Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	I²C, LINbus, SPI, UART/USART, USI
Peripherals	Brown-out Detect/Reset, POR, PWM, Temp Sensor, WDT
Number of I/O	16
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	20-TSSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny167-xu

– ‘*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 33.

4.3.2 CLKSELR Register

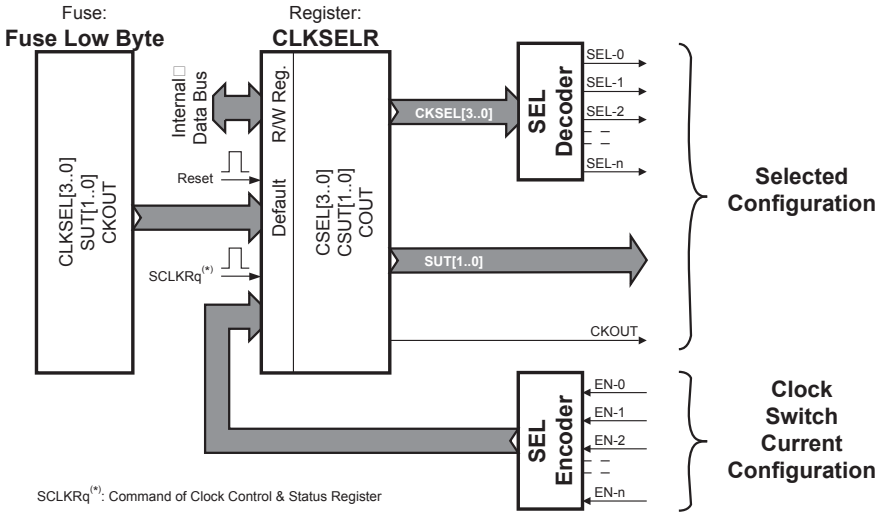
4.3.2.1 Fuses Substitution

At reset, bits of the Low Fuse Byte are copied into the CLKSELR register. The content of this register can subsequently be user modified to overwrite the default values from the Low Fuse Byte. CKSEL[3:0], SUT[1:0] and CKOUT fuses correspond respectively to CSEL[3:0], CSUT[1:0] and ~(COUT) bits of the CLKSELR register as shown in Figure 4-5 on page 32.

4.3.2.2 Source Selection

The available codes of clock source are given in Table 4-1 on page 25.

Figure 4-5. Fuses substitution and Clock Source Selection



The CLKSELR register contains the CSEL, CSUT and COUT values which will be used by the ‘Enable/Disable Clock Source’, ‘Request for Clock Availability’ or ‘Clock Source Switching’ commands.

4.3.2.3 Source Recovering

The ‘*Recover System Clock Source*’ command updates the CKSEL field of CLKSELR register (See “System Clock Source Recovering” on page 33.).

4.3.3 Enable/Disable Clock Source

The ‘*Enable Clock Source*’ command selects and enables a clock source configured by the settings in the CLKSELR register. CSEL[3:0] will select the clock source and CSUT[1:0] will select the start-up time (just as CKSEL and SUT fuse bits do). To be sure that a clock source is operating, the ‘*Request for Clock Availability* ’ command must be executed after the ‘*Enable Clock Source*’ command. This will indicate via the CLKRDY bit in the CLKCSR register that a valid clock source is available and operational.

range. Incrementing CAL[6:0] by 1 will give a frequency increment of less than 2% in the frequency range 7.3 - 8.1 MHz.

4.5.2 CLKPR – Clock Prescaler Register

Bit	7	6	5	4	3	2	1	0	
(0x61)	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	CLKPR
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

- **Bit 7 – CLKPCE: Clock Prescaler Change Enable**

The CLKPCE bit must be written to logic one to enable change of the CLKPS bits. The CLKPCE bit is only updated when the other bits in CLKPR are simultaneously written to zero. CLKPCE is cleared by hardware four cycles after it is written or when the CLKPS bits are written. Rewriting the CLKPCE bit within this time-out period does neither extend the time-out period, nor clear the CLKPCE bit.

- **Bits 6:4 – Res: Reserved Bits**

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

- **Bits 3:0 – CLKPS[3:0]: Clock Prescaler Select Bits 3 - 0**

These bits define the division factor between the selected clock source and the internal system clock. These bits can be written run-time to vary the clock frequency to suit the application requirements. As the divider divides the master clock input to the MCU, the speed of all synchronous peripherals is reduced when a division factor is used. The division factors are given in Table 4-10.

To avoid unintentional changes of clock frequency, a special write procedure must be followed to change the CLKPS bits:

1. Write the Clock Prescaler Change Enable (CLKPCE) bit to one and all other bits in CLKPR to zero.
2. Within four cycles, write the desired value to CLKPS while writing a zero to CLKPCE.

Interrupts must be disabled when changing prescaler setting in order not to disturb the procedure.

The CKDIV8 Fuse determines the initial value of the CLKPS bits. If CKDIV8 is unprogrammed, the CLKPS bits will be reset to “0000”. If CKDIV8 is programmed, CLKPS bits are reset to “0011”, giving a division factor of eight at start up. This feature should be used if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. Note that any value can be written to the CLKPS bits regardless of the CKDIV8 Fuse setting. The Application software must ensure that a sufficient division factor is chosen if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. The device is shipped with the CKDIV8 Fuse programmed.

the clock frequency and, of course, if the clock is alive. The user's code has to differentiate between '***no_clock_signal***' and '***clock_signal_not_yet_available***' condition.

- **Bits 3:0 – CLKC[3:0]: Clock Control Bits 3 - 0**

These bits define the command to provide to the '*Clock Switch*' module. The special write procedure must be followed to change the CLKC[3:0] bits (See "Bit 7 – CLKCCE: Clock Control Change Enable" on page 39.).

1. Write the Clock Control Change Enable (CLKCCE) bit to one and all other bits in CLKCSR to zero.
2. Within 4 cycles, write the desired value to CLKCSR register while clearing CLKCCE bit.

Interrupts should be disabled when setting CLKCSR register in order not to disturb the procedure.

Table 4-11. Clock command list.

Clock Command	CLKC[3:0]
No command	0000_b
Disable clock source	0001_b
Enable clock source	0010_b
Request for clock availability	0011_b
Clock source switch	0100_b
Recover system clock source code	0101_b
Enable watchdog in automatic reload mode	0110_b
CKOUT command	0111_b
No command	$1xxx_b$

4.5.4 CLKSELR – Clock Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x63)	–	COUT	CSUT1	CSUT0	CSEL3	CSEL2	CSEL1	CSEL0	CLKSELR
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	~ (CKOUT) fuse	SUT[1:0] fuses			CKSEL[3:0] fuses			

- **Bit 7 – Res: Reserved Bit**

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

- **Bit 6 – COUT: Clock Out**

The COUT bit is initialized with ~(CKOUT) Fuse bit.
The COUT bit is only used in case of '*CKOUT*' command. Refer to Section 4.2.7 "Clock Output Buffer" on page 31 for using.
In case of '*Recover System Clock Source*' command, COUT it is not affected (no recovering of this setting).

- **Bits 5:4 – CSUT[1:0]: Clock Start-up Time**

7. Interrupts

This section describes the specifics of the interrupt handling as performed in ATtiny87/167. For a general explanation of the AVR interrupt handling, refer to “Reset and Interrupt Handling” on page 12.

7.1 Interrupt Vectors in ATtiny87/167

Table 7-1. Reset and Interrupt Vectors in ATtiny87/167

Vector Nb.	Program Address		Source	Interrupt Definition
	ATtiny87	ATtiny167		
1	0x0000	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0001	0x0002	INT0	External Interrupt Request 0
3	0x0002	0x0004	INT1	External Interrupt Request 1
4	0x0003	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0004	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x0005	0x000A	WDT	Watchdog Time-out Interrupt
7	0x0006	0x000C	TIMER1 CAPT	Timer/Counter1 Capture Event
8	0x0007	0x000E	TIMER1 COMPA	Timer/Counter1 Compare Match A
9	0x0008	0x0010	TIMER1 COMPB	Timer/Coutner1 Compare Match B
10	0x0009	0x0012	TIMER1 OVF	Timer/Counter1 Overflow
11	0x000A	0x0014	TIMER0 COMPA	Timer/Counter0 Compare Match A
12	0x000B	0x0016	TIMER0 OVF	Timer/Counter0 Overflow
13	0x000C	0x0018	LIN TC	LIN/UART Transfer Complete
14	0x000D	0x001A	LIN ERR	LIN/UART Error
15	0x000E	0x001C	SPI, STC	SPI Serial Transfer Complete
16	0x000F	0x001E	ADC	ADC Conversion Complete
17	0x0010	0x0020	EE READY	EEPROM Ready
18	0x0011	0x0022	ANALOG COMP	Analog Comparator
19	0x0012	0x0024	USI START	USI Start Condition Detection
20	0x0013	0x0026	USI OVF	USI Counter Overflow

8. External Interrupts

8.1 Overview

The External Interrupts are triggered by the INT0 or INT1 pin or any of the PCINT[15:0] pins. Observe that, if enabled, the interrupts will trigger even if the INT0, INT1 or PCINT[15:0] pins are configured as outputs. This feature provides a way of generating a software interrupt.

The pin change interrupt PCINT1 will trigger if any enabled PCINT[15:8] pin toggles. The pin change interrupt PCINT0 will trigger if any enabled PCINT[7:0] pin toggles. The PCMSK1 and PCMSK0 Registers control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT[15:0] are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

The INT0 or INT1 interrupt can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the External Interrupt Control Register A – EICRA. When the INT0 or INT1 interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. The recognition of falling or rising edge interrupts on INT0 or INT1 requires the presence of an I/O clock, described in “Clock Systems and their Distribution” on page 24. Low level interrupts and the edge interrupt on INT0 or INT1 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode. The I/O clock is halted in all sleep modes except Idle mode.

Note that if a level triggered interrupt is used for wake-up from Power-down or Power-save, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no interrupt will be generated. The start-up time is defined by the SUT and CKSEL Fuses as described in “Clock Systems and their Distribution” on page 24.

8.2 Pin Change Interrupt Timing

An example of timing of a pin change interrupt is shown in Figure 8-1.

Figure 8-1. Timing of pin change interrupts

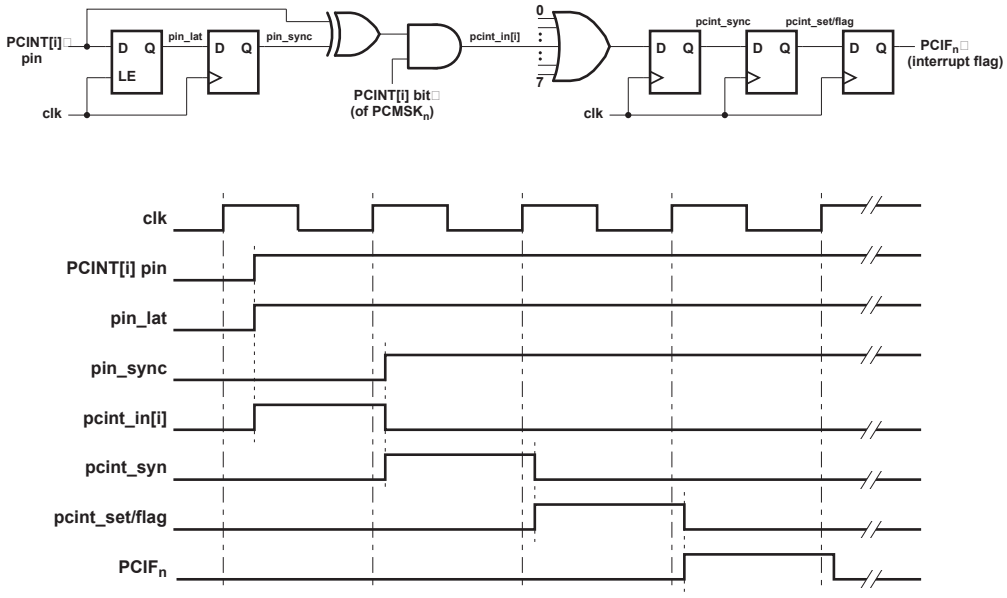
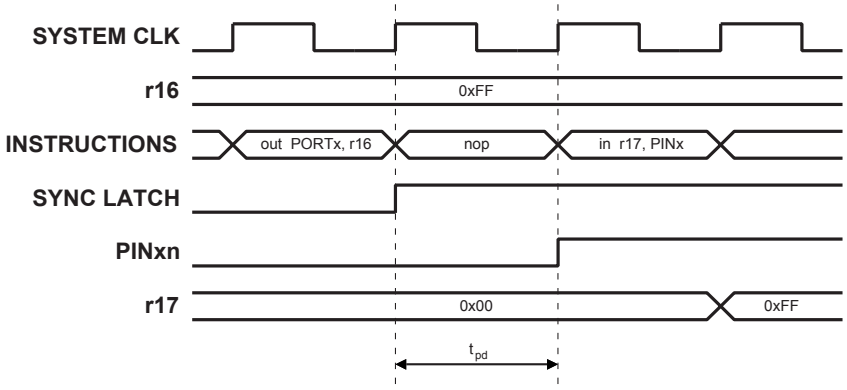


Figure 9-5. Synchronization when Reading a Software Assigned Pin Value



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]

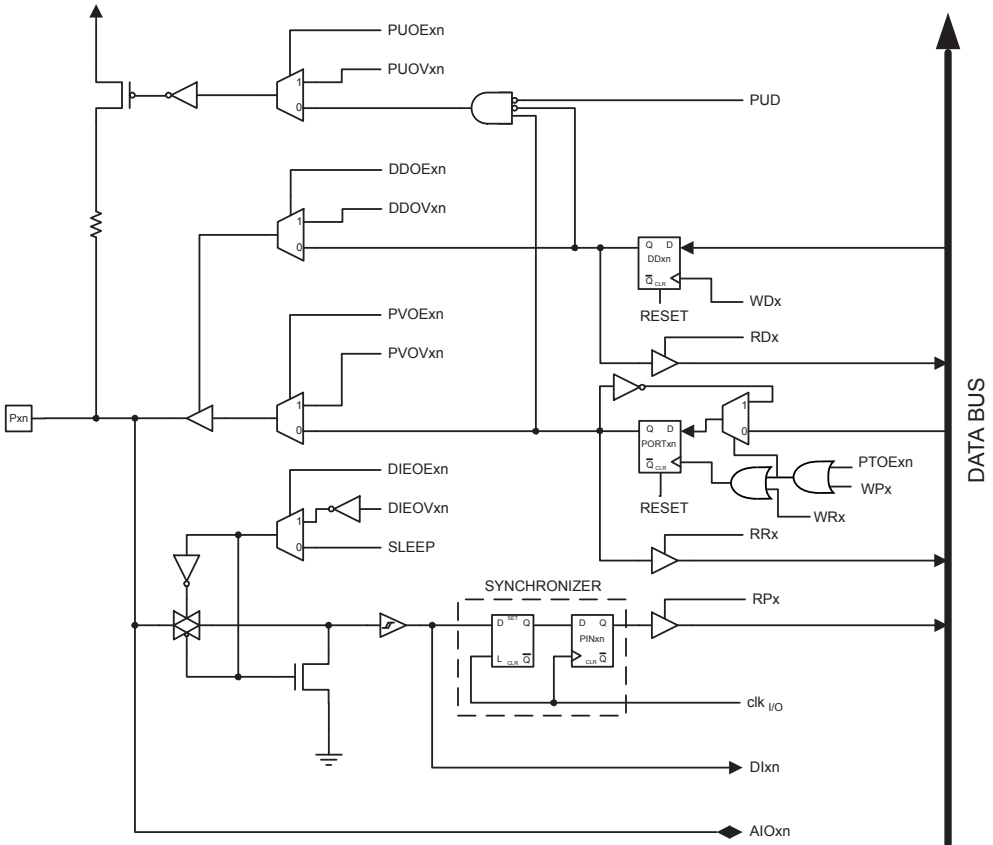
```
...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi   r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
ldi   r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
out   PORTB,r16
out   DDRB,r17
; Insert nop for synchronization
nop
; Read port pins
in    r16,PINB
...
```

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 9-6. Alternate Port Functions[1]



PUOExn: Pxn PULL-UP OVERRIDE ENABLE
PUOVxn: Pxn PULL-UP OVERRIDE VALUE
DDOExn: Pxn DATA DIRECTION OVERRIDE ENABLE
DDOVxn: Pxn DATA DIRECTION OVERRIDE VALUE
PVOExn: Pxn PORT VALUE OVERRIDE ENABLE
PVOVxn: Pxn PORT VALUE OVERRIDE VALUE
DIEOExn: Pxn DIGITAL INPUT-ENABLE OVERRIDE ENABLE
DIEOVxn: Pxn DIGITAL INPUT-ENABLE OVERRIDE VALUE
SLEEP: SLEEP CONTROL
PTOExn: Pxn, PORT TOGGLE OVERRIDE ENABLE

PUD: PULLUP DISABLE
WDx: WRITE DDRx
RDx: READ DDRx
RRx: READ PORTx REGISTER
WRx: WRITE PORTx
RPx: READ PORTx PIN
WPx: WRITE PINx
$clk_{I/O}$: I/O CLOCK
DIxn: DIGITAL INPUT PIN n ON PORTx
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.

Table 9-4 and Table 9-5 relate the alternate functions of Port A to the overriding signals shown in Figure 9-6 on page 73.

Table 9-4. Overriding Signals for Alternate Functions in PA[7:4]

Signal Name	PA7/PCINT7/ ADC7/AIN1/ XREF/AREF	PA6/PCINT6/ ADC6/AIN0/$\overline{SS}$	PA5/PCINT5/ ADC5/T1/USCK/ SCL/SCK	PA4/PCINT4/ ADC4/ICP1/DI/ SDA/MOSI
PUOE	0	SPE & $\overline{MSTR}$	SPE & $\overline{MSTR}$	SPE & $\overline{MSTR}$
PUOV	0	PORTA6 & $\overline{PUD}$	PORTA5 & $\overline{PUD}$	PORTA4 & $\overline{PUD}$
DDOE	0	SPE & $\overline{MSTR}$	(SPE & $\overline{MSTR}$) | (USI_2_WIRE & USIPOS)	(SPE & $\overline{MSTR}$) | (USI_2_WIRE & USIPOS)
DDOV	0	0	(USI_SCL_HOLD | $\overline{PORTA5}$) & DDRA6	{ (SPE & $\overline{MSTR}$) ? (0) : ($\overline{USI\_SHIFTOUT}$ | $\overline{PORTA4}$) & 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 -/- $\overline{SS}$	PCINT5 -/- T1 -/- USCK -/- SCL -/- SCK	PCINT4 -/- ICP1 -/- DI -/- SDA -/- MOSI
AIO	ADC7 -/- AIN1 -/- XREF -/- AREF	ADC6 -/- AIN0	ADC5	ADC4

In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC0A pin. Setting the COM0A[1:0] bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM0A[1:0] to three (See Table 10-2 on page 100). 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 setting (or clearing) the OC0A Register at the compare match between OCR0A and TCNT0, and clearing (or setting) the OC0A Register at the timer clock cycle the counter is cleared (changes from MAX to BOTTOM).

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

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

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 fast PWM mode. If the OCR0A is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR0A equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM0A[1:0] bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC0A to toggle its logical level on each compare match (COM0A[1:0] = 1). The waveform generated will have a maximum frequency of $f_{oc}0_A = f_{clk\_I/O}/2$ when OCR0A is set to zero. This feature is similar to the OC0A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

10.7.4 Phase Correct PWM Mode

The phase correct PWM mode (WGM0[1:0] = 1) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to MAX and then from MAX to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC0A) is cleared on the compare match between TCNT0 and OCR0A while upcounting, and set on the compare match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode is fixed to eight bits. In phase correct PWM mode the counter is incremented until the counter value matches MAX. When the counter reaches MAX, it changes the count direction. The TCNT0 value will be equal to MAX for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 10-7. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0A and TCNT0.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the Tn pin. The Clock Select logic block controls which clock source and edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{Tn}).

The double buffered Output Compare Registers (OCR1A/B) are compared with the Timer/Counter value at all time. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pins See ”Output Compare Units” on page 118.. The compare match event will also set the Compare Match Flag (OCF1A/B) which can be used to generate an Output Compare interrupt request.

The Input Capture Register can capture the Timer/Counter value at a given external (edge triggered) event on either the Input Capture pin (ICP1) or on the Analog Comparator pins (See ”AnaComp - Analog Comparator” on page 210.). The Input Capture unit includes a digital filtering unit (Noise Canceler) for reducing the chance of capturing noise spikes.

The TOP value, or maximum Timer/Counter value, can in some modes of operation be defined by either the OCR1A Register, the ICR1 Register, or by a set of fixed values. When using OCR1A as TOP value in a PWM mode, the OCR1A Register can not be used for generating a PWM output. However, the TOP value will in this case be double buffered allowing the TOP value to be changed in run time. If a fixed TOP value is required, the ICR1 Register can be used as an alternative, freeing the OCR1A to be used as PWM output.

12.2.2 Definitions

The following definitions are used extensively throughout the section:

BOTTOM	The counter reaches the BOTTOM when it becomes 0x0000.
MAX	The counter reaches its MAXimum when it becomes 0xFFFF (decimal 65,535).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be one of the fixed values: 0x00FF, 0x01FF, or 0x03FF, or to the value stored in the OCR1A or ICR1 Register. The assignment is dependent of the mode of operation.

12.3 Accessing 16-bit Registers

The TCNT1, OCR1A/B, and ICR1 are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. Each 16-bit timer has a single 8-bit register for temporary storing of the high byte of the 16-bit access. The same temporary register is shared between all 16-bit registers within each 16-bit timer. Accessing the low byte triggers the 16-bit read or write operation. When the low byte of a 16-bit register is written by the CPU, the high byte stored in the temporary register, and the low byte written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read.

Not all 16-bit accesses uses the temporary register for the high byte. Reading the OCR1A/B 16-bit registers does not involve using the temporary register.

To do a 16-bit write, the high byte must be written before the low byte. For a 16-bit read, the low byte must be read before the high byte.

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/B[1:0] to three (see Table 12-2 on page 132). 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:

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

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/B[1:0] bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC1A to toggle its logical level on each compare match (COM1A[1:0] = 1). The waveform generated will have a maximum frequency of $f_{OC1_A} = f_{clk\_I/O}/2$ when OCR1A is set to zero (0x0000). This feature is similar to the OC1A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

12.9.4 Phase Correct PWM Mode

The phase correct Pulse Width Modulation or phase correct PWM mode (WGM1[3:0] = 1, 2, 3, 10, or 11) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1A/B) is cleared on the compare match between TCNT1 and OCR1A/B while upcounting, and set on the compare match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode can be fixed to 8-, 9-, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated by using the following equation:

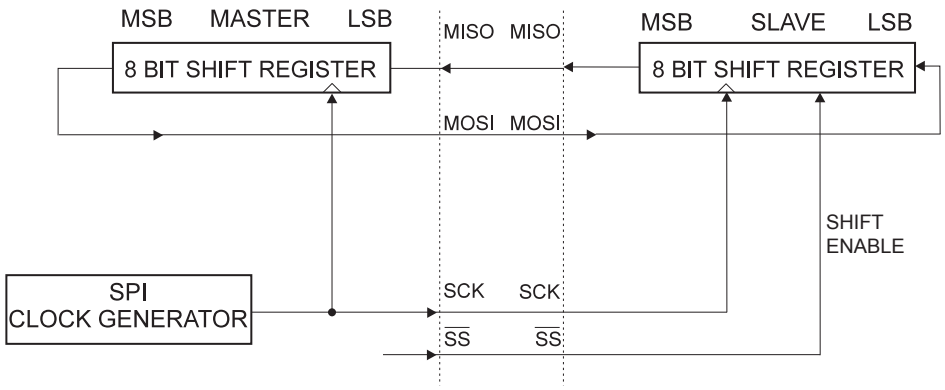
$$R_{PCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

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

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

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

Figure 13-2. SPI Master-slave Interconnection



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

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

13.2 $\overline{SS}$ Pin Functionality

13.2.1 Slave Mode

When the SPI is configured as a Slave, the Slave Select ($\overline{SS}$)pin is always input. When $\overline{SS}$ is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When $\overline{SS}$ is driven high, all pins are inputs, and the SPI is passive, which means that it will not receive incoming data. Note that the SPI logic will be reset once the $\overline{SS}$ pin is driven high.

The $\overline{SS}$ pin is useful for packet/byte synchronization to keep the slave bit counter synchronous with the master clock generator. When the $\overline{SS}$ pin is driven high, the SPI slave will immediately reset the send and receive logic, and drop any partially received data in the Shift Register.

13.2.2 Master Mode

When the SPI is configured as a Master (MSTR in SPCR is set), the user can determine the direction of the $\overline{SS}$ pin.

If $\overline{SS}$ is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the $\overline{SS}$ pin of the SPI Slave.

If $\overline{SS}$ is configured as an input, it must be held high to ensure Master SPI operation. If the $\overline{SS}$ pin is driven low by peripheral circuitry when the SPI is configured as a Master with the $\overline{SS}$ pin defined as an input, the SPI system interprets this as another master selecting the SPI as a slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. The MSTR bit in SPCR is cleared and the SPI system becomes a Slave. As a result of the SPI becoming a Slave, the MOSI and SCK pins become inputs.
2. The SPIF flag in SPSR is set, and if the SPI interrupt is enabled, and the I-bit in SREG is set, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in Master mode, and there exists a possibility that $\overline{SS}$ is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a slave select, it must be set by the user to re-enable SPI Master mode.

13.2.3 SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	**SPIE**	**SPE**	**DORD**	**MSTR**	**CPOL**	**CPHA**	**SPR1**	**SPR0**	**SPCR**
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 – SPIE: SPI Interrupt Enable**

This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and if the Global Interrupt Enable bit in SREG is set.

- **Bit 6 – SPE: SPI Enable**

When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

14.5.5 USIPP – USI Pin Position

Bit	7	6	5	4	3	2	1	0	
(0xBC)	–	–	–	–	–	–	–	USIPOS	USIPP
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:1 – Res: Reserved Bits**

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

- **Bit 0 – USIPOS: USI Pin Position**

Setting or clearing this bit changes the USI pin position.

Table 14-3. USI Pin Position

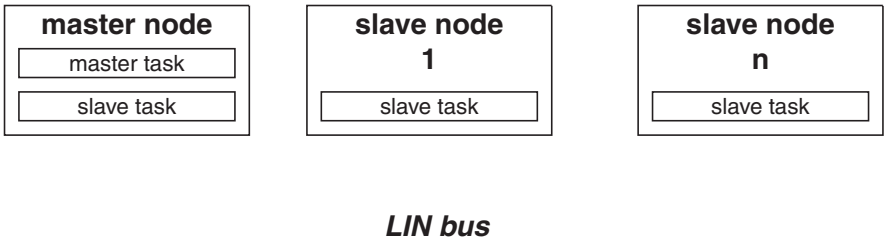
USIPOS		USI Pin Position	
0	PortB (Default)	DI, SDA	PB0 - (PCINT8/OC1AU)
		DO	PB1 - (PCINT9/OC1BU)
		USCK, SCL	PB2 - (PCINT10/OC1AV)
1	Port A (Alternate)	DI, SDA	PA4 - (PCINT4/ADC4/ICP1/MOSI)
		DO	PA2 - (PCINT2/ADC2/OC0A/MISO)
		USCK, SCL	PA5 - (PCINT5/ADC5/T1/SCK)

15.3 LIN Protocol

15.3.1 Master and Slave

A LIN cluster consists of one master task and several slave tasks. A master node contains the master task as well as a slave task. All other nodes contain a slave task only.

Figure 15-1. LIN cluster with one master node and “n” slave nodes



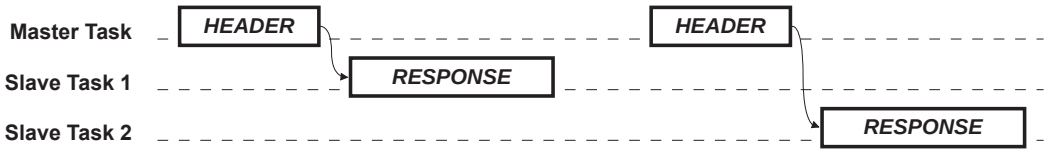
The master task decides when and which frame shall be transferred on the bus. The slave tasks provide the data transported by each frame. Both the master task and the slave task are parts of the Frame handler

15.3.2 Frames

A frame consists of a header (provided by the master task) and a response (provided by a slave task).

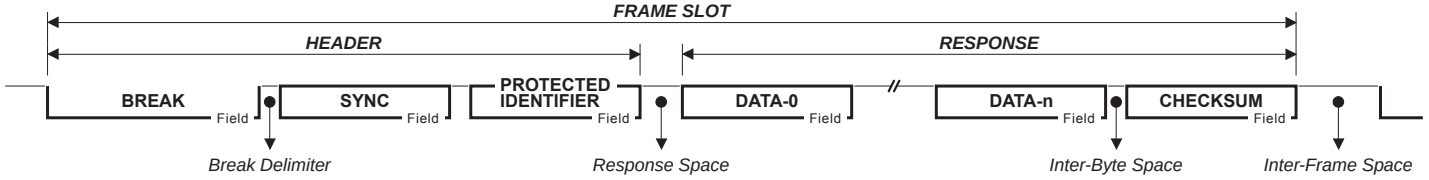
The header consists of a BREAK and SYNC pattern followed by a PROTECTED IDENTIFIER. The identifier uniquely defines the purpose of the frame. The slave task appointed for providing the response associated with the identifier transmits it. The response consists of a DATA field and a CHECKSUM field.

Figure 15-2. Master and slave tasks behavior in LIN frame



The slave tasks waiting for the data associated with the identifier receives the response and uses the data transported after verifying the checksum.

Figure 15-3. Structure of a LIN frame



Each byte field is transmitted as a serial byte, LSB first.

17.11.2 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
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 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set when an ADC conversion completes and the data registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

- **Bits 2:0 – ADPS[2:0]: ADC Prescaler Select Bits**

These bits determine the division factor between the system clock frequency and the input clock to the ADC.

Table 17-6. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8

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 225 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.

20.2.4 Reading the Signature Row from Software

To read the Signature Row from software, load the Z-pointer with the signature byte address given in Table 20-1 on page 220 and set the SIGRD and SPMEN bits in SPMCSR. When an LPM instruction is executed within three CPU cycles after the SIGRD and SPMEN bits are set in SPMCSR, the signature byte value will be loaded in the destination register. The SIGRD and SPMEN bits will auto-clear upon completion of reading the Signature Row Lock bits or if no LPM instruction is executed within three CPU cycles. When SIGRD and SPMEN are cleared, LPM will work as described in the Instruction set Manual.

Table 20-1. Signature Row Addressing

Signature Byte	Z-Pointer Address
Device Signature Byte 0	0x0000
Device Signature Byte 1	0x0002
Device Signature Byte 2	0x0004
8MHz RC Oscillator Calibration Byte	0x0001
TSOFFSET - Temp Sensor Offset	0x0003
TSGAIN - Temp Sensor Gain	0x0005

Note: All other addresses are reserved for future use.

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} < 12 MHz, **3** CPU clock cycles for f_{ck} >= 12 MHz

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

21.8.1 Serial Programming Algorithm

When writing serial data to the 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 for timing details.

To program and verify the ATtiny87/167 in the Serial Programming mode, the following sequence is recommended (see four byte instruction formats in Table 21-15 on page 239):

1. Power-up sequence:
   Apply power between V_{CC} 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 20 ms and enable serial programming by sending the Programming Enable serial instruction to pin MOSI.
3. The serial programming instructions will not work if the communication is out of synchronization. When in sync. the second byte (0x53), will echo back when issuing the third byte of the Programming Enable instruction. Whether the echo is correct or not, all four bytes of the instruction must be transmitted. If the 0x53 did not echo back, give $\overline{\text{RESET}}$ a positive pulse and issue a new Programming Enable command.
4. The Flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 5 LSB of the address and data together with the Load Program memory Page instruction. To ensure correct loading of the page, the data low byte must be loaded before data high byte is applied for a given address. The Program memory Page is stored by loading the Write Program memory Page instruction with the 6 MSB of the address. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least t_{WD_FLASH} before issuing the next page. (See Table 21-14) Accessing the serial programming interface before the Flash write operation completes can result in incorrect programming.
5. **A:** The EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte. (See Table 21-14) In a chip erased device, no 0xFFs in the data file(s) need to be programmed.
   B: The EEPROM array is programmed one page at a time. The Memory page is loaded one byte at a time by supplying the 2 LSB of the address and data together with the Load EEPROM Memory Page instruction. The EEPROM Memory Page is stored by loading the Write EEPROM Memory Page Instruction with the 6 MSB of the address. When using EEPROM page access only byte locations loaded with the Load EEPROM Memory Page instruction is altered. The remaining locations remain unchanged. If polling (RDY/$\overline{\text{BSY}}$) is not used, the used must wait at least t_{WD_EEPROM} before issuing the next page (See Table 21-8). In a chip erased device, no 0xFF in the data file(s) need to be programmed.
6. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO.

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

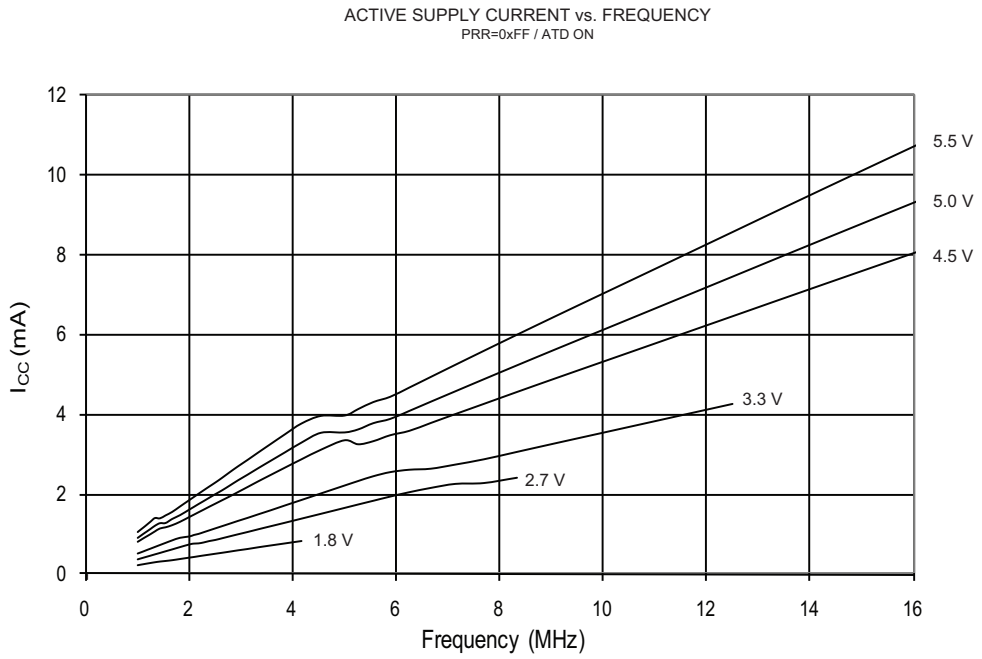
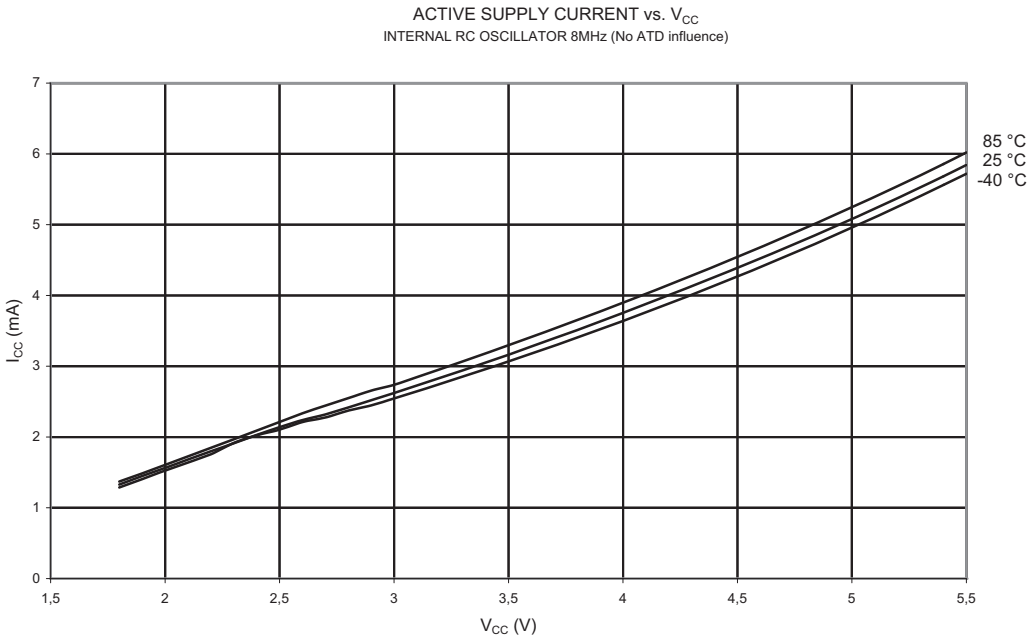


Figure 23-3. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)



Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**