Software Real-Time Clock

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

代码简述	本范例程序基于芯片内部 38.4 kHz低速RC振荡电路(LIRC)，以软件方式实现实时时钟(Real-time Clock, RTC)功能
BSP 版本	M031_Series_BSP_CMSIS_V3.01.000
开发平台	NuMaker-M031

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 功能介绍

1.1 简介

本文以NuMicro Cortex®-M0 M031芯片为例，透过芯片内部的低速RC振荡电路(LIRC)、定时器再搭配软件算法，即可实现具有计时与警报(Alarm)功能的软件实时时钟。除此之外，定时器工作在掉电模式下可以降低系统功耗，延长系统的使用时间。

本应用中，软件实时时钟提供了日期(年、月、日)与时间(时、分、秒)的信息，用户可以根据需求，自由选择24小时制与12小时制的输出，警报发生后，也能够于中断服务程序自定义处理程序，满足用户的应用需求。

1.2 实时时钟系统架构

实时时钟，需要具有计时、日期计算与记录秒数的功能。一般而言，硬件实时时钟具有下列基本单元：

1. 定时器单元 (Timer)：设定一段时间，达到设定时间时计数器累加，用于计算时间
2. 时间计算单元 (Calendar Calculate Unit)：随着计时计的累加，计算年月日时分秒的更新值
3. 时间信息储存单元 (Time Information Storage Unit)：储存年月日时分秒等信息

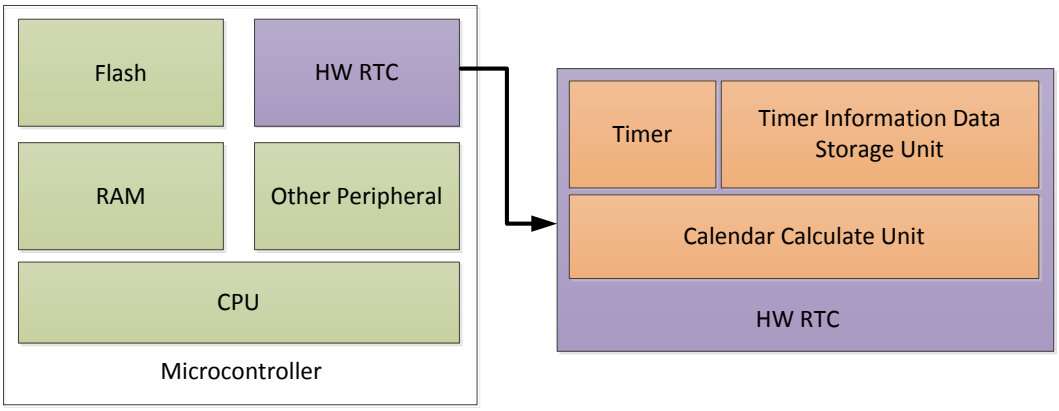


Figure 1-1 硬件实时时钟系统架构方块图

本文使用软件方式实现实时时钟，针对上述的硬件，本文采用下列方式实现：

1. 使用 M031 芯片内部的定时器实现硬件实时时钟中的定时器单元
2. 使用 M031 芯片的 CPU 透过时间更新算法实现时间计算单元
3. 使用 M031 芯片的 RAM 实现时间信息储存单元

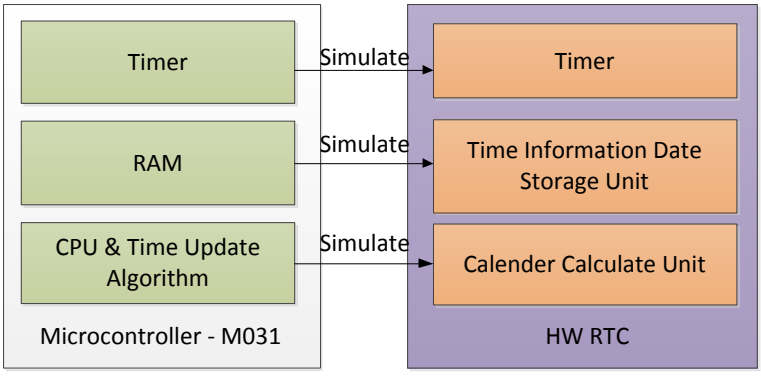


Figure 1-2 M031 软件实时时钟系统映像图

系统的架构如 Figure 1-3所示，低速内部RC振荡电路提供频率给定时器，定时器发送中断讯号给巢状向量中断控制器，巢状向量中断控制器会告知CPU中断事件产生，CPU再执行对应的中断子程序以更新时间信息。更新时间信息时，会先由随机存取内存中取出目前的时间信息，取出后计算新的时间，计算结束后存回随机存取内存，完成一次循环。

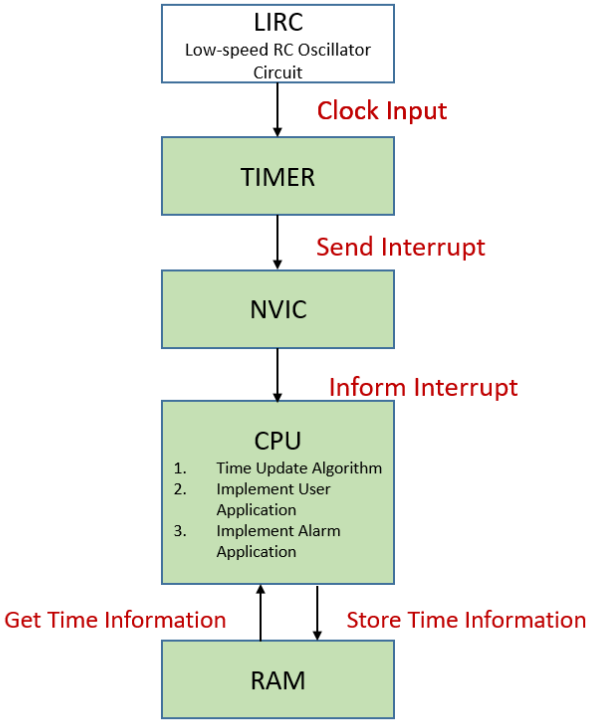


Figure 1-3 软件实时时钟架构图

1.3 实时时钟软件算法

1.3.1 时间更新算法

假设定时器唤醒芯片后距离上一次更新已经过1秒，在定时器的中断服务程序RTC_IRQ_Process()中，会呼叫_RTC_UpdateTime()进行时间信息的更新，以下为此函式的计算流程：

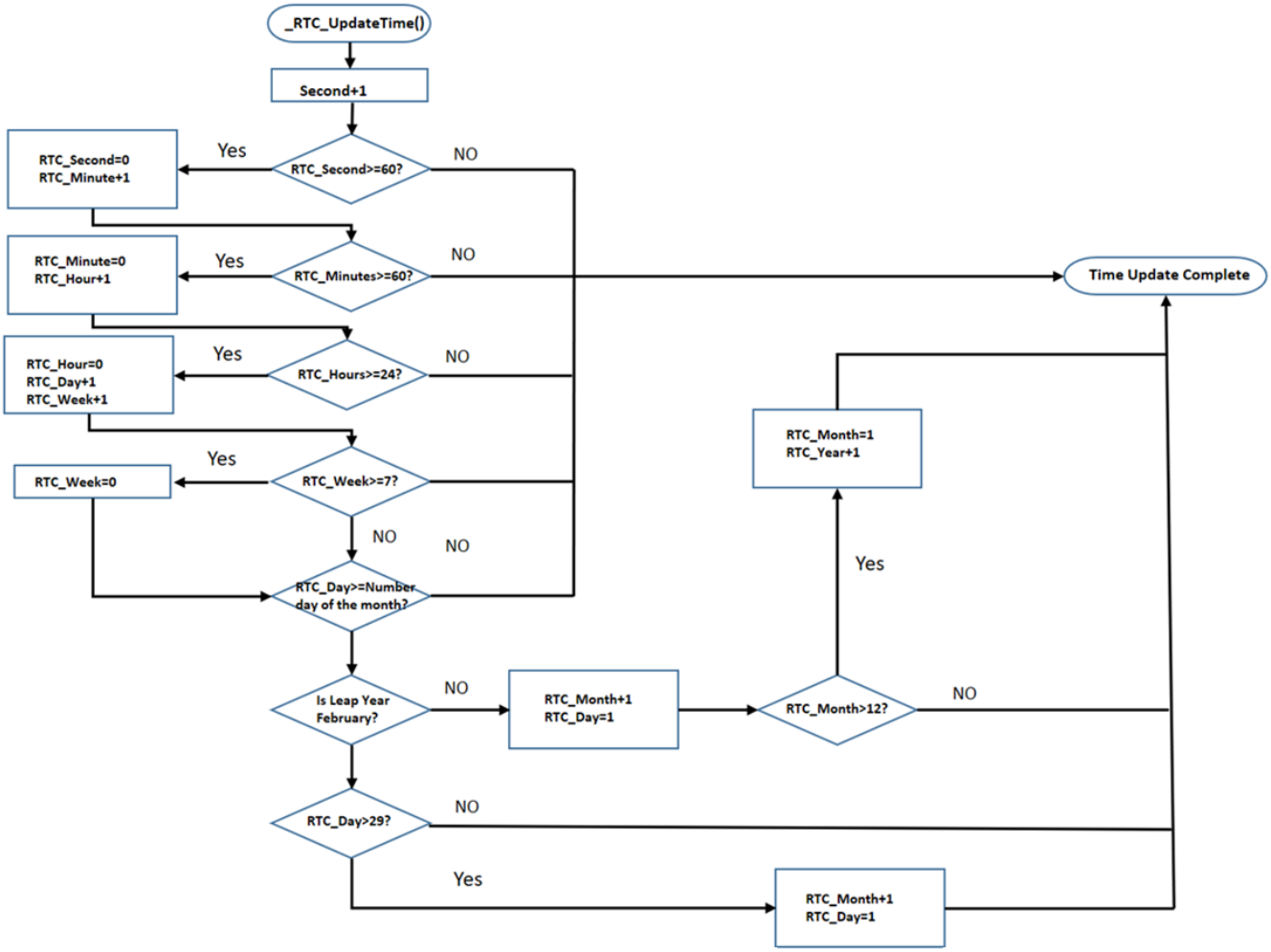


Figure 1-4 时间更新算法流程图

1.3.2 闰年判断算法

现今使用的Gregorianum历法与地球公转间存在着误差，此误差会透过闰年的方式做修正。每逢闰年，二月会比平年多一天，亦即二月会有29天。

Gregorianum历法对于闰年的定义如下：

1. 公元年分除以 4 不可整除，为平年
2. 公元年分除以 4 可整除但除以 100 不可整除，为闰年
3. 公元年分除以 100 可整除但除以 400 不可整除，为平年

上述条件可以用一句话来简化，「四年一闰、百年不闰、四百年再闰」。

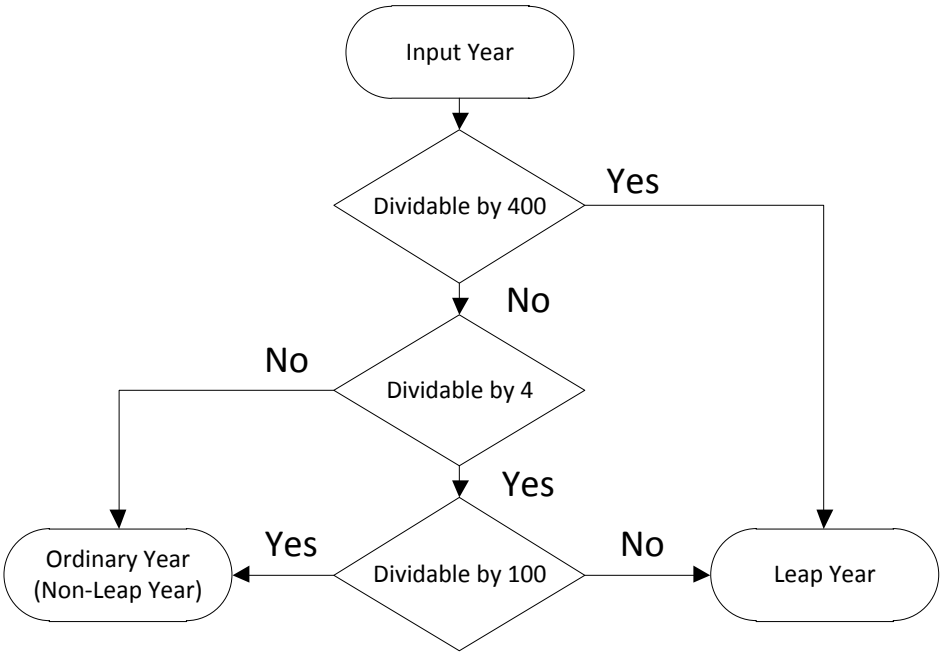


Figure 1-5 闰年判断流程图

1.3.3 星期计算算法

本文在星期计算的算法上，使用德国数学家Christian Zeller推算出来的Zeller公式，藉由此公式能够计算输入的日期是一周当中的星期几。

适用范围：1582年10月15日后的任何一天

注：罗马教宗额我略十三世颁布新历法(公历)，把1582年10月4日的后一天改为1582年10月15日，因此在这一天的前后会有日期上的断层

Zeller's congruence：

$$w = \left(y + \left[\frac{y}{4} \right] + \left[\frac{c}{4} \right] - 2c + \left[\frac{26(m+1)}{10} \right] + d - 1 \right) \bmod 7$$

w：星期，计算出的数值具有以下对应关系：0-星期日；1-星期一；2-星期二；3-星期三；4-星期四；5-星期五；6-星期六

c：年份前两位数，以2018年为例，c的值要取20

y：年份后两位数，以2018年为例，y的值要取18

m：月份，在Zeller公式中，1月与2月要视为上一年的13、14月来计算，以2018年1月1日为例，要看作2017年的13月1日来计算

d：日

[ ]：高斯符号，取不大于原数的最大整数

mod：同余符号，除以某数后取其余数，程序语言中使用%表示

由于Zeller公式的结果可能为负数，因此结果可以使用下列方式做补正，确保结果的范围限制在0-星期日 ~ 6-星期六：

$$\text{if } (w < 0) \ \ w \mathrel{+}= 7;$$

1.4 降低系统功耗

为了达到省电的目的，系统必须尽可能的待在掉电模式(Power-down Mode)，系统在掉电模式期间，用户应关闭所有不必要的设备以及外部的耗能组件。
在掉电模式下，38.4 kHz的内部低速RC振荡电路不会停止工作，因此采用此频率源输入的定时器会持续工作。定时器在送出中断讯号的同时会唤醒芯片，使芯片回到一般模式；在一般模式下，当实时时钟的时间更新完成后，芯片应尽快回到掉电模式，以减少系统的功耗。

1.5 实时时钟警报(Alarm)

用户需设定一个警报时间并开启警报功能，当实时时钟更新后的时间与警报设定的时间相等时，即可跳至对应的警报程序，用户可自行编辑警报子程序的程序满足应用需求。透过函式库中的指令RTC_SetRTCAlarmTime()与RTC_GetRTCInfo()，使用者可设定与读取警报的设定值。

1.6 软件实时时钟系统流程

透过软件实现实时时钟时，需要设定定时器频率、定时器中断、定时器唤醒功能等，另外为了降低系统功耗需要开启掉电模式。芯片在初始化之后，程序流程可以分为三个状态：芯片睡眠(掉电模式)、中断服务程序、主程序循环，以下针对这几个流程做简短说明。
初始化软硬件：

1. 初始化定时器，开启定时器中断与定时器唤醒功能
2. 初始化实时时钟时间、实时时钟警报时间
3. 选择是否开启实时时钟警报功能
4. 向实时时钟告知定时器一秒产生几次中断(中断频率)

进入程序主体后，会在芯片睡眠(掉电模式)、中断服务程序与主程序循环这三个状态重复循环：

1. 芯片睡眠(掉电模式)：芯片维持在低功耗状态等待中断事件发生
2. 中断服务程序：当定时器中断产生后，系统跳至定时器的中断服务程序，此时先将唤醒次数的缓存器数值加 1，再进行判断唤醒次数缓存器的值是否大于等于定时器一秒产生的中断次数。条件成立时代表时间经过一秒，此时会开始进行时间的更新，并在更新完成后将唤醒次数缓存器清除为 0；若是条件不成立，代表时间尚未累积到 1 秒，因此不会进行时间的更新。
   如果有进行时间更新，更新结束后会执行用户应用程序，此时若是有开启警报功能，会继续判断警报是否产生，并执行对应的警报应用程序，结束后离开中断服务程序，并回到主程序的循环中；如果没有进行时间更新，会直接离开中断服务程序回到主程序循环。
3. 主程序循环：用户的程序，执行结束后应进入芯片睡眠状态

定时器中断频率与唤醒次数的关系：中断频率的意思是定时器每一秒钟会产生几次中断讯号。中断频率2 Hz的意思是每秒唤醒芯片2次、发送2次中断讯号，每次的中断唤醒，都会累积唤醒次数1，因此当唤醒次数大于等于定时器的每秒唤醒次数时，代表时间已经经过了1秒，此时需要马上进行时间更新。

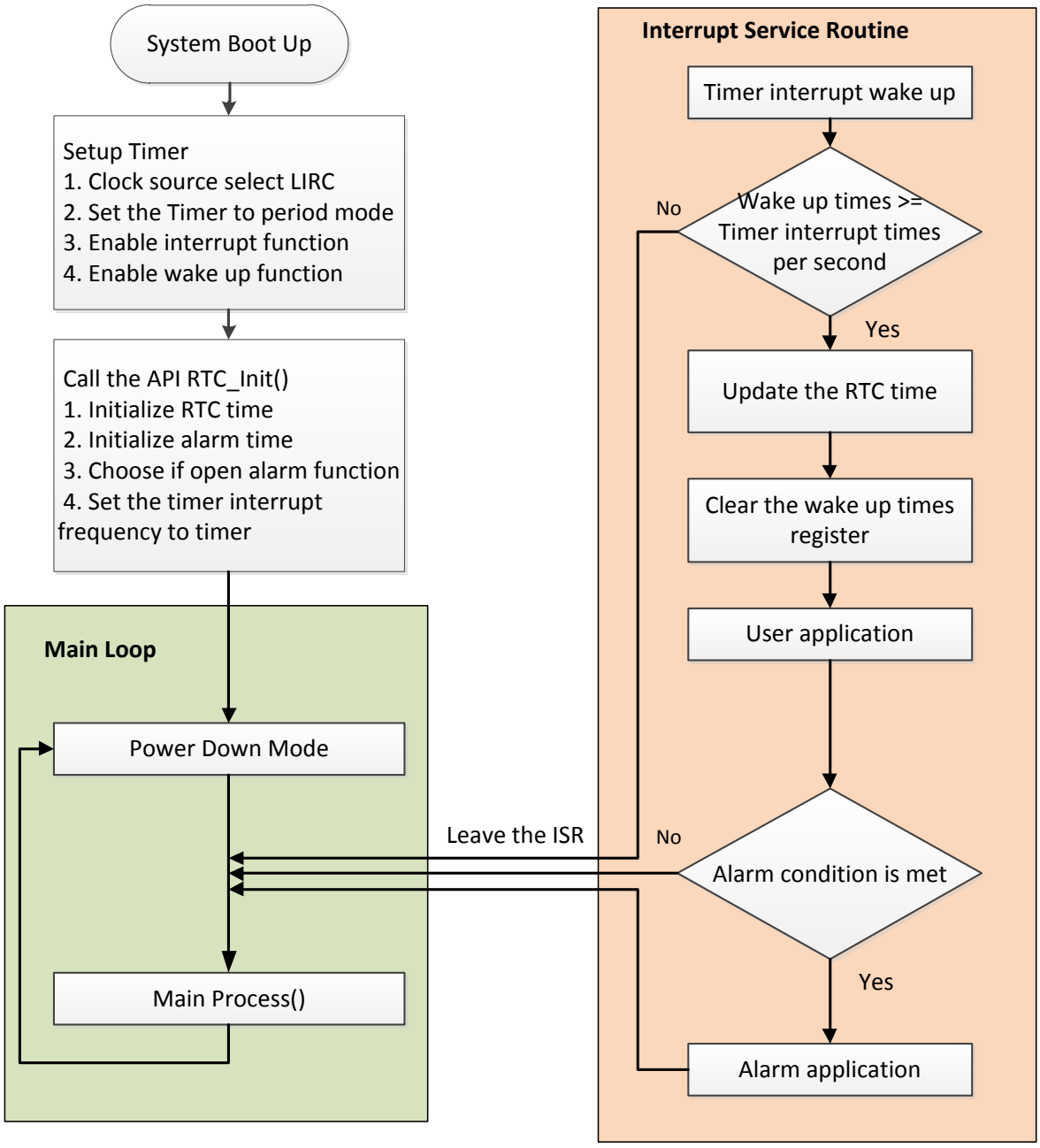
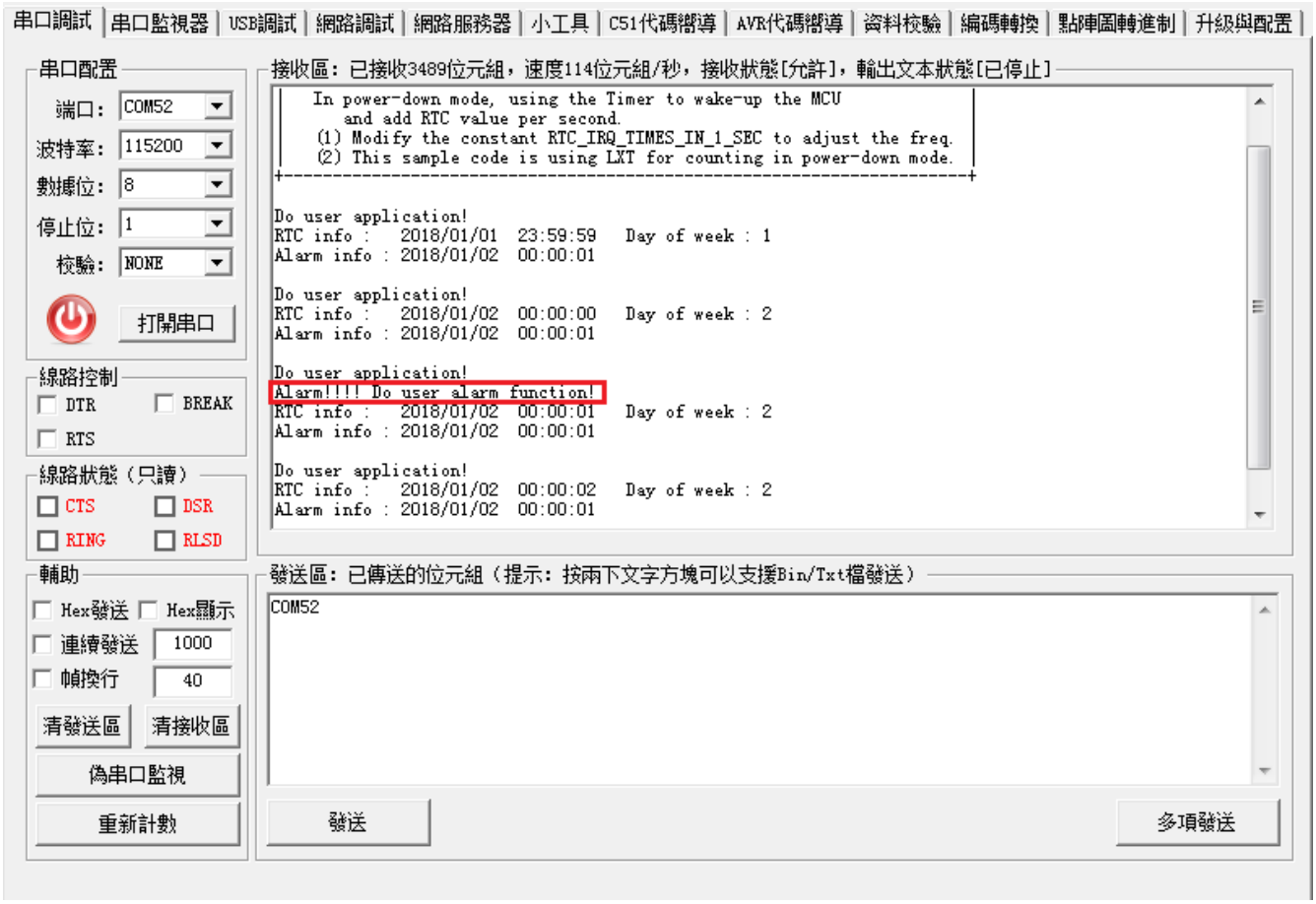


Figure 1-6 系统软件流程图

1.7 执行结果

实时时钟时间：2018/01/01 23:59:58
实时时钟警报时间：2018/01/02 00:00:01
当实时时钟的警报被触发时，会执行警报子程序，此外天数增加的同时星期也会随之改变。



2 代码介绍

2.1 main()

```
/*---------------------------------------------------------------------------*/
/*  Main Function                                                            */
/*---------------------------------------------------------------------------*/
int32_t main(void)
{
    /* Init System, IP clock and multi-function I/O. */
    SYS_Init();

    /* Init UART0 for printf */
    UART0_Init();

    /* Init SW RTC */
    SWRTC_Init();

    /* Init TIMER */
    TIMER_Init();

    /* Start Timer0 counting */
    TIMER_Start(TIMER0);

    printf("+-------------------------------------------------------------------+\n");
    printf("|   This sample code performs how to use software to simulate RTC.  |\n");
    printf("|   In power-down mode, using the Timer to wake-up the MCU          |\n");
    printf("|      and add RTC value per second.                                |\n");
    printf("|   (1) Modify the constant RTC_IRQ_TIMES_IN_1_SEC to adjust the freq. |\n");
    printf("|   (2) This sample code is using LIRC for counting in power-down mode. |\n");
    printf("+-------------------------------------------------------------------
+\n\n");

    /* Wait uart transfer message */
    while(!UART_IS_TX_EMPTY(UART0));

    while(1)
    {
        /* Unlock protected registers */
        SYS_UnlockReg();
        /* Make MCU power-down */
```

```
        CLK_PowerDown();
        /* lock protected registers */
        SYS_LockReg();

        #if 0 //Set new RTC Time
        TIMER_Stop(TIMER0);
        RTC_SetRTCTime(&g_sRTCInfo);
        TIMER_Open(TIMER0, TIMER_PERIODIC_MODE, RTC_IRQ_TIMES_IN_1_SEC);
        #endif

        /* Dump the RTC information and alarm information. */
        RTC_GetRTCInfo(&g_sRTCInfo);
        RTC_GetAlarmInfo(&g_sAlarmInfo);
        printf("RTC info :   %04d/%02d/%02d  %02d:%02d:%02d   Day of week : %d \n",
g_sRTCInfo.date.u32year, g_sRTCInfo.date.u8month, g_sRTCInfo.date.u8day,
g_sRTCInfo.time.u8hour, g_sRTCInfo.time.u8minutes, g_sRTCInfo.time.u8seconds,
g_sRTCInfo.date.u8dayofweek);
        printf("Alarm info : %04d/%02d/%02d  %02d:%02d:%02d \n\n",
g_sAlarmInfo.date.u32year, g_sAlarmInfo.date.u8month, g_sAlarmInfo.date.u8day,
g_sAlarmInfo.time.u8hour, g_sAlarmInfo.time.u8minutes, g_sAlarmInfo.time.u8seconds);

        /* Wait uart transfer message */
        while(!UART_IS_TX_EMPTY(UART0));
    }
}
```

2.2 SWRTC_Init()

```
void SWRTC_Init(void)
{
    /* Set RTC time and RTC alarm time. */
    g_sRTCInfo.date.u32year = 2018;
    g_sRTCInfo.date.u8month = 1;
    g_sRTCInfo.date.u8day = 1;
    g_sRTCInfo.time.u8hour = 23;
    g_sRTCInfo.time.u8minutes = 59;
    g_sRTCInfo.time.u8seconds = 58;

    g_sAlarmInfo.date.u32year = 2018;
    g_sAlarmInfo.date.u8month = 1;
    g_sAlarmInfo.date.u8day = 2;
```

```
    g_sAlarmInfo.time.u8hour = 0;
    g_sAlarmInfo.time.u8minutes = 0;
    g_sAlarmInfo.time.u8seconds = 1;

    /* Init SW RTC Setting */
    RTC_Init();
    /* Set RTC time */
    RTC_SetRTCTime(&sRTCInfo);
    /* Set Alarm time */
    RTC_SetAlarmTime(&sAlarmInfo);
    /* Enable Alarm function */
    RTC_SetAlarmFunc(eRTC_ALARM_ENABLE);
    /* Set interrupt frequency for calculating if it is passing 1 second and update RTC
time. */
    RTC_SetIRQFreq(WAKEUPFREQ);
}
```

2.3 Timer Initialize

```
void TIMER_Init(void)
{
    /* Initial Timer0 to periodic mode with WakeUpFreq Hz.
    User can modify the marco "WakeUpFreq" to change wake-up frequency */
    TIMER_Open(TIMER0, TIMER_PERIODIC_MODE, RTC_IRQ_TIMES_IN_1_SEC);

    /* Enable timer wake up system */
    TIMER_EnableWakeup(TIMER0);

    /* Enable Timer0 interrupt */
    TIMER_EnableInt(TIMER0);
    NVIC_EnableIRQ(TMR0_IRQn);
}
```

2.4 User Application

```
void User_Application(void)
{
    printf("Do user application!\n");
}
```

2.5 Alarm Application

```
void User_AlarmFunction(void)
{
    printf("Alarm!!!! Do user alarm function!\n");
}
```

2.6 Interrupt of Timer

```
void TMR0_IRQHandler(void)
{
    if(TIMER_GetIntFlag(TIMER0)==TRUE)
    {
        /* Clear wake up flag */
        TIMER_ClearWakeupFlag(TIMER0);
        /* Clear interrupt flag */
        TIMER_ClearIntFlag(TIMER0);
        /* Import callback function and run RTC process */
        RTC_IRQ_Process(User_Application, User_AlarmFunction);
    }
}
```

3 软件与硬件环境

- 软件环境
  - BSP 版本
    - M031_Series_BSP_CMSIS_V3.01.000
  - IDE 版本
    - Keil uVersion 5.26
- 硬件环境
  - 电路组件
    - NuMaker-M031
    - Nu-Bridge
  - 接线图

使用者可以将 M031 上的 UART_RX0(PB.12), UART_TX0(PB.13)接到 Nu-Bridge 上打印 UART 的除错讯息。使用者需要先打开 PC 上的设备管理器，查看“NuBridge Virtual Com Port (COMX)”，并打开终端机，将 Baudrate 设定为 115200 接收对应的 Com Port 讯息。

UART_RX0 (PB.12)	↔	PB5/ TX1 (Orange wire)
UART_TX0 (PB.13)	↔	PB4/RX1 (Brown wire)
GND	↔	GND (Black wire)
M031		Nu-Bridge

4 目录信息

- EC_M031_SW_RTC_V1.00
  - Library — Sample code header and source files
    - CMSIS — Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
    - Device — CMSIS compliant device header file
    - StdDriver — All peripheral driver header and source files
  - SampleCode
    - ExampleCode — Source file of example code

5 如何执行范例程序

1. 根据目录信息章节进入 ExampleCode/SWRTC_V1.00/KEIL 文件夹，双击 SW_RTC.uvproj 打开专案。
2. 进入编译模式接口
   a. 编译
   b. 下载代码至内存
   c. 进入 / 离开除错模式
3. 进入除错模式接口
   a. 执行代码

6 修订纪录

Date	Revision	Description
Oct. 8, 2019	1.00	1. 初版

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.