The following document contains information on Cypress products. Although the document is marked with the name "Spansion" and "Fujitsu", the company that originally developed the specification, Cypress will continue to offer these products to new and existing customers.

Continuity of Specifications
There is no change to this document as a result of offering the device as a Cypress product. Any changes that have been made are the result of normal document improvements and are noted in the document history page, where supported. Future revisions will occur when appropriate, and changes will be noted in a document history page.

Continuity of Ordering Part Numbers
Cypress continues to support existing part numbers. To order these products, please use only the Ordering Part Numbers listed in this document.

For More Information
Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress
Cypress (NASDAQ: CY) delivers high-performance, high-quality solutions at the heart of today's most advanced embedded systems, from automotive, industrial and networking platforms to highly interactive consumer and mobile devices. With a broad, differentiated product portfolio that includes NOR flash memories, F-RAM™ and SRAM, Traveo™ microcontrollers, the industry's only PSoC® programmable system-on-chip solutions, analog and PMIC Power Management ICs, CapSense® capacitive touch-sensing controllers, and Wireless BLE Bluetooth® Low-Energy and USB connectivity solutions, Cypress is committed to providing its customers worldwide with consistent innovation, best-in-class support and exceptional system value.

Colophon

The products described in this document are designed, developed and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for any use that includes fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for any use where chance of failure is intolerable (i.e., submersible repeater and artificial satellite). Please note that Spansion will not be liable to you and/or any third party for any claims or damages arising in connection with above-mentioned uses of the products. Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions. If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Law of Japan, the US Export Administration Regulations or the applicable laws of any other country, the prior authorization by the respective government entity will be required for export of those products.

Trademarks and Notice

The contents of this document are subject to change without notice. This document may contain information on a Spansion product under development by Spansion. Spansion reserves the right to change or discontinue work on any product without notice. The information in this document is provided as is without warranty or guarantee of any kind as to its accuracy, completeness, operability, fitness for particular purpose, merchantability, non-infringement of third-party rights, or any other warranty, express, implied, or statutory. Spansion assumes no liability for any damages of any kind arising out of the use of the information in this document.

Fujitsu Microelectronics Europe
Application Note

MCU-AN-391024-E-V11

FR FAMILY

32-BIT MICROCONTROLLER

MB91F369, MB91F364

UPDATE RTC

APPLICATION NOTE

Revision History

Date	Issue
2003-06-23	V 1.0, first draft, DFi
2003-06-27	V1.1, added description of RTC start/stop behaviour, introduced PRESCALER_FAST macro, added runtime table and timing diagram, DFi

This document contains 10 pages.

Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH restricts its warranties and its liability for **all products delivered free of charge** (eg. software include or header files, application examples, target boards, evaluation boards, engineering samples of IC's etc.), its performance and any consequential damages, on the use of the Product in accordance with (i) the terms of the License Agreement and the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials. In addition, to the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH disclaims all warranties and liabilities for the performance of the Product and any consequential damages in cases of unauthorised decompiling and/or reverse engineering and/or disassembling. **Note, all these products are intended and must only be used in an evaluation laboratory environment**.

1. Fujitsu Microelectronics Europe GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days form the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Microelectronics Europe GmbH warrants that the Product will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.

2. Should a Product turn out to be defect, Fujitsu Microelectronics Europe GmbH´s entire liability and the customer´s exclusive remedy shall be, at Fujitsu Microelectronics Europe GmbH´s sole discretion, either return of the purchase price and the license fee, or replacement of the Product or parts thereof, if the Product is returned to Fujitsu Microelectronics Europe GmbH in original packing and without further defects resulting from the customer´s use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to Fujitsu Microelectronics Europe GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Microelectronics Europe GmbH.

3. To the maximum extent permitted by applicable law Fujitsu Microelectronics Europe GmbH disclaims all other warranties, whether expressed or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the Product is not designated.

4. To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH´s and its suppliers´ liability is restricted to intention and gross negligence.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES

To the maximum extent permitted by applicable law, in no event shall Fujitsu Microelectronics Europe GmbH and its suppliers be liable for any damages whatsoever (including but without limitation, consequential and/or indirect damages for personal injury, assets of substantial value, loss of profits, interruption of business operation, loss of information, or any other monetary or pecuniary loss) arising from the use of the Product.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect

Contents

1 Introduction

The hour, minute and second values of the RTC module in the MB91360 series can be set/updated immediately by usage of the UPDT bit in the WTCR control register. However there are two restrictions with this approach:

1. For MB91360 series devices other than MB91F364G and MB91F369GA, the RBUS clock must be higher than the oscillation clock when using the UPDT bit (e.g. RBUS clock=2MHz, oscillation clock=4MHz is not allowed).
2. The time from updating the RTC value until the new value can be read out from the WTSR, WTMR, WTHR registers, is up to one second (until the next subsecond overflow) in MB91F364G and MB91F369GA.

For these cases, a workaround is described in this document in order to update the RTC quickly. In the workaround, only the WTCR.ST and WTCR.RUN bits are used to update the hour, minute and second registers. With the described routine, the RTC update takes approx. 15 µs. For detailed runtimes, refer to Table 1.

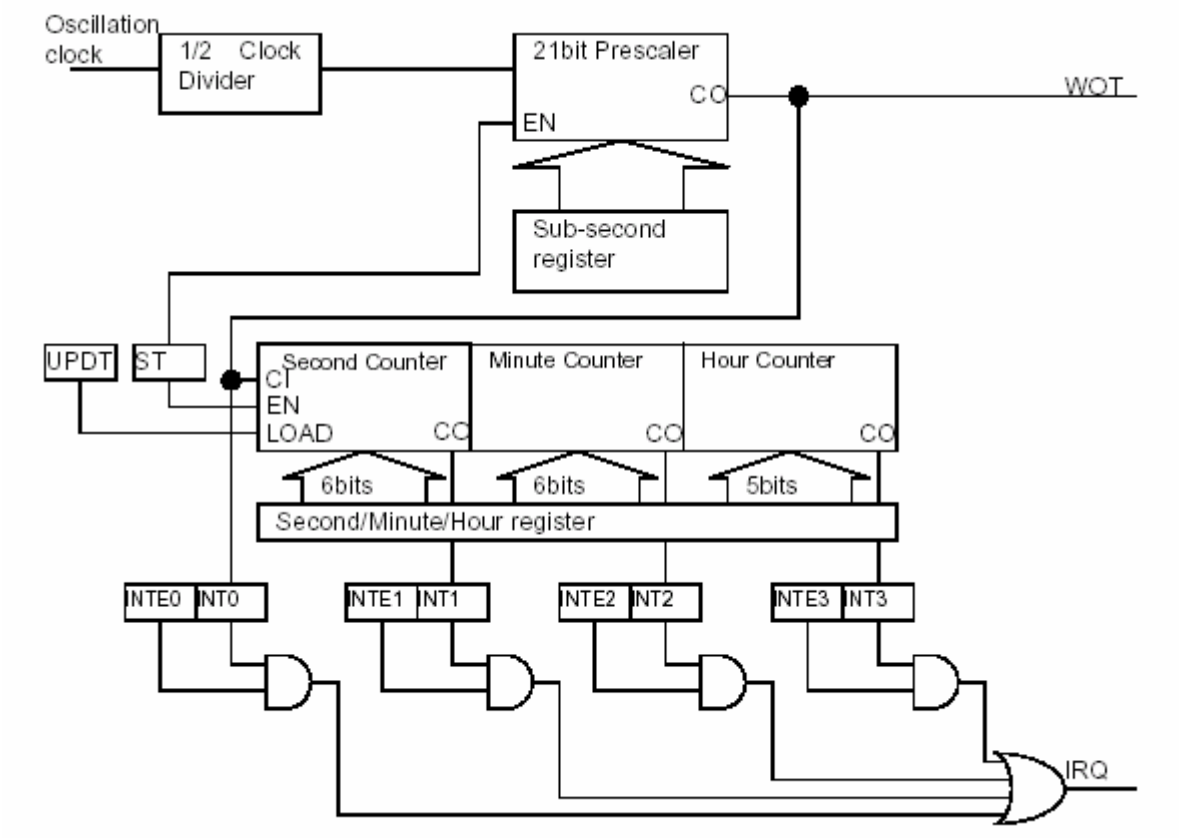
2 RTC update mechanism

This section describes the behaviour of the RTC during start/stop.

When starting the RTC by writing 1 to the WTCR_ST bit, the values, which were *written* into the hour, minute and second registers WTHR, WTMR and WTSR are transferred to the counter. Only after the next subsecond overflow (i.e. after one second), these values + 1 second can be read out by *read accessing* WTHR, WTMR and WTSR. However the WTHR, WTMR and WTSR input latches are not updated. I.e. if the RTC is stopped and restarted, the RTC is set to the hour, minute and second which was originally *written* into the WTHR, WTMR and WTSR register.

I.e. if 07:12:40 is written into the WTHR, WTMR and WTSR register and the RTC is started by writing 1 to the WTCR.ST bit, 07:12:41 can be read from WTHR, WTMR, WTSR *after* the next subsecond overflow. If the RTC is stopped after some time, e.g. at 07:13:58, by writing 0 into WTCR.ST, and restarted hereafter, the next value which can be read from WTHR, WTMR and WTSR (one subsecond overflow after WTCR.ST was set to 1) is 07:12:41.

2.1 RTC block diagram



3 Algorithm

This section describes the quick RTC update algorithm for MB91F364G and MB91F369GA. Although it can be used for all devices of the MB91360 series, the usage of the UPDT bit is more convenient, if certain requirements are fulfilled (please refer to section 1).

1) Start of set_rtc_routine (approx. time t1 in Figure 1)
2) Save prescaler register WTBR
3) Save control register WTCR
4) Stop RTC, disable interrupt (WTCR=0) (approx. time t1 in Figure 1)
5) Wait until RTC has stopped (wait until WTCR.RUN==0)
6) Write new value into hour, minute and second register (WTHR, WTMR and WTSR). Second value must be decremented by one, because the RTC runs until subsecond overflow occurs, thus incrementing the second register. Since the prescaler is set to PRESCALER_FAST (small prescaler value e.g 2), the susbescond overflow will occur some µs later and not one second later.
7) Set prescaler register WTBR to PRESCALER_FAST (small prescaler value e.g 2) in order to generate a subsecond overflow as fast as possible to update the WTHR, WTMR and WTSR output latch (read access value). Note that the prescaler must not be smaller than stated in Table 1, otherwise more than one subsecond overflow could occur before the RTC can be stopped by the software (step 11).
8) Disable interrupts in order to be able to stop RTC directly after subsecond overflow.
9) Start RTC (WTCR.ST=1) (approx. time t2 in Figure 1)
10) Wait for subsecond overflow (wait until WTCR.INT0==1) (overflow appearance approx. at time t3 in Figure 1)
11) Stop RTC (WTCR=0) (approx. time t3 in Figure 1)
12) Enable interrupts.
13) Wait until RTC has stopped (wait until WTCR.RUN==0)
14) Write new value into hour, minute and second register (WTHR, WTMR and WTSR) Second value must *not* be decremented by one. Since the prescaler is set to its original value, the second register is incremented automatically *after* one second.
15) Restore prescaler register WTBR (correct prescaler value e.g `0x1E847F` for 4MHz oscillation clock)
16) Restore control register WTCR (restart RTC) (approx. time t4 in Figure 1)
17) clear overflow flag WTCR.INT0

An implementation of this algorithm can be found in section 4. The runtime (time t4-t1 in Figure 1) of the `set_rtc_clock` function is listed in Table 1.

core clock CLKB [MHz]	resource clock CLKR [MHz]	oscillation clock [MHz]	PRESCALER_FAST	runtime `set_rtc_clock` [µs]
64	32	4	3	11 µs
64	16	4	3	12 µs
48	24	4	3	13,7 µs
48	16	4	3	14,2 µs
16	16	4	8	34,8 µs
2	2	4	50	269 µs

Table 1

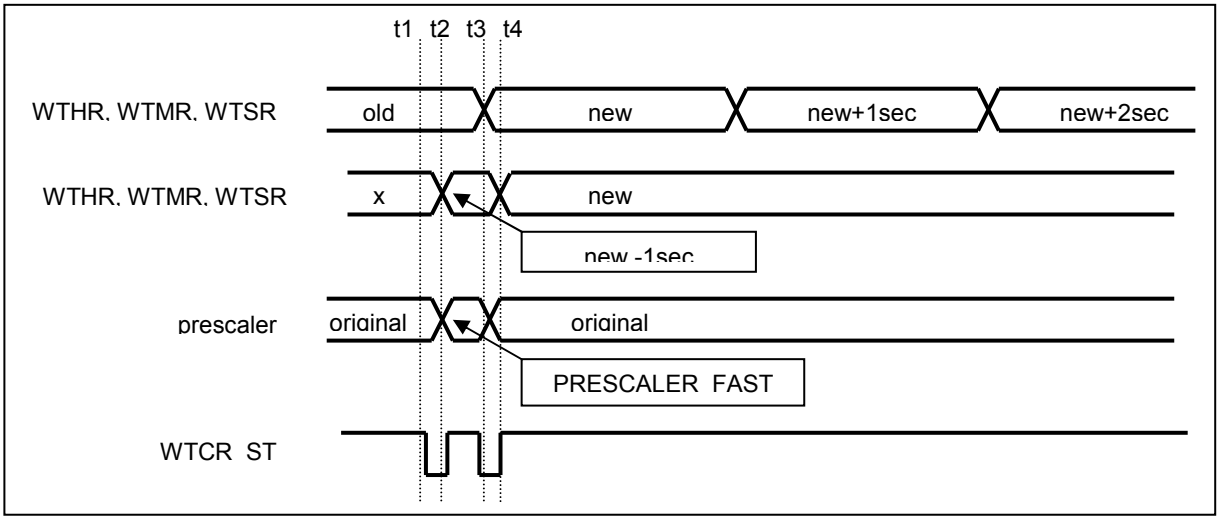


Figure 1 Note: The timescale of this figure is not exact.

4 Routines

```
/*****************************************************************************
 *     set rtc clock
 *     function :    set hour, minutes and second to RTC
 *                   For fast setting of the rtc the
 *                   prescaler is set to a minimum value to speed up
 *                   the clocking of the new values to RTC registers.
 *     par:
 *            in:    char h : hours for rtc
 *                   char m : minutes for rtc
 *                   char s : seconds for rtc
 *            out:   -
 *****************************************************************************/

#define PRESCALER_FAST 50

void set_rtc_clock (unsigned char h, unsigned char m, unsigned char s)
{

    WORD wtcr_original;
    DWORD prescaler_original;
    unsigned char hour, min, sec;

    sec=s;
    min=m;
    hour=h;

    prescaler_original = get_rtc_prescaler();
    wtcr_original = WTCR;            // save value before changing
    WTCR = 0x0000;                   // stop and disable interrupt reset int flag
    while (WTCR_RUN == 1);           // wait until rtc has stopped

    if (s!=0) s--;                   // subtract one second
                                     // to have the right time after clock
    else
    {
        s = 59;
        if (m!=0) m--;
        else
        {
             m = 59;
             if (h!=0) h--;
             else h=23;
        }
    }
    WTSR  = s;                       // set second (decremented by one)
    WTMR  = m;                       // set minute
    WTHR  = h;                       // set hour
    set_rtc_prescaler (PRESCALER_FAST);        // set fast clock
    __DI();                          // disable interrupts
    WTCR_ST = 1;                     // start rtc
    while (WTCR_INT0 == 0);          // wait for sec overflow
    WTCR_ST = 0;                     // stop rtc
    __EI();                          // enable interrupts
    while (WTCR_RUN == 1);           // wait until rtc has stopped

    WTSR  = sec;                     // set second
    WTMR  = min;                     // set minute
    WTHR  = hour;                    // set hour
    set_rtc_prescaler (prescaler_original);    // reload original settings
    WTCR = wtcr_original;                      // reload original settings
    WTCR_INT0 = 0;                   // clear overflow flag
    return;
}
```

```
/*****************************************************************************
 * set rtc prescaler
 * function : set the prescaler value to rtc
 *     par:
 *           in:          -
 *           out:   DWORD prescaler value
 *****************************************************************************/
void set_rtc_prescaler (DWORD x)
{
       WTBR_HIGH = (BYTE)(x / 0x10000) & 0xFF;
       WTBR_MID  = (BYTE)(x / 0x100) & 0xFF;
       WTBR_LOW  = (BYTE)(x / 0x1) & 0xFF;

}



/*****************************************************************************
 * get rtc prescaler
 * function : read out the prescaler value of rtc
 *     par:
 *           in:          -
 *           out:   DWORD prescaler value
 *****************************************************************************/
DWORD get_rtc_prescaler (void)
{
       return ((WTBR_HIGH&0x1F) *0x10000 + WTBR_MID * 0x100 + WTBR_LOW);
}
```

