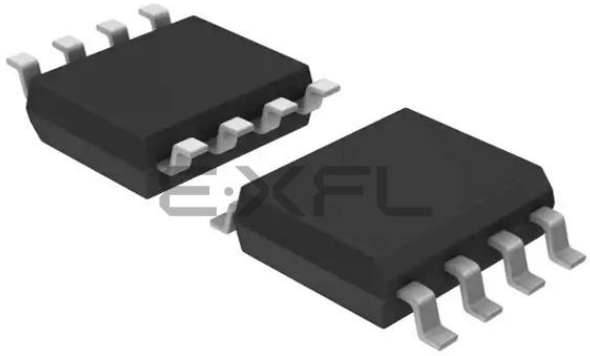


Welcome to E-XFL.COM

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	USI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	6
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 4x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	8-SOIC (0.209", 5.30mm Width)
Supplier Device Package	8-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny85-15sz

4.3.8 Erase

To erase a byte, the address must be written to EEAR. If the EEPMn bits are 0b01, writing the EEPE (within four cycles after EEMPE is written) will trigger the erase operation only (programming time is given in Table 19-1 on page 122). The EEPE bit remains set until the erase operation completes. While the device is busy programming, it is not possible to do any other EEPROM operations.

4.3.9 Write

To write a location, the user must write the address into EEAR and the data into EEDR. If the EEPMn bits are 0b10, writing the EEPE (within four cycles after EEMPE is written) will trigger the write operation only (programming time is given in Table 19-1 on page 122). The EEPE bit remains set until the write operation completes. If the location to be written has not been erased before write, the data that is stored must be considered as lost. While the device is busy with programming, it is not possible to do any other EEPROM operations.

The calibrated oscillator is used to time the EEPROM accesses. Make sure the oscillator frequency is within the requirements described in Section 5.6.1 “Oscillator Calibration Register – OSCCAL” on page 24.

The following code examples show one assembly and one C function for erase, write, or atomic write of the EEPROM. The examples assume that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts will occur during execution of these functions

Assembly Code Example

```
EEPROM_write:
        ; Wait for completion of previous write
        sbic    EECR,EEPE
        rjmp    EEPROM_write
        ; Set Programming mode
        ldi     r16, (0<<EEPM1)|(0<<EEPM0)
        out     EECR, r16
        ; Set up address (r17) in address register
        out     EEARL, r17
        ; Write data (r16) to data register
        out     EEDR,r16
        ; Write logical one to EEMWE
        sbi     EECR,EEMWE
        ; Start eeprom write by setting EEWE
        sbi     EECR,EEWE
        ret
```

C Code Example

```
void EEPROM_write(unsigned char ucAddress, unsigned char ucData)
{
        /* Wait for completion of previous write */
        while(EECR & (1<<EEPE))
        ;
        /* Set Programming mode */
        EECR = (0<<EEPM1)|(0>>EEPM0)
        /* Set up address and data registers */
        EEARL = ucAddress;
        EEDR = ucData;
        /* Write logical one to EEMWE */
        EECR |= (1<<EEMWE);
        /* Start eeprom write by setting EEWE */
        EECR |= (1<<EEWE);
}
```

9.2.1 Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in Section 9.4 “Register Description for I/O-Ports” on page 53, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

The DDxn bit in the DDRx register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when reset condition becomes active, even if no clocks are running.

If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

9.2.2 Toggling the Pin

Writing a logic one to PINxn toggles the value of PORTxn, independent on the value of DDRxn. Note that the SBI instruction can be used to toggle one single bit in a port.

9.2.3 Switching Between Input and Output

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled {DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the MCUCR Register can be set to disable all pull-ups in all ports.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b10) as an intermediate step.

Table 9-1 summarizes the control signals for the pin value.

Table 9-1. Port Pin Configurations

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output low (sink)
1	1	X	Output	No	Output high (source)

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 5 as input with a pull-up assigned to port pin 4. 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<<PB4)|(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<<PB4)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
/* Insert nop for synchronization*/
_NOP();
/* 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 and 4, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

9.2.5 Digital Input Enable and Sleep Modes

As shown in Figure 9-2, the digital input signal can be clamped to ground at the input of the schmitt-trigger. The signal denoted SLEEP in the figure, is set by the MCU sleep controller in power-down mode, power-save mode, and standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in Section 9.3 “Alternate Port Functions” on page 48.

If a logic high level (“one”) is present on an asynchronous external interrupt pin configured as “interrupt on rising edge, falling edge, or any logic change on pin” while the external interrupt is *not* enabled, the corresponding external interrupt flag will be set when resuming from the above mentioned sleep mode, as the clamping in these sleep mode produces the requested logic change.

9.2.6 Unconnected Pins

If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (reset, active mode and idle mode).

The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pulldown. Connecting unused pins directly to V_{CC} or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

Table 9-2 summarizes the function of the overriding signals. The pin and port indexes from Figure 9-5 on page 48 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 9-2. Generic Description of Overriding Signals for Alternate Functions

Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up override value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD register bits.
DDOE	Data direction override enable	If this signal is set, the output driver enable is controlled by the DDOV signal. If this signal is cleared, the output driver is enabled by the DDxn register bit.
DDOV	Data direction override value	If DDOE is set, the output driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn register bit.
PVOE	Port value override enable	If this signal is set and the output driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the output driver is enabled, the port value is controlled by the PORTxn register bit.
PVOV	Port value override value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn register bit.
PTOE	Port toggle override enable	If PTOE is set, the PORTxn register bit is inverted.
DIEOE	Digital input enable override enable	If this bit is set, the digital Input Enable is controlled by the DIEOV signal. If this signal is cleared, the Digital Input Enable is determined by MCU state (Normal mode, sleep mode).
DIEOV	Digital input enable Override value	If DIEOE is set, the digital input is enabled/disabled when DIEOV is set/cleared, regardless of the MCU state (Normal mode, sleep mode).
DI	Digital input	This is the digital input to alternate functions. In the figure, the signal is connected to the output of the schmitt-trigger but before the synchronizer. Unless the digital input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog input/output	This is the analog input/output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

The following subsections shortly describe the alternate functions for each port, and relate the overriding signals to the alternate function. Refer to the alternate function description for further details.

9.3.1 MCU Control Register – MCUCR

Bit	7	6	5	4	3	2	1	0	
	BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	MCUCR
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 6 – PUD: Pull-up Disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See Section 9.2.1 “Configuring the Pin” on page 45 for more details about this feature.

10. External Interrupts

The external interrupts are triggered by the INT0 pin or any of the PCINT5..0 pins. Observe that, if enabled, the interrupts will trigger even if the INT0 or PCINT5..0 pins are configured as outputs. This feature provides a way of generating a software interrupt. Pin change interrupts PCI will trigger if any enabled PCINT5..0 pin toggles. The PCMSK register control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT5..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 interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the MCU control register – MCUCR. When the INT0 interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT0 requires the presence of an I/O clock, described in Section 5.1 “Clock Systems and their Distribution” on page 19. Low level interrupt on INT0 is detected asynchronously. This implies that this interrupt 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, 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 Section 5. “System Clock and Clock Options” on page 19.

10.1 MCU Control Register – MCUCR

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

Bit	7	6	5	4	3	2	1	0	
	BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 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 10-1. 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 10-1. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

10.2 General Interrupt Mask Register – GIMSK

Bit	7	6	5	4	3	2	1	0	
	–	INT0	PCIE	–	–	–	–	–	GIMSK
Read/Write	R	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7, 4..0 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny25/45/85 and will always read as zero.

The OCR0x registers are double buffered when using any of the pulse width modulation (PWM) modes. For the normal and clear timer on compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR0x compare registers to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR0x register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR0x buffer register, and if double buffering is disabled the CPU will access the OCR0x directly.

11.4.1 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 (FOC0x) bit. Forcing compare match will not set the OCF0x flag or reload/clear the timer, but the OC0x pin will be updated as if a real compare match had occurred (the COM0x1:0 bits settings define whether the OC0x pin is set, cleared or toggled).

11.4.2 Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 register will block any compare match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0x to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

11.4.3 Using the Output Compare Unit

Since writing TCNT0 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT0 when using the output compare unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is down-counting.

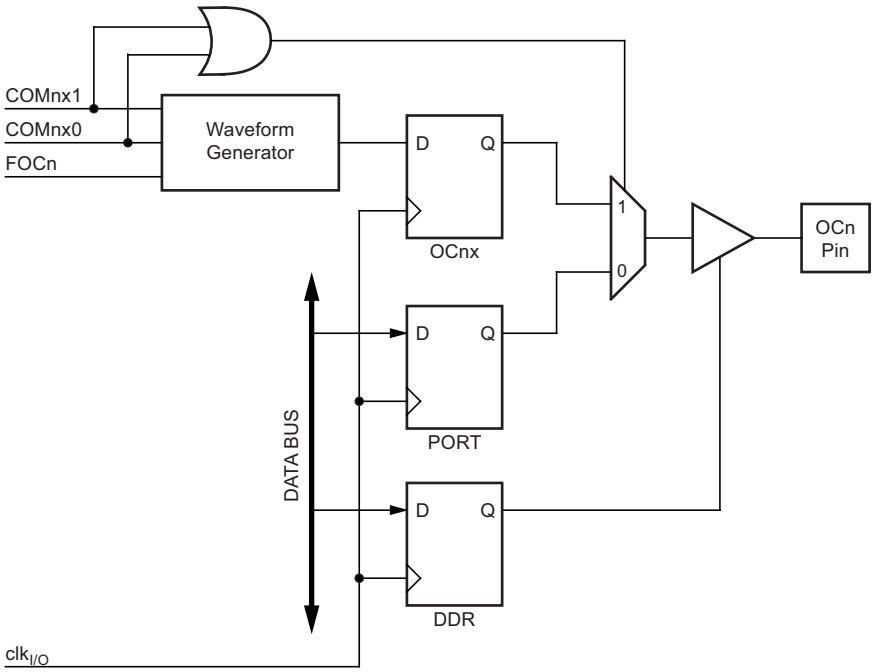
The setup of the OC0x should be performed before setting the data direction register for the port pin to output. The easiest way of setting the OC0x value is to use the force output compare (FOC0x) strobe bits in normal mode. The OC0x registers keep their values even when changing between waveform generation modes.

Be aware that the COM0x1:0 bits are not double buffered together with the compare value. Changing the COM0x1:0 bits will take effect immediately.

11.5 Compare Match Output Unit

The compare output mode (COM0x1:0) bits have two functions. The waveform generator uses the COM0x1:0 bits for defining the output compare (OC0x) state at the next compare match. Also, the COM0x1:0 bits control the OC0x pin output source. Figure 11-4 on page 60 shows a simplified schematic of the logic affected by the COM0x1:0 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 COM0x1:0 bits are shown. When referring to the OC0x state, the reference is for the internal OC0x register, not the OC0x pin. If a system reset occur, the OC0x register is reset to “0”.

Figure 11-4. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the output compare (OC0x) from the waveform generator if either of the COM0x1:0 bits are set. However, the OC0x pin direction (input or output) is still controlled by the data direction register (DDR) for the port pin. The Data direction register bit for the OC0x pin (DDR_OC0x) must be set as output before the OC0x value is visible on the pin. The port override function is independent of the waveform generation mode.

The design of the output compare pin logic allows initialization of the OC0x state before the output is enabled. Note that some COM0x1:0 bit settings are reserved for certain modes of operation. See Section 11.8 “8-bit Timer/Counter Register Description” on page 66

11.5.1 Compare Output Mode and Waveform Generation

The waveform generator uses the COM0x1:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COM0x1:0 = 0 tells the waveform generator that no action on the OC0x register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 11-2 on page 66. For fast PWM mode, refer to Table 11-3 on page 67, and for phase correct PWM refer to Table 11-4 on page 67.

A change of the COM0x1:0 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 FOC0x strobe bits.

11.7 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. The figures include information on when interrupt flags are set. Figure 11-8 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 11-8. Timer/Counter Timing Diagram, no Prescaling

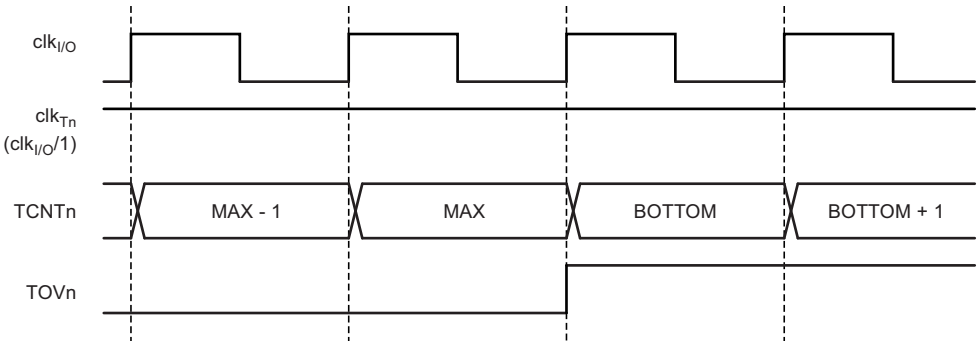


Figure 11-9 shows the same timing data, but with the prescaler enabled.

Figure 11-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)

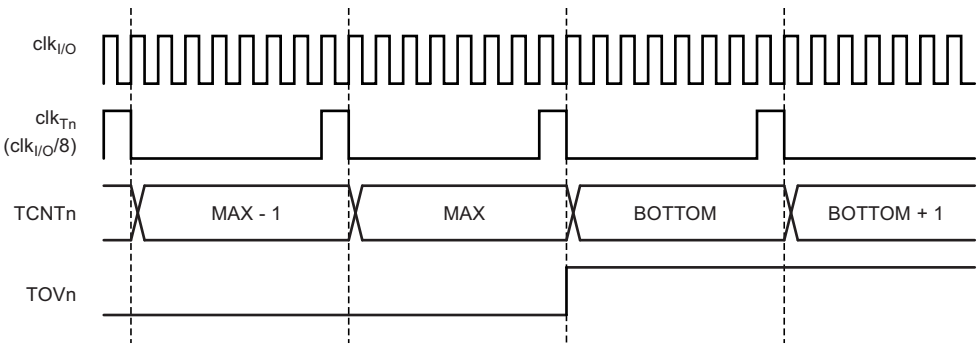


Figure 11-10 shows the setting of OCF0B in all modes and OCF0A in all modes except CTC mode and PWM mode, where OCR0A is TOP.

Figure 11-10. Timer/Counter Timing Diagram, Setting of OCF0x, with Prescaler ($f_{clk\_I/O}/8$)

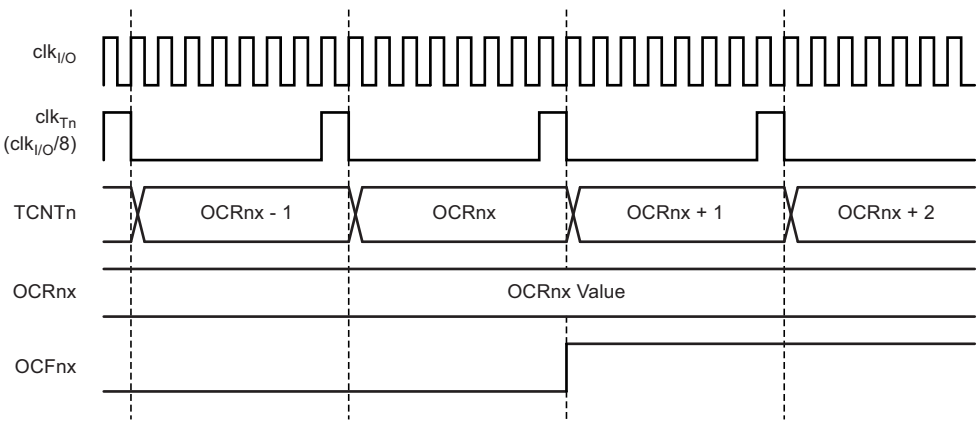
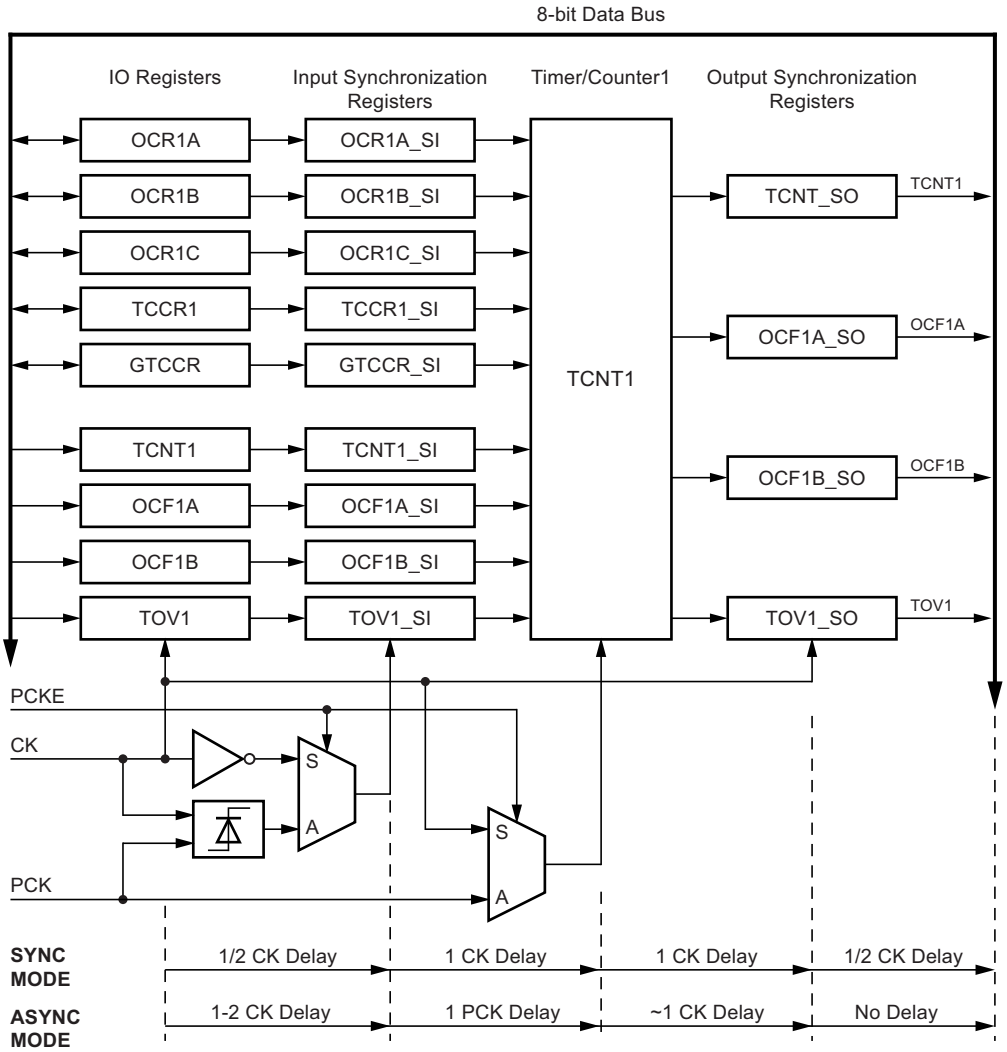


Figure 13-2. Timer/Counter 1 Synchronization Register Block Diagram



Timer/Counter1 and the prescaler allow running the CPU from any clock source while the prescaler is operating on the fast 64MHz (or 32MHz in low speed mode) PCK clock in the asynchronous mode.

Note that the system clock frequency must be lower than one third of the PCK frequency. The synchronization mechanism of the asynchronous Timer/Counter1 needs at least two edges of the PCK when the system clock is high. If the frequency of the system clock is too high, it is a risk that data or control values are lost.

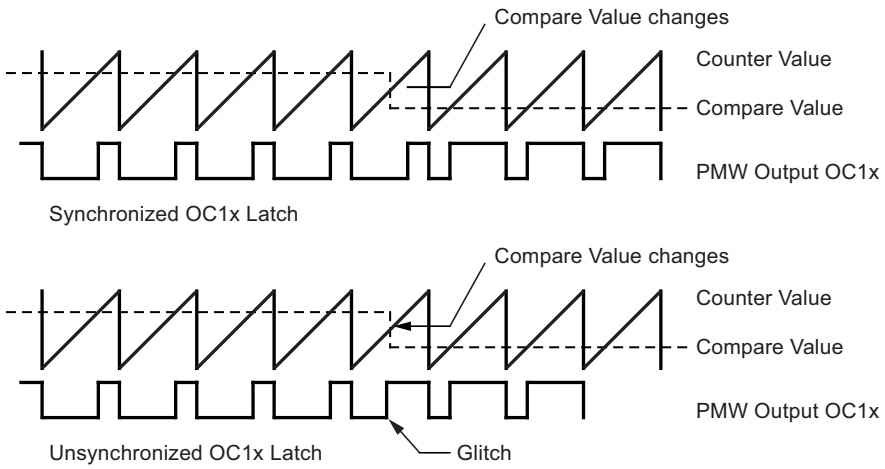
The following Figure 13-3 on page 76 shows the block diagram for Timer/Counter1.

Table 13-4. Compare Mode Select in PWM Mode

COM11	COM10	Effect on Output Compare Pins
0	0	OC1x not connected. $\overline{OC1x}$ not connected.
0	1	OC1x cleared on compare match. Set whenTCNT1 = $01. $\overline{OC1x}$ set on compare match. Cleared when TCNT1 = $00.
1	0	OC1x cleared on compare match. Set when TCNT1 = $01. $\overline{OC1x}$ not connected.
1	1	OC1x Set on compare match. Cleared when TCNT1= $01. $\overline{OC1x}$ not connected.

Note that in PWM mode, writing to the output compare registers OCR1A or OCR1B, the data value is first transferred to a temporary location. The value is latched into OCR1A or OCR1B when the Timer/Counter reaches OCR1C. This prevents the occurrence of odd-length PWM pulses (glitches) in the event of an unsynchronized OCR1A or OCR1B. See Figure 13-5 for an example.

Figure 13-5. Effects of Unsynchronized OCR Latching



During the time between the write and the latch operation, a read from OCR1A or OCR1B will read the contents of the temporary location. This means that the most recently written value always will read out of OCR1A or OCR1B.

When OCR1A or OCR1B contain $00 or the top value, as specified in OCR1C register, the output PB1(OC1A) or PB3(OC1B) is held low or high according to the settings of COM1A1/COM1A0. This is shown in Table 13-5 on page 83.

Table 13-5. PWM Outputs OCR1x = $00 or OCR1C, x = A or B

COM1x1	COM1x0	OCR1x	Output OC1x	Output $\overline{OC1x}$
0	1	$00	L	H
0	1	OCR1C	H	L
1	0	$00	L	Not connected.
1	0	OCR1C	H	Not connected.
1	1	$00	H	Not connected.
1	1	OCR1C	L	Not connected.

In PWM mode, the timer overflow flag - TOV1 is set when the TCNT1 counts to the OCR1C value and the TCNT1 is reset to $00. The timer overflow interrupt1 is executed when TOV1 is set provided that timer overflow interrupt and global interrupts are enabled. This also applies to the timer output compare flags and interrupts.

The frequency of the PWM will be timer clock 1 frequency divided by (OCR1C value + 1). See the following equation:

$$f_{PWM} = \frac{f_{TCK1}}{(OCR1C + 1)}$$

Resolution shows how many bit is required to express the value in the OCR1C register. It is calculated by following equation $Resolution_{PWM} = \log_2(OCR1C + 1)$.

Table 13-6. Timer/Counter1 Clock Prescale Select in the Asynchronous Mode

PWM Frequency	Clock Selection	CS13..CS10	OCR1C	RESOLUTION
20kHz	PCK/16	0101	199	7.6
30kHz	PCK/16	0101	132	7.1
40kHz	PCK/8	0100	199	7.6
50kHz	PCK/8	0100	159	7.3
60kHz	PCK/8	0100	132	7.1
70kHz	PCK/4	0011	228	7.8
80kHz	PCK/4	0011	199	7.6
90kHz	PCK/4	0011	177	7.5
100kHz	PCK/4	0011	159	7.3
110kHz	PCK/4	0011	144	7.2
120kHz	PCK/4	0011	132	7.1
130kHz	PCK/2	0010	245	7.9
140kHz	PCK/2	0010	228	7.8
150kHz	PCK/2	0010	212	7.7
160kHz	PCK/2	0010	199	7.6
170kHz	PCK/2	0010	187	7.6
180kHz	PCK/2	0010	177	7.5
190kHz	PCK/2	0010	167	7.4
200kHz	PCK/2	0010	159	7.3
250kHz	PCK	0001	255	8.0
300kHz	PCK	0001	212	7.7
350kHz	PCK	0001	182	7.5
400kHz	PCK	0001	159	7.3
450kHz	PCK	0001	141	7.1
500kHz	PCK	0001	127	7.0

The following code demonstrates how to use the USI module as a SPI master with maximum speed (fsck = fck/4):

```
SPITransfer_Fast:

       sts    USIDR,r16
       ldi    r16,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)
       ldi    r17,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)|(1<<USICLK)

       sts    USICR,r16; MSB
       sts    USICR,r17
       sts    USICR,r16
       sts    USICR,r17
       sts    USICR,r16
       sts    USICR,r17
       sts    USICR,r16
       sts    USICR,r17
       sts    USICR,r16
       sts    USICR,r17
       sts    USICR,r16
       sts    USICR,r17
       sts    USICR,r16
       sts    USICR,r17
       sts    USICR,r16; LSB
       sts    USICR,r17

       lds    r16,USIDR
ret
```

15.2.3 SPI Slave Operation Example

The following code demonstrates how to use the USI module as a SPI slave:

```
init:
       ldi    r16,(1<<USIWM0)|(1<<USICS1)
       sts    USICR,r16
...
SlaveSPITransfer:
       sts    USIDR,r16
       ldi    r16,(1<<USIOIF)
       sts    USISR,r16
SlaveSPITransfer_loop:
       lds    r16, USISR
       sbrs   r16, USIOIF
       rjmp   SlaveSPITransfer_loop
       lds    r16,USIDR
       ret
```

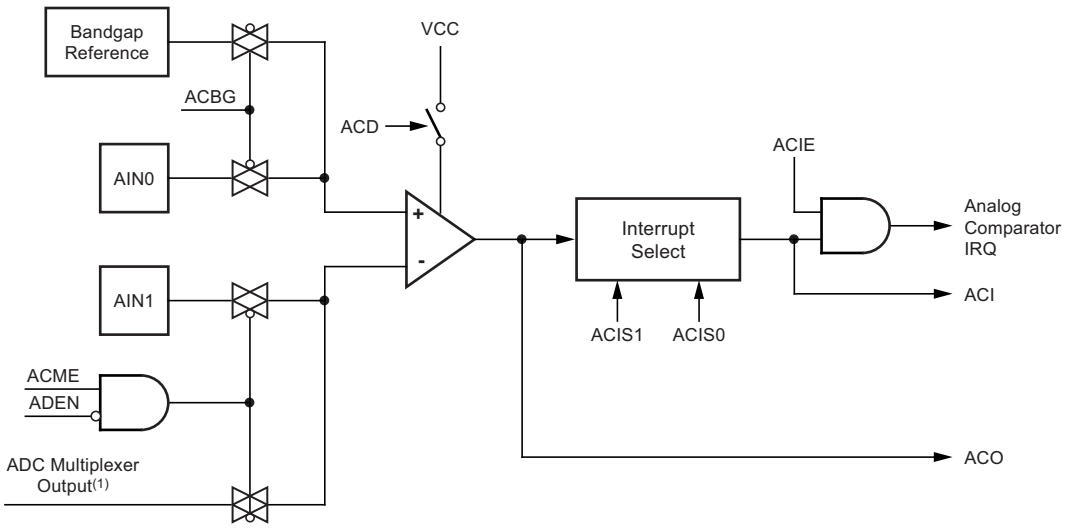
The code is size optimized using only eight instructions (+ ret). The code example assumes that the DO is configured as output and USCK pin is configured as input in the DDR register. The value stored in register r16 prior to the function is called is transferred to the master device, and when the transfer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

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

16. Analog Comparator

The analog comparator compares the input values on the positive pin AIN0 and negative pin AIN1. When the voltage on the positive pin AIN0 is higher than the voltage on the negative pin AIN1, the analog comparator output, ACO, is set. The comparator can trigger a separate interrupt, exclusive to the analog comparator. The user can select interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 16-1.

Figure 16-1. Analog Comparator Block Diagram(2)



Notes: 1. See Table 16-2 on page 100.

2. Refer to Figure 1 on page 2 and Table 9-5 on page 52 for analog comparator pin placement.

16.1 ADC Control and Status Register B – ADCSRB

Bit	7	6	5	4	3	2	1	0	
	BIN	ACME	IPR	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6 – ACME: Analog Comparator Multiplexer Enable**

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer selects the negative input to the analog comparator. When this bit is written logic zero, AIN1 is applied to the negative input of the analog comparator. For a detailed description of this bit, see Section 16.3 “Analog Comparator Multiplexed Input” on page 100.

16.2 Analog Comparator Control and Status Register – ACSR

Bit	7	6	5	4	3	2	1	0	
	ACD	**ACBG**	**ACO**	**ACI**	**ACIE**	**–**	**ACIS1**	**ACIS0**	**ACSR**
Read/Write	R/W	R/W	R	R/W	R/W	R	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 in ACSR. Otherwise an interrupt can occur when the bit is changed.

20. Memory Programming

This section describes the different methods for programming the Atmel® ATtiny25/45/85 memories.

20.1 Program And Data Memory Lock Bits

The ATtiny25/45/85 provides two lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional security listed in Table 20-2. The lock bits can only be erased to “1” with the chip erase command.

Program memory can be read out via the debugWIRE interface when the DWEN fuse is programmed, even if the lock bits are set. Thus, when lock bit security is required, should always debugWIRE be disabled by clearing the DWEN fuse.

Table 20-1. Lock Bit Byte(1)

Lock Bit Byte	Bit No	Description	Default Value
	7	–	1 (unprogrammed)
	6	–	1 (unprogrammed)
	5	–	1 (unprogrammed)
	4	–	1 (unprogrammed)
	3	–	1 (unprogrammed)
	2	–	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

Note: 1. “1” means unprogrammed, “0” means programmed

Table 20-2. Lock Bit Protection Modes(1)(2)

Memory Lock Bits			Protection Type
LB Mode	LB2	LB1	
1	1	1	No memory lock features enabled.
2	1	0	Further programming of the flash and EEPROM is disabled in high-voltage and serial programming mode. The fuse bits are locked in both serial and high-voltage programming mode(1). debugWire is disabled.
3	0	0	Further programming and verification of the flash and EEPROM is disabled in high-voltage and serial programming mode. The fuse bits are locked in both serial and high-voltage programming mode(1). debugWire is disabled.

Notes: 1. Program the fuse bits before programming the LB1 and LB2.
2. “1” means unprogrammed, “0” means programmed

Table 20-5. Fuse Low Byte

Fuse Low Byte	Bit No	Description	Default Value
CKDIV8(1)	7	Divide clock by 8	0 (unprogrammed)
CKOUT(2)	6	Clock output enable	1 (unprogrammed)
SUT1	5	Select start-up time	1 (unprogrammed)(3)
SUT0	4	Select start-up time	0 (programmed)(3)
CKSEL3	3	Select clock source	0 (programmed)(4)
CKSEL2	2	Select clock source	0 (programmed)(4)
CKSEL1	1	Select clock source	1 (unprogrammed)(4)
CKSEL0	0	Select clock source	0 (programmed)(4)

Notes:
1. See Section 5.10 “System Clock Prescaler” on page 26 for details.
2. The CKOUT fuse allows the system clock to be output on PORTB4. See “Section 5.9 “Clock Output Buffer” on page 26 for details.
3. The default value of SUT1..0 results in maximum start-up time for the default clock source. See Table 5-7 on page 23 for details.
4. The default setting of CKSEL1..0 results in internal RC oscillator at 8.0MHz. See Table 5-6 on page 23 for details.

The status of the fuse bits is not affected by chip erase. Note that the fuse bits are locked if lock bit1 (LB1) is programmed. program the fuse bits before programming the lock bits.

20.2.1 Latching of Fuses

The fuse values are latched when the device enters programming mode and changes of the fuse values will have no effect until the part leaves programming mode. This does not apply to the EESAVE fuse which will take effect once it is programmed. The fuses are also latched on power-up in normal mode.

20.3 Signature Bytes

All Atmel® microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and high-voltage programming mode, also when the device is locked. The three bytes reside in a separate address space.

20.3.1 ATtiny25 Signature Bytes

1. 0x000: 0x1E (indicates manufactured by Atmel).
2. 0x001: 0x91 (indicates 2KB flash memory).
3. 0x002: 0x08 (indicates ATtiny25 device when 0x001 is 0x91).

20.3.2 ATtiny45 Signature Bytes

1. 0x000: 0x1E (indicates manufactured by Atmel).
2. 0x001: 0x92 (indicates 4KB flash memory).
3. 0x002: 0x06 (indicates ATtiny45 device when 0x001 is 0x92).

20.3.3 ATtiny85 Signature Bytes

1. 0x000: 0x1E (indicates manufactured by Atmel).
2. 0x001: 0x93 (indicates 8KB flash memory).
3. 0x002: 0x0B (indicates ATtiny85 device when 0x001 is 0x93).

20.4 Calibration Byte

Signature area of the Atmel ATtiny25/45/85 has one byte of calibration data for the internal RC oscillator. This byte resides in the high byte of address 0x000. During reset, this byte is automatically written into the OSCCAL register to ensure correct frequency of the calibrated RC oscillator.

Table 20-9. Minimum Wait Delay Before Writing the Next Flash or EEPROM Location

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

Figure 20-2. Serial Programming Waveforms

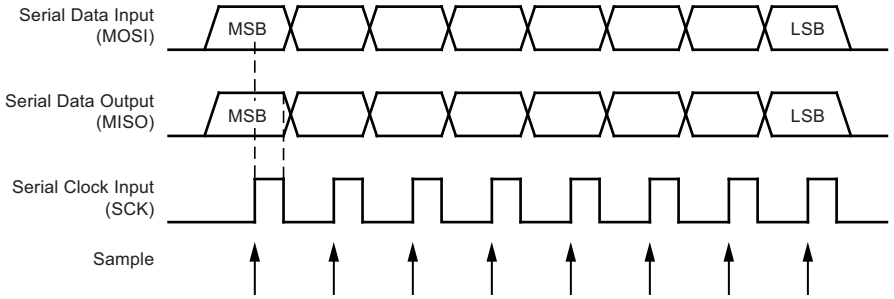


Table 20-10. Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable serial programming after RESET goes low.
Chip erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip erase EEPROM and flash.
Read program memory	0010 **H**000	0000 000**a**	**bbbb bbbb**	**oooo oooo**	Read **H** (high or low) data **o** from program memory at word address **a:b**.
Load program memory page	0100 **H**000	000x xxxx	xxx**b bbbb**	**iiii iiii**	Write **H** (high or low) data **i** to program memory page at word address **b**. Data low byte must be loaded before data high byte is applied within the same address.
Write program memory page	0100 1100	0000 000**a**	**bb**xx xxxx	xxxx xxxx	Write Program memory page at address **a:b**.
Read EEPROM memory	1010 0000	000x xxxx	xx**bb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **b**.
Write EEPROM memory	1100 0000	000x xxxx	xx**bb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **b**.
Load EEPROM memory page (Page access)	1100 0001	0000 0000	0000 00**bb**	**iiii iiii**	Load data **i** to EEPROM memory page buffer. After data is loaded, program EEPROM page.
Write EEPROM memory Page (page access)	1100 0010	00xx xxxx	xx**bb bb00**	xxxx xxxx	Write EEPROM page at address **b**.
Read lock bits	0101 1000	0000 0000	xxxx xxxx	xx**oo oooo**	Read Lock bits. “0” = programmed, “1” = unprogrammed. See Table 20-1 on page 123 for details.

Note: **a** = address high bits, **b** = address low bits, **H** = 0 – Low byte, 1 – high byte, **o** = data out, **i** = data in, x = don’t care

Table 20-16. High-voltage Serial Programming Instruction Set for ATtiny25/45/85

Instruction		Instruction Format				Operation Remarks
		Instr.1/5	Instr.2/6	Instr.3	Instr.4	
Chip erase	SDI SII SDO	0_1000_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx		Wait after Instr.3 until SDO goes high for the chip erase cycle to finish.
Load “write flash” command	SDI SII SDO	0_0001_0000_00 0_0100_1100_00 x_xxxx_xxxx_xx				Enter flash programming code.
Load flash page buffer	SDI SII SDO	0_ **bbbb_bbbb** _00 0_0000_1100_00 x_xxxx_xxxx_xx	0_**eeee_eeee**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_**dddd_dddd**_00 0_0011_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1101_00 x_xxxx_xxxx_xx	Repeat after Instr. 1 - 5 until the entire page buffer is filled or until all data within the page is filled. See note 1.
	SDI SII SDO	0_0000_0000_00 0_0111_1100_00 x_xxxx_xxxx_xx				Instr 5.
Load flash high address and program page	SDI SII SDO	0_0000_000**a**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx		Wait after Instr 3 until SDO goes high. Repeat Instr. 2 - 3 for each loaded flash page until the entire flash or all data is programmed. Repeat Instr. 1 for a new 256 byte page. See note 1.
Load “read flash” command	SDI SII SDO	0_0000_0010_00 0_0100_1100_00 x_xxxx_xxxx_xx				Enter flash read mode.
Read flash low and high bytes	SDI SII SDO	0_**bbbb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_0000_000**a**_00 0_0001_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 **q_qqqq_qqq**x_xx	Repeat Instr. 1, 3 - 6 for each new address. Repeat Instr. 2 for a new 256 byte page.
	SDI SII SDO	0_0000_0000_00 0_0111_1000_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0111_1100_00 **p_pppp_ppp**x_xx			Instr 5 - 6.
Load “write EEPROM” command	SDI SII SDO	0_0001_0001_00 0_0100_1100_00 x_xxxx_xxxx_xx				Enter EEPROM programming mode.
Load EEPROM page buffer	SDI SII SDO	0_00**bb_bbbb**_00 0_0000_1100_00 x_xxxx_xxxx_xx	0_**eeee_eeee**_00 0_0010_1100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1101_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx	Repeat Instr. 1 - 4 until the entire page buffer is filled or until all data within the page is filled. See note 2.
Program EEPROM page	SDI SII SDO	0_0000_0000_00 0_0110_0100_00 x_xxxx_xxxx_xx	0_0000_0000_00 0_0110_1100_00 x_xxxx_xxxx_xx			Wait after Instr. 2 until SDO goes high. Repeat Instr. 1 - 2 for each loaded EEPROM page until the entire EEPROM or all data is programmed.

Note: **a** = address high bits, **b** = address low bits, **d** = data in high bits, **e** = data in low bits, **p** = data out high bits, **q** = data out low bits,
x = don’t care, **1** = lock Bit1, **2** = lock bit2, **3** = CKSEL0 fuse, **4** = CKSEL1 fuse, **5** = SUT0 fuse, **6** = SUT1 fuse, **7** = CKDIV8, fuse, **8** = WDTON fuse, **9** = EESAVE fuse, **A** = SPIEN fuse, **B** = RSTDISBL fuse, **C** = BODLEVEL0 fuse, **D**= BODLEVEL1 fuse, **E** = MONEN fuse, **F** = SPMEN fuse

Notes:
1. For page sizes less than 256 words, parts of the address (bbbb_bbbb) will be parts of the page address.
2. For page sizes less than 256 bytes, parts of the address (bbbb_bbbb) will be parts of the page address.
3. The EEPROM is written page-wise. But only the bytes that are loaded into the page are actually written to the EEPROM. Page-wise EEPROM access is more efficient when multiple bytes are to be written to the same page. Note that auto-erase of EEPROM is not available in high-voltage serial programming, only in SPI programming.

Figure 22-27. I/O Pin Input Hysteresis versus V_{CC}

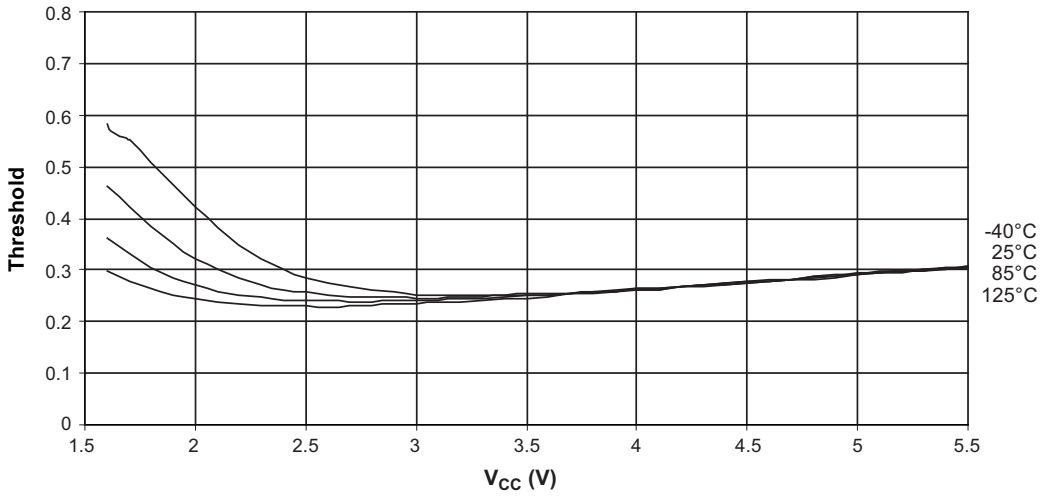


Figure 22-28. Reset Input Threshold Voltage versus V_{CC} (VIH, Reset Pin Read As '1')

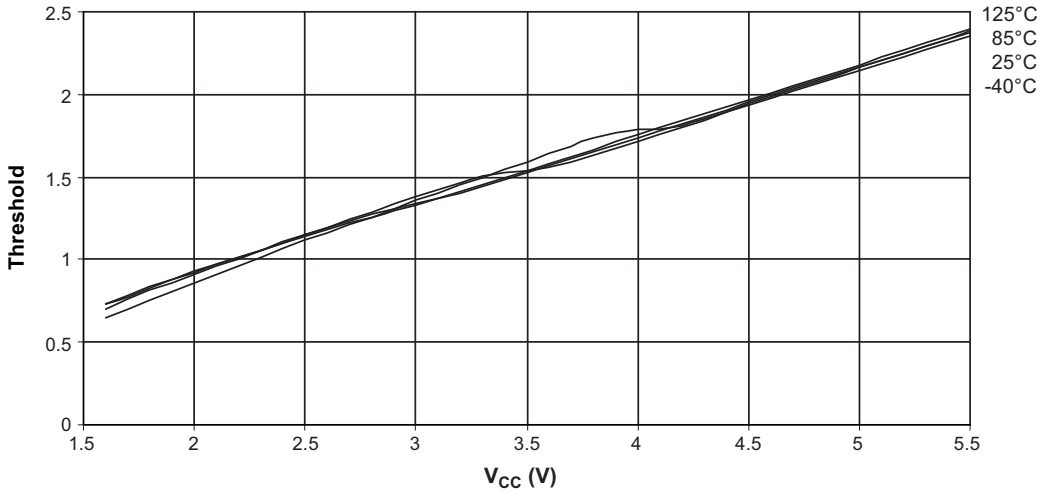
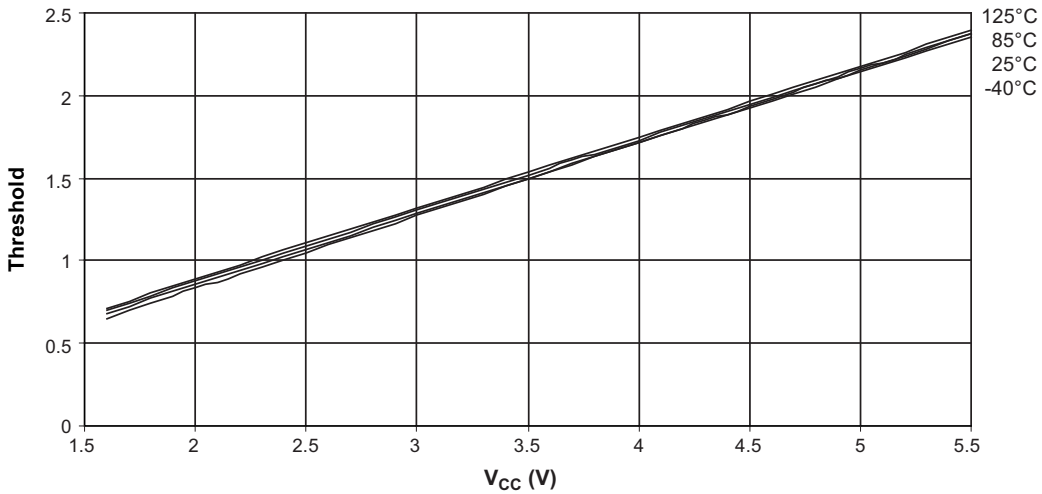


Figure 22-29. Reset Input Threshold Voltage versus V_{CC} (VIL, Reset Pin Read As '0')



22.9 Current Consumption of Peripheral Units

Figure 22-39. Brownout Detector Current versus V_{CC}

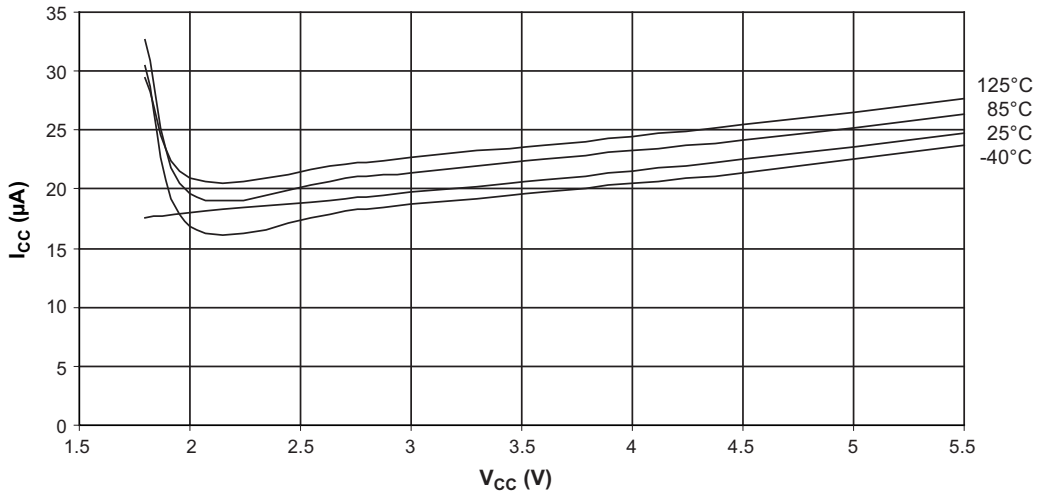


Figure 22-40. Analog Comparator Current versus V_{CC}

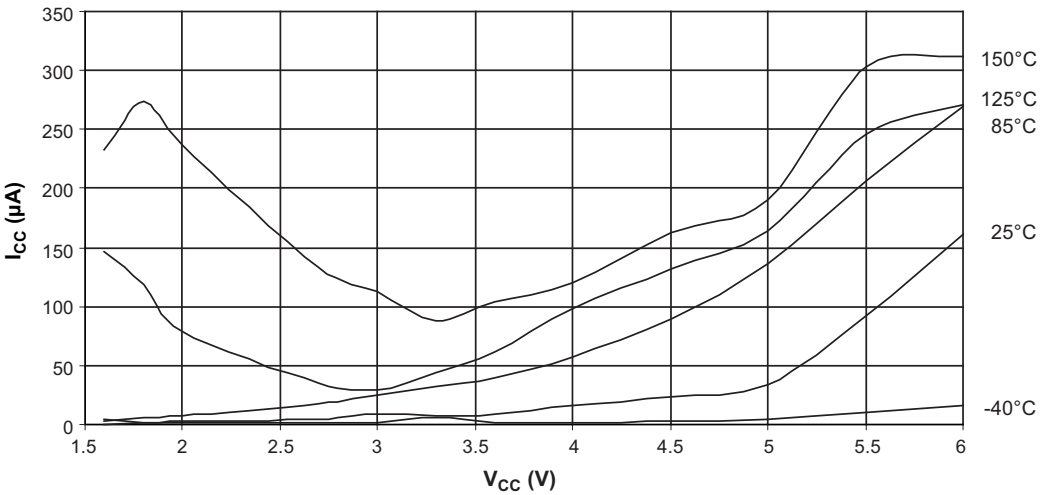
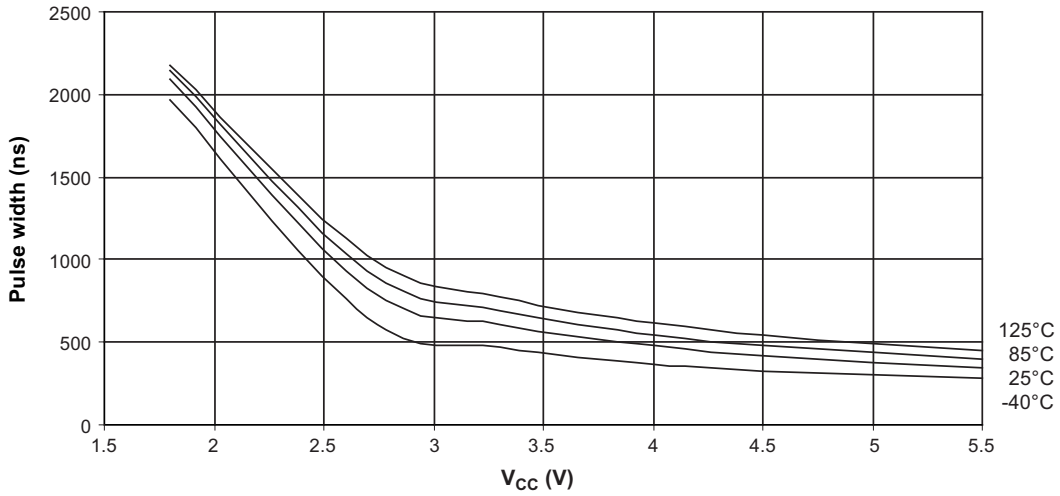


Figure 22-43. Reset Pulse Width versus V_{CC}



22.11 Analog to Digital Converter

Figure 22-44. Analog to Digital Converter Differential Mode OFFSET versus V_{CC}