
Long Delay Generation Using the AVR Microcontroller

Background

When a microcontroller-based application requires the implementation of long delays, the designer can choose among several solutions.

One solution consists of using the on-chip hardware timer. Since a typical timer is only 16 or 24 bits wide, this solution implies that the system clock be slow enough to cope with long delay. For example, a 24-bit timer running on a 1 MHz system clock can only lead to a delay of a few tens of seconds. This may be not enough in many applications. In addition, this solution impacts overall system performance since it imposes a processing speed.

Another solution involves a software overhead used to count a certain amount of timer overflows. However, it complicates the software and prevents the core from remaining in a low power mode while waiting for the delay.

A third solution relies on an external low speed oscillator. The timer is configured as an external event counter and can generate very long delays; however, system cost is impacted since additional components are necessary.

The solution presented here shows how the AVR AT90 series microcontrollers (AT90S2313, AT90S4414 and AT90S8515) generate and handle long delays. On-chip timers are used without any software intervention, thus allowing the core to be in a low-power mode during the delay. Since the timers are clocked by the system clock, there is no need of any additional components.

Due to the very long timing capability, this implementation combines high system performance with long delay generation. For example, an AVR Microcontroller running at 20 Mips can generate delays as long as half an hour.

Applications

The list below shows some applications where long delays are needed:

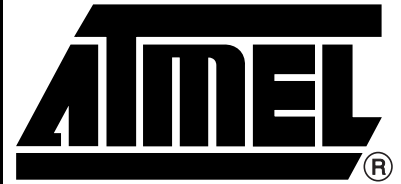
- Timeouts in man-machine interfaces
- Environmental measurement instruments (sound level, pollution)
- Regulation and process control

AVR Microcontrollers Timers/Counters

The following section briefly describes the timers used in AVR Microcontrollers. For more information, please refer to the AVR Enhanced RISC Microcontroller Data Book.

Timer/Counters

The AT90 series provides two general purpose Timer/Counters, one 8-bit T/C and one 16-bit T/C. The Timer/Counters have individual prescaling selection from the same 10-bit prescaling timer. Both Timer/Counters can either be used as a timer with an internal clock time base or as a counter with an external pin connection which triggers the counting.



AVR[®]
Embedded RISC
Microcontroller
Core

Application Note

Rev. 1268A-02/99

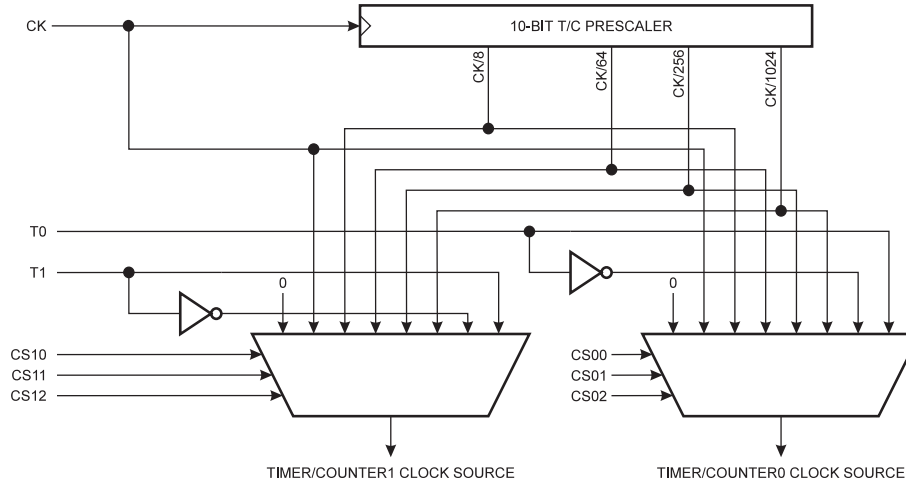


The Timer/Counter Prescaler

Figure 1 shows the general Timer/Counter prescaler. The four different prescaled selections are: CK/8, CK/64, CK/256 and CK/1024, where CK is the oscillator clock. For

the two Timer/Counters, added selections such as CK, external source and stop can be selected as clock sources.

Figure 1. Timer/Counter Prescaler.



The 8-Bit Timer/Counter0

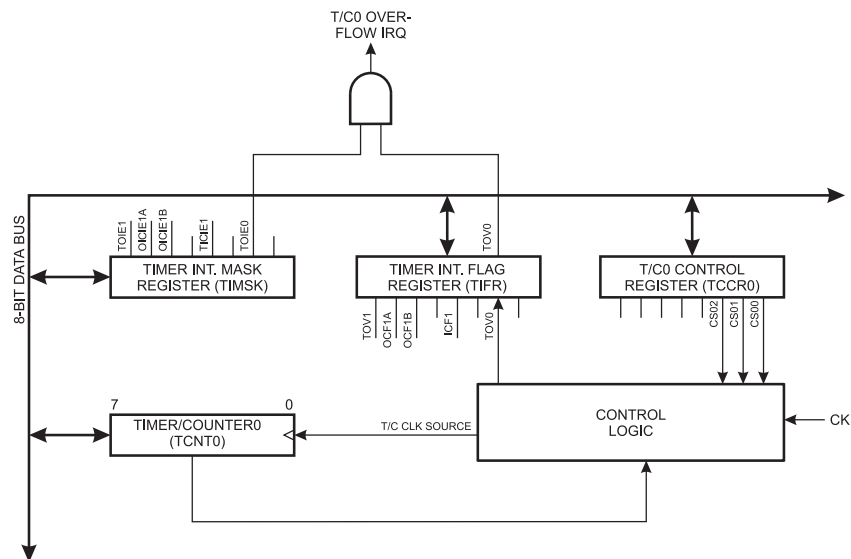
Figure 2 below shows the block diagram for Timer/Counter0.

The 8-bit Timer/Counter0 can select clock source from CK, prescaled CK, or an external pin. In addition it can be stopped as described in the specification for the Timer/Counter0 Control Register (TCCR0). The overflow status flag is found in the Timer/Counter Interrupt Flag Register (TIFR). Control signals are found in the Timer/Counter0 Control Register (TCCR0). The interrupt enable/disable settings for Timer/Counter0 are found in the Timer/Counter Interrupt Mask Register (TIMSK).

When Timer/Counter0 is externally clocked, the external signal is synchronized with the oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

The 8-bit Timer/Counter0 features both a high-resolution and a high-accuracy usage with lower prescaling opportunities. Similarly, high prescaling opportunities make the Timer/Counter0 useful for lower speed functions or exact timing functions with infrequent actions.

Figure 2. Timer/Counter0 Block Diagram.



The 16-Bit Timer/Counter1

Figure 3 shows the block diagram for Timer/Counter1.

The 16-bit Timer/Counter1 can select clock source from CK, prescaled CK, or an external pin. In addition it can be stopped as described in the specification for the Timer/Counter1 Control Registers (TCCR1A and TCCR1B). The different status flags (overflow, compare match and capture event) and control signals are found in the Timer/Counter1 Control Registers (TCCR1A and TCCR1B). The interrupt enable/disable settings for Timer/Counter1 are found in the Timer/Counter Interrupt Mask Register (TIMSK).

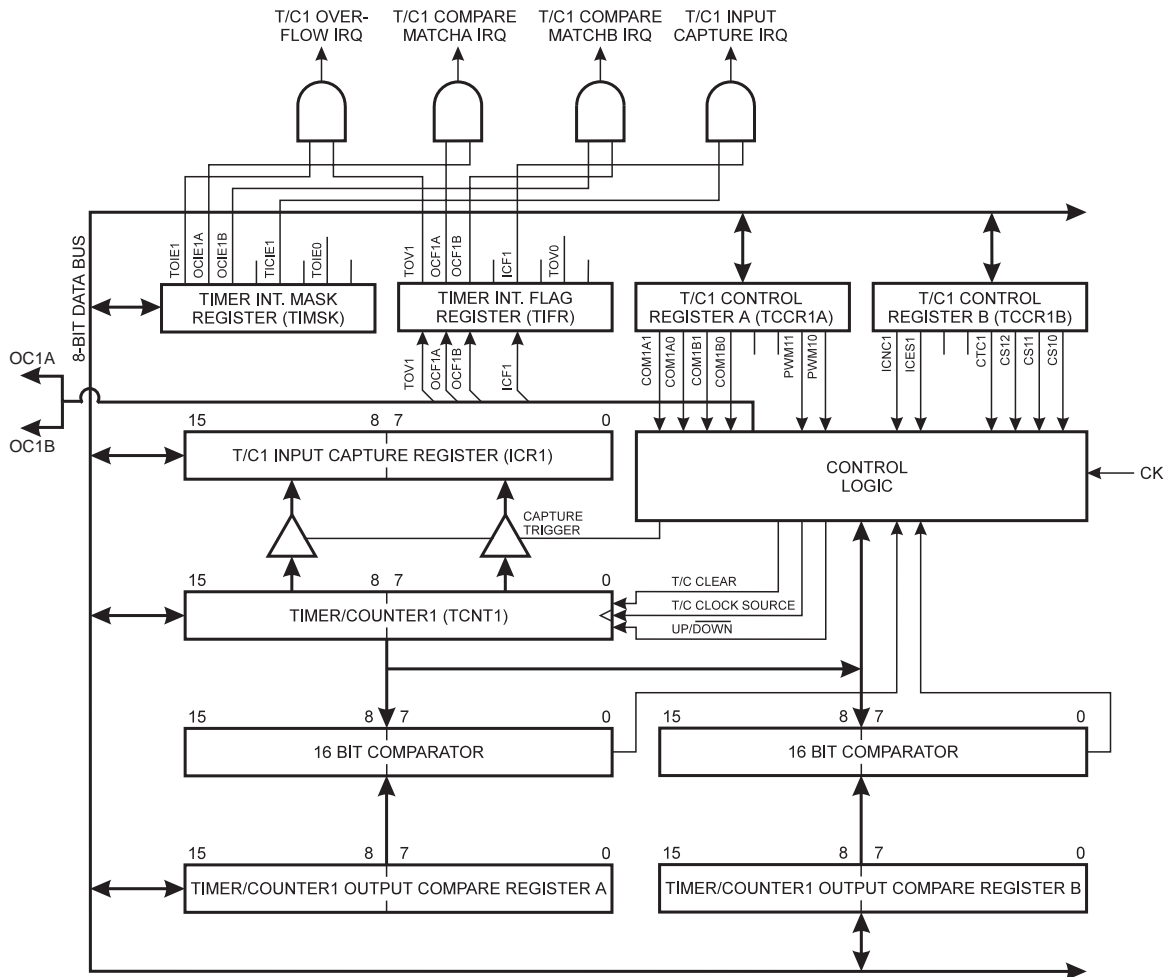
When Timer/Counter1 is externally clocked, the external signal is synchronized with the oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external

clock signal is sampled on the rising edge of the internal CPU clock.

The 16-bit Timer/Counter1 features both a high-resolution and a high-accuracy usage with lower prescaling opportunities. Similarly, high prescaling opportunities makes the Timer/Counter1 useful for lower speed functions or exact timing functions with infrequent actions.

The Timer/Counter1 supports two Output Compare functions using the Output Compare Register 1 A and B (OCR1A and OCR1B) as the data sources to be compared to the Timer/Counter1 contents. The Output Compare functions include optional clearing of the counter on compareA match, and actions on the Output Compare pins on both compare matches.

Figure 3. Timer/Counter1 Block Diagram



Hardware Configuration

To derive a long delay from a fast system clock, the prescaler and the two on-chip Timer/Counters are cascaded. The hardware configuration is illustrated in Figure 4.

The Timer/Counter1 is configured as a timer. It is programmed to divide the system clock by a user-defined ratio and toggles the OC1A pin each time it reaches the value written in the Timer/Counter0 output compare register (OCR1AH-OCR1AL). When OC1A toggles, the Timer/Counter1 register (TCNT1H-TCNT1L) is reloaded with 0000 and restarts counting.

Since OC1A is connected to T0 at the board level, the toggle on OC1A can trigger an event on Timer/Counter0. The latter is configured as a counter and increments at each rising edge on T0. When Timer/Counter0 overflows, a flag is set in TIFR, and an interrupt is eventually triggered. This indicates the programmed delay has elapsed.

The longest programmable delay can be calculated as follows:

Timer1 prescaler maximum ratio: 1024

Timer1 maximum division ratio: 65536

The toggle on OC1A implies an additional division ratio of 2

Counter0 maximum division ratio: 256.

More generally, the value of the programmed delay is given by the formula below:

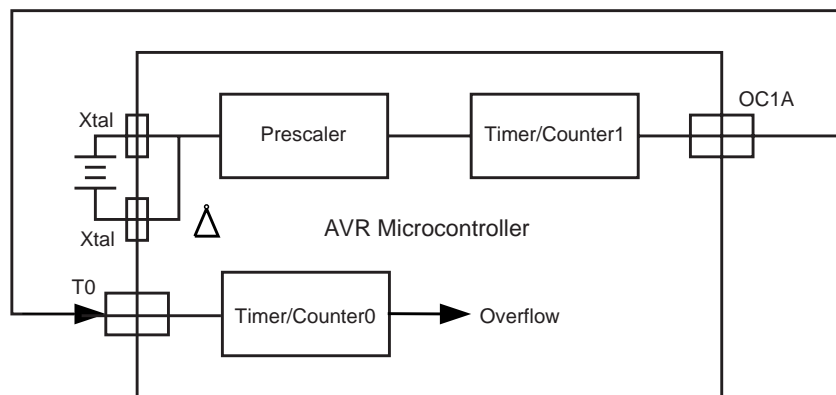
$$T = 2 / F_S \cdot T1P \cdot OCR1A \cdot (256 - TCNT0)$$

where:

F_S = System Clock Frequency

T1P = prescaler ratio defined in TCCR1B to be either 8, 64, 256 or 1024.

Figure 4. AVR Microcontroller Hardware Configuration for Long Delay Support



Example Software

This short program shows how to configure the Timer/Counters to implement a 1-hour delay with a 1 MHz system clock. (Processing power equivalent to 80C51.)

During the startup phase, the Timer/Counters and the interrupt controller are configured. Then idle mode is entered. After an hour, the delay elapses and an interrupt is trig

gered. This event wakes up the core which executes the user-defined task. In the example the task is just a toggle on an output pin (PA.0). When this task is complete, the core re-enters the idle mode.

This cycle continues indefinitely.

The commented source code follows.

```

; Constants definitions -----
.EQU PORTA = $1B
.EQU DDRA = $1A
.EQU DDRD = $11
.EQU TCCR1A = $2F
.EQU OCR1AH = $2B
.EQU OCR1AL = $2A
.EQU MCUCR = $35
.EQU TIMSK = $39
.EQU TCCR0 = $33
.EQU TCCR1B = $2E
.EQU TCNT1L = $2C
.EQU TCNT1H = $2D
.EQU TCNT0 = $32
.EQU SREG = $3F
.EQU SPH = $3E
.EQU SPL = $3D

; Interrupt service routines -----
.ORG $0000
    rjmp  start
    reti          ; INT0 service
    reti          ; INT1 service
    reti          ; T/C1 capture service
    reti          ; T/C1 compare match A service
    reti          ; T/C1 compare match B service
    reti          ; T/C1 overflow service
    reti          ; T/C0 overflow service
    reti          ; SPI transfer complete service
    reti          ; UART receive service
    reti          ; UART data reg empty service
    reti          ; UART transmit service
    reti          ; Analog comparator service

; Peripherals configuration -----
start:
    ldi  r17, $01    ; Register init

    ldi  r16, $01    ; Program PORTA.0 as an output
    out  DDRA, r16

    ldi  r16, $01    ; Initialize stack pointer ...
    out  SPH, r16    ; ... to 0x100
    ldi  r16, $00
    out  SPL, r16

```

```

ldi r16, $20 ; Program OC1A as an output
out DDRD, r16

ldi r16, $40 ; Program TCCR1A to toggle ...
out TCCR1A, r16 ; ... OC1A on each compare match

ldi r16, $6D ; Program the output compare ...
out OCR1AH, r16 ; ... register for a division ...
ldi r16, $DD ; ... ratio of 28125
out OCR1AL, r16

ldi r16, $20 ; Configure sleep mode
out MCUCR, r16

ldi r16, $02 ; Enable T/C0 interrupt
out TIMSK, r16

ldi r16, $80 ; Global interrupt enable
out SREG, r16

; Infinite loop -----
loop:
    rcall main ; Call the main routine

    ldi r16, $06 ; Reload counter 0 for a division ...
    out TCNT0, r16 ; ... ratio of 250

    ldi r16, $06 ; Start counter 0 for ...
    out TCCR0, r16 ; ... external pin T0 source

    ldi r16, $00 ; Reset timer 1 value
    out TCNT1H, r16
    ldi r16, $00
    out TCNT1L, r16

    ldi R16, $0C ; Start timer 1 for a prescale ...
    out TCCR1B, r16 ; ... ratio of 256

    sleep ; Wait for delay

    ldi r16, $00 ; Stop timer 1
    out TCCR1B, r16

    ldi r16, $00 ; Stop timer 0
    out TCCR0, r16

    rjmp loop

; Main routine -----
; This routine just toggles PORTA.0
main:
    in r16, PORTA
    eor r16, r17
    out PORTA, r16
    ret

```



Atmel Headquarters

Corporate Headquarters

2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe

Atmel U.K., Ltd.
Coliseum Business Centre
Riverside Way
Camberley, Surrey GU15 3YL
England
TEL (44) 1276-686677
FAX (44) 1276-686697

Asia

Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road
Tsimshatsui East
Kowloon, Hong Kong
TEL (852) 27219778
FAX (852) 27221369

Japan

Atmel Japan K.K.
Tonetsu Shinkawa Bldg., 9F
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Atmel Colorado Springs

1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Rousset

Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4 42 53 60 00
FAX (33) 4 42 53 60 01

Fax-on-Demand

North America:
1-(800) 292-8635
International:
1-(408) 441-0732

e-mail

literature@atmel.com

Web Site

<http://www.atmel.com>

BBS

1-(408) 436-4309

© Atmel Corporation 1999.

Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

Marks bearing ® and/or ™ are registered trademarks and trademarks of Atmel Corporation.

Terms and product names in this document may be trademarks of others.



Printed on recycled paper.

1268A-02/99/0M