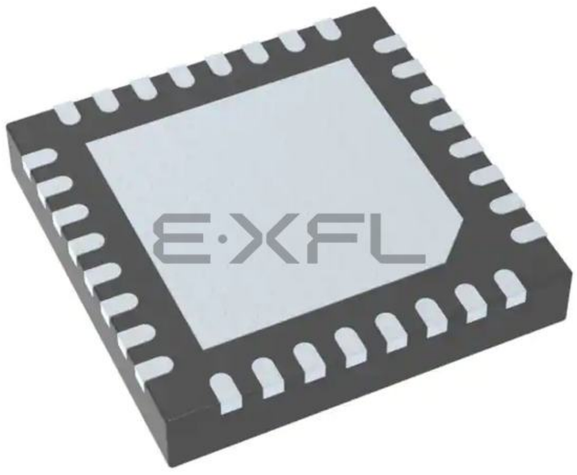


What is "Embedded - Microcontrollers"?

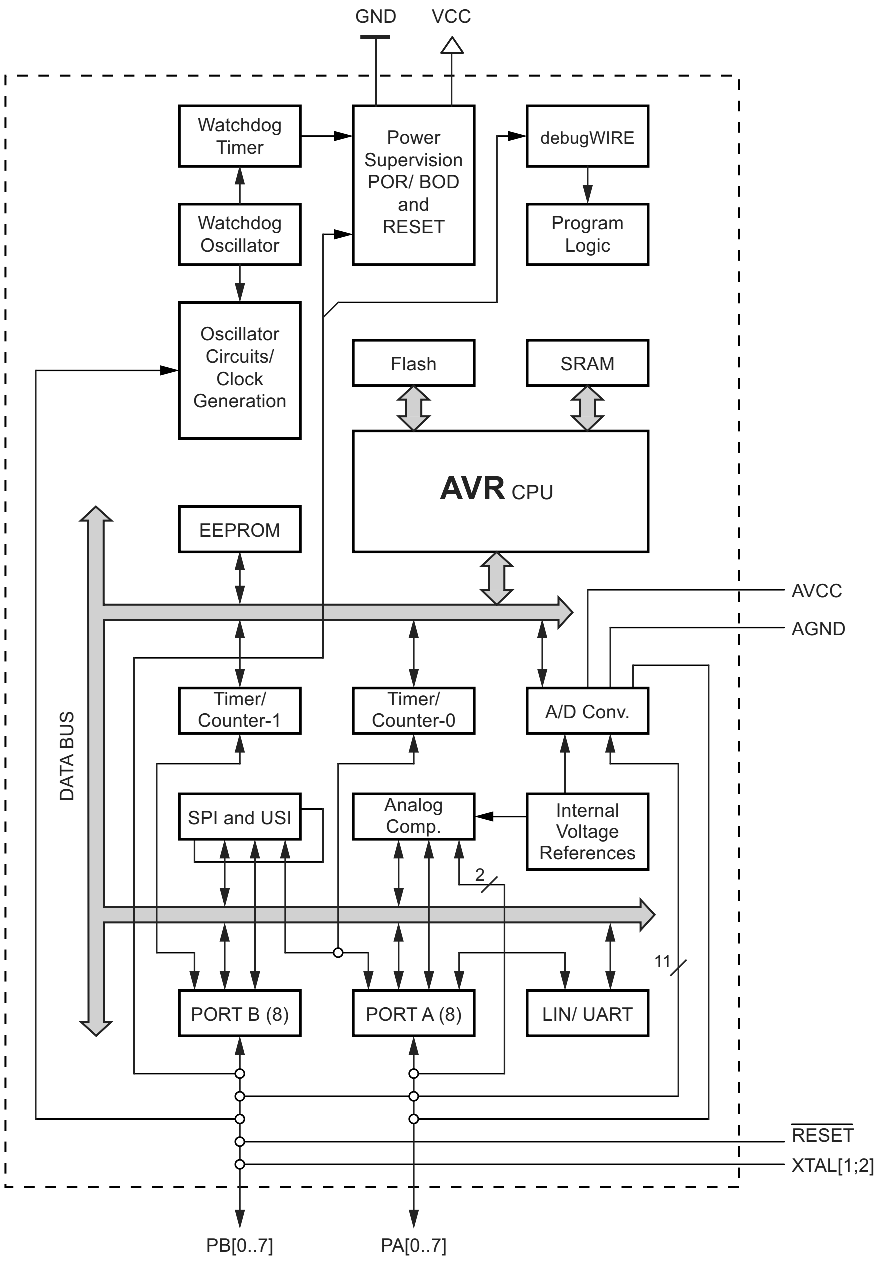
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded"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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	I²C, LINbus, SPI, UART/USART, USI
Peripherals	Brown-out Detect/Reset, POR, PWM, Temp Sensor, WDT
Number of I/O	16
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	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	32-VFQFN Exposed Pad
Supplier Device Package	32-QFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny167-a15mz

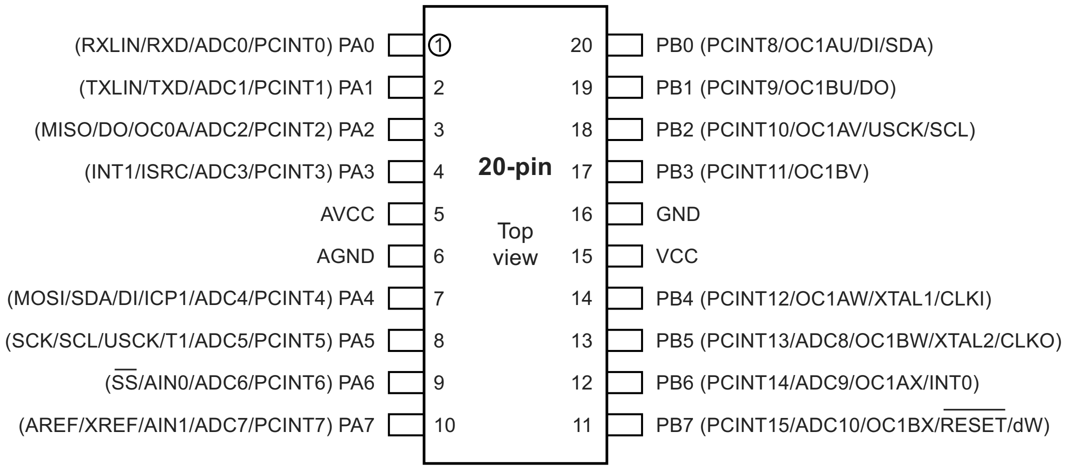
1.5 Block Diagram

Figure 1-1. Block Diagram



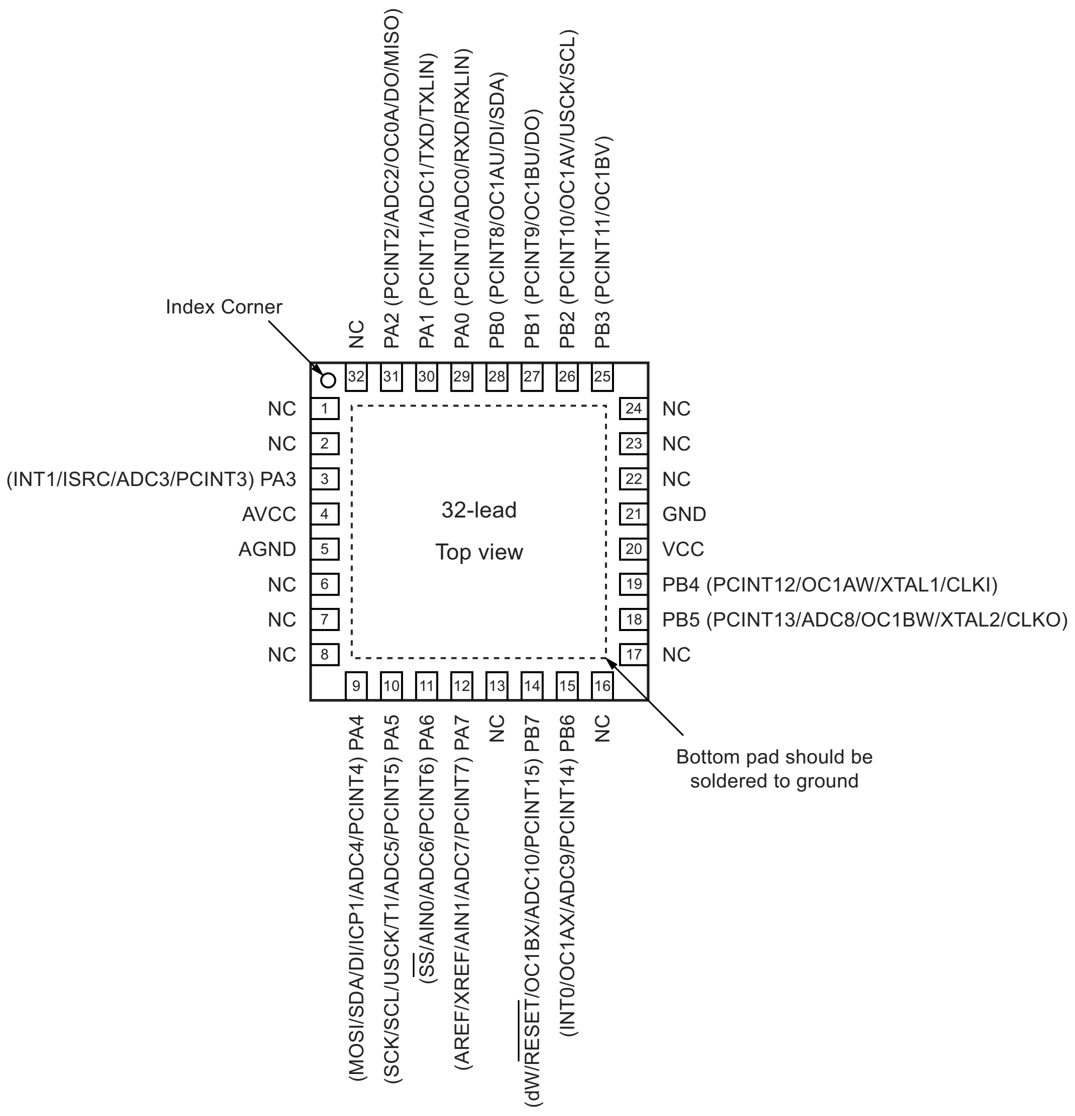
1.6 Pin Configuration

Figure 1-2. Pinout ATtiny87/167 - SOIC20 and TSSOP20



(RXLIN/RXD/ADC0/PCINT0) PA0 1
(TXLIN/TXD/ADC1/PCINT1) PA1 2
(MISO/DO/OC0A/ADC2/PCINT2) PA2 3
(INT1/ISRC/ADC3/PCINT3) PA3 4
AVCC 5
AGND 6
(MOSI/SDA/DI/ICP1/ADC4/PCINT4) PA4 7
(SCK/SCL/USCK/T1/ADC5/PCINT5) PA5 8
($\overline{SS}$/AIN0/ADC6/PCINT6) PA6 9
(AREF/XREF/AIN1/ADC7/PCINT7) PA7 10
20-pin
Top view
20 PB0 (PCINT8/OC1AU/DI/SDA)
19 PB1 (PCINT9/OC1BU/DO)
18 PB2 (PCINT10/OC1AV/USCK/SCL)
17 PB3 (PCINT11/OC1BV)
16 GND
15 VCC
14 PB4 (PCINT12/OC1AW/XTAL1/CLKI)
13 PB5 (PCINT13/ADC8/OC1BW/XTAL2/CLKO)
12 PB6 (PCINT14/ADC9/OC1AX/INT0)
11 PB7 (PCINT15/ADC10/OC1BX/$\overline{RESET}$/dW)

Figure 1-3. Pinout ATtiny87/167 - QFN32



Index Corner
NC 32
PA2 (PCINT2/ADC2/OC0A/DO/MISO) 31
PA1 (PCINT1/ADC1/TXD/TXLIN) 30
PA0 (PCINT0/ADC0/RXD/RXLIN) 29
PB0 (PCINT8/OC1AU/DI/SDA) 28
PB1 (PCINT9/OC1BU/DO) 27
PB2 (PCINT10/OC1AV/USCK/SCL) 26
PB3 (PCINT11/OC1BV) 25
NC 1
NC 2
(INT1/ISRC/ADC3/PCINT3) PA3 3
AVCC 4
AGND 5
NC 6
NC 7
NC 8
32-lead
Top view
24 NC
23 NC
22 NC
21 GND
20 VCC
19 PB4 (PCINT12/OC1AW/XTAL1/CLKI)
18 PB5 (PCINT13/ADC8/OC1BW/XTAL2/CLKO)
17 NC
9 (MOSI/SDA/DI/ICP1/ADC4/PCINT4) PA4
10 (SCK/SCL/USCK/T1/ADC5/PCINT5) PA5
11 ($\overline{SS}$/AIN0/ADC6/PCINT6) PA6
12 (AREF/XREF/AIN1/ADC7/PCINT7) PA7
13 NC
14 (dW/$\overline{RESET}$/OC1BX/ADC10/PCINT15) PB7
15 (INT0/OC1AX/ADC9/PCINT14) PB6
16 NC
Bottom pad should be soldered to ground

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) */
```

- **Bit 1 – EEPE: EEPROM Program Enable**

The EEPROM program enable signal EEPE is the programming enable signal to the EEPROM. When EEPE is written, the EEPROM will be programmed according to the EEPMn bits setting. The EEMPE bit must be written to one before a logical one is written to EEPE, otherwise no EEPROM write takes place. When the write access time has elapsed, the EEPE bit is cleared by hardware. When EEPE has been set, the CPU is halted for two cycles before the next instruction is executed.

- **Bit 0 – EERE: EEPROM Read Enable**

The EEPROM read enable signal – EERE – is the read strobe to the EEPROM. When the correct address is set up in the EEAR register, the EERE bit must be written to one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed. The user should poll the EEPE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR register.

3.5.4 General Purpose I/O Register 2 – GPIOR2

Bit	7	6	5	4	3	2	1	0	
	GPIOR27	GPIOR26	GPIOR25	GPIOR24	GPIOR23	GPIOR22	GPIOR21	GPIOR20	GPIOR2
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	

3.5.5 General Purpose I/O Register 1 – GPIOR1

Bit	7	6	5	4	3	2	1	0	
	GPIOR17	GPIOR16	GPIOR15	GPIOR14	GPIOR13	GPIOR12	GPIOR11	GPIOR10	GPIOR1
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	

3.5.6 General Purpose I/O Register 0 – GPIOR0

Bit	7	6	5	4	3	2	1	0	
	GPIOR07	GPIOR06	GPIOR05	GPIOR04	GPIOR03	GPIOR02	GPIOR01	GPIOR00	GPIOR0
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	

When this clock source is selected, start-up times are determined by the SUT Fuses or CSUT field as shown in Table 4-9.

This external clock can be used by the asynchronous timer if the high or low frequency crystal oscillator is not running (Section 4.3.3 “Enable/Disable Clock Source” on page 33). The asynchronous timer is then able to enable this input.

Table 4-9. Start-up Times for the External Clock Selection

SUT1..0[1] CSUT1..0[2]	Start-up Time from Power-down/save	Additional Delay from Reset (Vcc = 5.0V)	Recommended Usage
00	6CK	14CK (+ 4.1ms[3])	BOD enabled
01	6CK	14CK + 4.1ms	Fast rising power
10	6CK	14CK + 65ms	Slowly rising power
11	Reserved		

Notes: 1. Flash fuse bits.
2. CLKSELR register bits.
3. Additional delay (+ 4ms) available if RSTDISBL fuse is set.

Note that the system clock prescaler can be used to implement run-time changes of the internal clock frequency while still ensuring stable operation. Refer to Section 4.4 “System Clock Prescaler” on page 38 for details.

4.2.7 Clock Output Buffer

If not using a crystal oscillator, the device can output the system clock on the CLKO pin. To enable the output, the CKOUT fuse or COUT bit of CLKSELR register has to be programmed. This option is useful when the device clock is needed to drive other circuits on the system. Note that the clock will not be output during reset and the normal operation of I/O pin will be overridden when the fuses are programmed. If the System clock prescaler is used, it is the divided system clock that is output.

4.3 Dynamic Clock Switch

4.3.1 Features

The Atmel® ATtiny87/167 provides a powerful dynamic clock switch circuit that allows users to turn on and off clocks of the device on the fly. The built-in de-glitching circuitry allows clocks to be enabled or disabled asynchronously. This enables efficient power management schemes to be implemented easily and quickly. In a safety application, the dynamic clock switch circuit allows continuous monitoring of the external clock permitting a fallback scheme in case of clock failure.

The control of the dynamic clock switch circuit must be supervised by software. This operation is facilitated by the following features:

- **Safe commands**, to avoid unintentional commands, a special write procedure must be followed to change the CLKCSR register bits (Section 4.5.2 “CLKPR – Clock Prescaler Register” on page 38):
- **Exclusive action**, the actions are controlled by a decoding table (commands) written to the CLKCSR register. This ensures that only one command operation can be launched at any time. The main actions of the decoding table are:
  - ‘*Disable Clock Source*’,
  - ‘*Enable Clock Source*’,
  - ‘*Request Clock Availability*’,
  - ‘*Clock Source Switching*’,
  - ‘*Recover System Clock Source*’,
  - ‘*Enable Watchdog in Automatic Reload Mode*’.
- **Command status return.** The ‘*request clock availability* ’ command returns status via the CLKRDY bit in the CLKCSR register. The ‘*recover system clock source* ’ command returns a code of the current clock source in the CLKSELR register. This information is used in the supervisory software routines as shown in Section 4.3.7 on page 34.

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();
}
```

Note: 1. See Section 1.9 “About Code Examples” on page 7.

Note that 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.

8.3 External Interrupts Register Description

8.3.1 External Interrupt Control Register A – EICRA

The external interrupt control register A contains control bits for interrupt sense control.

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..4 – Res: Reserved Bits**

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

- **Bit 3, 2 – ISC11, ISC10: Interrupt Sense Control 1 Bit 1 and Bit 0**

The external interrupt 1 is activated by the external pin INT1 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT1 pin that activate the interrupt are defined in Table 8-1. The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

- **Bit 1, 0 – ISC01, ISC00: Interrupt Sense Control 0 Bit 1 and Bit 0**

The external interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 8-1. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 8-1. Interrupt Sense Control

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Any logical change on INTn generates an interrupt request.
1	0	The falling edge of INTn generates an interrupt request.
1	1	The rising edge of INTn generates an interrupt request.

8.3.2 External Interrupt Mask Register – EIMSK

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	INT1	INT0	EIMSK
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 – INT1: External Interrupt Request 1 Enable**

When the INT1 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 control1 bits 1/0 (ISC11 and ISC10) in the external interrupt control register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of external interrupt request 1 is executed from the INT1 interrupt vector.

Table 9-4 and Table 9-5 on page 77 relate the alternate functions of Port A to the overriding signals shown in Figure 9-6 on page 70.

Table 9-4. Overriding Signals for Alternate Functions in PA7..PA4

Signal Name	PA7/PCINT7/ ADC7/AIN1 /XREF/AREF	PA6/PCINT6/ ADC6/AIN0/$\overline{\text{SS}}$	PA5/PCINT5/ADC5/ T1/USCK/SCL/SCK	PA4/PCINT4/ADC4/ ICP1/DI/SDA/MOSI
PUOE	0	SPE & $\overline{\text{MSTR}}$	SPE & $\overline{\text{MSTR}}$	SPE & $\overline{\text{MSTR}}$
PUOV	0	PORTA6 & $\overline{\text{PUD}}$	PORTA5 & $\overline{\text{PUD}}$	PORTA4 & $\overline{\text{PUD}}$
DDOE	0	SPE & $\overline{\text{MSTR}}$	(SPE & $\overline{\text{MSTR}}$) | (USI_2_WIRE & USIPOS)	(SPE & $\overline{\text{MSTR}}$) | (USI_2_WIRE & USIPOS)
DDOV	0	0	(USI_SCL_HOLD | $\overline{\text{PORTA5}}$) & DDRA6	{ (SPE & $\overline{\text{MSTR}}$) ? (0) : $\overline{\text{(USI\_SHIFTOUT}}$ | $\overline{\text{PORTA4}}$) & DDRA4) }
PVOE	0	0	(SPE & MSTR) | (USI_2_WIRE & USIPOS & DDRA5)	(SPE & MSTR) | (USI_2_WIRE & USIPOS & DDRA4)
PVOV	0	0	{ (SPE & MSTR) ? (SCK_OUTPUT) : ~ (USI_2_WIRE & USIPOS & DDRA5) }	{ (SPE & MSTR) ? (MOSI_OUTPUT) : ~ (USI_2_WIRE & USIPOS & DDRA4) }
PTOE	0	0	USI_PTOE & USIPOS	0
DIEOE	ADC7D | (PCIE0 & PCMSK07)	ADC6D | (PCIE0 & PCMSK06)	ADC5D | (USISIE & USIPOS) | (PCIE0 & PCMSK05)	ADC4D | (USISIE & USIPOS) | (PCIE0 & PCMSK04)
DIEOV	PCIE0 & PCMSK07	PCIE0 & PCMSK06	(USISIE & USIPOS) | (PCIE0 & PCMSK05)	(USISIE & USIPOS) | (PCIE0 & PCMSK04)
DI	PCINT7	PCINT6 -/- $\overline{\text{SS}}$	PCINT5 -/- T1 -/- USCK -/- SCL -/- SCK	PCINT4 -/- ICP1 -/- DI -/- SDA -/- MOSI
AIO	ADC7 -/- AIN1 -/- XREF -/- AREF	ADC6 -/- AIN0	ADC5	ADC4

- If timer/counter0 is used to wake the device up from power-save mode, precautions must be taken if the user wants to re-enter one of these modes: The interrupt logic needs one TOSC1 cycle to be reset. If the time between wake-up and re-entering sleep mode is less than one TOSC1 cycle, the interrupt will not occur, and the device will fail to wake up. If the user is in doubt whether the time before re-entering power-save mode is sufficient, the following algorithm can be used to ensure that one TOSC1 cycle has elapsed:
    a. Write a value to TCCR0A, TCNT0, or OCR0A.
    b. Wait until the corresponding update busy flag in ASSR returns to zero.
    c. Enter power-save or ADC noise reduction mode.
- When the asynchronous operation is selected, the oscillator for timer/counter0 is always running, except in power-down mode. After a power-up reset or wake-up from power-down mode, the user should be aware of the fact that this oscillator might take as long as one second to stabilize. The user is advised to wait for at least one second before using timer/counter0 after power-up or wake-up from power-down mode. The contents of all timer/counter0 registers must be considered lost after a wake-up from power-down mode due to unstable clock signal upon start-up, no matter whether the oscillator is in use or a clock signal is applied to the XTAL1 pin.
- Description of wake up from power-save mode when the timer is clocked asynchronously: When the interrupt condition is met, the wake up process is started on the following cycle of the timer clock, that is, the timer is always advanced by at least one before the processor can read the counter value. After wake-up, the MCU is halted for four cycles, it executes the interrupt routine, and resumes execution from the instruction following SLEEP.
- Reading of the TCNT0 register shortly after wake-up from power-save may give an incorrect result. Since TCNT0 is clocked on the asynchronous clock, reading TCNT0 must be done through a register synchronized to the internal I/O clock domain (CPU main clock). Synchronization takes place for every rising XTAL1 edge. When waking up from power-save mode, and the I/O clock ($clk_{I/O}$) again becomes active, TCNT0 will read as the previous value (before entering sleep) until the next rising XTAL1 edge. The phase of the XTAL1 clock after waking up from power-save mode is essentially unpredictable, as it depends on the wake-up time. The recommended procedure for reading TCNT0 is thus as follows:
    a. Write any value to either of the registers OCR0A or TCCR0A.
    b. Wait for the corresponding update busy flag to be cleared.
    c. Read TCNT0.
- During asynchronous operation, the synchronization of the interrupt flags for the asynchronous timer takes 3 processor cycles plus one timer cycle. The timer is therefore advanced by at least one before the processor can read the timer value causing the setting of the interrupt flag. The output compare pin is changed on the timer clock and is not synchronized to the processor clock.

10.11.2 Timer/Counter0 Register – TCNT0

Bit	7	6	5	4	3	2	1	0	
	TCNT07	**TCNT06**	**TCNT05**	**TCNT04**	**TCNT03**	**TCNT02**	**TCNT01**	**TCNT00**	**TCNT0**
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 timer/counter register gives direct access, both for read and write operations, to the timer/counter unit 8-bit counter. writing to the TCNT0 register blocks (removes) the compare match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a compare match between TCNT0 and the OCR0x register.

10.11.3 Output Compare Register A – OCR0A

Bit	7	6	5	4	3	2	1	0	
	OCR0A7	**OCR0A6**	**OCR0A5**	**OCR0A4**	**OCR0A3**	**OCR0A2**	**OCR0A1**	**OCR0A0**	**OCR0A**
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 output compare register A contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0A pin.

10.11.4 Asynchronous Status Register – ASSR

Bit	7	6	5	4	3	2	1	0	
	–	**EXCLK**	**AS0**	**TCN0UB**	**OCR0AUB**	**–**	**TCR0AUB**	**TCR0BUB**	**ASSR**
Read/Write	R	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – Res: Reserved Bit**

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

- **Bit 6 – EXCLK: Enable External Clock Input**

When EXCLK is written to one, and asynchronous clock is selected, the external clock input buffer is enabled and an external clock can be input on XTAL1 pin instead of an external crystal. Writing to EXCLK should be done before asynchronous operation is selected. Note that the crystal oscillator will only run when this bit is zero.

- **Bit 5 – AS0: Asynchronous Timer/Counter0**

When AS0 is written to zero, timer/counter0 is clocked from the I/O clock, clkI/O and the timer/counter0 acts as a synchronous peripheral.

When AS0 is written to one, timer/counter0 is clocked from the low-frequency crystal oscillator (see Section 4.2.5 “Low-frequency Crystal Oscillator” on page 31) or from external clock on XTAL1 pin (see Section 4.2.6 “External Clock” on page 31) depending on EXCLK setting. When the value of AS0 is changed, the contents of TCNT0, OCR0A, and TCCR0A might be corrupted.

AS0 also acts as a flag: timer/counter0 is clocked from the low-frequency crystal or from external clock ONLY IF the calibrated internal RC oscillator or the internal watchdog oscillator is used to drive the system clock. After setting AS0, if the switching is available, AS0 remains to 1, else it is forced to 0.

- **Bit 4 – TCN0UB: Timer/Counter0 Update Busy**

When timer/counter0 operates asynchronously and TCNT0 is written, this bit becomes set. When TCNT0 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCNT0 is ready to be updated with a new value.

12.7.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the force output compare (FOC1A/B) bit. Forcing compare match will not set the OCF1A/B flag or reload/clear the timer, but the OC1A/Bi pins will be updated as if a real compare match had occurred (the COM1A/B1:0 bits settings define whether the OC1A/Bi pins are set, cleared or toggled - if the respective OCnxi bit is set).

12.7.2 Compare Match Blocking by TCNT1 Write

All CPU writes to the TCNT1 register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR1A/B to be initialized to the same value as TCNT1 without triggering an interrupt when the timer/counter clock is enabled.

12.7.3 Using the Output Compare Unit

Since writing TCNT1 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT1 when using any of the output compare channels, independent of whether the timer/counter is running or not. If the value written to TCNT1 equals the OCR1A/B value, the compare match will be missed, resulting in incorrect waveform generation. Do not write the TCNT1 equal to TOP in PWM modes with variable TOP values. The compare match for the TOP will be ignored and the counter will continue to 0xFFFF. Similarly, do not write the TCNT1 value equal to BOTTOM when the counter is down counting.

The setup of the OC1A/B should be performed before setting the data direction register for the port pin to output. The easiest way of setting the OC1A/B value is to use the force output compare (FOC1A/B) strobe bits in normal mode. The OC1A/B register keeps its value even when changing between waveform generation modes.

Be aware that the COM1A/B1:0 bits are not double buffered together with the compare value. Changing the COM1A/B1:0 bits will take effect immediately.

12.8 Compare Match Output Unit

The compare output mode (COM1A/B1:0) bits have two functions. The waveform generator uses the COM1A/B1:0 bits for defining the output compare (OC1A/B) state at the next compare match. Secondly the COM1A/B1:0 and OCnxi bits control the OC1A/Bi pin output source. Figure 12-6 on page 115 shows a simplified schematic of the logic affected by the COM1A/B1:0 and OCnxi bit setting. The I/O registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers (DDR and PORT) that are affected by the COM1A/B1:0 and OCnxi bits are shown. When referring to the OC1A/B state, the reference is for the internal OC1A/B register, not the OC1A/Bi pin. If a system reset occur, the OC1A/B register is reset to “0”.

Figure 12-13 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR1A/B Register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV1 flag at BOTTOM.

Figure 12-13.Timer/Counter Timing Diagram, no Prescaling

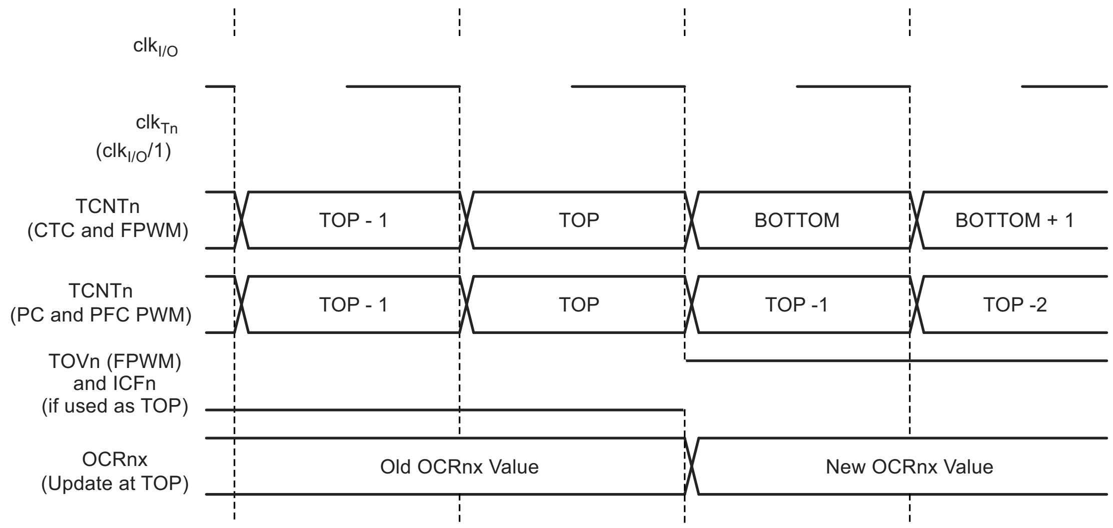
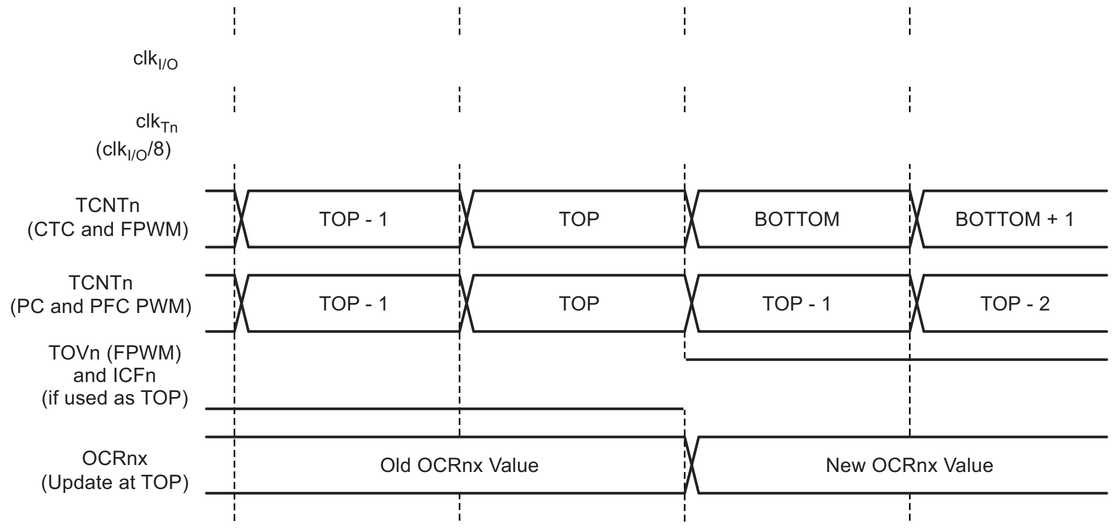


Figure 12-14 shows the same timing data, but with the prescaler enabled.

Figure 12-14.Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)



12.11.3 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.

12.11.4 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 12-6 on page 115.

- **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 12-6 on page 115.

12.11.5 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 12.3 “Accessing 16-bit Registers” on page 105.

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.

16. ISRC - Current Source

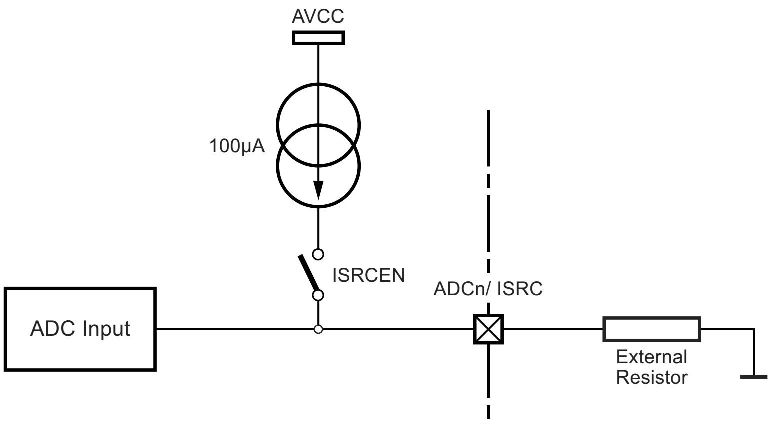
16.1 Features

- 100µA Constant current source
- ±10% Absolute Accuracy

The Atmel® ATtiny87/167 features a 100µA ±10% Current Source. Up on request, the current is flowing through an external resistor. The voltage can be measured on the dedicated pin shared with the ADC. Using a resistor in series with a ≤ 0.5% tolerance is recommended. To protect the device against big values, the ADC must be configured with AVcc as internal reference to perform the first measurement. Afterwards, another internal reference can be chosen according to the previous measured value to refine the result.

When ISRCEN bit is set, the ISRC pin sources 100µA. Otherwise this pin keeps its initial function.

Figure 16-1. Current Source Block Diagram



16.2 Typical applications

16.2.1 LIN Current Source

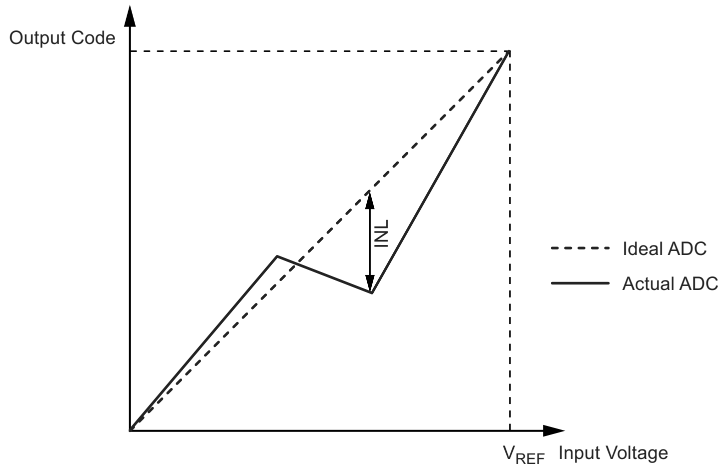
During the configuration of a LIN node in a cluster, it may be necessary to attribute dynamically an unique physical address to every cluster node. The way to do it is not described in the LIN protocol.

The current source offers an excellent solution to associate a physical address to the application supported by the LIN node. A full dynamic node configuration can be used to set-up the LIN nodes in a cluster.

Atmel ATtiny87/167 proposes to have an external resistor used in conjunction with the current source. The device measures the voltage to the boundaries of the resistance via the analog to digital converter. The resulting voltage defines the physical address that the communication handler will use when the node will participate in LIN communication.

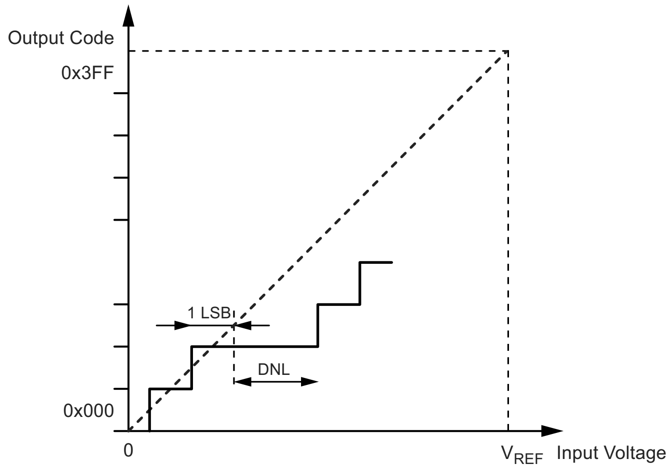
- Integral non-linearity (INL): After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

Figure 17-11. Integral Non-linearity (INL)



- Differential non-linearity (DNL): The maximum deviation of the actual code width (the interval between two adjacent transitions) from the ideal code width (1 LSB). Ideal value: 0 LSB.

Figure 17-12. Differential Non-linearity (DNL)



- Quantization error: Due to the quantization of the input voltage into a finite number of codes, a range of input voltages (1 LSB wide) will code to the same value. Always ±0.5 LSB.
- Absolute accuracy: The maximum deviation of an actual (unadjusted) transition compared to an ideal transition for any code. This is the compound effect of offset, gain error, differential error, non-linearity, and quantization error. Ideal value: ±0.5 LSB.

C. Load Data Low Byte

1. Set XA1, XA0 to “0,1”. This enables data loading.
2. Set DATA = data low byte (0x00 - 0xFF).
3. Give XTAL1 a positive pulse. This loads the data byte.

D. Load Data High Byte

1. Set BS1 to “1”. This selects high data byte.
2. Set XA1, XA0 to “0,1”. This enables data loading.
3. Set DATA = data high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the data byte.

E. Latch Data

1. Set BS1 to “1”. This selects high data byte.
2. Give PAGEL a positive pulse. This latches the data bytes. (See Figure 21-3 on page 214 for signal waveforms)

F. Repeat B through E until the entire buffer is filled or until all data within the page is loaded.

While the lower bits in the address are mapped to words within the page, the higher bits address the pages within the FLASH. This is illustrated in Figure 21-2. Note that if less than eight bits are required to address words in the page (pagesize < 256), the most significant bit(s) in the address low byte are used to address the page when performing a page write.

G. Load Address High byte

1. Set XA1, XA0 to “0,0”. This enables address loading.
2. Set BS1 to “1”. This selects high address.
3. Set DATA = address high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the address high byte.

H. Program Page

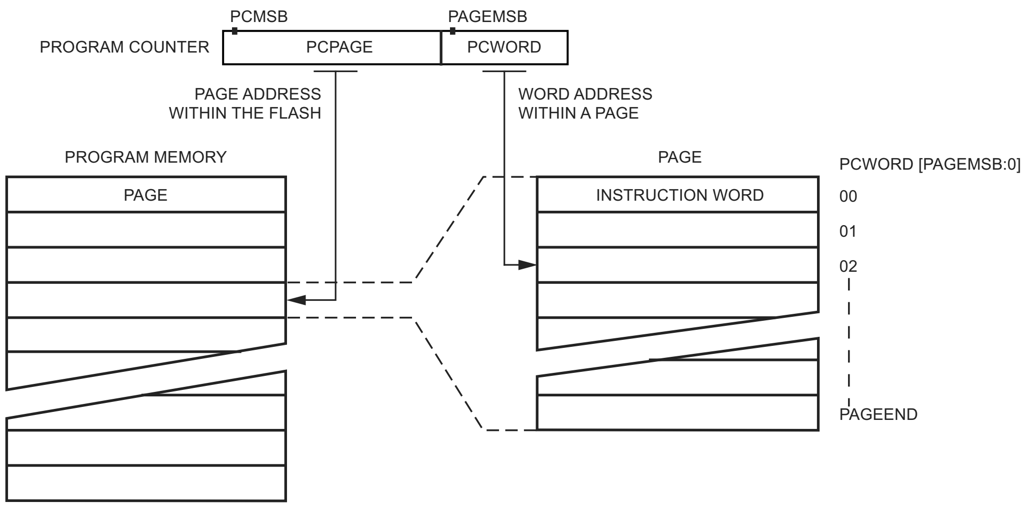
1. Give $\overline{\text{WR}}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{\text{BSY}}$ goes low.
2. Wait until RDY/$\overline{\text{BSY}}$ goes high (See Figure 21-3 on page 214 for signal waveforms).

I. Repeat B through H until the entire flash is programmed or until all data has been programmed.

J. End Page Programming

1. 1. Set XA1, XA0 to “1,0”. This enables command loading.
2. Set DATA to “0000 0000 $_b$”. This is the command for no operation.
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

Figure 21-2. Addressing the Flash Which is Organized in Pages

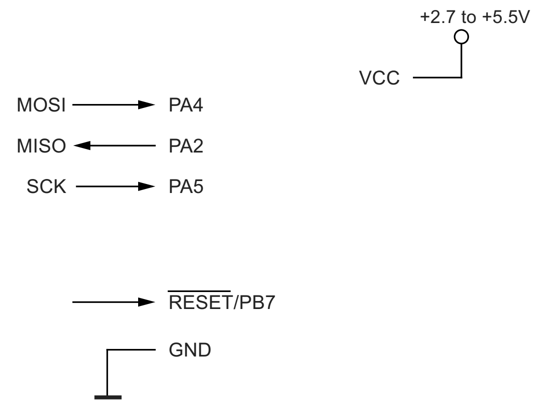


21.8 Serial Downloading

Both the flash and EEPROM memory arrays can be programmed using the serial SPI bus while $\overline{\text{RESET}}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{\text{RESET}}$ is set low, the programming enable instruction needs to be executed first before program/erase operations can be executed.

Note: In Table 21-13 on page 218, the pin mapping for SPI programming is listed. Not all parts use the SPI pins dedicated for the internal SPI interface.

Figure 21-7. Serial Programming and Verify [(1)]



Note: 1. If the device is clocked by the internal oscillator, it is no need to connect a clock source to the XTAL1 pin

Table 21-13. Pin Mapping Serial Programming

Symbol	Pin Name	I/O	Function
MOSI	PA4	I	Serial data in
MISO	PA2	O	Serial data out
SCK	PA5	I	Serial clock

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the serial mode **ONLY**) and there is no need to first execute the chip erase instruction. The chip erase operation turns the content of every memory location in both the program and EEPROM arrays into 0xFF.

Depending on CKSEL fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > **2** CPU clock cycles for f_{ck} < 12MHz, **3** CPU clock cycles for f_{ck} ≥ 12Hz

High: > **2** CPU clock cycles for f_{ck} < 12MHz, **3** CPU clock cycles for f_{ck} ≥ 12MHz

21.8.1 Serial Programming Algorithm

When writing serial data to the Atmel® ATtiny87/167, data is clocked on the rising edge of SCK.

When reading data from the ATtiny87/167, data is clocked on the falling edge of SCK. See Figure 21-7 and Figure 21-8 on page 221 for timing details.

To program and verify the Atmel ATtiny87/167 in the serial programming mode, the following sequence is recommended (see four byte instruction formats in Table 21-15 on page 220):

1. Power-up sequence: Apply power between Vcc and GND while $\overline{\text{RESET}}$ and SCK are set to "0". In some systems, the programmer can not guarantee that SCK is held low during power-up. In this case, $\overline{\text{RESET}}$ must be given a positive pulse of at least two CPU clock cycles duration after SCK has been set to "0".
2. Wait for at least 20ms and enable serial programming by sending the programming enable serial instruction to pin MOSI.

22.9 Parallel Programming Characteristics

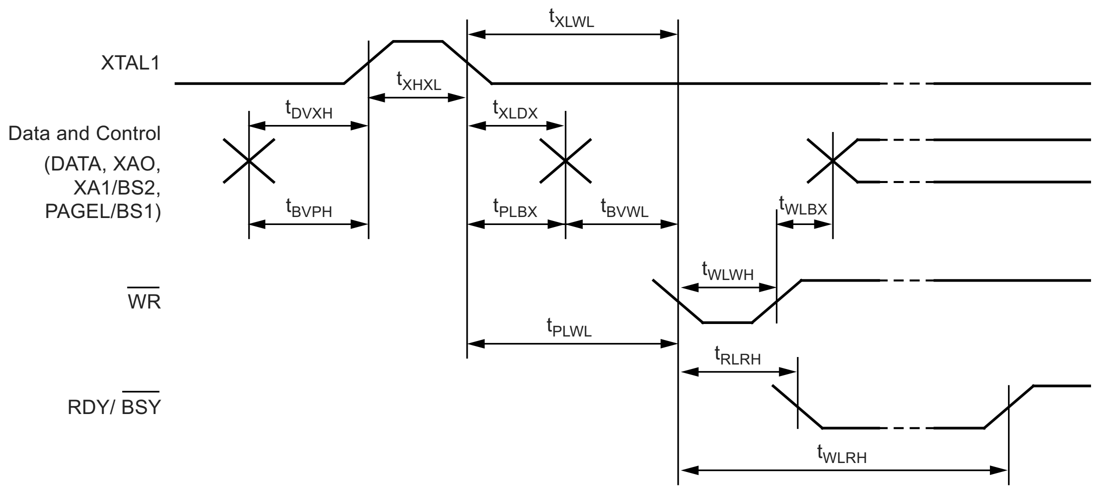


Figure 22-3. Parallel Programming Timing, Including some General Timing Requirements

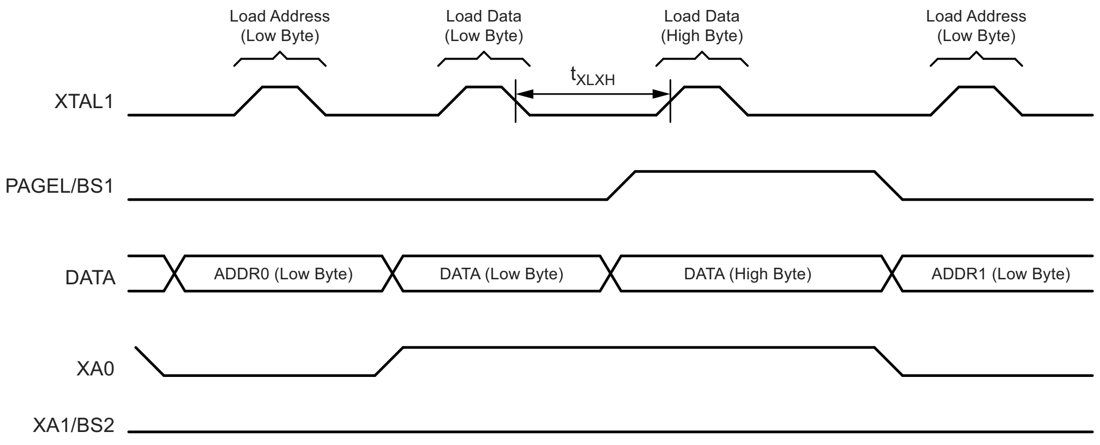
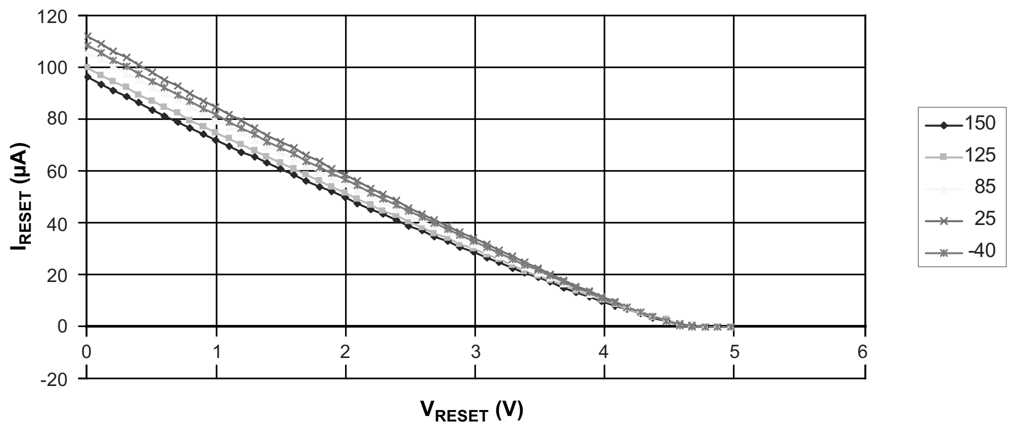


Figure 22-4. Parallel Programming Timing, Loading Sequence with Timing Requirements[1]

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.

Figure 24-13. Reset Pull-up Resistor Current versus Reset Pin Voltage (VCC = 5V)



24.6 Pin Driver Strength

Figure 24-14. I/O Pin Output Voltage versus Sink Current (VCC = 3V)

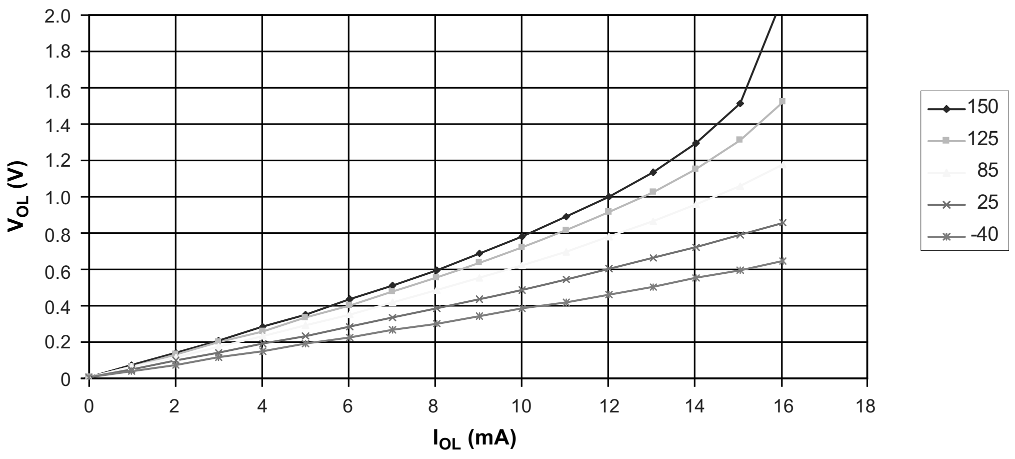
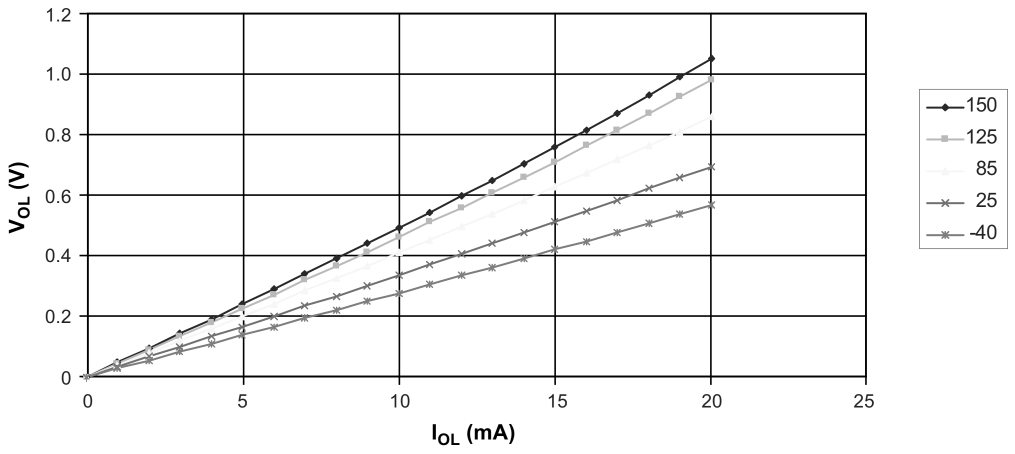


Figure 24-15. I/O Pin Output Voltage versus Sink Current (VCC = 5V)



25. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xFF)	Reserved									
(0xFE)	Reserved									
(0xFD)	Reserved									
(0xFC)	Reserved									
(0xFB)	Reserved									
(0xFA)	Reserved									
(0xF9)	Reserved									
(0xF8)	Reserved									
(0xF7)	Reserved									
(0xF6)	Reserved									
(0xF5)	Reserved									
(0xF4)	Reserved									
(0xF3)	Reserved									
(0xF2)	Reserved									
(0xF1)	Reserved									
(0xF0)	Reserved									
(0xEF)	Reserved									
(0xEE)	Reserved									
(0xED)	Reserved									
(0xEC)	Reserved									
(0xEB)	Reserved									
(0xEA)	Reserved									
(0xE9)	Reserved									
(0xE8)	Reserved									
(0xE7)	Reserved									
(0xE6)	Reserved									
(0xE5)	Reserved									
(0xE4)	Reserved									
(0xE3)	Reserved									
(0xE2)	Reserved									
(0xE1)	Reserved									
(0xE0)	Reserved									
(0xDF)	Reserved									
(0xDE)	Reserved									
(0xDD)	Reserved									
(0xDC)	Reserved									

Notes:
1. Address bits exceeding EEAMSB (Table 21-8 on page 210) are don't care.
2. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
3. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
4. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
5. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The Atmel® ATtiny87/167 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in opcode for the IN and OUT instructions. For the extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.