APPLICATION NOTE

Atmel AVR134: Real Time Clock (RTC) Using the Asynchronous Timer

Atmel AVR 8-bit Microcontroller

Introduction

This application note describes how to implement a real time counter (RTC) on Atmel® AVR® microcontrollers that features the RTC module. The implementation requires only one discrete component – a 32.768kHz watch crystal. The application has very low power consumption because the microcontroller operates in Power-save mode most of the time. In Power-save mode the AVR controller is sleeping with only a timer running. The Timer is clocked by the external crystal. On every Timer overflow the time, date, month, and year are counted. This RTC implementation is written for the Atmel ATmega128, and can easily be ported to other AVRs with RTC Module. The advantages of implementing an RTC in software compared to an external hardware RTC are obvious:

- Lower cost
- Few external components
- Lower power
- Greater flexibility

Features

- Real Time Clock with Very Low Power Consumption (10µA @ 3.3V)
- Very Low Cost Solution
- Adjustable Prescaler to Adjust Precision
- Counts Time, Date, Month, and Year with Auto Leap Year Configuration
- Year 2000 Compliant Date Format
- Can be used on all AVR Controllers with RTC Module
- C-code for ATmega128 Included

Table of Contents

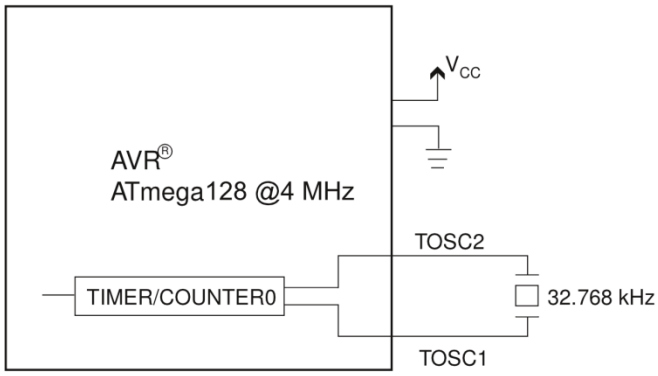
1. Theory of Operation

The implementation of an RTC utilizes the asynchronous operation of the RTC module. In this mode, Timer/Counter0 runs independently from the CPU clock.

Figure 1-1 shows the Atmel AVR Controller operating at 4MHz, from the divided internal calibrated RC-oscillator, when in active mode. When low power operation is desired, the AVR operates in power-down mode, with only the asynchronous timer running from an external 32.768kHz crystal.

The software Real Time Clock (RTC) is implemented using an 8-bit Timer/Counter with overflow interrupts enabled. Each timer overflow interrupt triggers an update of the software variables second, minute, hour, date, month, and year at the correct intervals.

Figure 1-1. Oscillator Connection for Real Time Clock



Because the amount of time needed to make the Timer/Counter overflow is constant, each of these timer variables will be incremented by a fixed number, for each timer overflow. This happens in the timer overflow interrupt routine.

To reduce power consumption, the AVR spends most of its time in Power-save mode, in which all on-chip modules are disabled, except for the RTC. As shown in Table 1-1, the MCU typically consumes less than 10µA in this mode. Each timer overflow interrupt will bring the device out of power-save mode, and back into active mode. When in active mode, the timer overflow interrupt routine is executed, before the device re-enters power-save mode.

To calculate the total power consumption, the power consumption in power-save mode should be added to the power consumption in active mode. However, the code executed when the device is awake is expected to be executed in less than 100 clock cycles, keeping the device awake for no more than 25µs (@4MHz). The current consumed in this period can therefore be considered neglectable.

According to the above and the current consumptions listed in Table 1-1, the current consumption can be estimated to be less than 10µA for this application, as opposed to 5mA if the device never was put in power-save mode.

Table 1-1. Current Consumption for the Atmel ATmega128 AVR Controller

Mode	Typical	Maximum
Active 4MHz, $3V_{CC}$	5mA	5.5mA
Idle 4MHz, $3V_{CC}$	2mA	2.5mA
Power-down, $3V_{CC}$	<5µA	10µA
Power-save, $3V_{CC}$	<10µA	-

2. Calculation

Given the frequency of the watch crystal, the user can determine the time for each Timer/Counter tick, by selecting the desired pre-scaling factor. As shown in Table 2-1, CS02, CS01, and CS00 in the TCCR0 (Timer/Counter0 Control Register) define the Timer/Counter pre-scaler source. In the table below clkT0S denotes the frequency of the Timer/Counter0 clock source. For example, if clkT0S equals 32.768kHz, as in our case, the Timer/Counter will tick at a frequency of 256Hz with a prescaler of 128.

Table 2-1. Timer/Counter0 Prescaler Select

CS02	CS01	CS00	Description (1)	Overflow period
0	0	0	Timer/Counter0 is stopped	–
0	0	1	clkTOS	1/128s
0	1	0	clkTOS /8	1/16s
0	1	1	clkTOS /32	1/4s
1	0	0	clkTOS /64	1/2s
1	0	1	clkTOS /128	1s
1	1	0	clkTOS /256	2s
1	1	1	clkTOS /1024	8s

Note: 1. clkTOS = 32.768kHz.

3. Configuration Example

As shown in Figure 1-1, the crystal should be connected directly between the pins TOSC1 and TOSC2. Some devices might require additional external capacitors on these pins, as the internal oscillator characteristics can vary. Refer to the device datasheet for details on crystal connections. The oscillator is optimized for use with a 32.768kHz watch crystal, or an external clock signal in the interval of 0Hz - 256kHz. In this example, the eight LEDs in port B are used to display the value of the RTC. The LED on pin B0 will change state every second. The next six LEDs represent the minutes in binary, and the LED on pin B7 alternates every hour.

Some considerations must be taken into account when clocking the Timer/Counter from an asynchronous clock source. A 32.768kHz crystal has a stabilization time of up to one second after power-up. The controller should therefore not enter power-save mode less than one second after power-up. Care should also be taken when configuring the timer/counter to operate in asynchronous mode. See the data sheet for detailed instructions. When updating the timer register, the data is transferred to a temporary register and latched after two external clock cycles. The Asynchronous Status Register (ASSR) contains status flags that can be checked to verify that the written register is updated.

4. Implementation

The software consists of two subroutines, a FUSES section and one Interrupt Service Routine (ISR).

The init() routine configures the device to the needs of this application.

- The AS0 bit in the ASSR (Asynchronous Status Register) is set to configure Timer/Counter0 to be clocked from an external clock source. Only this timer can perform asynchronous operations
- The start value for the timer is reset
- The desired prescaler is selected
- To synchronize with the external clock signal, the program will wait for the ASSR register to be updated

- The TOIE0 bit in the TIMSK (Timer/Counter Interrupt Mask Register) is set to enable the Timer/Counter0 Overflow Interrupt
- The Global Interrupt Enable bit in SREG (Status Register) is set to enable interrupts. Setting this bit is done calling the sei() macro
- Calling the set_sleep_mode(…) macro with the desired sleep mode as argument sets the SM1 and SM0 bit in MCUCR (MCU Control Register) to select what sleep mode the device should enter

The other sub routine, “not_leap”, corrects the date for leap years. The routine is described in detail in the flow diagram provided in Section 4.2.

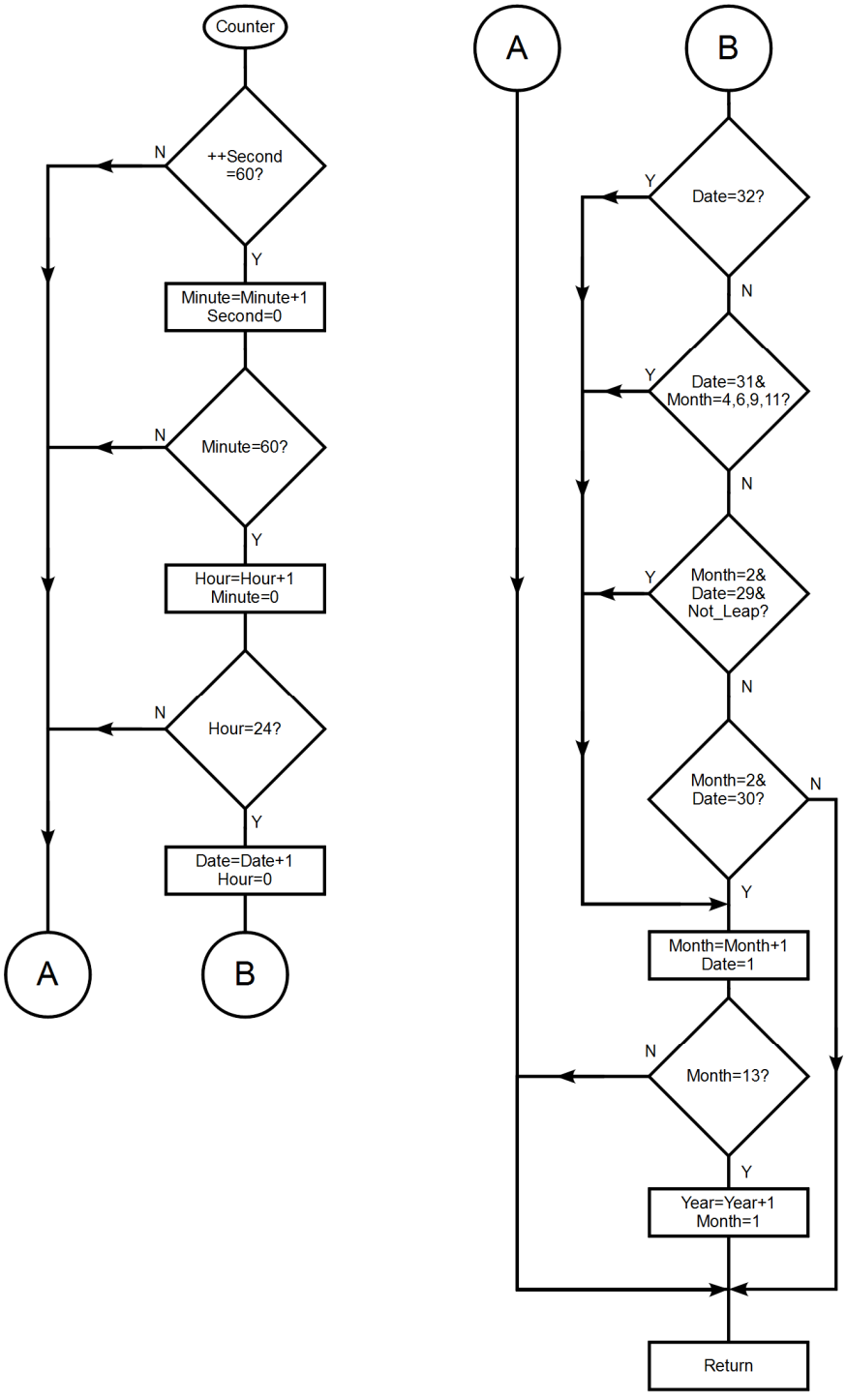
The FUSES section modifies the device fuses to make it run from the internal calibrated RC oscillator, with a prescaler of two. This results in a clocking speed of 4MHz.

The main program controls the power down sequence. The sleep_mode() macro will place the controller in sleep mode. To make sure the interrupt is cleared upon wakeup, the main routine also makes sure the device never re-enters sleep mode within less than one TOSC cycle.

4.1 Timer/Counter0 Overflow Interrupt Service Routine

The interrupt service routine is executed every time TCC0 overflows. The interrupt request wakes the MCU to update the timer variables. An interrupt routine cannot return or accept any variables. A global structure with timer variables are declared to keep track of time: “second”, “minute”, “hour”, “date”, “month” and “year”. Since the time required to complete one timer overflow is known, second will be incremented by a fixed number for every interrupt. Once it reaches 60, minute is incremented by 1 and second is set to 0.

Figure 4-1. Flow Chart, TCC0_Overflo ISR

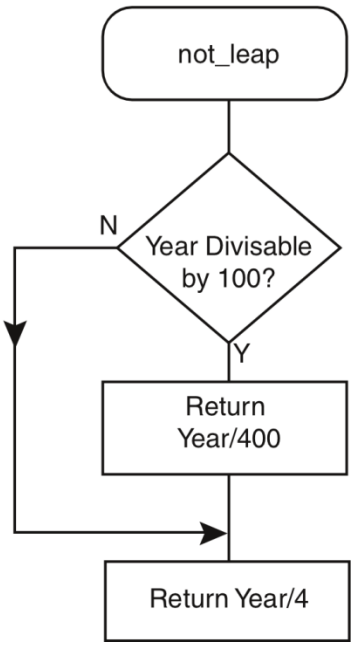


4.2 “not_leap” Subroutine

This routine checks whether or not a year is a leap year. It returns false if a year is leap, and true if not. A year is considered to be leap if both of the following conditions are met:

1. The year is divisible with four, and
2. If the year is divisible with 100, it also has to be divisible by 400.

Figure 4-2. Flow Chart, “not_leap” Subroutine



5. Accuracy

The RTC on the AVR controller maintains high accuracy as long as the watch crystal is accurate. Asynchronous operation allows the timer to run without any delays, even when the CPU is under heavy load. However, a small neglectable discrepancy does occur because the timer variables are not updated in parallel. By the time they are finished updating, they deviate from the Timer/Counter very slightly. The largest discrepancy occurs when all the timer variables are overflowed. At this moment, “second” is 59, “minute” is 59, “hour” is 23, and so on. It takes 94 cycles for the MCU to complete the update. At a 4MHz CPU clock, the error between the RTC and the watch crystal will not exceed 23.5µs found by 94/(4 * 106). A typical error should be 6µs since 24 cycles are needed to update “second”. This error does not accumulate since the Timer is always synchronous with the watch crystal.

6. Revision History

Doc. Rev.	Date	Comments
1259H	01/2014	New template and bugfixing
1259G	04/2009	Bugfixing
1259F	08/2006	Bugfixing
1259E	Unknown	
1259D	Unknown	
1259C	Unknown	
1259B	Unknown	
1259A	Unknown	Initial document release

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Building
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370