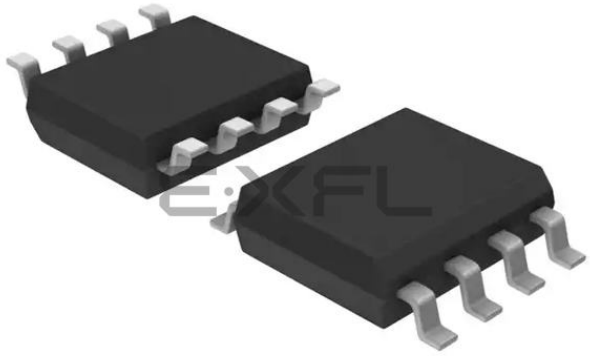


Welcome to E-XFL.COM

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 ~ 105°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/attiny85-15st1

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the arithmetic logic unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The Atmel® ATtiny25/45/85 provides the following features: 2/4/8K byte of in-system programmable flash, 128/256/512 bytes EEPROM, 128/256/256 bytes SRAM, 6 general purpose I/O lines, 32 general purpose working registers, one 8-bit Timer/Counter with compare modes, one 8-bit high speed Timer/Counter, universal serial interface, internal and external interrupts, a 4-channel, 10-bit ADC, a programmable watchdog timer with internal oscillator, and three software selectable power saving modes. The idle mode stops the CPU while allowing the SRAM, Timer/Counter, ADC, analog comparator, and interrupt system to continue functioning. The power-down mode saves the register contents, disabling all chip functions until the next interrupt or hardware reset. The ADC noise reduction mode stops the CPU and all I/O modules except ADC, to minimize switching noise during ADC conversions.

The device is manufactured using Atmel's high density non-volatile memory technology. The on-chip ISP flash allows the program memory to be re-programmed in-system through an SPI serial interface, by a conventional non-volatile memory programmer or by an on-chip boot code running on the AVR core.

The ATtiny25/45/85 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

1.2 Automotive Quality Grade

The ATtiny25/45/85 have been developed and manufactured according to the most stringent requirements of the international standard ISO-TS-16949. This data sheet contains limit values extracted from the results of extensive characterization (temperature and voltage). The quality and reliability of the ATtiny25/45/85 have been verified during regular product qualification as per AEC-Q100 grade 1.

As indicated in the ordering information paragraph, the products are available in three different temperature grades, but with equivalent quality and reliability objectives. Different temperature identifiers have been defined as listed in Table 1-1.

Table 1-1. Temperature Grade Identification for Automotive Products

Temperature	Temperature Identifier	Comments
–40; +85	T	Similar to industrial temperature grade but with automotive quality
–40; +105	T1	Reduced automotive temperature range
–40; +125	Z	Full automotive temperature range

1.3 Pin Descriptions

1.3.1 VCC

Supply voltage.

1.3.2 GND

Ground.

1.3.3 Port B (PB5..PB0)

Port B is a 6-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the Atmel ATtiny25/45/85 as listed on Section 9.3.2 "Alternate Functions of Port B" on page 50.

1.3.4 $\overline{\text{RESET}}$

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table on page 34. Shorter pulses are not guaranteed to generate a reset.

5.5 Low-frequency Crystal Oscillator

To use a 32.768kHz watch crystal as the clock source for the device, the low-frequency crystal oscillator must be selected by setting CKSEL fuses to '0110'. The crystal should be connected as shown in Figure 5-3 on page 22. Refer to the 32kHz crystal oscillator application note for details on oscillator operation and how to choose appropriate values for C1 and C2.

When this oscillator is selected, start-up times are determined by the SUT fuses as shown in Table 5-5.

Table 5-5. Start-up Times for the Low Frequency Crystal Oscillator Clock Selection

SUT1..0	Start-up Time from Power Down and Power Save	Additional Delay from Power On Reset (V_{CC} = 5.0V)	Recommended usage
00	1K CK(1)	4ms	Fast rising power or BOD enabled
01	1K CK(1)	64ms	Slowly rising power
10	32K CK	64ms	Stable frequency at start-up
11	Reserved		

Note: 1. These options should only be used if frequency stability at start-up is not important for the application.

5.6 Calibrated Internal RC Oscillator

The calibrated internal RC oscillator provides an 8.0MHz clock. The frequency is the nominal value at 3V and 25°C. If the frequency exceeds the specification of the device (depends on V_{CC}), the CKDIV8 fuse must be programmed in order to divide the internal frequency by 8 during start-up. See Section 5.10 "System Clock Prescaler" on page 26 for more details. This clock may be selected as the system clock by programming the CKSEL fuses as shown in Table 5-6. If selected, it will operate with no external components. During reset, hardware loads the calibration byte into the OSCCAL register and thereby automatically calibrates the RC oscillator. At 3V and 25°C, this calibration gives a frequency within ±1% of the nominal frequency. When this oscillator is used as the chip clock, the watchdog oscillator will still be used for the watchdog timer and for the reset time-out. For more information on the pre-programmed calibration value, see Section 20.4 "Calibration Byte" on page 125.

Table 5-6. Internal Calibrated RC Oscillator Operating Modes

CKSEL3..0	Nominal Frequency
0010(1)	8.0MHz

Note: 1. The device is shipped with this option selected.

When this oscillator is selected, start-up times are determined by the SUT fuses as shown in Table 5-7.

Table 5-7. Start-up Times for the Internal Calibrated RC Oscillator Clock Selection

SUT1..0	Start-up Time from Power-down	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	6CK	14CK + 4ms	BOD enabled
01	6CK	14CK + 4ms	Fast rising power
10(1)	6CK	14CK + 64ms	Slowly rising power
11	Reserved		

Note: 1. The device is shipped with this option selected.

6.2 Idle Mode

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

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

6.3 ADC Noise Reduction Mode

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

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

6.4 Power-down Mode

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

Note that if a level triggered interrupt is used for wake-up from power-down mode, the changed level must be held for some time to wake up the MCU. Refer to Section 10. “External Interrupts” on page 54 for details.

Table 6-2. Active Clock Domains and Wake-up Sources in the Different Sleep Modes

	Active Clock Domains					Oscillators	Wake-up Sources					
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{PCK}	Main Clock Source Enabled	INT0 and Pin Change	SPM/ EEPROM Ready	USI Start Condition	ADC	Other I/O	Watchdog Interrupt
Idle			X	X	X	X	X	X	X	X	X	X
ADC Noise Reduction				X		X	X(1)	X	X	X		X
Power-down							X(1)		X			X

Note: 1. For INT0, only level interrupt.

6.5 Limitations

BOD disable functionality has been implemented in the following devices, only:

- ATtiny25, revision D, and newer
- ATtiny45, revision D, and newer
- ATtiny85, revision C, and newer

7. System Control and Reset

7.1 Resetting the AVR

During reset, all I/O registers are set to their initial values, and the program starts execution from the reset vector. The instruction placed at the reset vector must be a RJMP – relative jump – instruction to the reset handling routine. If the program never enables an interrupt source, the interrupt vectors are not used, and regular program code can be placed at these locations. The circuit diagram in Figure 7-1 on page 33 shows the reset logic. Table on page 34 defines the electrical parameters of the reset circuitry.

The I/O ports of the AVR® are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running.

After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the SUT and CKSEL fuses. The different selections for the delay period are presented in Section 5.2 “Clock Sources” on page 21.

7.2 Reset Sources

The Atmel® ATtiny25/45/85 has four sources of reset:

- Power-on reset. The MCU is reset when the supply voltage is below the power-on reset threshold (V_{POT}).
- External reset. The MCU is reset when a low level is present on the $\overline{\text{RESET}}$ pin for longer than the minimum pulse length.
- Watchdog reset. The MCU is reset when the watchdog timer period expires and the watchdog is enabled.
- Brown-out reset. The MCU is reset when the supply voltage V_{CC} is below the brown-out reset threshold (V_{BOT}) and the brown-out detector is enabled.

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.

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.

11. 8-bit Timer/Counter0 with PWM

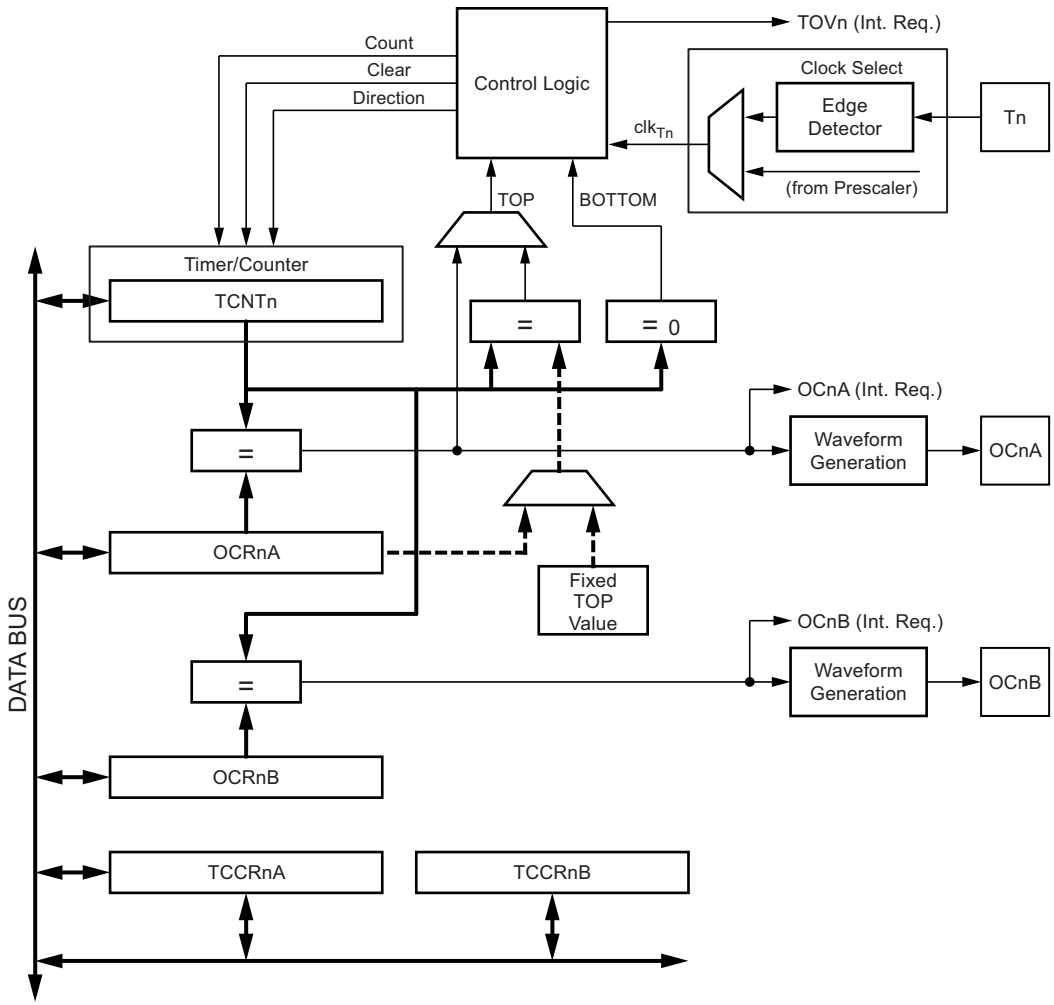
Timer/Counter0 is a general purpose 8-bit Timer/Counter module, with two independent output compare units, and with PWM support. It allows accurate program execution timing (event management) and wave generation. The main features are:

- Two independent output compare units
- Double buffered output compare registers
- Clear timer on compare match (auto reload)
- Glitch free, phase correct pulse width modulator (PWM)
- Variable PWM period
- Frequency generator
- Three independent interrupt sources (TOV0, OCF0A, and OCF0B)

11.1 Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 11-1. For the actual placement of I/O pins, refer to Figure 1 on page 2. CPU accessible I/O registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O register and bit locations are listed in Section 11.8 “8-bit Timer/Counter Register Description” on page 66.

Figure 11-1. 8-bit Timer/Counter Block Diagram



Signal description (internal signals):

count	Increment or decrement TCNT0 by 1.
direction	Select between increment and decrement.
clear	Clear TCNT0 (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{T0} in the following.
top	Signalize that TCNT0 has reached maximum value.
bottom	Signalize that TCNT0 has reached minimum value (zero).

Depending of the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T0}). clk_{T0} can be generated from an external or internal clock source, selected by the clock select bits (CS02:0). When no clock source is selected (CS02:0 = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether clk_{T0} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter control register (TCCR0A) and the WGM02 bit located in the Timer/Counter control register B (TCCR0B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the output compare output OC0A. For more details about advanced counting sequences and waveform generation, see Section 11.6 “Modes of Operation” on page 61.

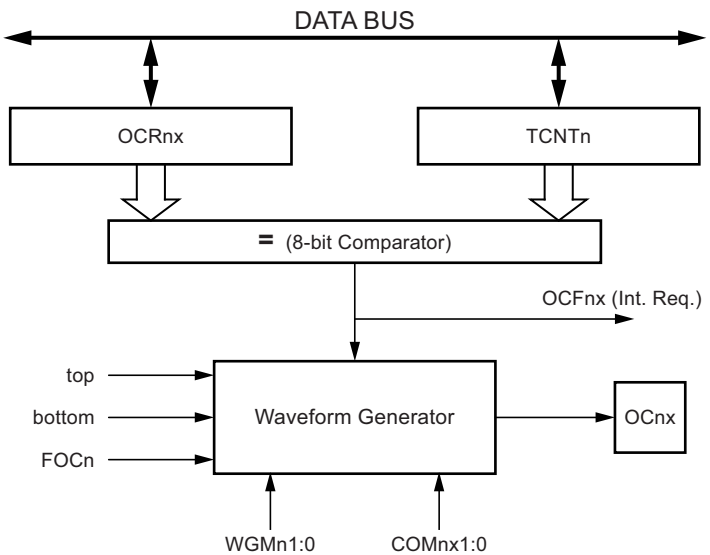
The Timer/Counter overflow flag (TOV0) is set according to the mode of operation selected by the WGM01:0 bits. TOV0 can be used for generating a CPU interrupt.

11.4 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the output compare registers (OCR0A and OCR0B). Whenever TCNT0 equals OCR0A or OCR0B, the comparator signals a match. A match will set the output compare flag (OCF0A or OCF0B) at the next timer clock cycle. If the corresponding interrupt is enabled, the output compare flag generates an output compare interrupt. The output compare flag is automatically cleared when the interrupt is executed. Alternatively, the flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the WGM02:0 bits and compare output mode (COM0x1:0) bits. The max and bottom signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation (see Section 11.6 “Modes of Operation” on page 61).

Figure 11-3 shows a block diagram of the output compare unit.

Figure 11-3. Output Compare Unit, Block Diagram



12. Timer/Counter Prescaler

The Timer/Counter can be clocked directly by the system clock (by setting the CSn2:0 = 1). This provides the fastest operation, with a maximum Timer/Counter clock frequency equal to system clock frequency ($f_{CLK\_I/O}$). Alternatively, one of four taps from the prescaler can be used as a clock source. The prescaled clock has a frequency of either $f_{CLK\_I/O}/8$, $f_{CLK\_I/O}/64$, $f_{CLK\_I/O}/256$, or $f_{CLK\_I/O}/1024$.

12.1 Prescaler Reset

The prescaler is free running, i.e., operates independently of the clock select logic of the Timer/Counter. Since the prescaler is not affected by the Timer/Counter's clock select, the state of the prescaler will have implications for situations where a prescaled clock is used. One example of prescaling artifacts occurs when the timer is enabled and clocked by the prescaler (6 > CSn2:0 > 1). The number of system clock cycles from when the timer is enabled to the first count occurs can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (8, 64, 256, or 1024).

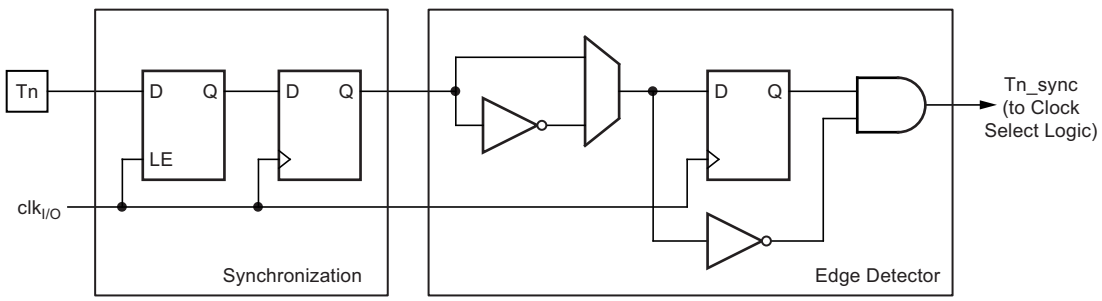
It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution.

12.2 External Clock Source

An external clock source applied to the T0 pin can be used as Timer/Counter clock (clk_{T0}). The T0 pin is sampled once every system clock cycle by the pin synchronization logic. The synchronized (sampled) signal is then passed through the edge detector. Figure 12-1 shows a functional equivalent block diagram of the T0 synchronization and edge detector logic. The registers are clocked at the positive edge of the internal system clock ($clk_{I/O}$). The latch is transparent in the high period of the internal system clock.

The edge detector generates one clk_{T0} pulse for each positive (CSn2:0 = 7) or negative (CSn2:0 = 6) edge it detects.

Figure 12-1. T0 Pin Sampling



The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T0 pin to the counter is updated.

Enabling and disabling of the clock input must be done when T0 has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{ExtClk} < f_{clk\_I/O}/2$) given a 50/50% duty cycle. Since the edge detector uses sampling, the maximum frequency of an external clock it can detect is half the sampling frequency (nyquist sampling theorem). However, due to variation of the system clock frequency and duty cycle caused by oscillator source (crystal, resonator, and capacitors) tolerances, it is recommended that maximum frequency of an external clock source is less than $f_{clk\_I/O}/2.5$.

An external clock source can not be prescaled.

13.1.6 Timer/Counter1 Output Compare RegisterC - OCR1C

Bit	7	6	5	4	3	2	1	0	
$2B ($4B)	MSB							LSB	OCR1C
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	1	1	1	1	1	1	1	1	

The output compare register C is an 8-bit read/write register.

The Timer/Counter output compare register C contains data to be continuously compared with Timer/Counter1. A compare match does only occur if Timer/Counter1 counts to the OCR1C value. A software write that sets TCNT1 and OCR1C to the same value does not generate a compare match. If the CTC1 bit in TCCR1 is set, a compare match will clear TCNT1.

This register has the same function in normal mode and PWM mode.

13.1.7 Timer/Counter Interrupt Mask Register - TIMSK

Bit	7	6	5	4	3	2	1	0	
$39 ($59)	-	OCIE1A	OCIE1B	OCIE0A	OCIE0B	TOIE1	TOIE0	-	TIMSK
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 - Res: Reserved Bit**

This bit is a reserved bit in the Atmel® ATtiny25/45/85 and always reads as zero.

- **Bit 6 - OCIE1A: Timer/Counter1 Output Compare Interrupt Enable**

When the OCIE1A bit is set (one) and the I-bit in the status register is set (one), the Timer/Counter1 compare match A, interrupt is enabled. The corresponding interrupt at vector $003 is executed if a compare match A occurs. The compare flag in Timer/Counter1 is set (one) in the Timer/Counter interrupt flag register.

- **Bit 5 - OCIE1B: Timer/Counter1 Output Compare Interrupt Enable**

When the OCIE1B bit is set (one) and the I-bit in the status register is set (one), the Timer/Counter1 compare match B, interrupt is enabled. The corresponding interrupt at vector $009 is executed if a compare match B occurs. The compare flag in Timer/Counter1 is set (one) in the Timer/Counter interrupt flag register.

- **Bit 4– OCIE0A: Timer/Counter Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to one, and the I-bit in the status register is set, the Timer/Counter compare match A interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter occurs, i.e., when the OCF0A bit is set in the Timer/Counter interrupt flag register – TIFR0.

- **Bit 3 – OCIE0B: Timer/Counter Output Compare Match B Interrupt Enable**

When the OCIE0B bit is written to one, and the I-bit in the status register is set, the Timer/Counter compare match B interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter occurs, i.e., when the OCF0B bit is set in the Timer/Counter interrupt flag register – TIFR0.

- **Bit 2 - TOIE1: Timer/Counter1 Overflow Interrupt Enable**

When the TOIE1 bit is set (one) and the I-bit in the status register is set (one), the Timer/Counter1 overflow interrupt is enabled. The corresponding interrupt (at vector $004) is executed if an overflow in Timer/Counter1 occurs. The overflow flag (timer1) is set (one) in the Timer/Counter interrupt flag register - TIFR.

- **Bit 0 - Res: Reserved Bit**

This bit is a reserved bit in the Atmel ATtiny25/45/85 and always reads as zero.

15. Universal Serial Interface – USI

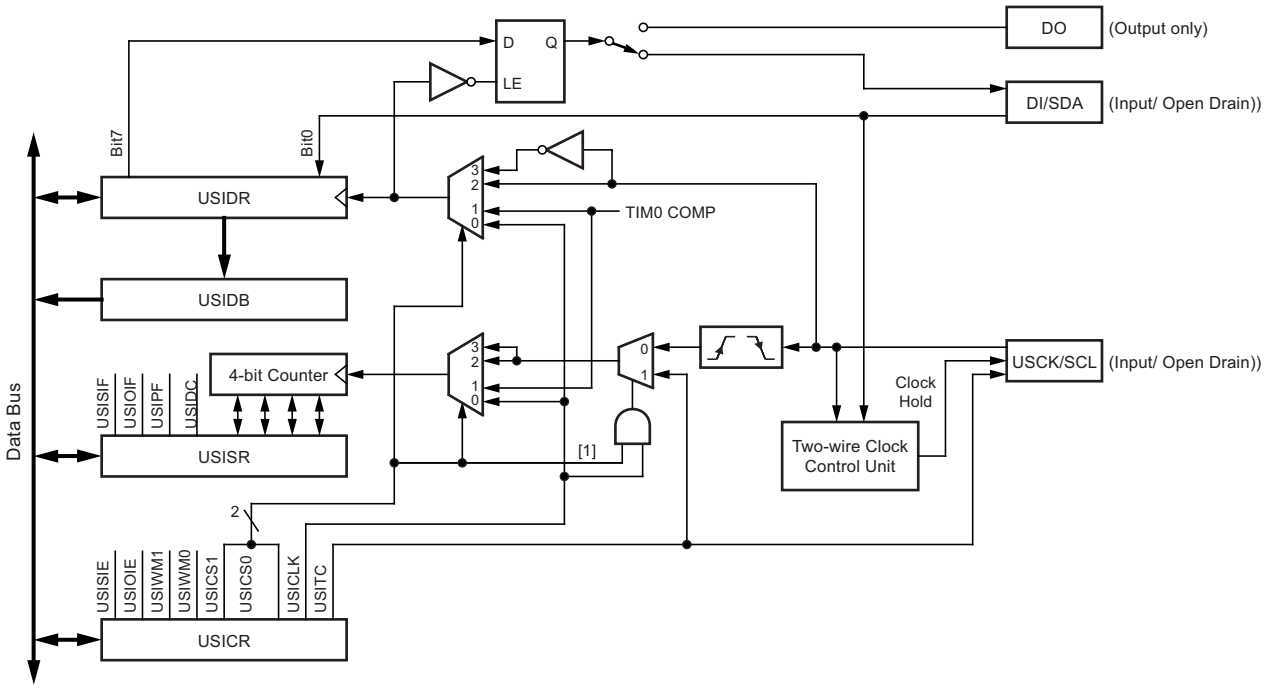
The universal serial interface, or USI, provides the basic hardware resources needed for serial communication. Combined with a minimum of control software, the USI allows significantly higher transfer rates and uses less code space than solutions based on software only. Interrupts are included to minimize the processor load. The main features of the USI are:

- Two-wire synchronous data transfer (master or slave, $f_{SCLmax} = f_{CK}/16$)
- Three-wire synchronous data transfer (master or slave $f_{SCKmax} = f_{CK}/4$)
- Data received interrupt
- Wakeup from idle mode
- In two-wire mode: Wake-up from all sleep modes, including power-down mode
- Two-wire start condition detector with interrupt capability

15.1 Overview

A simplified block diagram of the USI is shown on Figure 15-1 For the actual placement of I/O pins, refer to Figure 1 on page 2. CPU accessible I/O registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O register and bit locations are listed in the Section 15.4 “USI Register Descriptions” on page 94.

Figure 15-1. Universal Serial Interface, Block Diagram



The 8-bit shift register is directly accessible via the data bus and contains the incoming and outgoing data. The register has no buffering so the data must be read as quickly as possible to ensure that no data is lost. The most significant bit is connected to one of two output pins depending of the wire mode configuration. A transparent latch is inserted between the serial register output and output pin, which delays the change of data output to the opposite clock edge of the data input sampling. The serial input is always sampled from the data input (DI) pin independent of the configuration.

The 4-bit counter can be both read and written via the data bus, and can generate an overflow interrupt. Both the serial register and the counter are clocked simultaneously by the same clock source.

This allows the counter to count the number of bits received or transmitted and generate an interrupt when the transfer is complete. Note that when an external clock source is selected the counter counts both clock edges. In this case the counter counts the number of edges, and not the number of bits. The clock can be selected from three different sources: The USCK pin, Timer/Counter0 compare match or from software.

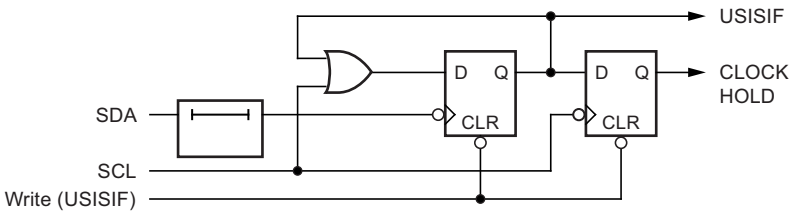
The two-wire clock control unit can generate an interrupt when a start condition is detected on the two-wire bus. It can also generate wait states by holding the clock pin low after a start condition is detected, or after the counter overflows.

Referring to the timing diagram (Figure 15-5 on page 92), a bus transfer involves the following steps:

1. The a start condition is generated by the master by forcing the SDA low line while the SCL line is high (A). SDA can be forced low either by writing a zero to bit 7 of the shift register, or by setting the corresponding bit in the PORT register to zero. Note that the data direction register bit must be set to one for the output to be enabled. The slave device's start detector logic (Figure 15-5) detects the start condition and sets the USISIF flag. The flag can generate an interrupt if necessary.
2. In addition, the start detector will hold the SCL line low after the master has forced an negative edge on this line (B). This allows the Slave to wake up from sleep or complete its other tasks before setting up the shift register to receive the address. This is done by clearing the start condition flag and reset the counter.
3. The master set the first bit to be transferred and releases the SCL line (C). The slave samples the data and shift it into the serial register at the positive edge of the SCL clock.
4. After eight bits are transferred containing slave address and data direction (read or write), the slave counter overflows and the SCL line is forced low (D). If the slave is not the one the master has addressed, it releases the SCL line and waits for a new start condition.
5. If the slave is addressed it holds the SDA line low during the acknowledgment cycle before holding the SCL line low again (i.e., the counter register must be set to 14 before releasing SCL at (D)). Depending of the R/W bit the master or slave enables its output. If the bit is set, a master read operation is in progress (i.e., the slave drives the SDA line) The slave can hold the SCL line low after the acknowledge (E).
6. Multiple bytes can now be transmitted, all in same direction, until a stop condition is given by the master (F). Or a new start condition is given.

If the slave is not able to receive more data it does not acknowledge the data byte it has last received. When the master does a read operation it must terminate the operation by force the acknowledge bit low after the last byte transmitted.

Figure 15-6. Start Condition Detector, Logic Diagram



15.2.5 Start Condition Detector

The start condition detector is shown in Figure 15-6 The SDA line is delayed (in the range of 50 to 300 ns) to ensure valid sampling of the SCL line. The start condition detector is only enabled in two-wire mode.

The start condition detector is working asynchronously and can therefore wake up the processor from the power-down sleep mode. However, the protocol used might have restrictions on the SCL hold time. Therefore, when using this feature in this case the oscillator start-up time set by the CKSEL fuses (see Section 5.1 "Clock Systems and their Distribution" on page 19) must also be taken into the consideration. Refer to the USISIF bit description on page 95 for further details.

15.3 Alternative USI Usage

When the USI unit is not used for serial communication, it can be set up to do alternative tasks due to its flexible design.

15.3.1 Half-duplex Asynchronous Data Transfer

By utilizing the shift register in three-wire mode, it is possible to implement a more compact and higher performance UART than by software only.

15.3.2 4-bit Counter

The 4-bit counter can be used as a stand-alone counter with overflow interrupt. Note that if the counter is clocked externally, both clock edges will generate an increment.

17.5 Changing Channel or Reference Selection

The MUX3..0 and REFS2..0 bits in the ADMUX register are single buffered through a temporary register to which the CPU has random access. This ensures that the channels and voltage reference selection only takes place at a safe point during the conversion. The channel and voltage reference selection is continuously updated until a conversion is started. Once the conversion starts, the channel and voltage reference selection is locked to ensure a sufficient sampling time for the ADC. Continuous updating resumes in the last ADC clock cycle before the conversion completes (ADIF in ADCSRA is set). Note that the conversion starts on the following rising ADC clock edge after ADSC is written. The user is thus advised not to write new channel or voltage reference selection values to ADMUX until one ADC clock cycle after ADSC is written.

If auto triggering is used, the exact time of the triggering event can be indeterministic. Special care must be taken when updating the ADMUX register, in order to control which conversion will be affected by the new settings.

If both ADATE and ADEN is written to one, an interrupt event can occur at any time. If the ADMUX register is changed in this period, the user cannot tell if the next conversion is based on the old or the new settings. ADMUX can be safely updated in the following ways:

a. When ADATE or ADEN is cleared.
b. During conversion, minimum one ADC clock cycle after the trigger event.
c. After a conversion, before the Interrupt Flag used as trigger source is cleared.

When updating ADMUX in one of these conditions, the new settings will affect the next ADC conversion.

17.5.1 ADC Input Channels

When changing channel selections, the user should observe the following guidelines to ensure that the correct channel is selected:

In single conversion mode, always select the channel before starting the conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the conversion to complete before changing the channel selection.

In free running mode, always select the channel before starting the first conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the first conversion to complete, and then change the channel selection. Since the next conversion has already started automatically, the next result will reflect the previous channel selection. Subsequent conversions will reflect the new channel selection.

17.5.2 ADC Voltage Reference

The voltage reference for the ADC (V_{REF}) indicates the conversion range for the ADC. Single ended channels that exceed V_{REF} will result in codes close to 0x3FF. V_{REF} can be selected as either V_{CC}, or internal 1.1V/2.56V voltage reference, or external AREF pin. The first ADC conversion result after switching voltage reference source may be inaccurate, and the user is advised to discard this result.

17.6 ADC Noise Canceler

The ADC features a noise canceler that enables conversion during sleep mode to reduce noise induced from the CPU core and other I/O peripherals. The noise canceler can be used with ADC noise reduction and Idle mode. To make use of this feature, the following procedure should be used:

a. Make sure that the ADC is enabled and is not busy converting. Single conversion mode must be selected and the ADC conversion complete interrupt must be enabled.
b. Enter ADC noise reduction mode (or idle mode). The ADC will start a conversion once the CPU has been halted.
c. If no other interrupts occur before the ADC conversion completes, the ADC interrupt will wake up the CPU and execute the ADC conversion complete interrupt routine. If another interrupt wakes up the CPU before the ADC conversion is complete, that interrupt will be executed, and an ADC conversion complete interrupt request will be generated when the ADC conversion completes. The CPU will remain in active mode until a new sleep command is executed.

Note that the ADC will not be automatically turned off when entering other sleep modes than Idle mode and ADC noise reduction mode. The user is advised to write zero to ADEN before entering such sleep modes to avoid excessive power consumption.

20.2 Fuse Bytes

The Atmel® ATtiny25/45/85 has three fuse bytes. Table 20-4, Table 20-5 and Table 20-6 on page 126 describe briefly the functionality of all the fuses and how they are mapped into the fuse bytes. Note that the fuses are read as logical zero, “0”, if they are programmed.

Table 20-3. Fuse Extended Byte

Fuse High Byte	Bit No	Description	Default Value
	7	-	1 (unprogrammed)
	6	-	1 (unprogrammed)
	5	-	1 (unprogrammed)
	4	-	1 (unprogrammed)
	3	-	1 (unprogrammed)
	2	-	1 (unprogrammed)
	1	-	1 (unprogrammed)
SELFPRGEN	0	Self-programming enable	1 (unprogrammed)

Table 20-4. Fuse High Byte

Fuse High Byte	Bit No	Description	Default Value
RSTDISBL(1)	7	External reset disable	1 (unprogrammed)
DWEN(2)	6	DebugWIRE enable	1 (unprogrammed)
SPIEN(3)	5	Enable serial program and data downloading	0 (programmed, SPI prog. enabled)
WDTON(4)	4	Watchdog timer always on	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the chip erase	1 (unprogrammed, EEPROM not preserved)
BODLEVEL2(5)	2	Brown-out detector trigger level	1 (unprogrammed)
BODLEVEL1(5)	1	Brown-out detector trigger level	1 (unprogrammed)
BODLEVEL0(5)	0	Brown-out detector trigger level	1 (unprogrammed)

Notes:
1. See Section 9.3.2 “Alternate Functions of Port B” on page 50 for description of RSTDISBL and DWEN fuses.
2. DWEN must be unprogrammed when lock bit security is required.
   See Section 20.1 “Program And Data Memory Lock Bits” on page 123
3. The SPIEN fuse is not accessible in SPI programming mode.
4. See Section 7.9.1 “Watchdog Timer Control Register – WDTCR” on page 38 for details.
5. See Table 7-2 on page 35 for BODLEVEL fuse decoding.
6. When programming the RSTDISBL fuse, high-voltage serial programming has to be used to change fuses to perform further programming.

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

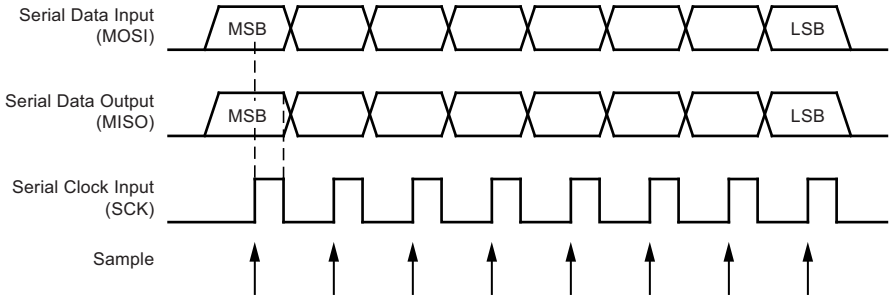


Table 20-10. Serial Programming Instruction Set

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 20-10. Serial Programming Instruction Set (Continued)

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Write lock bits	1010 1100	111x xxxx	xxxx xxxx	11**ii iiii**	Write lock bits. Set bits = “0” to program lock bits. See Table 20-1 on page 123 for details.
Read signature byte	0011 0000	000x xxxx	xxxx xx**bb**	**oooo oooo**	Read signature byte **o** at address **b**.
Write fuse bits	1010 1100	1010 0000	xxxx xxxx	**iiii iiii**	Set bits = “0” to program, “1” to unprogram. See Table 20-5 on page 125 for details.
Write fuse high bits	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	Set bits = “0” to program, “1” to unprogram. See Table 20-4 on page 124 for details.
Write extended fuse bits	1010 1100	1010 0100	xxxx xxxx	**xxxx xxxi**	Set bits = “0” to program, “1” to unprogram. See Table 20-3 on page 124 for details.
Read fuse bits	0101 0000	0000 0000	xxxx xxxx	**oooo oooo**	Read fuse bits. “0” = programmed, “1” = unprogrammed. See Table 20-5 on page 125 for details.
Read fuse high bits	0101 1000	0000 1000	xxxx xxxx	**oooo oooo**	Read fuse high bits. “0” = pro-grammed, “1” = unprogrammed. See Table 20-4 on page 124 for details.
Read extended fuse bits	0101 0000	0000 1000	xxxx xxxx	**oooo oooo**	Read extended fuse bits. “0” = pro-grammed, “1” = unprogrammed. See Table 20-3 on page 124 for details.
Read calibration byte	0011 1000	000x xxxx	0000 0000	**oooo oooo**	Read calibration byte
Poll RDY/$\overline{\text{BSY}}$	1111 0000	0000 0000	xxxx xxxx	xxxx xxx**o**	If **o** = “1”, a programming operation is still busy. Wait until this bit returns to “0” before applying another command.

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

20.6.2 Serial Programming Characteristics

Figure 20-3. Serial Programming Timing

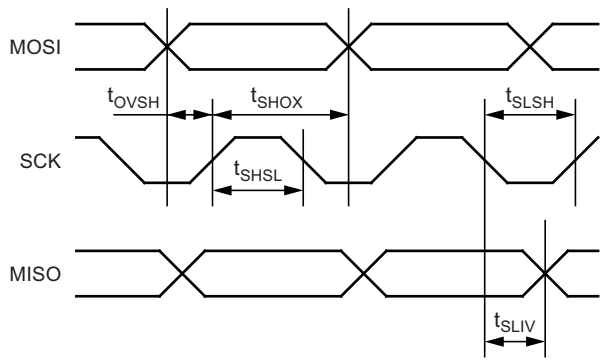


Figure 22-27. I/O Pin Input Hysteresis versus V_{CC}

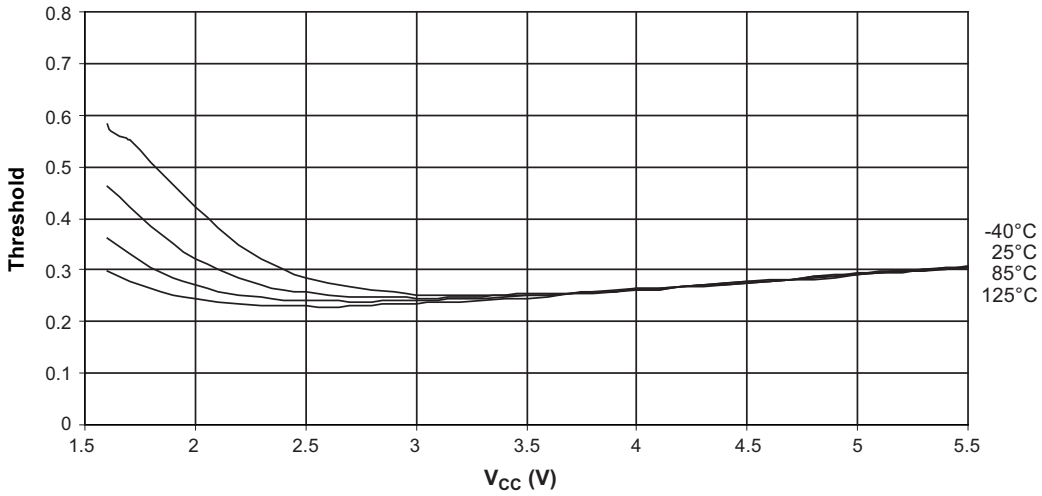


Figure 22-28. Reset Input Threshold Voltage versus V_{CC} (VIH, Reset Pin Read As '1')

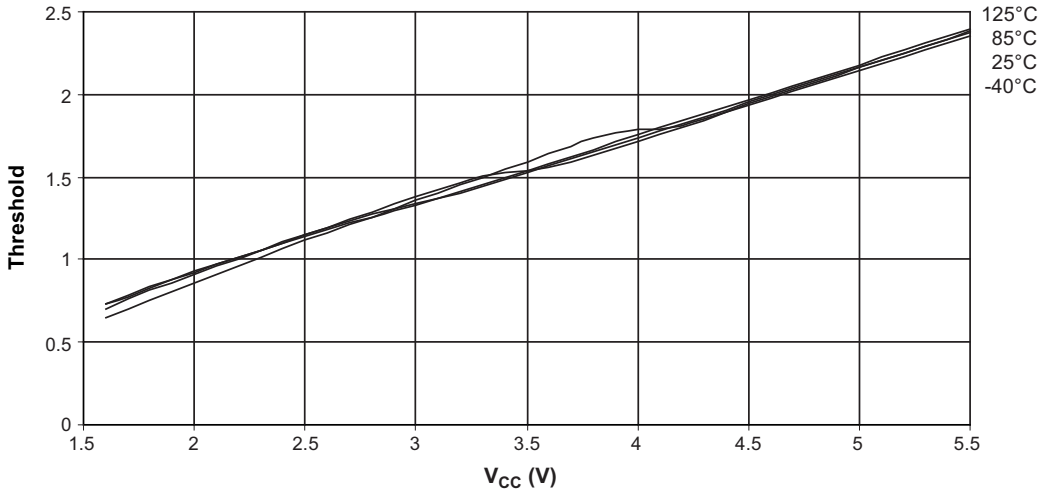
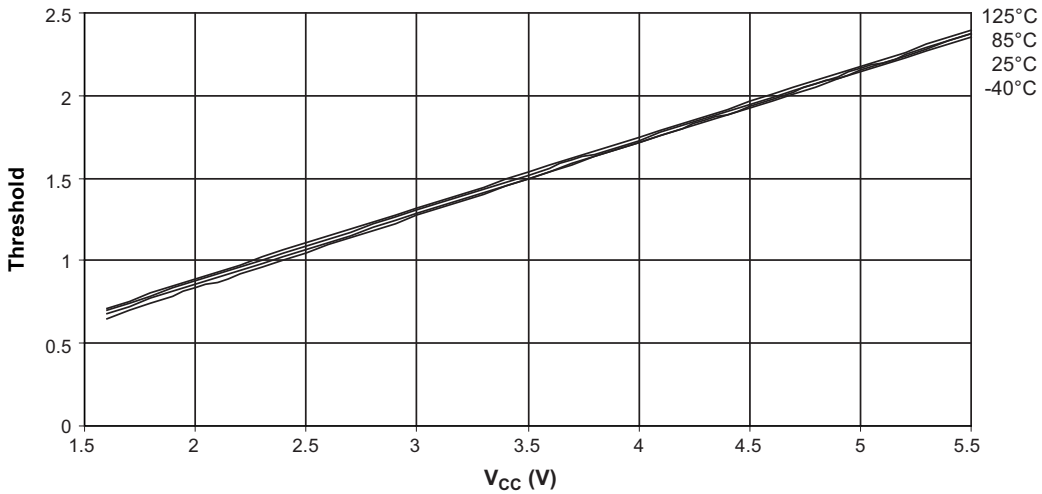


Figure 22-29. Reset Input Threshold Voltage versus V_{CC} (VIL, Reset Pin Read As '0')



22.8 Internal Oscillator Speed

Figure 22-34. Watchdog Oscillator Frequency versus V_{CC})

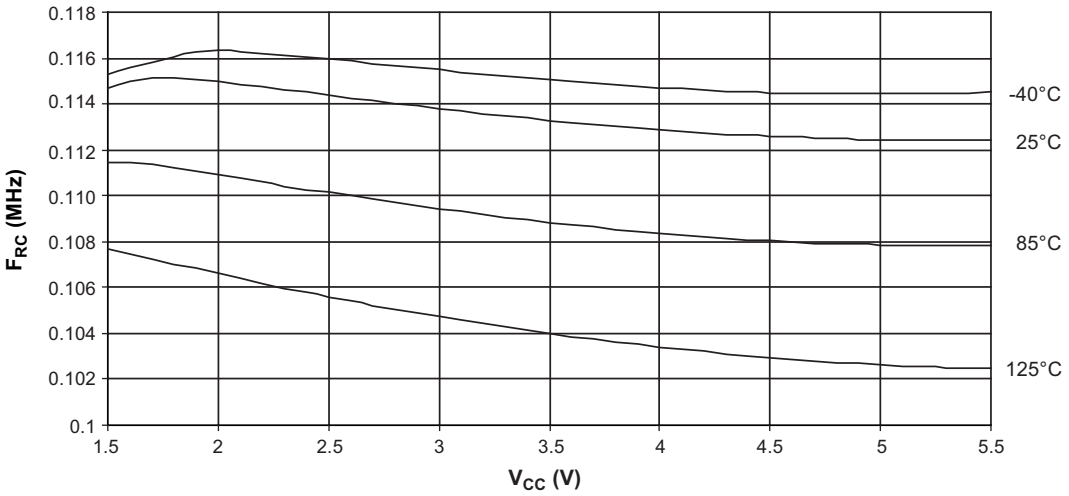


Figure 22-35. Watchdog Oscillator Frequency versus Temperature