**EEAR8**	EEARH
	EEAR7	**EEAR6**	**EEAR5**	**EEAR4**	**EEAR3**	**EEAR2**	**EEAR1**	**EEAR0**	EEARL
Bit	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
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	X	
Initial Value	X	X	X	X	X	X	X	X	

- **Bit 7:1 – Reserved Bits**

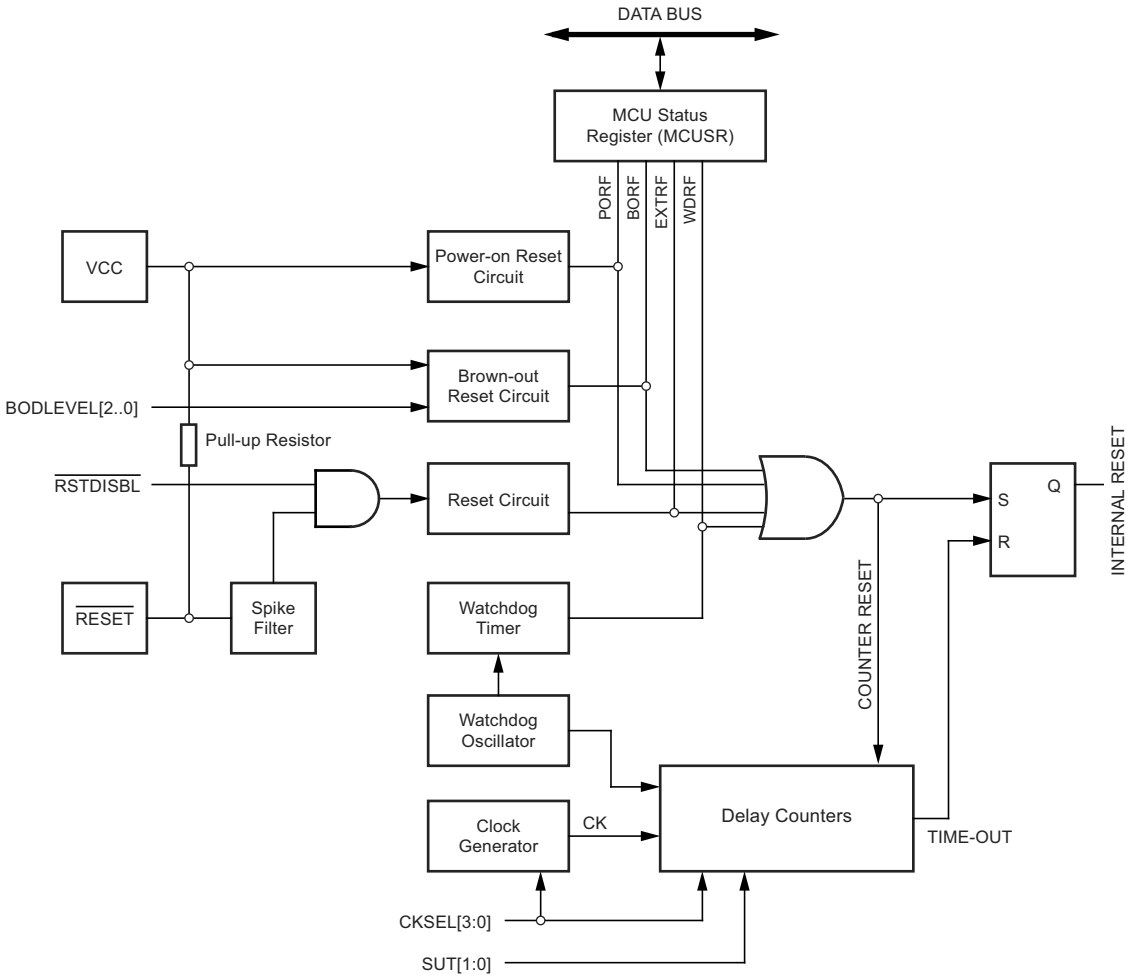
These bits are reserved for future use and will always read as 0 in ATtiny87/167.

- **Bits 8:0 – EEAR8:0: EEPROM Address**

The EEPROM address registers – EEARH and EEARL – specifies the high EEPROM address in the EEPROM space (see "E2 size" in Table 3-1 on page 16). The EEPROM data bytes are addressed linearly between 0 and "E2 size". The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

Note: For information only - ATtiny47: EEAR8 exists as register bit but it is not used for addressing.

Figure 6-1. Reset Circuit

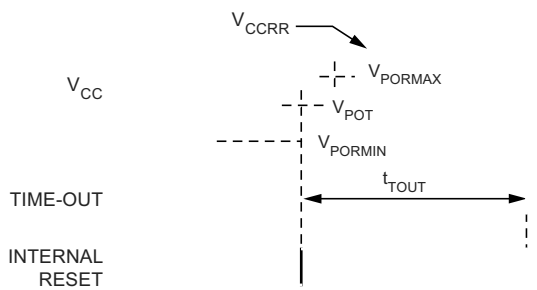


6.1.3 Power-on Reset

A power-on reset (POR) pulse is generated by an on-chip detection circuit. The detection level is defined in Table 22-4 on page 225. The POR is activated whenever Vcc is below the detection level. The POR circuit can be used to trigger the start-up reset, as well as to detect a failure in supply voltage.

A power-on reset (POR) circuit ensures that the device is reset from power-on. Reaching the power-on reset threshold voltage invokes the delay counter, which determines how long the device is kept in RESET after Vcc rise. The RESET signal is activated again, without any delay, when Vcc decreases below the detection level.

Figure 6-2. MCU Start-up, $\overline{\text{RESET}}$ Tied to Vcc



Bit 0 – INT0: External Interrupt Request 0 Enable

When the INT0 bit is set (one) and the I-bit in the status register (SREG) is set (one), the external pin interrupt is enabled. The interrupt sense control0 bits 1/0 (ISC01 and ISC00) in the external interrupt control register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of external interrupt request 0 is executed from the INT0 interrupt vector.

8.3.3 External Interrupt Flag Register – EIFR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	INTF1	INTF0	EIFR
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 – INTF1: External Interrupt Flag 1**

When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in EIMSK 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. This flag is always cleared when INT1 is configured as a level interrupt.

- **Bit 0 – INTF0: External Interrupt Flag 0**

When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in EIMSK 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. This flag is always cleared when INT0 is configured as a level interrupt.

8.3.4 Pin Change Interrupt Control Register – PCICR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	PCIE1	PCIE0	PCICR
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 - PCIE1: Pin Change Interrupt Enable 1**

When the PCIE1 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 1 is enabled. Any change on any enabled PCINT15..8 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI1 interrupt vector. PCINT15..8 pins are enabled individually by the PCMSK1 register.

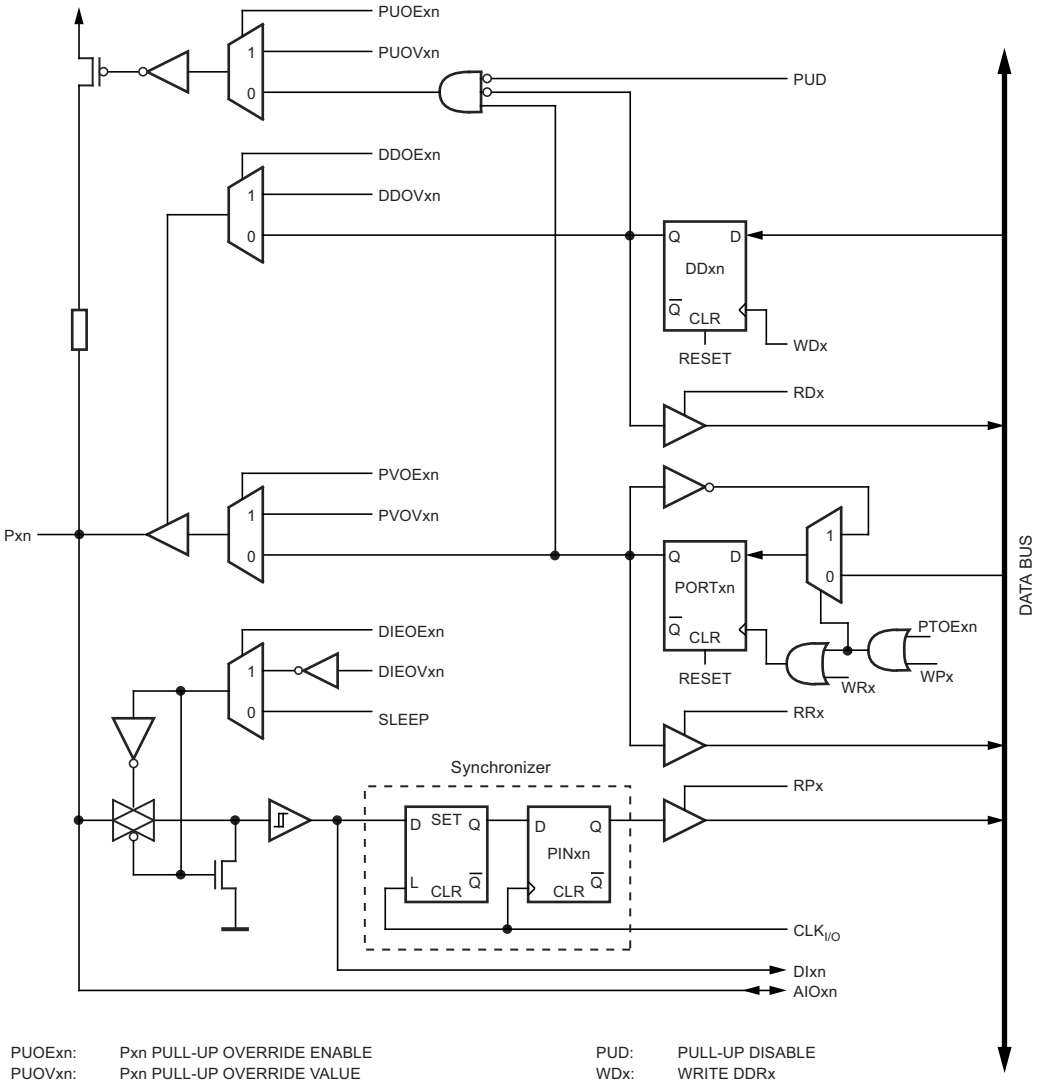
- **Bit 0 - PCIE0: Pin Change Interrupt Enable 0**

When the PCIE0 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 0 is enabled. Any change on any enabled PCINT7..0 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI0 Interrupt Vector. PCINT7..0 pins are enabled individually by the PCMSK0 register.

9.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 9-6 shows how the port pin control signals from the simplified Figure 9-2 on page 65 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR microcontroller family.

Figure 9-6. Alternate Port Functions(1)



PUOExn:	Pxn PULL-UP OVERRIDE ENABLE	PUD:	PULL-UP DISABLE
PUOVxn:	Pxn PULL-UP OVERRIDE VALUE	WDx:	WRITE DDRx
DDOExn:	Pxn DATA DIRECTION OVERRIDE ENABLE	RDx:	READ DDRx
DDOVxn:	Pxn DATA DIRECTION OVERRIDE VALUE	RRx:	READ PORTx REGISTER
PVOExn:	Pxn PORT VALUE OVERRIDE ENABLE	WRx:	WRITE PORTx
PVOVxn:	Pxn PORT VALUE OVERRIDE VALUE	RPx:	READ PORTx PIN
DIEOExn:	Pxn DIGITAL INPUT ENABLE OVERRIDE ENABLE	WPx:	WRITE PINx
DIEOVxn:	Pxn DIGITAL INPUT ENABLE OVERRIDE VALUE	$CLK_{I/O}$:	I/O CLOCK
SLEEP:	SLEEP CONTROL	DIxn:	DIGITAL INPUT PIN n ON PORTx
PTOExn:	Pxn, PORT TOGGLE OVERRIDE ENABLE	AIOxn:	ANALOG INPUT/OUTPUT PIN n ON PORTx

Note: 1. WRx, WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports. All other signals are unique for each pin.

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

PCINT10: pin change interrupt, source 10.

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

USCK: three-wire mode USI clock input.

SCL: two-wire mode USI clock input.

- **PCINT9/OC1BU/DO – Port B, Bit 1**

PCINT9: pin change interrupt, source 9.

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

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

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

IPCINT8: pin change interrupt, source 8.

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

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

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

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

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

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

10.6.2 Compare Output Mode and Waveform Generation

The waveform generator uses the COM0A1:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COM0A1:0 = 0 tells the waveform generator that no action on the OC0A register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 10-1 on page 96. For fast PWM mode, refer to Table 10-2 on page 96, and for phase correct PWM refer to Table 10-3 on page 96.

A change of the COM0A1:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC0A strobe bits.

10.7 Modes of Operation

The mode of operation, i.e., the behavior of the timer/counter and the output compare pins, is defined by the combination of the waveform generation mode (WGM01:0) and compare output mode (COM0A1:0) bits. The compare output mode bits do not affect the counting sequence, while the waveform generation mode bits do. The COM0A1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM0A1:0 bits control whether the output should be set, cleared, or toggled at a compare match (Section 10.6 “Compare Match Output Unit” on page 87).

For detailed timing information refer to Section 10.8 “Timer/Counter Timing Diagrams” on page 92.

10.7.1 Normal Mode

The simplest mode of operation is the normal mode (WGM01:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the timer/counter overflow flag (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero. The TOV0 flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 flag, the timer resolution can be increased by software. There are no special cases to consider in the normal mode, a new counter value can be written anytime.

The output compare unit can be used to generate interrupts at some given time. Using the output compare to generate waveforms in normal mode is not recommended, since this will occupy too much of the CPU time.

10.7.2 Clear Timer on Compare Match (CTC) Mode

In clear timer on compare or CTC mode (WGM01:0 = 2), the OCR0A register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches the OCR0A. The OCR0A defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 10-5. The counter value (TCNT0) increases until a compare match occurs between TCNT0 and OCR0A, and then counter (TCNT0) is cleared.

Figure 10-5. CTC Mode, Timing Diagram

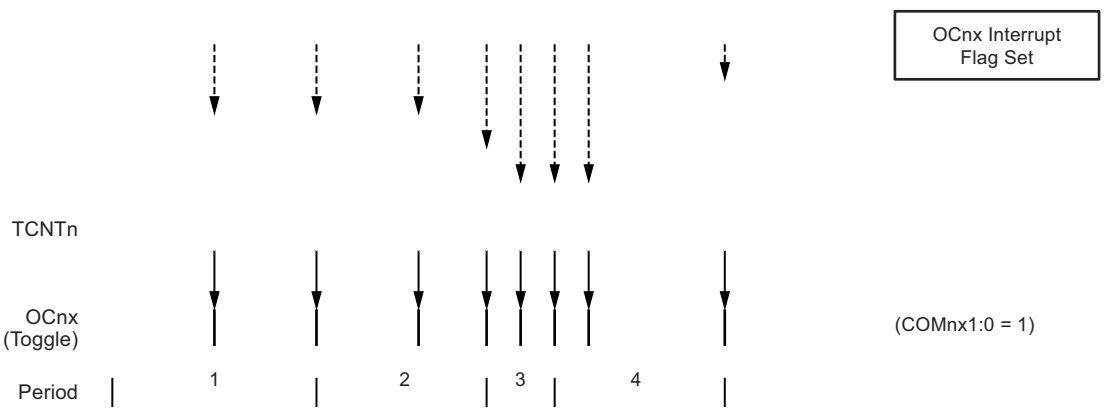
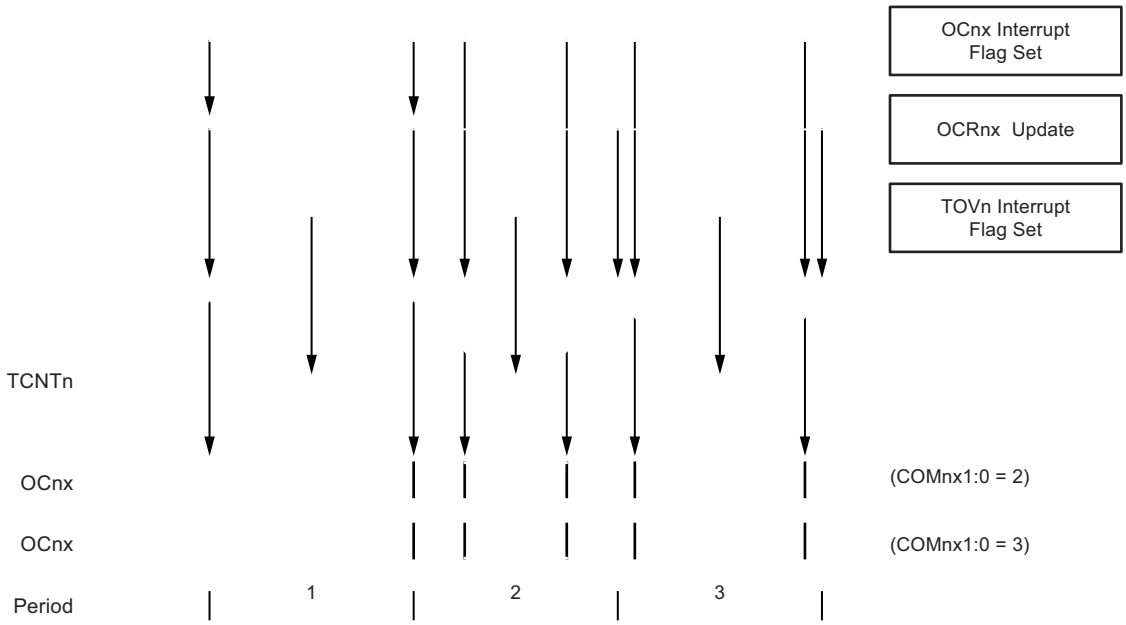


Figure 10-7. Phase Correct PWM Mode, Timing Diagram



The timer/counter overflow flag (TOV0) is set each time the counter reaches BOTTOM. The interrupt flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC0A pin. Setting the COM0A1:0 bits to two will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM0A1:0 to three (See Table 10-3 on page 96). The actual OC0A value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC0A Register at the compare match between OCR0A and TCNT0 when the counter increments, and setting (or clearing) the OC0A register at compare match between OCR0A and TCNT0 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

The extreme values for the OCR0A register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR0A is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

12.9.1 Normal Mode

The simplest mode of operation is the normal mode (WGM13:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the BOTTOM (0x0000). In normal operation the timer/counter overflow flag (TOV1) will be set in the same timer clock cycle as the TCNT1 becomes zero. The TOV1 flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV1 flag, the timer resolution can be increased by software. There are no special cases to consider in the normal mode, a new counter value can be written anytime.

The input capture unit is easy to use in normal mode. However, observe that the maximum interval between the external events must not exceed the resolution of the counter. If the interval between events are too long, the timer overflow interrupt or the prescaler must be used to extend the resolution for the capture unit.

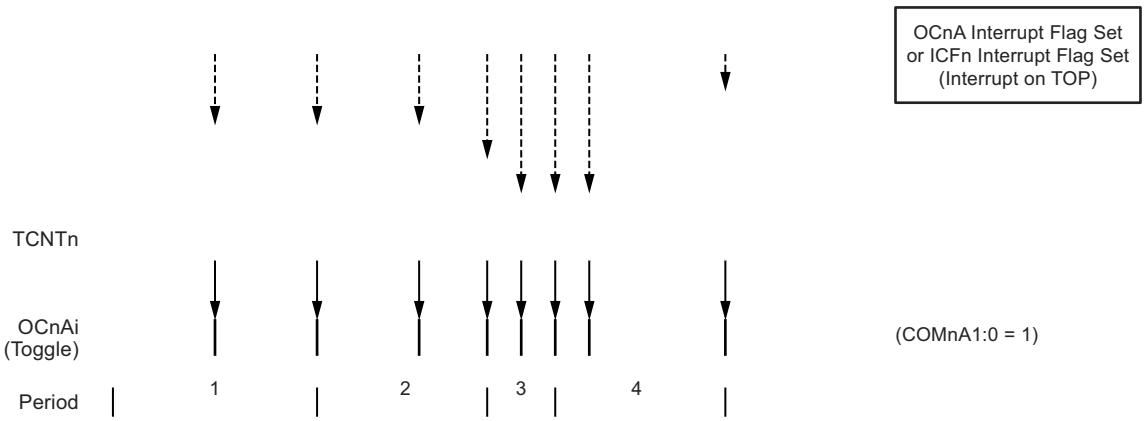
The output compare units can be used to generate interrupts at some given time. Using the output compare to generate waveforms in normal mode is not recommended, since this will occupy too much of the CPU time.

12.9.2 Clear Timer on Compare Match (CTC) Mode

In clear timer on compare or CTC mode (WGM13:0 = 4 or 12), the OCR1A or ICR1 register are used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT1) matches either the OCR1A (WGM13:0 = 4) or the ICR1 (WGM13:0 = 12). The OCR1A or ICR1 define the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 12-7. The counter value (TCNT1) increases until a compare match occurs with either OCR1A or ICR1, and then counter (TCNT1) is cleared.

Figure 12-7. CTC Mode, Timing Diagram



An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCF1A or ICF1 flag according to the register used to define the TOP value. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR1A or ICR1 is lower than the current value of TCNT1, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. In many cases this feature is not desirable. An alternative will then be to use the fast PWM mode using OCR1A for defining TOP (WGM13:0 = 15) since the OCR1A then will be double buffered.

For generating a waveform output in CTC mode, the OC1A output can be set to toggle its logical level on each compare match by setting the compare output mode bits to toggle mode (COM1A1:0 = 1). The OC1A value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OC1A = 1) and OC1Ai is set. The waveform generated will have a maximum frequency of $f_{OC}1_A = f_{clk\_I/O}/2$ when OCR1A is set to zero (0x0000). The waveform frequency is defined by the following equation:

$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

Table 14-1. Relations between USIWM1..0 and the USI Operation

USIWM1	USIWM0	Description
0	0	Outputs, clock hold, and start detector disabled. Port pins operates as normal.
0	1	Three-wire mode. Uses DO, DI, and USCK pins. The *data output* (DO) pin overrides the corresponding bit in the PORT register in this mode. However, the corresponding DDR bit still controls the data direction. When the port pin is set as input the pins pull-up is controlled by the PORT bit. The *data input* (DI) and *serial clock* (USCK) pins do not affect the normal port operation. When operating as master, clock pulses are software generated by toggling the PORT register, while the data direction is set to output. The USITC bit in the USICR register can be used for this purpose.
1	0	Two-wire mode. Uses SDA (DI) and SCL (USCK) pins(1). The *serial data* (SDA) and the *serial clock* (SCL) pins are bi-directional and uses open-collector output drives. The output drivers are enabled by setting the corresponding bit for SDA and SCL in the DDR register. When the output driver is enabled for the SDA pin, the output driver will force the line SDA low if the output of the USI data register or the corresponding bit in the PORT register is zero. Otherwise the SDA line will not be driven (i.e., it is released). When the SCL pin output driver is enabled the SCL line will be forced low if the corresponding bit in the PORT register is zero, or by the start detector. Otherwise the SCL line will not be driven. The SCL line is held low when a start detector detects a start condition and the output is enabled. Clearing the start condition flag (USISIF) releases the line. The SDA and SCL pin inputs is not affected by enabling this mode. Pull-ups on the SDA and SCL port pin are disabled in Two-wire mode.
1	1	Two-wire mode. Uses SDA and SCL pins. Same operation as for the two-wire mode described above, except that the SCL line is also held low when a counter overflow occurs, and is held low until the counter overflow flag (USIOIF) is cleared.

Note: 1. The DI and USCK pins are renamed to *serial data* (SDA) and *serial clock* (SCL) respectively to avoid confusion between the modes of operation.

- **Bit 3:2 – USICS1:0: Clock Source Select**

These bits set the clock source for the USI data register and counter. The data output latch ensures that the output is changed at the opposite edge of the sampling of the data input (DI/SDA) when using external clock source (USCK/SCL). When software strobe or timer/counter0 compare match clock option is selected, the output latch is transparent and therefore the output is changed immediately. Clearing the USICS1:0 bits enables software strobe option. When using this option, writing a one to the USICLK bit clocks both the USI data register and the counter. For external clock source (USICS1 = 1), the USICLK bit is no longer used as a strobe, but selects between external clocking and software clocking by the USITC strobe bit.

Table 14-2 on page 148 shows the relationship between the USICS1..0 and USICLK setting and clock source used for the USI data register and the 4-bit counter.

15.5.4 Configuration

Depending on the mode (LIN or UART), LCONF[1..0] bits of the LINCR register set the controller in the following configuration (see Table 15-3).

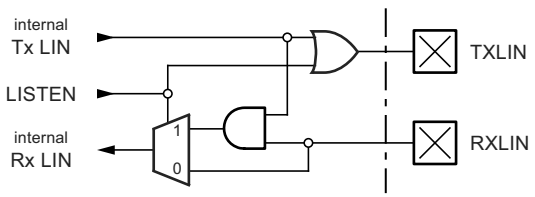
Table 15-3. Configuration Table versus Mode

Mode	LCONF[1..0]	Configuration
LIN	00_b	LIN standard configuration (default)
	01_b	No CRC field detection or transmission
	10_b	Frame_time_out disable
	11_b	Listening mode
UART	00_b	8-bit data, no parity and 1 stop-bit
	01_b	8-bit data, even parity and 1 stop-bit
	10_b	8-bit data, odd parity and 1 stop-bit
	11_b	Listening mode, 8-bit data, no parity and 1 stop-bit

The LIN configuration is independent of the programmed LIN protocol.

The listening mode connects the internal Tx LIN and the internal Rx LIN together. In this mode, the TXLIN output pin is disabled and the RXLIN input pin is always enabled. The same scheme is available in UART mode.

Figure 15-6. Listening Mode

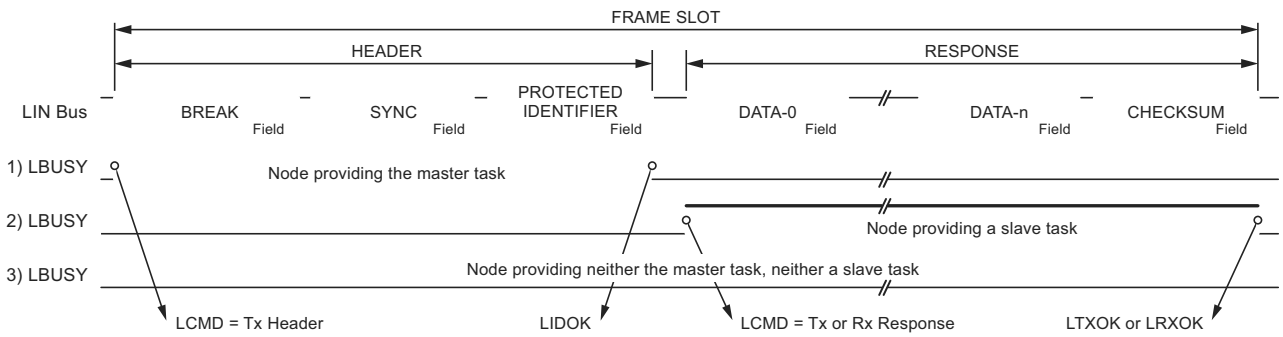


15.5.5 Busy Signal

LBUSY bit flag in LINSIR register is the image of the BUSY signal. It is set and cleared by hardware. It signals that the controller is busy with LIN or UART communication.

15.5.5.1 Busy Signal in LIN Mode

Figure 15-7. Busy Signal in LIN Mode



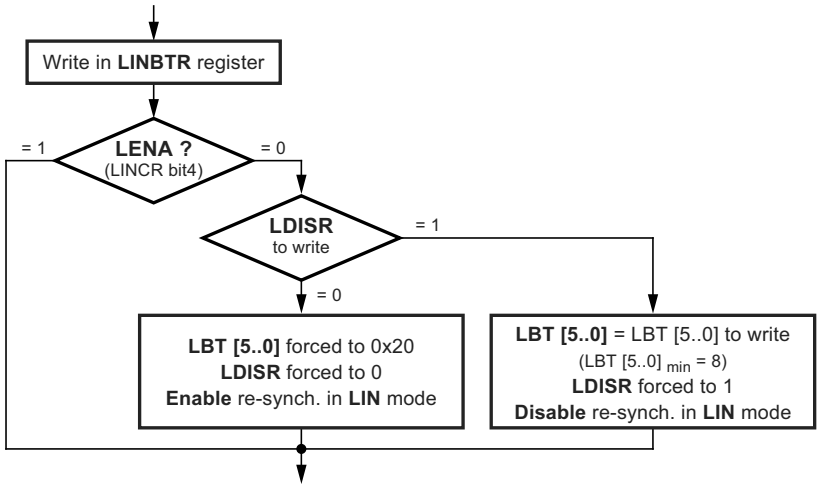
15.5.6.3 Handling LBT[5..0]

LDISR bit of LINBTR register is used to:

- Disable the re-synchronization (for instance in the case of LIN MASTER node),
- To enable the setting of LBT[5..0] (to manually adjust the baud rate especially in the case of UART mode). A minimum of 8 is required for LBT[5..0] due to the sampling operation.

Note that the LENA bit of LINCR register is important for this handling (see Figure 15-8).

Figure 15-8. Handling LBT[5..0]



15.5.7 Data Length

Section 15.4.6 “LIN Commands” on page 154 describes how to set or how are automatically set the LRXDL[3..0] or LTXDL[3..0] fields of LINDLR register before receiving or transmitting a response.

In the case of Tx response the LRXDL[3..0] will be used by the hardware to count the number of bytes already successfully sent.

In the case of Rx response the LTXDL[3..0] will be used by the hardware to count the number of bytes already successfully received.

If an error occurs, this information is useful to the programmer to recover the LIN messages.

15.5.7.1 Data Length in LIN 2.1

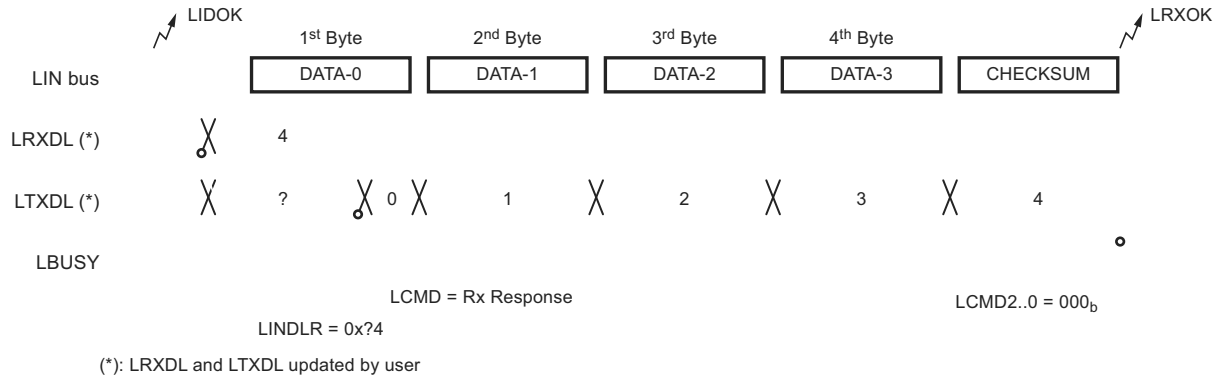
- If LTXDL[3..0]=0 only the CHECKSUM will be sent,
- If LRXDL[3..0]=0 the first byte received will be interpreted as the CHECKSUM,
- If LTXDL[3..0] or LRXDL[3..0] >8, values will be forced to 8 after the command setting and before sending or receiving of the first byte.

15.5.7.2 Data Length in LIN 1.3

- LRXDL and LTXDL fields are both hardware updated before setting LIDOK by decoding the data length code contained in the received PROTECTED IDENTIFIER (LRXDL = LTXDL).
- Via the above mechanism, a length of 0 or >8 is not possible.

15.5.7.3 Data Length in Rx Response

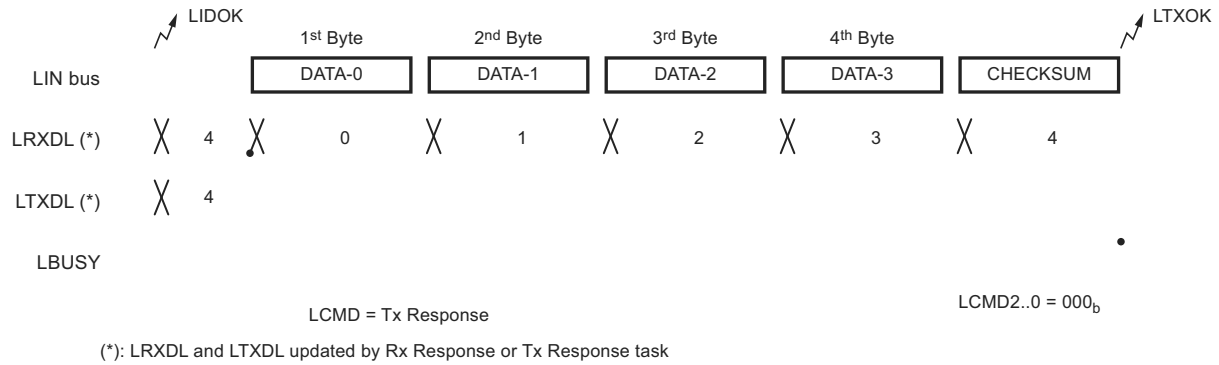
Figure 15-9. LIN2.1 - Rx Response - No error



- The user initializes LRXDL field before setting the Rx response command,
- After setting the Rx response command, LTXDL is reset by hardware,
- LRXDL field will remain unchanged during Rx (during busy signal),
- LTXDL field will count the number of received bytes (during busy signal),
- If an error occurs, Rx stops, the corresponding error flag is set and LTXDL will give the number of received bytes without error,
- If no error occurs, LRXOK is set after the reception of the CHECKSUM, LRXDL will be unchanged (and LTXDL = LRXDL).

15.5.7.4 Data Length in Tx Response

Figure 15-10. LIN1.3 - Tx Response - No Error



- The user initializes LTXDL field before setting the Tx response command,
- After setting the Tx response command, LRXDL is reset by hardware,
- LTXDL will remain unchanged during Tx (during busy signal),
- LRXDL will count the number of transmitted bytes (during busy signal),
- If an error occurs, Tx stops, the corresponding error flag is set and LRXDL will give the number of transmitted bytes without error,
- If no error occurs, LTXOK is set after the transmission of the CHECKSUM, LTXDL will be unchanged (and LRXDL = LTXDL).

- **Bits 5, 4 – ACIR1, ACIR0: Analog Comparator Internal Voltage Reference Select**

When ACIRS bit is set in ADCSRA register, these bits select a voltage reference for the negative input to the analog comparator, see Table 18-3 on page 197.

18.1.2 ACSR – Analog Comparator Control and Status Register

Bit	7	6	5	4	3	2	1	0	
	ACD	**ACIRS**	**ACO**	**ACI**	**ACIE**	**ACIC**	**ACIS1**	**ACIS0**	**ACSR**
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

- **Bit 7 – ACD: Analog Comparator Disable**

When this bit is written logic one, the power to the analog comparator is switched off. This bit can be set at any time to turn off the analog comparator. This will reduce power consumption in active and idle mode. When changing the ACD bit, the analog comparator interrupt must be disabled by clearing the ACIE bit of ACSR register. Otherwise an interrupt can occur when the bit is changed.

- **Bit 6 – ACIRS: Analog Comparator Internal Reference Select**

When this bit is set an internal reference voltage replaces the negative input to the analog comparator (c.f. Table 18-3 on page 197). If ACIRS is cleared, AIN0 is applied to the negative input to the analog comparator.

- **Bit 5 – ACO: Analog Comparator Output**

The output of the analog comparator is synchronized and then directly connected to ACO. The synchronization introduces a delay of 1 - 2 clock cycles.

- **Bit 4 – ACI: Analog Comparator Interrupt Flag**

This bit is set by hardware when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The analog comparator interrupt routine is executed if the ACIE bit is set and the I-bit in SREG is set. ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

- **Bit 3 – ACIE: Analog Comparator Interrupt Enable**

When the ACIE bit is written logic one and the I-bit in the status register is set, the analog comparator interrupt is activated. When written logic zero, the interrupt is disabled.

- **Bit 2 – ACIC: Analog Comparator Input Capture Enable**

When written logic one, this bit enables the input capture function in Timer/counter1 to be triggered by the analog comparator. The comparator output is in this case directly connected to the input capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/counter1 Input capture interrupt. When written logic zero, no connection between the analog comparator and the input capture function exists. To make the comparator trigger the Timer/counter1 input capture interrupt, the ICIE1 bit in the timer interrupt mask register (TIMSK1) must be set.

- **Bits 1, 0 – ACIS1, ACIS0: Analog Comparator Interrupt Mode Select**

These bits determine which comparator events that trigger the analog comparator interrupt. The different settings are shown in Table 18-1.

Table 18-1. ACIS1 / ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator interrupt on output toggle.
0	1	Reserved
1	0	Comparator interrupt on falling output edge.
1	1	Comparator interrupt on rising output edge.

Note: When changing the ACIS1/ACIS0 bits, the Analog Comparator Interrupt must be disabled by clearing its interrupt enable bit in the ACSR register. Otherwise an interrupt can occur when the bits are changed.

20.2.2 EEPROM Write Prevents Writing to SPMCSR

Note that an EEPROM write operation will block all software programming to flash. Reading the fuses and lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EEPE) in the EECR register and verifies that the bit is cleared before writing to the SPMCSR register.

20.2.3 Reading the Fuse and Lock Bits from Software

It is possible to read both the fuse and lock bits from software. To read the lock bits, load the Z-pointer with 0x0001 and set the RFLB and SPMEN bits in SPMCSR. When an LPM instruction is executed within three CPU cycles after the RFLB and SPMEN bits are set in SPMCSR, the value of the lock bits will be loaded in the destination register. The RFLB and SPMEN bits will auto-clear upon completion of reading the lock bits or if no LPM instruction is executed within three CPU cycles or no SPM instruction is executed within four CPU cycles. When RFLB and SPMEN are cleared, LPM will work as described in the instruction set manual.

Bit	7	6	5	4	3	2	1	0
Rd (Z=0x0001)	**–**	**–**	**–**	**–**	**–**	**–**	**LB2**	**LB1**

The algorithm for reading the fuse low byte is similar to the one described above for reading the lock bits. To read the fuse low byte, load the Z-pointer with 0x0000 and set the RFLB and SPMEN bits in SPMCSR. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the fuse low byte (FLB) will be loaded in the destination register as shown below. See Table 21-5 on page 209 for a detailed description and mapping of the fuse low byte.

Bit	7	6	5	4	3	2	1	0
Rd (Z=0x0000)	**FLB7**	**FLB6**	**FLB5**	**FLB4**	**FLB3**	**FLB2**	**FLB1**	**FLB0**

Similarly, when reading the fuse high byte (FHB), load 0x0003 in the Z-pointer. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the fuse high byte will be loaded in the destination register as shown below. See Table 21-4 on page 208 for detailed description and mapping of the fuse high byte.

Bit	7	6	5	4	3	2	1	0
Rd (Z=0x0003)	**FHB7**	**FHB6**	**FHB5**	**FHB4**	**FHB3**	**FHB2**	**FHB1**	**FHB0**

Similarly, when reading the extended fuse byte (EFB), load 0x0002 in the Z-pointer. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the extended fuse byte will be loaded in the destination register as shown below. See Table 21-3 on page 208 for detailed description and mapping of the extended fuse byte.

Bit	7	6	5	4	3	2	1	0
Rd (Z=0x0002)	**–**	**–**	**–**	**–**	**–**	**–**	**–**	**EFB0**

Fuse and lock bits that are programmed, will be read as zero. Fuse and lock bits that are unprogrammed, will be read as one.

20.2.7 Simple Assembly Code Example for a Boot Loader

Note that the RWWSB bit will always be read as zero in Atmel® ATtiny87/167. Nevertheless, it is recommended to check this bit as shown in the code example, to ensure compatibility with devices supporting read-while-write.

```
;- The routine writes one page of data from RAM to Flash
;  the first data location in RAM is pointed to by the Y-pointer
;  the first data location in Flash is pointed to by the Z-pointer
;- Error handling is not included
;- Registers used: r0, r1, temp1 (r16), temp2 (r17), looplo (r24),
;                  loophi (r25), spmcsrval (r20)
; -Storing and restoring of registers is not included in the routine
;  register usage can be optimized at the expense of code size

.equ PAGESIZEB = PAGESIZE*2   ; AGESIZEB is page size in BYTES, not words
.org SMALLBOOTSTART

Write_page:
  ; Page Erase
  ldi    spmcsrval, (1<<PGERS) | (1<<SELFPGEN)
  rcall  Do_spm

  ; Clear temporary page buffer
  ldi    spmcsrval, (1<<CPTB) | (1<<SELFPGEN)
  rcall  Do_spm

  ; Transfer data from RAM to Flash temporary page buffer
  ldi    looplo, low(PAGESIZEB)   ; init loop variable
  ldi    loophi, high(PAGESIZEB)  ; not required for PAGESIZEB<=256

Wrloop:
  ld     r0, Y+
  ld     r1, Y+
  ldi    spmcsrval, (1<<SELFPGEN)
  rcall  Do_spm
  adiw   ZH:ZL, 2
  sbiw   loophi:looplo, 2         ; use subi for PAGESIZEB<=256
  brne   Wrloop

  ; Execute Page Write
  subi   ZL, low(PAGESIZEB)       ; restore pointer
  sbci   ZH, high(PAGESIZEB)      ; not required for PAGESIZEB<=256
  ldi    spmcsrval, (1<<PGWRT) | (1<<SELFPGEN)
  rcall  Do_spm

  ; Clear temporary page buffer
  ldi    spmcsrval, (1<<CPTB) | (1<<SELFPGEN)
  rcall  Do_spm

  ; Read back and check, optional
  ldi    looplo, low(PAGESIZEB)   ; init loop variable
  ldi    loophi, high(PAGESIZEB)  ; not required for PAGESIZEB<=256
  subi   YL, low(PAGESIZEB)       ; restore pointer
  sbci   YH, high(PAGESIZEB)
```

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.

22.4.3 External Clock Drive

Table 22-2. External Clock Drive

Parameter	Symbol	Vcc = 2.7 - 5.5V		Vcc = 4.5 - 5.5V		Units
		Min.	Max.	Min.	Max.	
Oscillator frequency	$1/t_{CLCL}$	0	8	0	16	MHz
Clock period	t_{CLCL}	125		62.5		ns
High time	t_{CHCX}	50		25		ns
Low time	t_{CLCX}	50		25		ns
Rise time	t_{CLCH}		1.6		0.5	ms
Fall time	t_{CHCL}		1.6		0.5	ms
Change in period from one clock cycle to the next	Δt_{CLCL}		2		2	%

22.5 RESET Characteristics

Table 22-3. External Reset Characteristics

Parameter	Condition	Symbol	Min	Typ	Max	Units
$\overline{\text{RESET}}$ pin threshold voltage	V_{CC} = 5V	V_{RST}	0.1 Vcc		0.9 Vcc	V
Minimum pulse width on $\overline{\text{RESET}}$ pin	V_{CC} = 5V	t_{RST}			2.5	µs
Bandgap reference voltage	V_{CC} = 2.7V, T_A = 25°C	V_{BG}	1.0	1.1	1.2	V
Bandgap reference start-up time	V_{CC} = 2.7V, T_A = 25°C	t_{BG}		40	70	µs
Bandgap reference current consumption	V_{CC} = 2.7V, T_A = 25°C	I_{BG}		15		µA

Table 22-4. Power On Reset Characteristics

Parameter	Symbol	Min	Typ	Max	Units
Power-on reset threshold voltage (rising)	VPOT		1.4		V
Power-on reset threshold voltage (falling)(1)		1.0	1.3	1.6	V
VCC max. start voltage to ensure internal power-on reset signal	VPORMAX			0.4	V
VCC Min. start voltage to ensure internal power-on reset signal	VPORMIN	–0.1			V
VCC rise rate to ensure power-on reset	VCCRR	0.01			V/ms
$\overline{\text{RESET}}$ pin threshold voltage	VRST	0.1 Vcc		0.9 Vcc	V

Note: 1. Before rising, the supply has to be between VPORMIN and VPORMAX to ensure a reset.

Table 22-10. ADC Characteristics, Differential Channels (–40°C/+125°C) (Continued)

Parameter	Condition	Symbol	Min	Typ	Max	Units
Differential non linearity	Gain = 8x, BIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz	DNL		0.3	0.8	LSB
	Gain = 20x, BIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz			0.3	0.8	
	Gain = 8x, UNIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz			0.4	0.8	
	Gain = 20x, UNIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz			0.6	1.6	
Gain error	Gain = 8x, BIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz		-3.0	1.0	3.0	LSB
	Gain = 20x, BIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz		–4.0	1.5	4.0	
	Gain = 8x, UNIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz		–5.0	–2.5	0.0	
	Gain = 20x, UNIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz		–4.0	–0.5	4.0	
Offset error	Gain = 8x or 20x, BIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz		–2.0	0.5	2.0	LSB
	Gain = 8x or 20x, UNIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz		–2.0	0.5	2.0	
Reference voltage		V_{REF}	2.56		AVCC - 0.5	V
Input differential voltage		VDIFF	–VREF/Gain		+VREF/Gain	V
Analog supply voltage		AVcc	Vcc – 0.3		Vcc + 0.3	V
Input voltage	Differential conversion	VIN	0		AVcc	V
ADC conversion output			–511		+511	LSB
Input bandwidth	Differential conversion			4		kHz
Internal voltage reference		VINT	2.4	2.56	2.7	V
Reference input resistance		RREF		32		kΩ
Analog input resistance		RAIN		100		MΩ

22.9 Parallel Programming Characteristics

Figure 22-3. Parallel Programming Timing, Including some General Timing Requirements

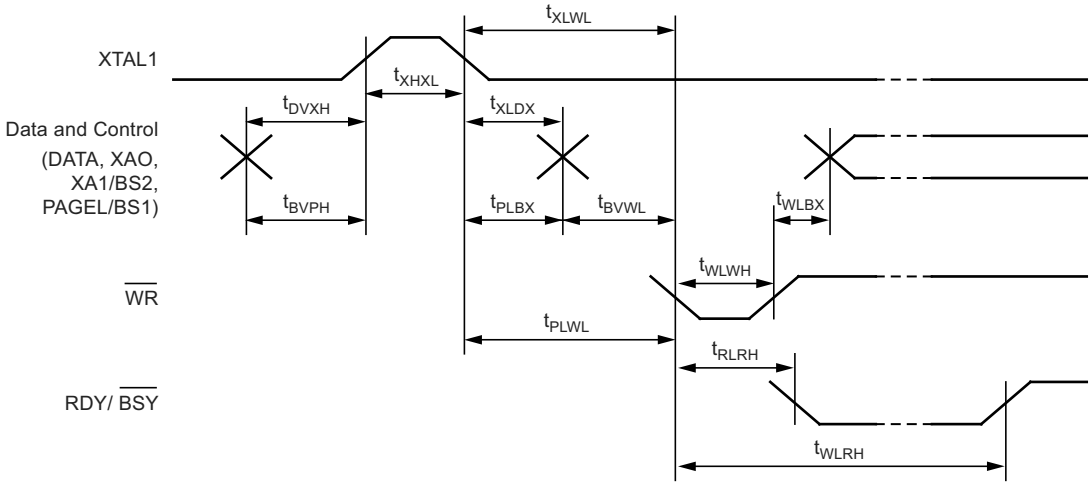
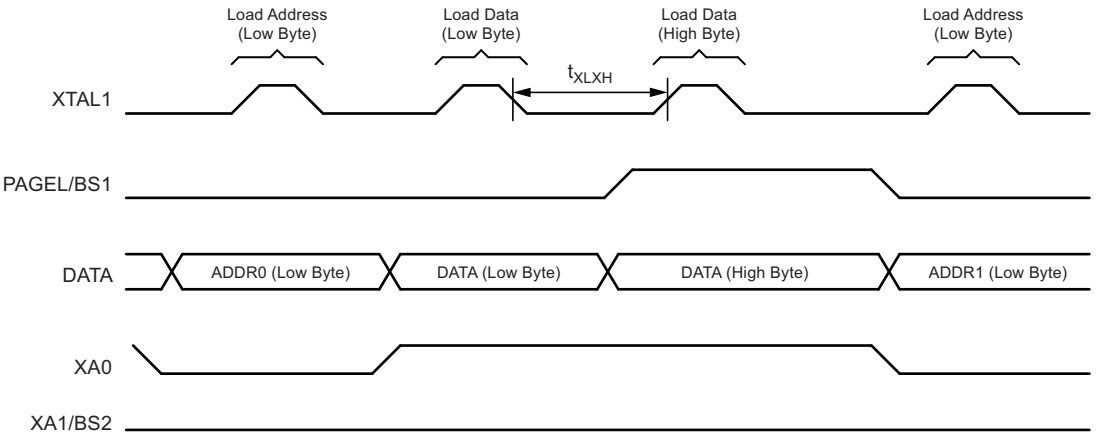


Figure 22-4. Parallel Programming Timing, Loading Sequence with Timing Requirements(1)



Note: 1. The timing requirements shown in Figure 22-3 on page 229 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.