AVR1321: Using the Atmel AVR XMEGA 32-bit Real Time Counter and Battery Backup System

8-bit AVR® Microcontrollers

Application Note

Features

- **32-bit Real Time Counter (RTC)**
  - **32-bit counter**
  - **Selectable clock source**
    - **1.024kHz**
    - **1Hz**
  - **Long overflow time**
    - **More than 136 years when using 1Hz as the clock source**
  - **Ultra low power consumption**
  - **Able to wakeup MCU from Power Save mode**
  - **Optional Interrupt/Event on overflow and compare match**
- **Battery Backup System**
  - **Supplies the 32-bit RTC and two backup registers with supply voltage from V_{BAT} power pin when main power is lost**
  - **Automatic switching from main power to battery power at brown-out detection of main power**
  - **Automatic switching from battery backup power to main power when main power is available again**
  - **Crystal failure detection monitor**

1 Introduction

The Atmel®AVR®XMEGA® 32-bit real time counter module is designed to count 1.024kHz or 1Hz clock source cycles. It can be used to wake up the MCU from low power modes on specified intervals by overflow interrupt or (and) compare match interrupt. Thanks to the 32-bit resolution, the maximum overflow time is longer than 136 years when using a 1Hz clock source. With a software algorithm, the RTC can easily be used to provide calendar functionality, and derive time of day, week, and year.

The 32-bit RTC is contained within a battery backup system which is a separate power domain that can be powered from a battery backup when the main power is lost. The transition from main power to backup power is done automatically by the backup system and thus ensures safe operation of the RTC. The backup system with the RTC forms an ideal solution for systems that need to keep track of time even while the rest of the system is without power. For example imagine a battery powered handheld device, at some point the user may need to swap batteries but this would leave to a loss of time when no backup system is available. The fully integrated solution removes the cost of an external real time clock and/or power switches.

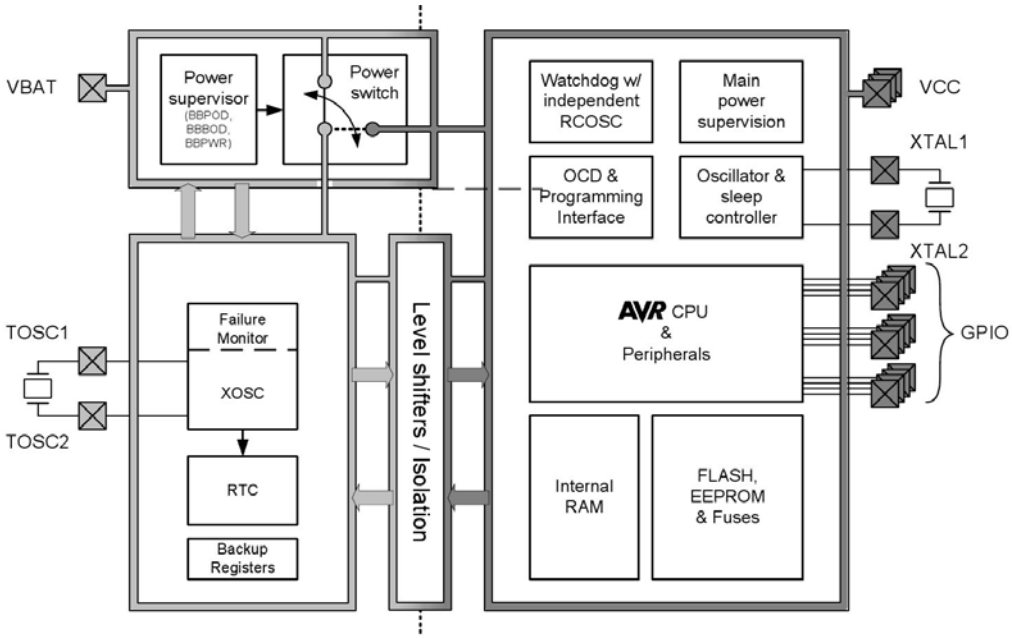
This application note covers the use of the 32-bit Real Time Counter and the battery backup system that is available on some XMEGA devices in detail.

Rev. 8307B-AVR-02/11

2 Battery Backup System

The battery backup system provides the option to supply all peripherals that are contained within its power domain with power from an optional external supply while main power is lost.

Figure 2-1. Battery Backup System overview



Following modules are located inside the backup power domain:

- Battery backup system supervisor
- Power switch
- Low power 32.768kHz crystal oscillator
- Oscillator failure detection monitor
- 32-bit Real Time Counter (RTC)
- 2 backup registers
- Battery backup brown-out detector (BBBOD)
- Battery backup power-on detector (BBPOD)

The battery backup system does not provide power to other parts outside of its power domain like volatile memory in the device such as SRAM and I/O registers!

The device uses its BOD (Brown-out Detector) on the main supply to detect main power loss, and switches automatically to the backup supply that is connected to the V_{BAT}. In addition, the backup system BOD is turned on to monitor the backup supply. After main power is restored, the system will automatically switch back to the main power.

The battery backup is only drained when main power is not present. This ensures maximum battery life time for the battery backup.

2.1 Power supervisor

The power supervisor handles the startup of the backup system and updates the status of the backup system status flags. The supervisor has a power-on and brown-out detector and from these the status flags BBPWR, BBPODF and BBODF are generated as follows:

The power detection (BBPWR) function checks the VBAT voltage each time the device leaves a reset state. If no voltage is present on the VBAT pin, the Battery Backup Power (BBPWR) flag will be set. This indicates that the backup battery has been removed or drained while main power was lost. Since this flag is updated only during the startup sequence of the system it is not possible to use this flag as a backup battery power monitor.

The power-on detection (BBPOD) function detects when power is applied to the VBAT pin, i.e. when the backup battery is inserted. When this happens the battery backup power-on detection flag (BBPODF) is set, and the power switch is disconnected to prevent the backup battery to be drained before the device is configured. This flag is only updated during a reset of the device; take a look at Chapter 2.2 for more information. Since this flag is only updated once, a disconnection of the backup power will only be detected after a reset of the device (such as re-applying the main power).

The brown-out detection (BBBOD) function monitors the VBAT voltage level when the battery backup system is powered from the VBAT pin. This detector is not active when main power is available, so it is not possible to detect a low battery backup voltage while running from main supply with this detector. When running from the backup supply and if then the VBAT voltage drops below a threshold voltage, the battery backup BOD flag (BBBODF) will be set. The BBBOD samples the VBAT voltage level at around 1Hz rate and is designed for detecting slow voltage changes.

2.2 Backup System Startup Sequence

Each time the main power has been restored, the backup system will run its startup sequence. Note that this reset sequence is not triggered by a software reset of the backup system; it is triggered only by a reset of the device. Following actions are performed internally in the backup system:

1) The brown-out detector at the VBAT input is sampled to check if the backup battery has sufficient power. If the VBAT input voltage is not sufficient the BBPWR flag is set
2) The BBPWR flag is stored in the status register
3) If sufficient power is measured on VBAT the flags BBBODF and BBPODF are stored in the status registers, if no valid power (BBPWR) is detected, these flags are invalid
4) The VBAT system switches to the main power, disconnects VBAT input and enters normal mode

Since BBPWR, BBPODF and BBBODF are only updated during the reset sequence of the backup system it is not possible to use these flags to monitor the backup battery state while running from main power.

2.3 Crystal oscillator and failure detection monitor

The Crystal Oscillator (XOSC) supports connection of an external 32.768kHz crystal. It provides a pre-scaled clock with either 1.024kHz or 1Hz output.

The crystal oscillator is designed for ultra low power consumption and therefore has limited drive strength to achieve this. Reduced drive strength can also make the oscillator more susceptible to noise or other influences. Because of that the oscillator offers a configuration option for a “high ESR” mode, which will increase the drive strength at the expense of power consumption.

The crystal oscillator failure monitor will detect if the crystal is permanent or temporary stopped, and then set the Crystal Oscillator Failure flag. The failure detection mechanism has an integrated low power RC oscillator that is fed into a counter. The counter will be reset each time a clock tick, if the crystal oscillator is detected. If the counter is not reset before an overflow, the failure detection flag will be set. A failure will be detected if during 64 cycles of an internal 32kHz RC oscillator no cycle of the crystal oscillator was detected. The supervision oscillator has an accuracy of ~30%.

2.4 Backup registers

The backup system provides two 8-bit registers that can be used to store information that should be backed up during a loss of main power.

3 32-bit RTC

The XMEGA 32-bit RTC contains a 32-bit counter and a single compare channel. An optional overflow interrupt can be generated when the counter value reaches the specified period time. In addition, a compare match interrupt can be generated when the counter value is equal to the specified compare value.

3.1 Registers

The RTC count, period, and compare value are all 32-bit values. Since the data bus is 8-bit wide, four registers are used to represent a 32-bit value. The register name with a “3” suffix is the most significant byte. The 32-bit registers for this RTC are CNT[3:0], PER[3:0], CMP[3:0].

3.1.1 Byte access

Most C-compilers will handle the 32-bit access automatically when symbolic names of the 32-bit registers are used. It is also possible to access the 8-bit register individually.

To write a 32-bit register of 32-bit RTC module in byte-accessed way, the least significant byte must be written first. After writing the most significant byte, the new value will be available after several RTC clock cycles. Refer to Chapter 3.4 for more detailed information about the synchronization of the clock domains.

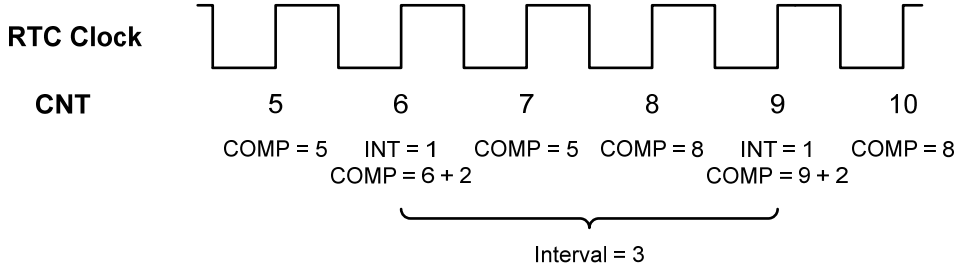
3.2 Interrupts

The RTC can generate two interrupts, an overflow interrupt, and one compare interrupt. The overflow interval is controlled by PER register. This interrupt can be used to wake the MCU or generate an event. The compare register offers a way to set a variable timeout interrupt without changing the period or reloading the timer’s count registers (CNT). Users can for example utilize this interrupt to wake MCU from Power Save mode every minute, to update the time display on LCD, or to implement a calendar alarm function.

The compare match interrupt will be raised on the next count after compare match condition occurs. That means an addition cycle will be added in to the compare match

intervals. For instance, suppose the compare period is set to 2 RTC clock cycles (COMP = CNT + 2), the compare match interval will be 3 RTC clock cycles.

Figure 3-1. Additional cycle in compare match interval



3.3 Clock sources

The 32-bit RTC can only be clocked by one of two outputs of the oscillator; these pre-scaled clocks are 1Hz and 1.024kHz.

To reduce the power consumption and get the longest time before overflow, the 1Hz output can be used. In this case, the timer tick resolution is 1 second, so that the maximum overflow time will be 2^{32} seconds which is more than 136 years. The benefit of using the 1.024kHz output as the clock source is a higher resolution of approximately 1ms.

3.4 Clock domains

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.

There are four registers that need synchronization, CTRL, CNT, PER and COMP. The synchronization is controlled by hardware. That means, by changing the value of the 8-bit register (CTRL) or writing the most significant byte of the 32-bit register (CNT, PER, COMP), synchronization is triggered. For 32-bit registers, writing the lower bytes has no effect. It is therefore relevant to access the registers in correct order – writing the lower bytes first and then the most significant byte.

The synchronization from RTC to CPU domain is triggered by the SYNCCNT bit of SYNCCTRL register. Note that only CNT synchronization is needed.

To monitor if synchronization is completed the SYNCBUSY flag in the SYNCCTRL register can be inspected. Only CTRL or CNT synchronization will affect this flag.

The synchronization time varies from register to register. Described in the table below:

Table 3-1. RTC registers synchronization cycle

Register Name	Write Synchronization Cycle	Read Synchronization Cycle
CTRL	0.5 RTC clock cycle	
CNT	12 peripheral clock cycles	8 peripheral clock cycles
PER	2 RTC clock cycles	
COMP	2 RTC clock cycles	

To ensure the correct operation, the peripheral clock must be eight times faster than the RTC clock (1.024kHz or 1Hz) when any of the control and count register is accessed, expect 12 times faster when the count register is written.

3.5 Special concerns for sleep mode

Due to synchronization it is impossible to wake up from sleep mode by RTC interrupts more frequent than every 3 RTC clock cycles, that is 2 RTC cycles for synchronization, 1 addition RTC cycle for interrupt delay (refer to Chapter 2.2 for more details). If the compare interval is set to 2 RTC clock cycles (COMP = CNT + 1) or less, when the new compare value is synchronized, the actual CNT value is larger than the compare value, so that the compare match interrupt will not be happened as expected.

3.6 Connecting the RTC to the Event System

It is possible for the 32-bit RTC to generate events on overflow and compare match. For details about the event system, please refer to the datasheet and application note AVR1001.

Note: The event system is not operating in other sleep modes than Idle mode.

3.7 Operation of the RTC in debugging mode

The RTC clock is blocked when code execution is stopped in debugging mode for example by a breakpoint. This ensures that interrupts are not generated continuously in a debug session where single stepping is desired. However, the timing of the RTC will be affected when single stepping the code because the RTC clock source is asynchronous to the CPU clock.

4 Configuration of battery backup system and RTC

4.1 Precautions

4.1.1 BOD configuration

Because the backup system needs to know when main power is lost, the activation of the main BOD is mandatory. Atmel®AVR®XMEGA® has two fuses for the BOD configuration which offers different configurations for the operation modes of the device. Following fuse settings are necessary to operate the backup system:

- The BOD fuse for active and idle mode must be set to continuous BOD
- The BOD fuse for the power-down modes must be set to either continuous mode or sampled mode

4.1.2 Check power-on slope

Activation of the main supply should result in a monotonic rising slope. Otherwise it could happen that the device is reset twice when the slope has some drop in-between. The consequence of a double reset will be the clearing of the BBPODF and BBBODF flags, that is, a double reset will look like the backup battery was never inserted and it is not known whether the backup system had a brown-out.

In order to avoid this, it is recommended to check that the main supply rising slope is monotonic.

In addition, it is best to check for the BBPODF and BBBODF flags very early in the application code. When tested early it is less likely that these are missed due to another reset such as power supply drop or watchdog timer.

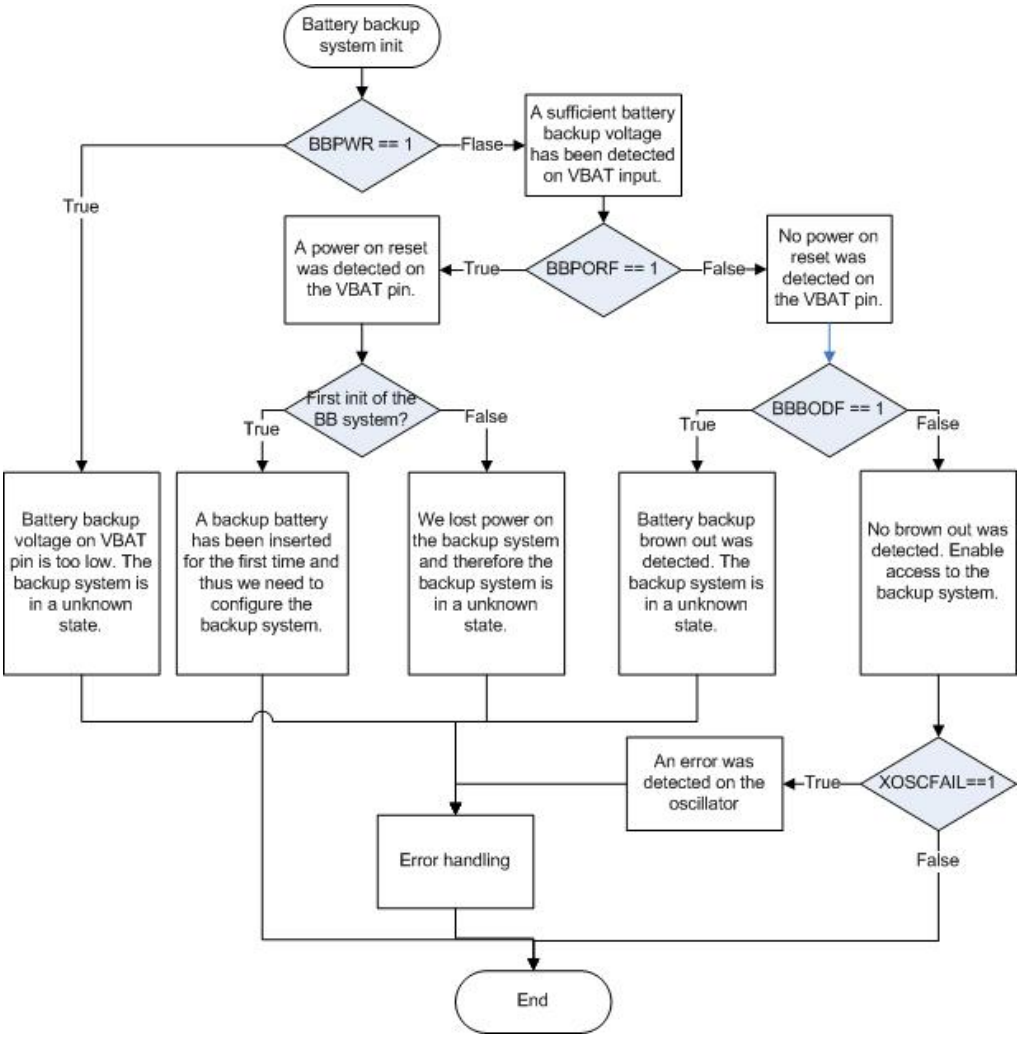
4.1.3 Additional safety with the oscillator failure detection mechanism

For applications where a correct time is critical, it is recommended to enable the oscillator failure detection mechanism. This feature will detect if the oscillator has stopped working for at least 64 cycles. An oscillator issue can thus be detected by checking the XOSCFAIL flag at start-up and by monitoring the flag in the application.

4.2 Backup system check after start-up

When the backup functionality is used, it is necessary to check the backup system state when main power is applied in order to evaluate if there was any issue during the absence of the main power. This check should be one of the first actions in the application as discussed in chapter Check power-on slope.

Figure 4-1. Recommended procedure to check the backup system state



The recommended way to determine the backup system state is outlined in the Figure 4-1. In order to be able to know whether the system is initialized the first time or a

power loss was detected on VBAT, it is necessary to store this information permanently, for example in EEPROM.

4.3 Initialization

Whether or not the backup functionality is used, the initialization of the oscillator and the RTC is always the same. Following steps are needed to configure them:

1. Perform a battery backup system RESET
2. Set the ACCEN bit in CTRL register of battery backup system
3. Enable oscillator failure detection
4. Optionally select input clock to the RTC
5. Optionally enable high ESR mode for oscillator
6. Wait 200µs
7. Enable oscillator
8. Wait until crystal is stable by polling XOSCRDY flag (the flag will be set after ~8k cycles of the oscillator have been detected)
9. Set period value by writing PER register in RTC module
10. Optionally set compare match value (COMP) and initial count value (CNT)
11. Optionally set interrupt level for compare match and overflow interrupt
12. Enable the 32-bit RTC by setting ENABLE bit in CTRL register of RTC module

4.4 Change RTC period

1. Disable the RTC module
2. Wait until synchronization finished by checking SYNCBUSY bit in SYNCCTRL register of RTC module
3. Write PER registers
4. Re-enable RTC module

4.5 Read count value

1. Set SYNCCNT bit of SYNCCTRL register
2. Wait until synchronization finished
3. Read CNT registers

4.6 Change compare match or count value

1. Wait until synchronization finished
2. Write either COMP or CNT register

5 Application Implementation

5.1 Drivers

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 the number of function calls.

5.2 Examples

The source code package consists of two examples. Below are the files:

- rtc32_driver.c – driver source file
- rtc32_driver.h – driver header file
- vbat.c – driver source for VBAT system
- vbat.h – header file for VBAT driver
- rtc32_example1.c – example code using the driver

5.2.1 Example 1

This example shows how to use 32-bit RTC, and wake up the MCU from Power Save mode every 3seconds.

5.2.1.1 Power consumption calculation

In most applications, power consumption is a very important parameter. We need to calculate it very carefully to know if the average power consumption is low enough to provide the desired lifetime of a battery. The average current is defined by the following equation:

Equation 5-1. Average current equation

$$I_{average} = \frac{T_{active} \cdot I_{active} + T_{sleep} \cdot I_{sleep}}{T_{active} + T_{sleep}}$$

The T_{active} represents time consumption in active mode. It can be calculated by the following equation:

Equation 5-2.

$$T_{active} = \frac{N}{f_{clk}}$$

The N variable represents how many clock cycles will be executed in active mode.

Hereunder is a typical loop cycle which is disassembled from the object code of Example 1, generated by IAR™ compiler.

Code example:

```
RETI                    Interrupt return
RJMP      PC-0x000D     Relative jump
LDI       R16,0x02      Load immediate
LDI       R17,0x00      Load immediate
LDI       R18,0x00      Load immediate
LDI       R19,0x00      Load immediate
CALL      0x000001DB    Call subroutine
```

```
LDI       R30,0x21       Load immediate
LDI       R31,0x04       Load immediate
LDD       R20,Z+0        Load indirect with displacement
ORI       R20,0x10       Logical OR with immediate
STD       Z+0,R20        Store indirect with displacement
LDD       R20,Z+0        Load indirect with displacement
SBRC      R20,4          Skip if bit in register cleared
LDI       R30,0x24       Load immediate
LDD       R20,Z+0        Load indirect with displacement
LDD       R21,Z+1        Load indirect with displacement
LDD       R22,Z+2        Load indirect with displacement
LDD       R23,Z+3        Load indirect with displacement
ADD       R20,R16        Add without carry
ADC       R21,R17        Add with carry
ADC       R22,R18        Add with carry
ADC       R23,R19        Add with carry
LDI       R30,0x28       Load immediate
LDD       R16,Z+0        Load indirect with displacement
LDD       R17,Z+1        Load indirect with displacement
LDD       R18,Z+2        Load indirect with displacement
LDD       R19,Z+3        Load indirect with displacement
CP        R16,R20        Compare
CPC       R17,R21        Compare with carry
CPC       R18,R22        Compare with carry
CPC       R19,R23        Compare with carry
BRCC      PC+0x09        Branch if carry cleared
LDI       R30,0x2C       Load immediate
STD       Z+0,R20        Store indirect with displacement
STD       Z+1,R21        Store indirect with displacement
STD       Z+2,R22        Store indirect with displacement
STD       Z+3,R23        Store indirect with displacement
RET                      Subroutine return
LDI       R16,0x02       Load immediate
STS       0x0607,R16     Store direct to data space
LDI       R16,0x07       Load immediate
STS       0x0048,R16     Store direct to data space
SLEEP                    Sleep
```

The instructions above expend 73 clock cycles. If the main clock is 32MHz, according to Equation 5-2, we can get:

$$T_{active} = \frac{73}{32MHz} = 2.28125us$$

In case of 3.0V power supply, according to the ATxmega256A3B's datasheet, the I_{active} = 15.7mA, I_{sleep} = 0.65µA. So the average current can be calculated as:

$$I_{average} = \frac{2.28125us \times 15700uA + \left(3 \times 10^6 us - 2.28125us\right) \times 0.65uA}{3 \times 10^6 us} \approx 0.66uA$$

5.3 Doxygen

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.

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

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Milennium 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
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chou-ku, Tokyo 104-0033
JAPAN
Tel: (+81) 3523-3551
Fax: (+81) 3523-7581

8307B-AVR-02/11