AVR1314: Using the XMEGA Real Time Counter

Features

- **16- bit RTC with programmable period and alarm.**
- **Clock source options**
  - **Accurate Internal 32 kHz RC oscillator**
  - **Ultra Low Power Internal 32 kHz RC oscillator**
  - **External 32 kHz crystal oscillator.**
- **Optional division of 32 kHz clock to 1kHz to save power**
- **RTC clock prescaling factor between 1 and 1024.**
  - **Timer tick between 31us and 1 second**

1 Introduction

In many embedded systems a real time counter is used to keep track of time. This can be to schedule application functions, like the duration of a washing cycle in a washing machine. It can also be used to wake up the MCU from low power modes on (ir)regular intervals The ability to use long time out periods reduces the battery power consumption in applications like e.g. thermostats, wireless applications and sensor/actuators.

This application note covers the use of the 16-bit Real Time Counter (RTC) in the XMEGA™.

8-bit AVR® Microcontrollers

Application Note

Rev. 8047A-AVR-02/08

2 Real time counter overview

The XMEGA RTC is a 16-bit timer with programmable period time and a single compare channel.

2.1 Registers

The RTC count, period, and compare value are all 16-bit values. Since the data bus is only 8-bit wide, two registers are used to represent a 16-bit value. The high byte is referred to with an “H” suffix and low byte with an “L”. Most C-compilers will handle the 16-bit access automatically when symbolic names of the 16-bit registers are used. It is also possible to access the high and low word individually. The 16-bit registers for the RTC are CNT[H:L], PER[H:L], and CMP[H:L].

2.2 Interrupts

The RTC can generate two interrupts, an overflow interrupt, and one compare interrupt. The overflow interrupt will have a fixed frequency, as long as the period register (PER) is not changed: it can e.g. interrupt with 1 Hz interval – to update the clock digit on an LCD. The compare register (COMP) offers a way to set a variable timeout interrupt, without changing the period or reloading the timer’s count registers (CNT).

2.3 Clock sources

The RTC can be clocked from of three clock sources:

- Internal 32 kHz Ultra-Low Power (ULP) RC oscillator
- Internal 32kHz Calibrated RC oscillator (more accurate than the ULP RC, but draws more current).
- An external 32.768kHz watch crystal oscillator (highly accurate).

To reduce the power consumption the clock system can divide the 32 kHz clock sources to 1 kHz before providing it to the RTC module. An RTC clock cycle is therefore either 1ms or 31us. The RTC clock must be enabled in the clock system before the RTC can be used. Please refer to the application note AVR1003 and the datasheet for details on the clock system.

The RTC module has an internal prescaling option. Prescaling factors up to 1024 are available. The prescaler factor is controlled by the CTRL register.

The maximum timeout period and time resolution for the RTC is determined by the RTC clock frequency. If a fast clock is used this gives a fine timer resolution, but at the expense of fairly short timeout periods. If on the other hand the clock frequency is increased, the timeout period can be quite long, but with lower resolution. A few examples are given in the table below:

Table 2-1. RTC clock frequency, time resolution and timeout period

Scaled RTC clock	Timer tick resolution	Timeout period
32.768 kHz	31 us	2 sec.
1.024 kHz	1 ms	64 sec
1 Hz	1 sec	18 hours 12 min

This means that ultra low power battery applications can spend very long periods in sleep mode. This reduces power consumption, as frequent wakeups only to update the real time variables are not needed. The MCU only wakes up when processing is required.

2.4 Clock domains – synchronized and unsynchronized RTC registers

Since the RTC is operating from a different clock than the CPU, it has another "clock domain" than the CPU. When the CPU and RTC exchange information across the clock domain boundary, synchronization between the two clock domains is required. The synchronization is hardware controlled. When data is transferred from CPU into RTC domain, the synchronization takes 3 clock cycles in the RTC clock domain. Note that, when referring clock cycles in the RTC clock domain, it is the unscaled clock input to the RTC module that it referred to. This is either 32 kHz or 1 kHz.

The synchronization from RTC to CPU domain is 3 CPU cycles. This means that unless the CPU clock is very slow the duration of the synchronization in this direction is negligible.

The four synchronized registers are CTRL, CNT, PER and COMP. Note that the PER register does not have a separate synchronizer; the synchronization of PER is triggered by synchronization of the CNT, CTRL or COMP registers.

Synchronization from CPU into RTC domain is triggered when writing the high byte of the COUNT, COMP or PER registers. Writing the low bytes does not trigger synchronization. When writing new values it is therefore relevant to access the registers in correct order – write the low byte first and then the high byte. If the PER register is updated, one of the other registers must also be updated to synchronize PER value into RTC domain The mechanism is designed in this ways to ensure that corresponding PER and COMP values can be loaded in the same RTC cycle, while the RTC clock is running.

To monitor if synchronization is completed the SYNCBUSY flag in the STATUS register can be inspected. The flag is set while synchronization is ongoing. Note that once the synchronization of a register is triggered, this register cannot be written again until the synchronization is completed. This applies to all four synchronized registers. Writing to one register does not block access to other registers.

2.5 Special concerns for sleep mode

Due to synchronization it is impossible to wake up from sleep mode by RTC interrupts more frequent than every 4 RTC clock cycles (unscaled clock). If for instance the compare period is set to 3 RTC timer ticks (and the clock is not prescaled in the RTC module), the synchronization will cause every second interrupt to be missed. If this interrupt is used to wake up the part from sleep this would be observed as if the wake-up from sleep occurs at half the expected frequency. This can be avoided by selecting an appropriate time base for the RTC with a higher frequency.

2.6 Connecting the RTC to the Event System

It is possible for the RTC to generate events on overflow and compare match – for details about the event system please refer to the datasheet and application note AVR1001. If combined with the Timer/Counters, this can be used to form a 24 hour timer: The RTC can generate an event every 60 seconds. The “minute-event” is used as clock source to a Timer Counter, which in turn generates overflow event to the event system after 60 minutes. The “hour-overflow” event is used at clock for another Timer Counter, which keeps tracks of the hours.

Note that the event system is not operating in other sleep modes than Idle mode. Idle and Extended Power Save are however quite similar in power consumption when peripherals are not enabled.

2.7 Operation of the RTC in debugging mode

The RTC clock is blocked when breaking code execution in debugging mode. This ensures that interrupts are not generated continuously when single stepping. However, the timing of the RTC will be affected when single stepping the code as the RTC clock source is asynchronous to the CPU clock.

3 Required methods of use

The following subsections describe the recommended procedures for initialization and configuration of the RTC. All the procedures below assume that clock system is set up to provide a clock for the RTC module.

3.1.1 Initialization (RTC not already running)

1. Write period value by writing to the PER register.
2. Write compare match and count values (COMP and CNT).
3. Set interrupt level for compare match and overflow interrupt.
4. Set clock prescaler selection in CTRL register.

3.1.2 Re-initialization (RTC running)

1. Stop RTC clock, by clearing prescaler.
2. Wait for BSY flag to clear.
3. Write period value by writing to the PER registers.
4. Write compare match and count values (COMP and CNT).
5. Set interrupt level for compare match and overflow interrupt.
6. Set prescaler setting in CTRL register.

3.1.3 Change RTC period

1. Wait for SYNCBUSY flag to clear.
2. Write period value by writing to the PER registers.
3. Write either CNT or COMP register.

3.1.4 Changing compare match or count value

1. Wait for SYNCBUSY flag to clear.
2. Write either CNT or COMP register.

3.1.5 Enter sleep mode

1. Wait for SYNCBUSY flag to clear.
2. Enter sleep mode.

4 Driver Implementation

This application note includes a source code package with a basic driver implemented in C.

Note that this driver is written to be highly readable and as general example how to use the peripheral module. If using the driver in an application it may be desirable to copy relevant parts of the code to where it is needed, to reduce to number of function calls. This will both speed up the code and reduce the code footprint.

4.1 Files

The source code package consists of three files:

- *rtc_driver.c* – driver source file
- *rtc_driver.h* – driver header file
- *rtc_example.c* – Example code using the driver

For an overview of the available driver interface functions and their use, please refer to the source code documentation.

4.2 Doxygen documentation

All source code is prepared for automatic documentation generation using Doxygen. Doxygen is a tool for generating documentation from source code by analyzing the source code and using special keywords. For more details about Doxygen please visit http://www.doxygen.org. Precompiled Doxygen documentation is also supplied with the source code accompanying this application note, available from the *readme.html* file in the source code folder.

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

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

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature