The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

**How to Check the Ordering Part Number**

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

**For More Information**

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

**About Cypress**

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.

AN205371

# FR Family, MB91460 Microcontroller Real-Time Clock

This application note describes the functionality of the Real-Time Clock and gives some examples.

## Contents



# 1 Introduction

This application note describes the functionality of the Real-Time Clock and gives some examples.

## 1.1 Key Features

- Clock selectable as Main Clock, Sub Clock and RC Clock – 100KHz
- RTC is not deactivated during Stop Mode
- Interrupts selectable for: ½ second, 1 second, 1 minute, 1 hour, and 1 day

# 2 The Real-Time Clock

The Basic Functionality of the Real-Time Clock

## 2.1 Block Diagrams

Figure 1 shows the internal block diagram of the Real-Time Clock.

Figure 1. Real Time Clock block diagram

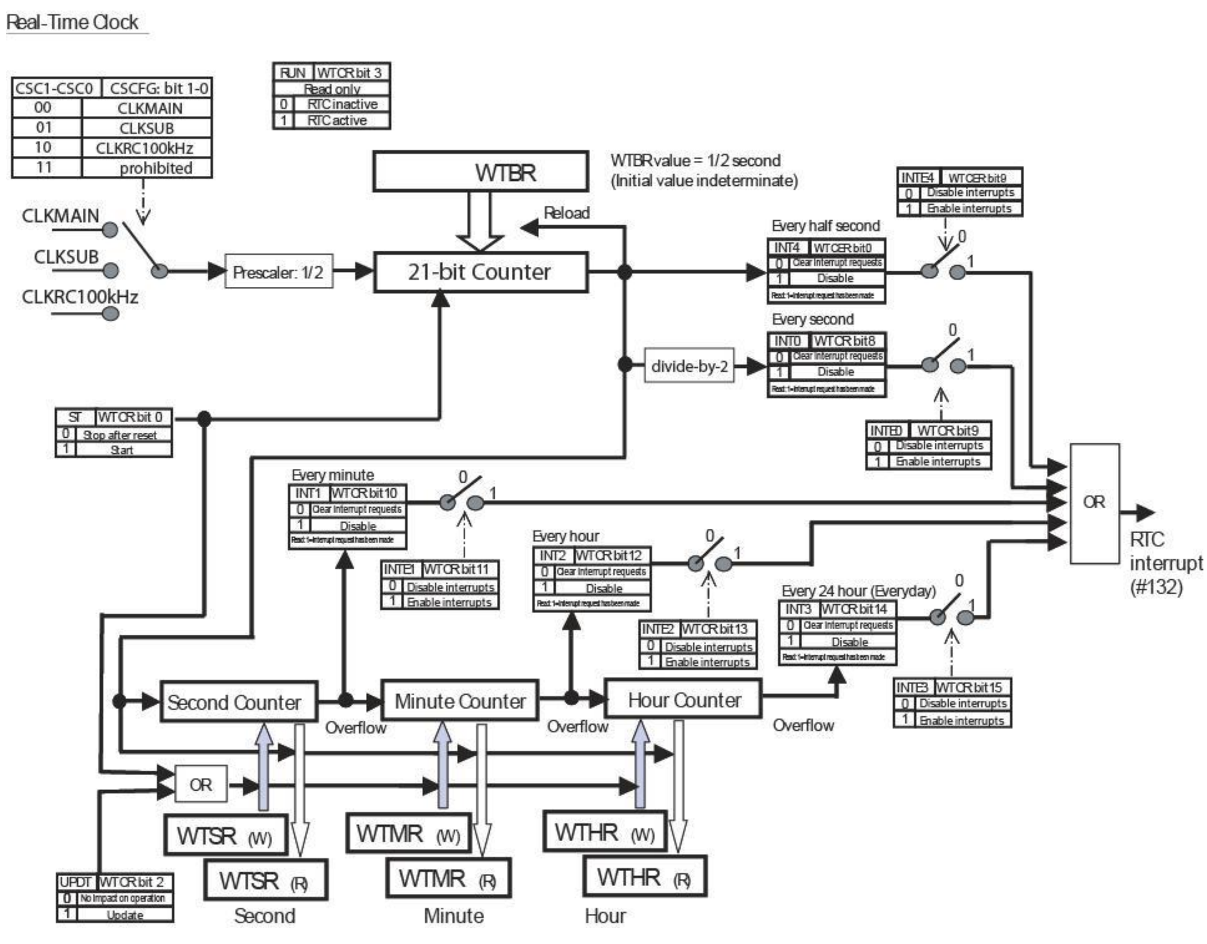
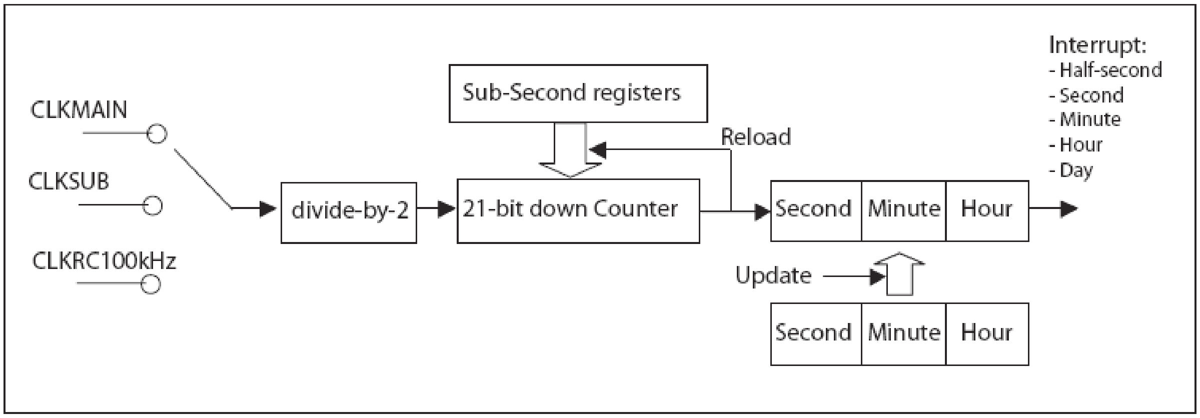


Figure 2 shows the simplified block diagram of the Real Time Clock.

Figure 2. Simplified Real Time Clock block diagram



## 2.2 Registers

Please write always “0” to the “Reserved” bits when accessing a register, if not stated otherwise.

### 2.2.1 Timer Control Register (WTCR)

Table 1. WTCR

| Bit No. | Name | Explanation | Value | Operation |
|---|---|---|---|---|
| 15 | INTE3 | Interrupt Request at 1-Day (24 Hour) | 0 | Interrupt disabled |
| | | | 1 | Interrupt, if 24 Hour (1-Day)Counter overflow |
| 14 | INT3 | 1-Day Interrupt Flag/Clear | 0 | Write: Clear Request |
| | | | 1 | Interrupt Request |
| 13 | INTE2 | Interrupt Request at 1-Hour | 0 | Interrupt disabled |
| | | | 1 | Interrupt, if 1 Hour Counter overflow |
| 12 | INT2 | 1-Hour Interrupt Flag/Clear | 0 | Write: Clear Request |
| | | | 1 | Interrupt Request |
| 11 | INTE1 | Interrupt Request at 1-Minute | 0 | Interrupt disabled |
| | | | 1 | Interrupt, if Minute Counter overflow |
| 10 | INT1 | 1-Minute Interrupt Flag/Clear | 0 | Write: Clear Request |
| | | | 1 | Interrupt Request |
| 9 | INTE0 | Interrupt Request at 1-Second | 0 | Interrupt disabled |
| | | | 1 | Interrupt, if Second Counter overflow |
| 8 | INT0 | 1-Second Interrupt Flag/Clear | 0 | Write: Clear Request |
| | | | 1 | Interrupt Request |
| 7-5 | – | *Reserved* | - | - |
| 4 | – | *Undefined* | - | - |
| 3 | RUN | Operation Status | 0 | RTC inactive |
| | | | 1 | RTC is active |
| 2 | UPDT | Update Counter with written Values into WTSR, WTHR, and WTMR | 0 | Write: No effect |
| | | | 1 | Write: Update |
| 1 | – | *Undefined* | - | - |
| 0 | ST | Start[1] | 0 | Clear Clock and Stop it |
| | | | 1 | Load Values and Start clock |

[1] It is recommended to set the ST-Bit to “0”, when changing the clock time of the RTC.

### 2.2.2 Timer Control Extended Register (WTCER)

Table 2. WTCER

| Bit No. | Name | Explanation | Value | Operation |
|---|---|---|---|---|
| 7 | – | Undefined | - | - |
| 6 | – | Undefined | - | - |
| 5 | – | Undefined | - | - |
| 4 | – | Undefined | - | - |
| 3 | – | Undefined | - | - |
| 2 | – | Undefined | - | - |
| 1 | INTE4 | Enable Interrupt Request at Half-Second (500 ms) | 0 | Interrupt disabled |
| | | | 1 | Interrupt, if Half-Second Counter overflow |
| 0 | INT4 | Half-Second Interrupt Flag/Clear | 0 | Write: Clear Request |
| | | | 1 | Interrupt Request |

### 2.2.3 Sub-Second Register (WTBR)

This register contains the 21-Bit reload value, which divides the clock source. The value to be programmed in WTBR register should be actual source clock-frequency (one among CLKMAIN, CLKSUB or CLKRC100KHz) divide by 4 in order to get half a second interrupt (0.5 s). This is also because there is a default prescaler of divide by 2 (please refer Figure 2).

The lower 8 Bits of this value is stored in WTBR2, the middle 8 Bits of this value is stored in WTBR1 and the remaining upper 5 Bits of the value in the lower 5 Bits of WTBR0.

The following table shows example values for different clock sources.

Table 3. Clock Source and WTBR values

| Clock Source | WTBR decimal | WTBR hexadecimal |
|---|---|---|
| Main Clock 4 MHz | 1000000 | 0x0F4240 |
| RC Clock 100 kHz | 25000 | 0x0061A8 |
| Sub Clock 32768 Hz | 8192 | 0x002000 |

Sub-Second Registers can be accessed via byte/half-word/word access.

### 2.2.4 Second/Minute/Hour Registers (WTSR, WTMR, WTHR)

The lower 6 Bits of the WTSR contains the actual second counter value. Writing a value to it memorizes the value. The Second-Counter is updated with this value by writing “1” to WTCR_UPDT.

The lower 6 Bits of the WTMR contains the actual minute counter value. Writing and reading has the same behavior like for WTSR.

The lower 5 Bits of the WTHR contains the actual hour counter value. Writing and reading has the same behavior like for WTSR.

Please store only reasonable values to these registers. If values, that do not present a clock time, are used, the behavior of the RTC will be undefined.

These registers can be accessed via byte/half-word access.

### 2.2.5 Clock Source Selection

The RTC clock source can be selected using CSC[1:0] bits of Clock Source Configuration Register (CSCFG).

#### Clock Source Configuration Register (CSCFG)

Table 4. WTCKSR

| Bit No. | Name | Explanation | Value | Operation |
|---|---|---|---|---|
| 1, 0 | CKC1,0 | RTC Clock Select Selection | 0, 0 | Main Clock (CLKMAIN) |
| | | | 0, 1 | Sub Clock (CLKSUB) |
| | | | 1, 0 | RC Clock (CLKRC100kHz) |
| | | | 1, 1 | *prohibited* |

The other bits of CSCFG register are not discussed here.

# 3 Real Time Clock Examples

Examples for the Real Time Clock

## 3.1 RTC with main clock source and without interrupts

The above example demonstrates how to initialize RTC immediately after a Reset. Here it is considered that the Main Clock is 4 MHz.

```
#define DividerMC 1000000

void InitRTCAfterReset (void)
{
    WTBR = DividerMC;       // Set Sub-Second Prescaler

    WTCR_INTE0 = 0;         // No Interrupts
    WTCR_INTE1 = 0;
    WTCR_INTE2 = 0;
    WTCR_INTE3 = 0;
    WTCER_INTE4 = 0;
    CSCFG_CSC = 0;          // Main Clock Source

    WTSR = 56;              // Seconds: 56
    WTMR = 34;              // Minutes: 34
    WTHR = 12;              // Hours:   12

    WTCR_ST = 1;            // ... and go!
       }

void main(void)
{
    . . .

    InitRTCAfterReset ();  // Init and start the RTC

    . . .
       }
```

```
#define DividerSC 8192

void InitRTC (void)
{
    WTCR_ST = 0;            // Stop the RTC
    while (WTCR_RUN != 0); // Wait till the RTC stops

    WTBR2 = (0xFFFF & DividerSC);  // Set Sub-Second Prescaler
    WTBR1 = (DividerSC >> 16);

    WTSR = 28;              // Seconds: 28
    WTMR = 59;              // Minutes: 59
    WTHR = 18;              // Hours:   18

    CSCFG_CKC = 1;                 // Sub Clock Source
    WTCR_ST = 1;                   // ... and go!
       }
```

## 3.2 Re-Initialize the RTC with Sub clock

The above examples demonstrates how to initialize the Sub-Second, Second, Minute & Hour registers if the RTC is already running. Here it is considered that the Sub Clock is 32.768 kHz.

### 3.3 Initialize the Sub-Second Register in ISR

```
#define DividerRC 25000

       __interrupt void RealTimeClock (void)
       {
    WTCR_INT0 = 0;                  // Clear the interrupt flag
    WTBR2 = (0xFFFF & DividerRC);   // Set Sub-Second Prescaler
    WTBR1 = (DividerRC >> 16);

       }
```

The above examples demonstrates how to initialize the Sub_Second register within the RTC interrupt service routine (if RTC is already running). Here the RTC does not need to be stopped since there is enough time to securely modify the registers until the next reload operation (next second interrupt). It should be noted that the RTC Second interrupt needs to be enabled.

```
void InitIrqLevels(void)
{
      . . .

      ICR58 = 30;    /* Real Time Clock                */
                     /* Calibration Unit               */
      . . .
}

__interrupt void RealTimeClock (void);  // Prototype

   . . .

#pragma intvect RealTimeClock 132   /* Real Time Clock              */
```

Please note, that the corresponding interrupt vector and level has to be defined in the *vectors.c* module of our standard template project.

```
extern unsigned char second, minute, hour;

        __interrupt void RealTimeClock (void)
        {
                WTCR_INT0 = 0;                          // Clear the interrupt flag
                second = WTSR;
                minute = WTMR;
                hour = WTHR;

        }
```

## 3.4 Read the Time inside ISR

It is recommended that the interrupts (INT0-4) should be used to read the time information, as this would eliminate the possibility of reading incorrect values from time (hour/minute/second) registers (in case of carry while reading). In the above example the Time information is read in the RTC Second (INT0) interrupt service routine.

Please note that the corresponding interrupt vector and level has to be defined in the *vectors.c* module as shown in the above example.

### 3.5 Read the Time inside application

```
#define TRUE        1
#define FALSE       0

unsigned char second, minute, hour;
unsigned char second1, minute1, hour1;
unsigned int time, time1;

void ReadTime (void)
{
	unsigned char result = FALSE;

	/* Normally this loop would exit at 1st iteration (e.g. 02:59:59 ->
	 03:00:00). In some cases only this loop will have 2 iterations at the max
	(e.g. 02:59:59 -> 03:59:59 (1st Iteration), 03:00:00 -> 03:00:00 (2nd
	 Iteration)).*/
	while (result != TRUE)
	{
		/* First Set of Time */
		second = WTSR;
		minute = WTMR;
		hour = WTHR;

		/* Second Set of Time */
		second1 = WTSR;
		minute1 = WTMR;
		hour1 = WTHR;

		/* Calculating absolute seconds for first set of time */
		time = hour*3600 + minute*60 + second;

		/* Calculating absolute seconds for second set of time */
		time1 = hour1*3600 + minute1*60 + second1;

		/* If the difference in the first & second set is 0 or 1 then the
		second set contains the latest accurate time information */
		if (((time1 - time) == 1) || ((time1 - time) == 0)))
		{
			result = TRUE;
		}
		/* The following condition takes care of day change situation
		23:59:59 -> 00:00:00, and the second set contains the latest
		accurate time information */
		else if (time == 86399 && time1 == 0)
		{
			result = TRUE;      // 23:59:59 -> 00:00:00
		}
	}
}
```

The above example demonstrates how the time can be read inside an application (without using interrupts). This also takes care of reading the time (hour/minute/second) registers at the very timing of changing over the hour or minute boundary.

Here the time registers are read twice. Then it is converted into absolute seconds value and if difference between the old and the new value is 0/1 or the old value is 86399 & the new value is 0 (i.e. 23:59:59 -> 00:00:00), then the second set (second1/minute1/hour1) is considered to contain the correct time information.

# 4 Additional Information

Information about CYPRESS Microcontrollers can be found on the following Internet page:

http://www.cypress.com/cypress-microcontrollers

The software example related to this application note is:

*91460_rtc_init_read*

It can be found on the following Internet page:

http://www.cypress.com/16fx

# Document History

Document Title: AN205371 - FR Family, MB91460 Microcontroller Real-Time Clock

Document Number: 002-05371

| Revision | ECN | Orig. of Change | Submission Date | Description of Change |
|---|---|---|---|---|
| ** | – | NOFL | 06/05/2008 | First Version; MPi |
| *A | 5128745 | NOFL | 02/07/2016 | Converted Spansion Application Note “MCU-AN-300075-E-V10” to Cypress template. |
| *B | 5870296 | AESATMP9 | 09/01/2017 | Updated logo and copyright. |
| *C | 6060603 | NOFL | 02/06/2018 | Updated hyperlinks across the document.<br>Updated to new template.<br>Completing Sunset Review. |

## Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

## Products

| | |
|---|---|
| ARM® Cortex® Microcontrollers | cypress.com/arm |
| Automotive | cypress.com/automotive |
| Clocks & Buffers | cypress.com/clocks |
| Interface | cypress.com/interface |
| Internet of Things | cypress.com/iot |
| Memory | cypress.com/memory |
| Microcontrollers | cypress.com/mcu |
| PSoC | cypress.com/psoc |
| Power Management ICs | cypress.com/pmic |
| Touch Sensing | cypress.com/touch |
| USB Controllers | cypress.com/usb |
| Wireless Connectivity | cypress.com/wireless |

## PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

## Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

## Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709