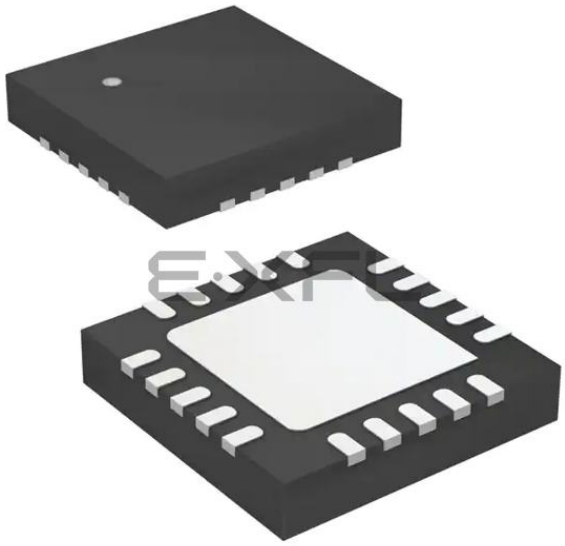


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

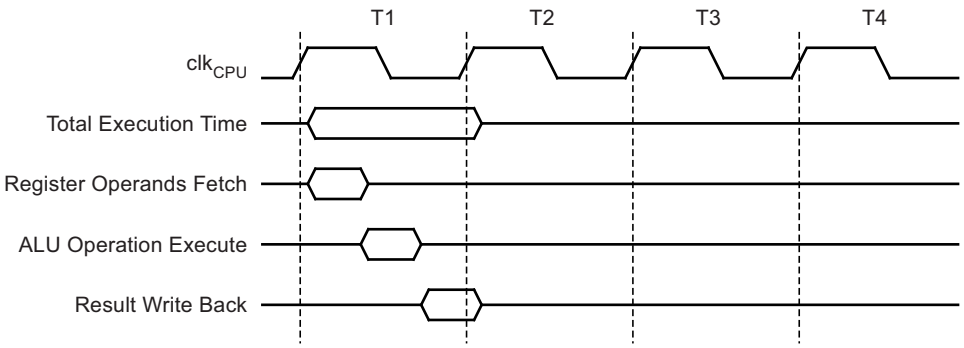
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	8MHz
Connectivity	USI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	6
Program Memory Size	2KB (1K x 16)
Program Memory Type	FLASH
EEPROM Size	128 x 8
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 4x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-WFQFN Exposed Pad
Supplier Device Package	20-WQFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny25v-15mt

Figure 3-5 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 3-5. Single Cycle ALU Operation



3.8 Reset and Interrupt Handling

The AVR® provides several different interrupt sources. These interrupts and the separate reset vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the global interrupt enable bit in the status register in order to enable the interrupt.

The lowest addresses in the program memory space are by default defined as the reset and interrupt vectors. The complete list of vectors is shown in Section 8. “Interrupts” on page 42. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 – the external interrupt request 0.

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

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

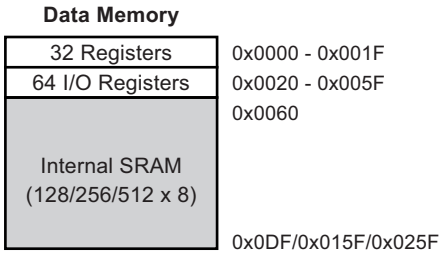
The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have interrupt flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.

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

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

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

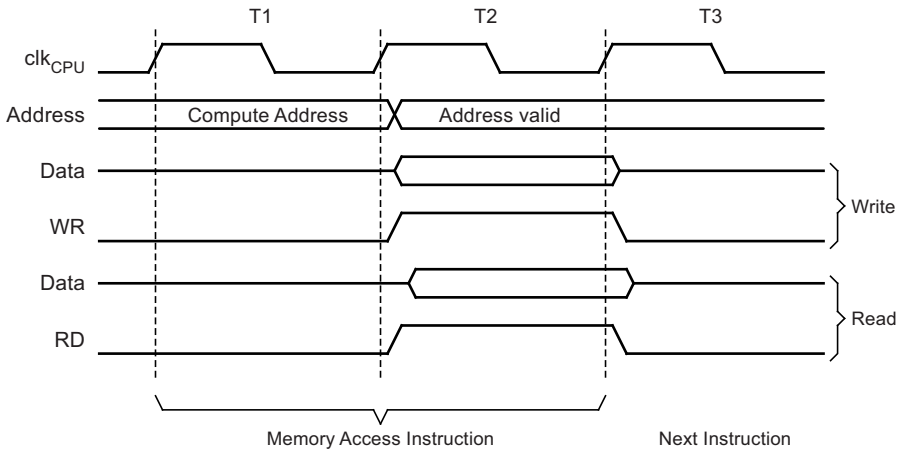
Figure 4-2. Data Memory Map



4.2.1 Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in Figure 4-3.

Figure 4-3. On-chip Data SRAM Access Cycles



4.3 EEPROM Data Memory

The Atmel® ATtiny25/45/85 contains 128/256/512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM address registers, the EEPROM data register, and the EEPROM control register. For a detailed description of serial data downloading to the EEPROM, see Section 20.6 “Serial Downloading” on page 126.

4.3.1 EEPROM Read/Write Access

The EEPROM access registers are accessible in the I/O space.

The write access times for the EEPROM are given in Table 4-1 on page 15. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See Section 4.3.10 “Preventing EEPROM Corruption” on page 17 for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to Section 4.3.6 “Atomic Byte Programming” on page 15 and Section 4.3.7 “Split Byte Programming” on page 15 for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

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.

5.9 Clock Output Buffer

The device can output the system clock on the CLKO pin. To enable the output, the CKOUT fuse has to be programmed. This mode is suitable when the chip clock is used to drive other circuits on the system. Note that the clock will not be output during reset and the normal operation of I/O pin will be overridden when the fuse is programmed. Any clock source, including the internal RC oscillator, can be selected when the clock is output on CLKO. If the system clock prescaler is used, it is the divided system clock that is output.

5.10 System Clock Prescaler

The Atmel® ATtiny25/45/85 system clock can be divided by setting the clock prescale register – CLKPR. This feature can be used to decrease power consumption when the requirement for processing power is low. This can be used with all clock source options, and it will affect the clock frequency of the CPU and all synchronous peripherals. $clk_{I/O}$, clk_{ADC}, clk_{CPU}, and clk_{FLASH} are divided by a factor as shown in Table 5-13 on page 27.

5.10.1 Clock Prescale Register – CLKPR

Bit	7	6	5	4	3	2	1	0	
	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 Atmel ATtiny25/45/85 and will always read as zero.

- **Bits 3..0 – CLKPS3..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 5-13 on page 27.

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.

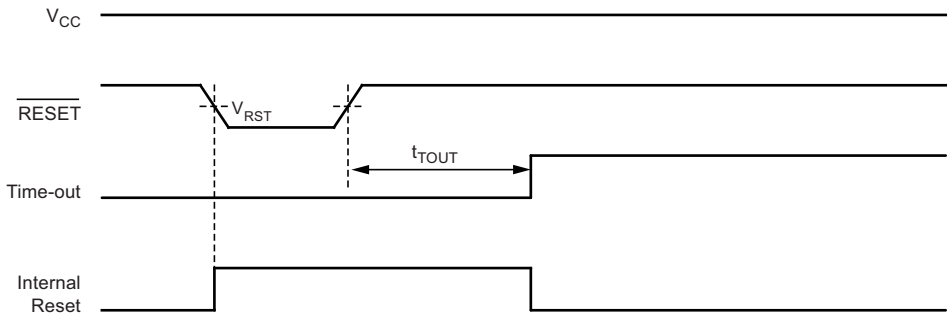
Interrupts must be disabled when changing prescaler setting to make sure the write procedure is not interrupted.

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.

7.4 External Reset

An external reset is generated by a low level on the $\overline{\text{RESET}}$ pin if enabled. Reset pulses longer than the minimum pulse width (see Table on page 34) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the reset threshold voltage – V_{RST} – on its positive edge, the delay counter starts the MCU after the time-out period – t_{TOUT} – has expired.

Figure 7-4. External Reset During Operation



7.5 Brown-out Detection

ATtiny25/45/85 has an on-chip brown-out detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the BODLEVEL fuses. The trigger level has a hysteresis to ensure spike free brown-out detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

Table 7-2. BODLEVEL Fuse Coding(1)

BODLEVEL [2..0] Fuses	Min V_{BOT}	Typ V_{BOT}	Max V_{BOT}	Units
111	BOD Disabled			
110	1.7	1.8	2.0	V
101	2.5	2.7	2.9	
100	4.0	4.3	4.6	
011		2.3(2)		
010		2.2(2)		
001		1.9(2)		
000		2.0(2)		

Notes: 1. V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to $V_{CC} = V_{BOT}$ during the production test. This guarantees that a brown-out reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed.

2. Centered value, not tested.

Table 7-3. Brown-out Characteristics

Parameter	Symbol	Min	Typ	Max	Units
RAM retention voltage(1)	V_{RAM}		50		mV
Brown-out detector hysteresis	V_{HYST}		50		mV
Min pulse width on brown-out reset	t_{BOD}		2		µs

Note: 1. This is the limit to which VDD can be lowered without losing RAM data

When the BOD is enabled, and V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 7-5), the brown-out reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 7-5), the delay counter starts the MCU after the time-out period t_{TOUT} has expired.

The following code example shows one assembly and one C function for turning off the WDT. The example assumes that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts will occur during execution of these functions

Assembly Code Example[1]

```
WDT_off:
WDR
        ; Clear WDRF in MCUSR
        ldi     r16, (0<<WDRF)
        out     MCUSR, r16
        ; Write logical one to WDCE and WDE
        ; Keep old prescaler setting to prevent unintentional Watchdog
Reset
        in      r16, WDTCR
        ori     r16, (1<<WDCE)|(1<<WDE)
        out     WDTCR, r16
        ; Turn off WDT
        ldi     r16, (0<<WDE)
        out     WDTCR, r16
        ret
```

C Code Example[1]

```
void WDT_off(void)
{
        _WDR();
        /* Clear WDRF in MCUSR */
        MCUSR = 0x00
        /* Write logical one to WDCE and WDE */
        WDTCR |= (1<<WDCE) | (1<<WDE);
        /* Turn off WDT */
        WDTCR = 0x00;
}
```

Note: 1. The example code assumes that the part specific header file is included.

7.10 Timed Sequences for Changing the Configuration of the Watchdog Timer

The sequence for changing configuration differs slightly between the two safety levels. Separate procedures are described for each level.

7.10.1 Safety Level 1

In this mode, the watchdog timer is initially disabled, but can be enabled by writing the WDE bit to one without any restriction. A timed sequence is needed when disabling an enabled watchdog timer. To disable an enabled watchdog timer, the following procedure must be followed:

1. In the same operation, write a logic one to WDCE and WDE. A logic one must be written to WDE regardless of the previous value of the WDE bit.
2. Within the next four clock cycles, in the same operation, write the WDE and WDP bits as desired, but with the WDCE bit cleared.

7.10.2 Safety Level 2

In this mode, the watchdog timer is always enabled, and the WDE bit will always read as one. A timed sequence is needed when changing the watchdog time-out period. To change the watchdog time-out, the following procedure must be followed:

1. In the same operation, write a logical one to WDCE and WDE. Even though the WDE always is set, the WDE must be written to one to start the timed sequence.
2. Within the next four clock cycles, in the same operation, write the WDP bits as desired, but with the WDCE bit cleared. The value written to the WDE bit is irrelevant.

11. 8-bit Timer/Counter0 with PWM

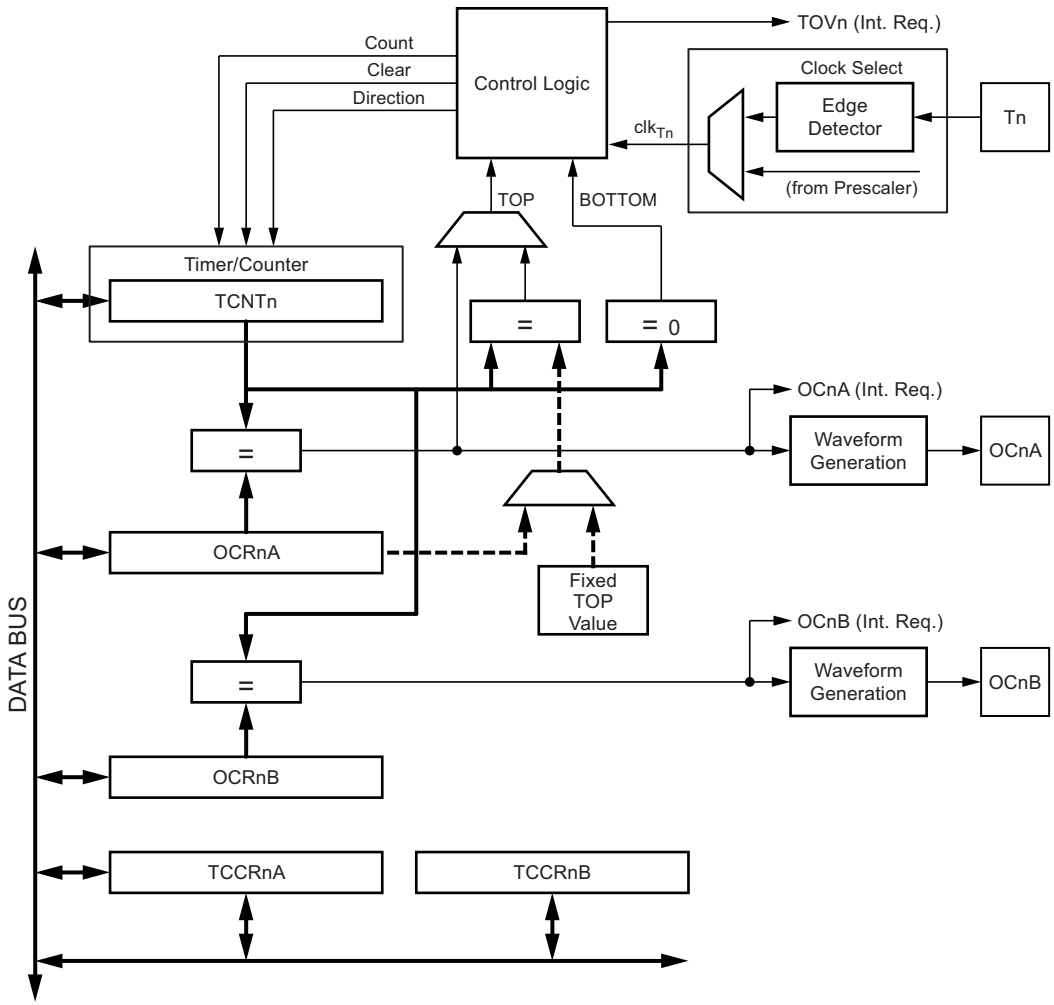
Timer/Counter0 is a general purpose 8-bit Timer/Counter module, with two independent output compare units, and with PWM support. It allows accurate program execution timing (event management) and wave generation. The main features are:

- 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.

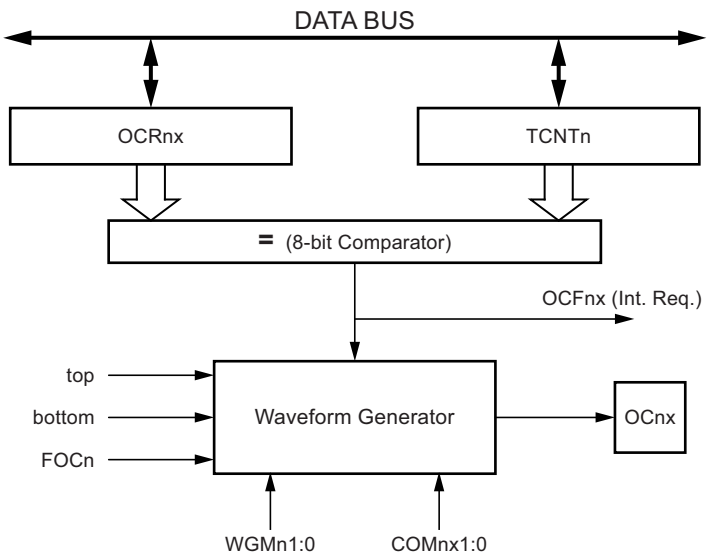
The Timer/Counter overflow flag (TOV0) is set according to the mode of operation selected by the WGM01:0 bits. TOV0 can be used for generating a CPU interrupt.

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



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.

For generating a waveform output in CTC mode, the OC0A output can be set to toggle its logical level on each compare match by setting the compare output mode bits to toggle mode (COM0A1:0 = 1). The OC0A value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0A is set to zero (0x00). The waveform frequency is defined by the following equation:

$$f_{OCnx} = \frac{f_{clk\ I/O}}{2 \times N \times (1 + OCRnx)}$$

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

As for the normal mode of operation, the TOV0 flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

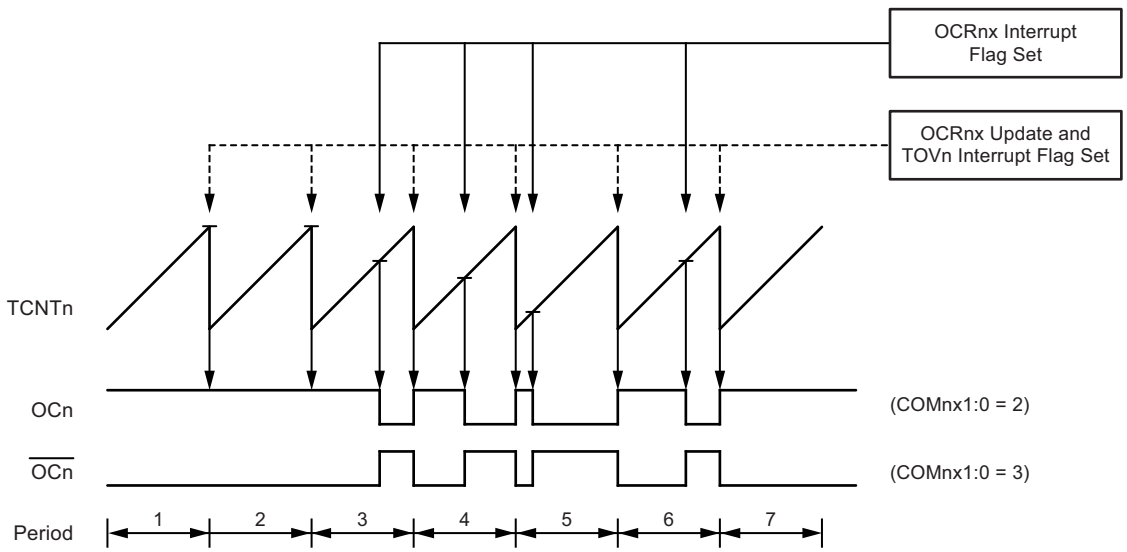
11.6.3 Fast PWM Mode

The fast pulse width modulation or fast PWM mode (WGM02:0 = 3 or 7) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. TOP is defined as 0xFF when WGM2:0 = 3, and OCR0A when WGM2:0 = 7. In non-inverting compare output mode, the output compare (OC0x) is cleared on the compare match between TCNT0 and OCR0x, and set at BOTTOM. In inverting compare output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that use dual-slope operation.

This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 11-6 on page 62. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0x and TCNT0.

Figure 11-6. Fast PWM Mode, Timing Diagram



The Timer/Counter overflow flag (TOV0) is set each time the counter reaches TOP. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

15.3.3 12-bit Timer/Counter

Combining the USI 4-bit counter and Timer/Counter0 allows them to be used as a 12-bit counter.

15.3.4 Edge Triggered External Interrupt

By setting the counter to maximum value (F) it can function as an additional external interrupt. The overflow flag and interrupt Enable bit are then used for the external interrupt. This feature is selected by the USICS1 bit.

15.3.5 Software Interrupt

The counter overflow interrupt can be used as a software interrupt triggered by a clock strobe.

15.4 USI Register Descriptions

15.4.1 USI Data Register – USIDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	USIDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

When accessing the USI data register (USIDR) the serial register can be accessed directly. If a serial clock occurs at the same cycle the register is written, the register will contain the value written and no shift is performed. A (left) shift operation is performed depending of the USICS1..0 bits setting. The shift operation can be controlled by an external clock edge, by a Timer/Counter0 compare match, or directly by software using the USICLK strobe bit. Note that even when no wire mode is selected (USIWM1..0 = 0) both the external data input (DI/SDA) and the external clock input (USCK/SCL) can still be used by the shift register.

The output pin in use, DO or SDA depending on the wire mode, is connected via the output latch to the most significant bit (bit 7) of the data register. The output latch is open (transparent) during the first half of a serial clock cycle when an external clock source is selected (USICS1 = 1), and constantly open when an internal clock source is used (USICS1 = 0). The output will be changed immediately when a new MSB written as long as the latch is open. The latch ensures that data input is sampled and data output is changed on opposite clock edges.

Note that the corresponding data direction register to the pin must be set to one for enabling data output from the shift register.

15.4.2 USI Buffer Register – USIBR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	USIBR
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

The content of the serial register is loaded to the USI buffer register when the trasfer is completed, and instead of accessing the USI data register (the serial register) the USI data buffer can be accessed when the CPU reads the received data. This gives the CPU time to handle other program tasks too as the controlling of the USI is not so timing critical. The USI flags as set same as when reading the USIDR register.

15.4.3 USI Status Register – USISR

Bit	7	6	5	4	3	2	1	0	
	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	USISR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The status register contains interrupt flags, line status flags and the counter value.

- **Bit 7 – USISIF: Start Condition Interrupt Flag**

When two-wire mode is selected, the USISIF flag is set (to one) when a start condition is detected. When output disable mode or three-wire mode is selected and (USICSx = 0b11 and USICLK = 0) or (USICS = 0b10 and USICLK = 0), any edge on the SCK pin sets the flag.

An interrupt will be generated when the flag is set while the USISIE bit in USICR and the global interrupt enable flag are set. The flag will only be cleared by writing a logical one to the USISIF bit. Clearing this bit will release the start detection hold of USCL in two-wire mode.

A start condition interrupt will wakeup the processor from all sleep modes.

- **Bit 6 – USIOIF: Counter Overflow Interrupt Flag**

This flag is set (one) when the 4-bit counter overflows (i.e., at the transition from 15 to 0). An interrupt will be generated when the flag is set while the USIOIE bit in USICR and the global interrupt enable flag are set. The flag will only be cleared if a one is written to the USIOIF bit. Clearing this bit will release the counter overflow hold of SCL in two-wire mode.

A counter overflow interrupt will wakeup the processor from Idle sleep mode.

- **Bit 5 – USIPF: Stop Condition Flag**

When two-wire mode is selected, the USIPF Flag is set (one) when a stop condition is detected. The flag is cleared by writing a one to this bit. Note that this is not an interrupt flag. This signal is useful when implementing two-wire bus master arbitration.

- **Bit 4 – USIDC: Data Output Collision**

This bit is logical one when bit 7 in the shift register differs from the physical pin value. The flag is only valid when two-wire mode is used. This signal is useful when implementing two-wire bus master arbitration.

- **Bits 3..0 – USICNT3..0: Counter Value**

These bits reflect the current 4-bit counter value. The 4-bit counter value can directly be read or written by the CPU.

The 4-bit counter increments by one for each clock generated either by the external clock edge detector, by a Timer/Counter0 compare match, or by software using USICLK or USITC strobe bits. The clock source depends of the setting of the USICS1..0 bits. For external clock operation a special feature is added that allows the clock to be generated by writing to the USITC strobe bit. This feature is enabled by write a one to the USICLK bit while setting an external clock source (USICS1 = 1).

Note that even when no wire mode is selected (USIWM1..0 = 0) the external clock input (USCK/SCL) are can still be used by the counter.

16.3 Analog Comparator Multiplexed Input

It is possible to select any of the ADC3..0 pins to replace the negative input to the analog comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the analog comparator multiplexer enable bit (ACME in ADCSRB) is set and the ADC is switched off (ADEN in ADCSRA is zero), MUX1..0 in ADMUX select the input pin to replace the negative input to the analog comparator, as shown in Table 16-2. If ACME is cleared or ADEN is set, AIN1 is applied to the negative input to the analog comparator.

Table 16-2. Analog Comparator Multiplexed Input

ACME	ADEN	MUX1..0	Analog Comparator Negative Input
0	x	xx	AIN1
1	1	xx	AIN1
1	0	00	ADC0
1	0	01	ADC1
1	0	10	ADC2
1	0	11	ADC3

16.3.1 Digital Input Disable Register 0 – DIDR0

Bit	7	6	5	4	3	2	1	0	
	–	–	ADC0D	ADC2D	ADC3D	ADC1D	AIN1D	AIN0D	DIDR0
Read/Write	R	R	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 – AIN1D, AIN0D: AIN1, AIN0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the AIN1/0 pin is disabled. The corresponding PIN register bit will always read as zero when this bit is set. When an analog signal is applied to the AIN1/0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

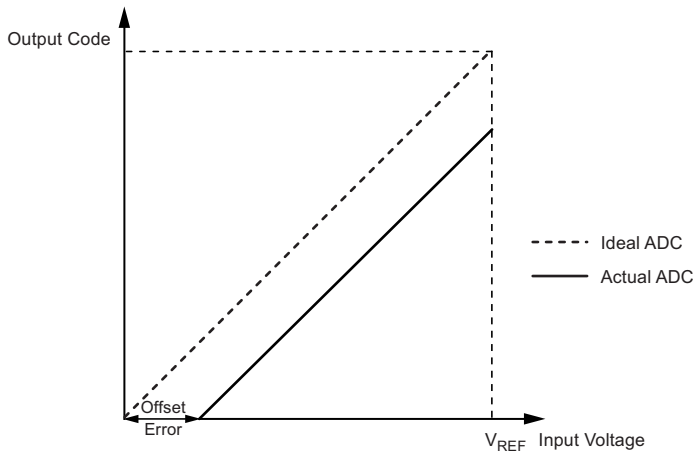
17.6.3 ADC Accuracy Definitions

An n-bit single-ended ADC converts a voltage linearly between GND and V_{REF} in 2^n steps (LSBs). The lowest code is read as 0, and the highest code is read as 2^n–1.

Several parameters describe the deviation from the ideal behavior:

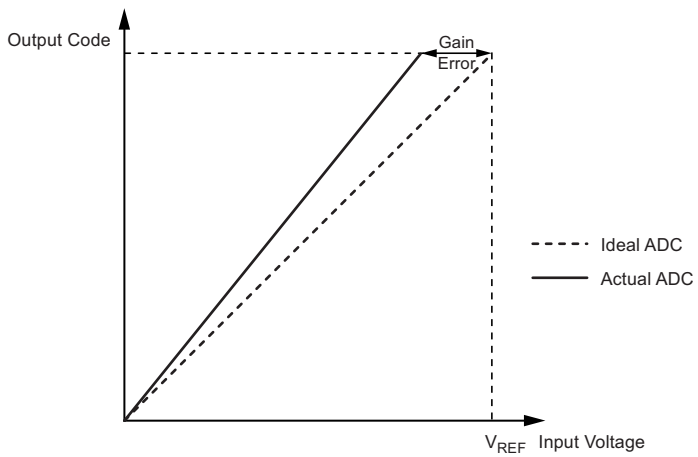
- Offset: The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

Figure 17-9. Offset Error



- Gain error: After adjusting for offset, the gain error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB

Figure 17-10. Gain Error



17.7 ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC result registers (ADCL, ADCH). The form of the conversion result depends on the type of the conversion as there are three types of conversions: single ended conversion, unipolar differential conversion and bipolar differential conversion.

17.7.1 Single Ended Conversion

For single ended conversion, the result is

$$ADC = \frac{V_{IN} \times 1024}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see Table 17-3 on page 112 and Table 17-4 on page 113). 0x000 represents analog ground, and 0x3FF represents the selected voltage reference minus one LSB. The result is presented in one-sided form, from 0x3FF to 0x000.

17.7.2 Unipolar Differential Conversion

If differential channels and an unipolar input mode are used, the result is

$$ADC = \frac{(V_{POS} - V_{NEG}) \times 1024}{V_{REF}} \times GAIN$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, and V_{REF} the selected voltage reference (see Table 17-3 on page 112 and Table 17-4 on page 113). The voltage on the positive pin must always be larger than the voltage on the negative pin or otherwise the voltage difference is saturated to zero. The result is presented in one-sided form, from 0x000 (0d) to 0x3FF (+1023d). The GAIN is either 1x or 20x.

17.7.3 Bipolar Differential Conversion

As default the ADC converter operates in the unipolar input mode, but the bipolar input mode can be selected by writing the BIN bit in the ADCSRB to one. In the bipolar input mode two-sided voltage differences are allowed and thus the voltage on the negative input pin can also be larger than the voltage on the positive input pin. If differential channels and a bipolar input mode are used, the result is

$$ADC = \frac{(V_{POS} - V_{NEG}) \times 512}{V_{REF}} \times GAIN$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, and V_{REF} the selected voltage reference. The result is presented in two's complement form, from 0x200 (–512d) through 0x000 (+0d) to 0x1FF (+511d). The GAIN is either 1x or 20x.

However, if the signal is not bipolar by nature (9 bits + sign as the 10th bit), this scheme loses one bit of the converter dynamic range. Then, if the user wants to perform the conversion with the maximum dynamic range, the user can perform a quick polarity check of the result and use the unipolar differential conversion with selectable differential input pairs (see the input polarity reversal mode ie. the IPR bit in the Section 17.7.8 "ADC Control and Status Register B – ADCSRB" on page 115). When the polarity check is performed, it is sufficient to read the MSB of the result (ADC9 in ADCH). If the bit is one, the result is negative, and if this bit is zero, the result is positive.

19.4.3 Reading the Fuse and Lock Bits from Software

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

Bit	7	6	5	4	3	2	1	0
Rd	–	–	–	–	–	–	LB2	LB1

The algorithm for reading the fuse low byte is similar to the one described above for reading the lock bits. To read the fuse low byte, load the Z-pointer with 0x0000 and set the RFLB and SPMEN bits in SPMCSR. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the fuse low byte (FLB) will be loaded in the destination register as shown below. Refer to Table 20-5 on page 125 for a detailed description and mapping of the fuse low byte.

Bit	7	6	5	4	3	2	1	0
Rd	FLB7	FLB6	FLB5	FLB4	FLB3	FLB2	FLB1	FLB0

Similarly, when reading the fuse high byte, load 0x0003 in the Z-pointer. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the fuse high byte (FHB) will be loaded in the destination register as shown below. Refer to Table 20-4 for detailed description and mapping of the fuse high byte.

Bit	7	6	5	4	3	2	1	0
Rd	FHB7	FHB6	FHB5	FHB4	FHB3	FHB2	FHB1	FHB0

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

19.4.4 Preventing Flash Corruption

During periods of low V_{CC}, the flash program can be corrupted because the supply voltage is too low for the CPU and the flash to operate properly. These issues are the same as for board level systems using the flash, and the same design solutions should be applied.

A flash program corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the flash requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage for executing instructions is too low.

Flash corruption can easily be avoided by following these design recommendations (one is sufficient):

1. Keep the AVR® RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal brown-out detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{CC} reset protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.
2. Keep the AVR core in power-down sleep mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCSR register and thus the flash from unintentional writes.

19.4.5 Programming Time for Flash when Using SPM

The calibrated RC oscillator is used to time flash accesses. Table 19-1 shows the typical programming time for flash accesses from the CPU.

Table 19-1. SPM Programming Time

Symbol	Min Programming Time	Max Programming Time
Flash write (page erase, page write, and write lock bits by SPM)	3.7ms	4.5ms

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

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

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

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

20.6.1 Serial Programming Algorithm

When writing serial data to the Atmel® ATtiny25/45/85, data is clocked on the rising edge of SCK.

When reading data from the Atmel ATtiny25/45/85, data is clocked on the falling edge of SCK. See Figure 20-2 on page 128 and Figure 20-3 on page 129 for timing details.

To program and verify the ATtiny25/45/85 in the serial programming mode, the following sequence is recommended (see four byte instruction formats in Table 20-10 on page 128):

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 20ms 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 20-9 on page 128.) 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 20-9 on page 128.) 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 20-7 on page 126). 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.
7. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
8. Power-off sequence (if needed):
   Set $\overline{\text{RESET}}$ to “1”.
   Turn V_{CC} power off.

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

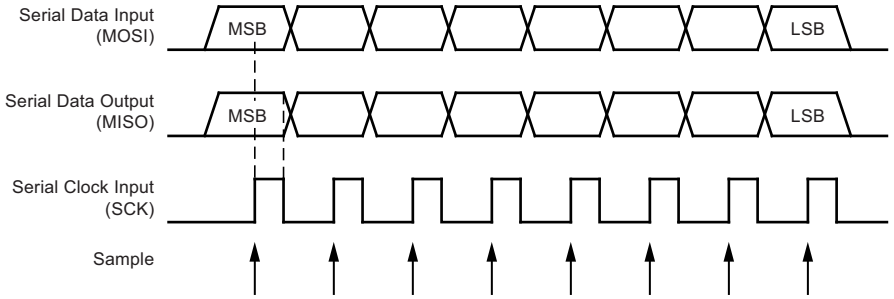


Table 20-10. Serial Programming Instruction Set

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 20-11. Serial Programming Characteristics, T_A = –40°C to 125°C, V_{CC} = 2.7 to 5.5V (Unless Otherwise Noted)

Parameter	Symbol	Min	Typ	Max	Units
Oscillator frequency (ATtiny25/45/85V)	$1/t_{CLCL}$	0		4	MHz
Oscillator period (ATtiny25/45/85V)	t_{CLCL}	250			ns
Oscillator frequency (ATtiny25/45/85L, VCC = 2.7 to 5.5V)	$1/t_{CLCL}$	0		10	MHz
Oscillator period (ATtiny25/45/85L, VCC = 2.7 to 5.5V)	t_{CLCL}	100			ns
Oscillator frequency (ATtiny25/45/85, V_{CC} = 4.5V to 5.5V)	$1/t_{CLCL}$	0		20	MHz
Oscillator period (ATtiny25/45/85, V_{CC} = 4.5V to 5.5V)	t_{CLCL}	50			ns
SCK pulse width high	t_{SHSL}	2 t_{CLCL}*			ns
SCK pulse width low	t_{SLSH}	2 t_{CLCL}*			ns
MOSI setup to SCK high	t_{OVSH}	t_{CLCL}			ns
MOSI hold after SCK high	t_{SHOX}	2 t_{CLCL}			ns

Note: 2 t_{CLCL} for f_{ck} < 12MHz, 3 t_{CLCL} for f_{ck} ≥12MHz

20.7 High-voltage Serial Programming

This section describes how to program and verify flash program memory, EEPROM data memory, lock bits and fuse bits in the Atmel® ATtiny25/45/85.

Figure 20-4. High-voltage Serial Programming

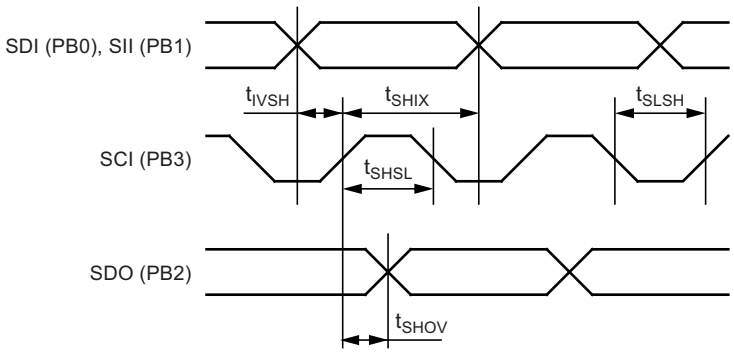


Table 20-12. Pin Name Mapping

Signal Name in High-voltage Serial Programming Mode	Pin Name	I/O	Function
SDI	PB0	I	Serial data input
SII	PB1	I	Serial instruction input
SDO	PB2	O	Serial data output
SCI	PB3	I	Serial clock input (min. 220ns period)

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

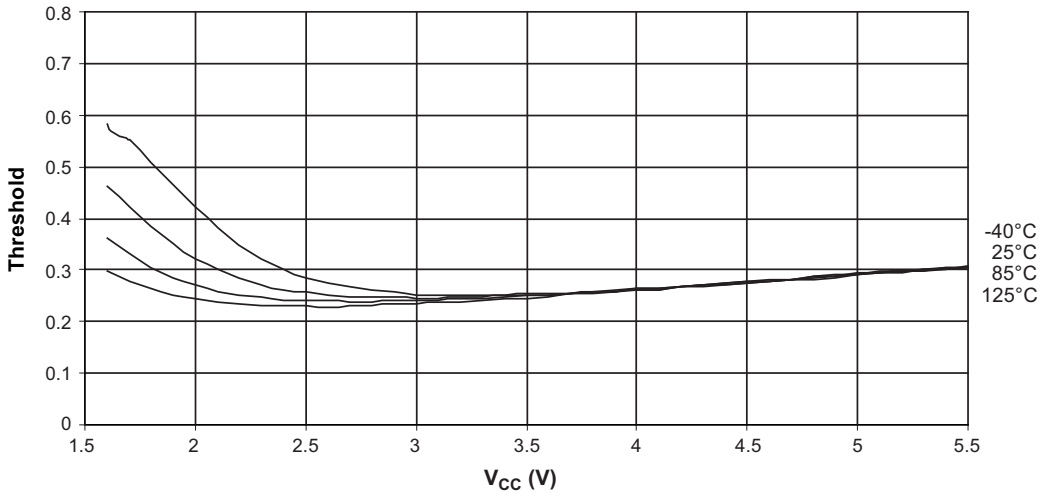


Figure 22-28. Reset Input Threshold Voltage versus V_{CC} (VIH, Reset Pin Read As '1')

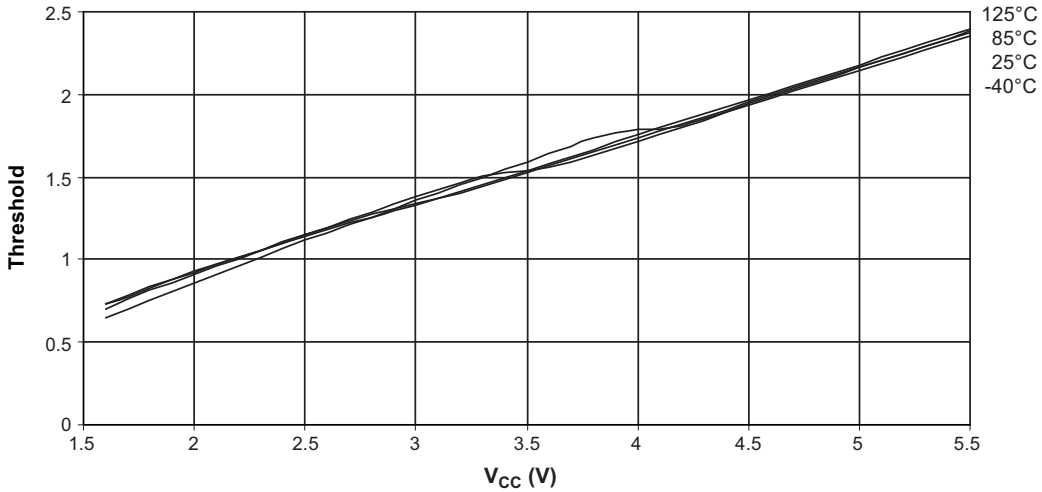


Figure 22-29. Reset Input Threshold Voltage versus V_{CC} (VIL, Reset Pin Read As '0')