Software Real-Time Clock

NuMicro® 32 位系列微控制器範例代碼介紹

文件資訊

代碼簡述	本範例程式基於晶片內部 38.4 kHz低速RC振盪電路(LIRC)，以軟體方式實現實時時鐘(Real-time Clock, RTC)功能
BSP 版本	M031_Series_BSP_CMSIS_V3.01.000
開發平台	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 功能介紹

1.1 簡介

本文以NuMicro Cortex®-M0 M031晶片為例，透過晶片內部的低速RC振盪電路(LIRC)、計時器再搭配軟體演算法，即可實現具有計時與警報(Alarm)功能的軟體實時時鐘。除此之外，計時器工作在掉電模式下可以降低系統功耗，延長系統的使用時間。

本應用中，軟體實時時鐘提供了日期(年、月、日)與時間(時、分、秒)的資訊，使用者可以根據需求，自由選擇24小時制與12小時制的輸出，警報發生後，也能夠於中斷服務程序自定義處理程序，滿足使用者的應用需求。

1.2 實時時鐘系統架構

實時時鐘，需要具有計時、日期計算與記錄秒數的功能。一般而言，硬體實時時鐘具有下列基本單元：

1. 計時器單元 (Timer)：設定一段時間，達到設定時間時計數器累加，用於計算時間
2. 時間計算單元 (Calendar Calculate Unit)：隨著計時計的累加，計算年月日時分秒的更新值
3. 時間資訊儲存單元 (Time Information Storage Unit)：儲存年月日時分秒等資訊

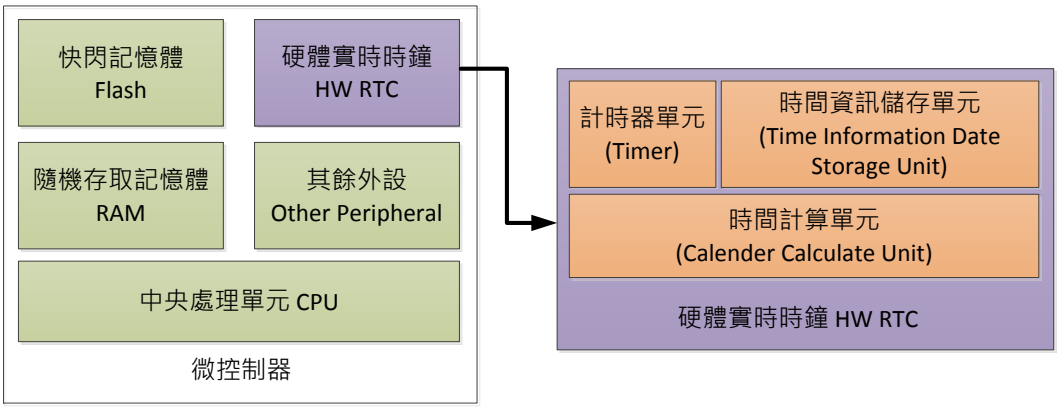


Figure 1-1 硬體實時時鐘系統架構方塊圖

本文使用軟體方式實現實時時鐘，針對上述的硬體，本文採用下列方式實現：

1. 使用 M031 晶片內部的計時器實現硬體實時時鐘中的計時器單元
2. 使用 M031 晶片的 CPU 透過時間更新演算法實現時間計算單元
3. 使用 M031 晶片的 RAM 實現時間資訊儲存單元

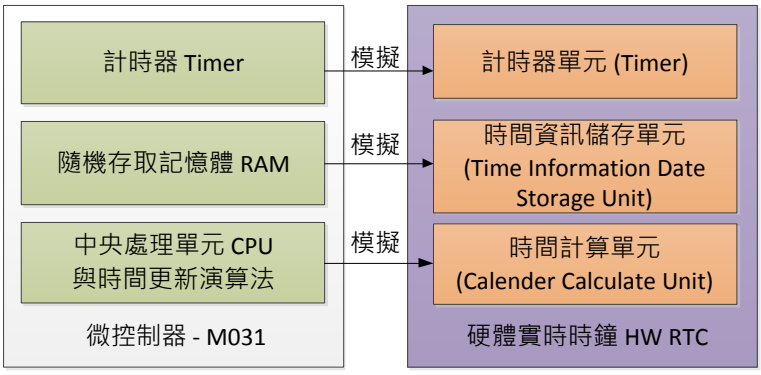


Figure 1-2 M031 軟體實時時鐘系統映射圖

系統的架構如 Figure 1-3軟體實時時鐘架構圖所示，低速內部RC振盪電路提供時脈給計時器，計時器發送中斷訊號給巢狀向量中斷控制器，巢狀向量中斷控制器會告知CPU中斷事件產生，CPU再執行對應的中斷副程式以更新時間資訊。更新時間資訊時，會先由隨機存取記憶體中取出目前的時間資訊，取出後計算新的時間，計算結束後存回隨機存取記憶體，完成一次循環。

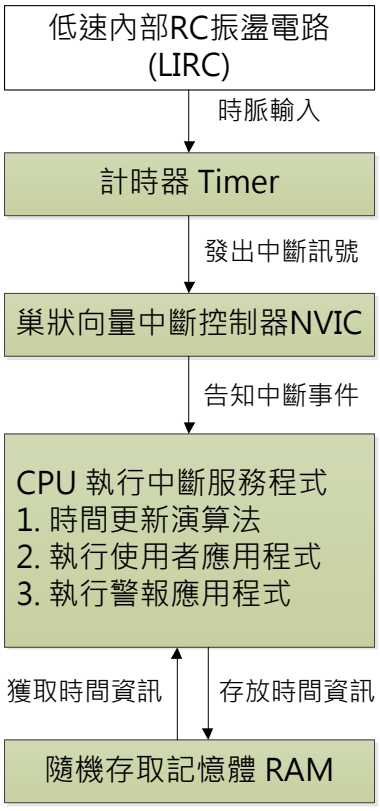


Figure 1-3 軟體實時時鐘架構圖

1.3 實時時鐘軟體演算法

1.3.1 時間更新演算法

假設計時器喚醒晶片後距離上一次更新已經過1秒，在計時器的中斷服務程序RTC_IRQ_Process()中，會呼叫_RTC_UpdateTime()進行時間資訊的更新，以下為此函式的計算流程：

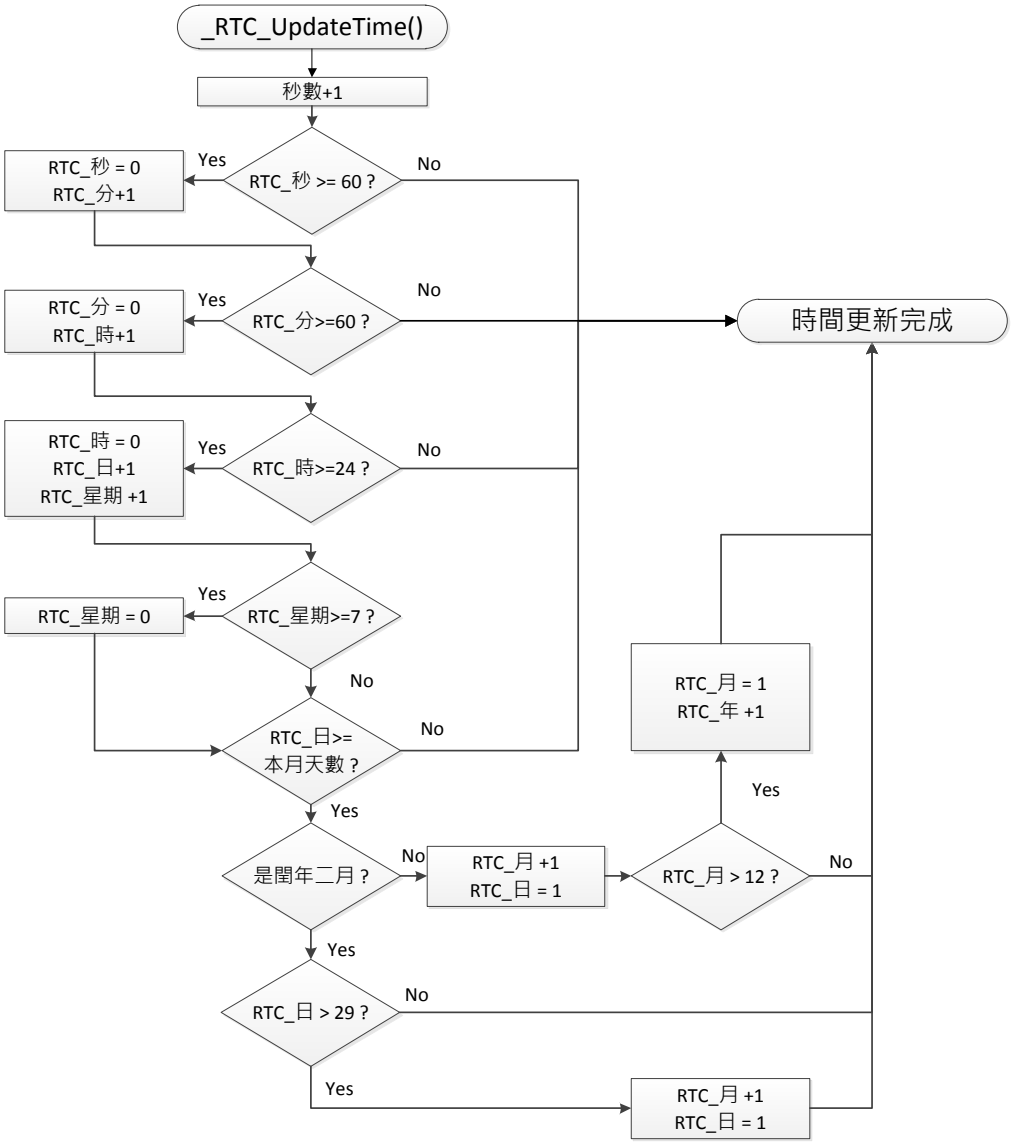


Figure 1-4 時間更新演算法流程圖

1.3.2 閏年判斷演算法

現今使用的Gregorianum曆法與地球公轉間存在著誤差，此誤差會透過閏年的方式做修正。每逢閏年，二月會比平年多一天，亦即二月會有29天。

Gregorianum曆法對於閏年的定義如下：

1. 公元年分除以 4 不可整除，為平年
2. 公元年分除以 4 可整除但除以 100 不可整除，為閏年
3. 公元年分除以 100 可整除但除以 400 不可整除，為平年

上述條件可以用一句話來簡化，「四年一閏、百年不閏、四百年再閏」。

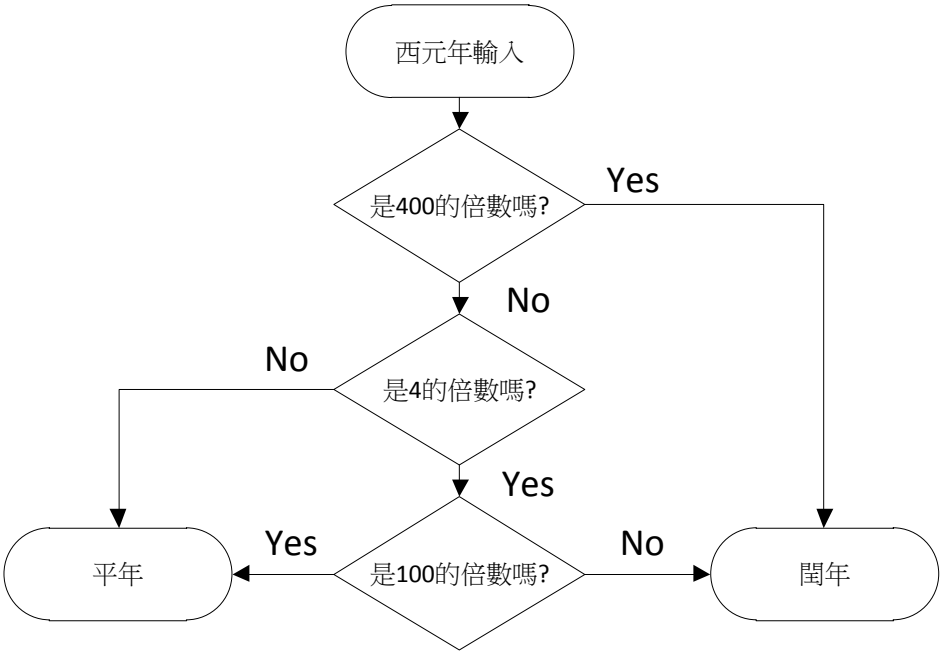


Figure 1-5 閏年判斷流程圖

1.3.3 星期計算演算法

本文在星期計算的演算法上，使用德國數學家Christian Zeller推算出來的Zeller公式，藉由此公式能夠計算輸入的日期是一週當中的星期幾。

適用範圍：1582年10月15日後的任何一天

注：羅馬教宗額我略十三世頒布新曆法(公曆)，把1582年10月4日的後一天改為1582年10月15日，因此在這一天的前後會有日期上的斷層

Zeller's congruence：

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

w：星期，計算出的數值具有以下對應關係：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：年份前兩位數，以2018年為例，c的值要取20

y：年份後兩位數，以2018年為例，y的值要取18

m：月份，在Zeller公式中，1月與2月要視為上一年的13、14月來計算，以2018年1月1日為例，要看作2017年的13月1日來計算

d：日

[ ]：高斯符號，取不大於原數的最大整數

mod：同餘符號，除以某數後取其餘數，程式語言中使用%表示

由於Zeller公式的結果可能為負數，因此結果可以使用下列方式做補正，確保結果的範圍限制在0-星期日 ~ 6-星期六：

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

1.4 降低系統功耗

為了達到省電的目的，系統必須盡可能的待在掉電模式(Power-down Mode)，系統在掉電模式期間，使用者應關閉所有不必要的設備以及外部的耗能元件。

在掉電模式下，38.4 kHz的內部低速RC振盪電路不會停止工作，因此採用此時脈源輸入的計時器會持續工作。計時器在送出中斷訊號的同時會喚醒晶片，使晶片回到一般模式；在一般模式下，當實時時鐘的時間更新完成後，晶片應盡快回到掉電模式，以減少系統的功耗。

1.5 實時時鐘警報(Alarm)

使用者需設定一個警報時間並開啟警報功能，當實時時鐘更新後的時間與警報設定的時間相等時，即可跳至對應的警報程序，使用者可自行編輯警報副程式的程序滿足應用需求。透過函式庫中的指令RTC_SetRTCAlarmTime()與RTC_GetRTCInfo()，使用者可設定與讀取警報的設定值。

1.6 軟體實時時鐘系統流程

透過軟體實現實時時鐘時，需要設定計時器頻率、計時器中斷、計時器喚醒功能等，另外為了降低系統功耗需要開啟掉電模式。晶片在初始化之後，程式流程可以分為三個狀態：晶片睡眠(掉電模式)、中斷服務程序、主程式迴圈，以下針對這幾個流程做簡短說明。

初始化軟硬體：

1. 初始化計時器，開啟計時器中斷與計時器喚醒功能
2. 初始化實時時鐘時間、實時時鐘警報時間
3. 選擇是否開啟實時時鐘警報功能
4. 向實時時鐘告知計時器一秒產生幾次中斷(中斷頻率)

進入程式主體後，會在晶片睡眠(掉電模式)、中斷服務程序與主程式迴圈這三個狀態重複循環：

1. 晶片睡眠(掉電模式)：晶片維持在低功耗狀態等待中斷事件發生
2. 中斷服務程序：當計時器中斷產生後，系統跳至計時器的中斷服務程序，此時先將喚醒次數的暫存器數值加 1，再進行判斷喚醒次數暫存器的值是否大於等於計時器一秒產生的中斷次數。條件成立時代表時間經過一秒，此時會開始進行時間的更新，並在更新完成後將喚醒次數暫存器清除為 0；若是條件不成立，代表時間尚未累積到 1 秒，因此不會進行時間的更新。

   如果有進行時間更新，更新結束後會執行使用者應用程式，此時若是有開啟警報功能，會繼續判斷警報是否產生，並執行對應的警報應用程式，結束後離開中斷服務程序，並回到主程式的迴圈中；如果沒有進行時間更新，會直接離開中斷服務程序回到主程式迴圈。
3. 主程式迴圈：使用者的程序，執行結束後應進入晶片睡眠狀態

計時器中斷頻率與喚醒次數的關係：中斷頻率的意思是計時器每一秒鐘會產生幾次中斷訊號。中斷頻率2 Hz的意思是每秒喚醒晶片2次、發送2次中斷訊號，每次的中斷喚醒，都會累積喚醒次數1，因此當喚醒次數大於等於計時器的每秒喚醒次數時，代表時間已經經過了1秒，此時需要馬上進行時間更新。

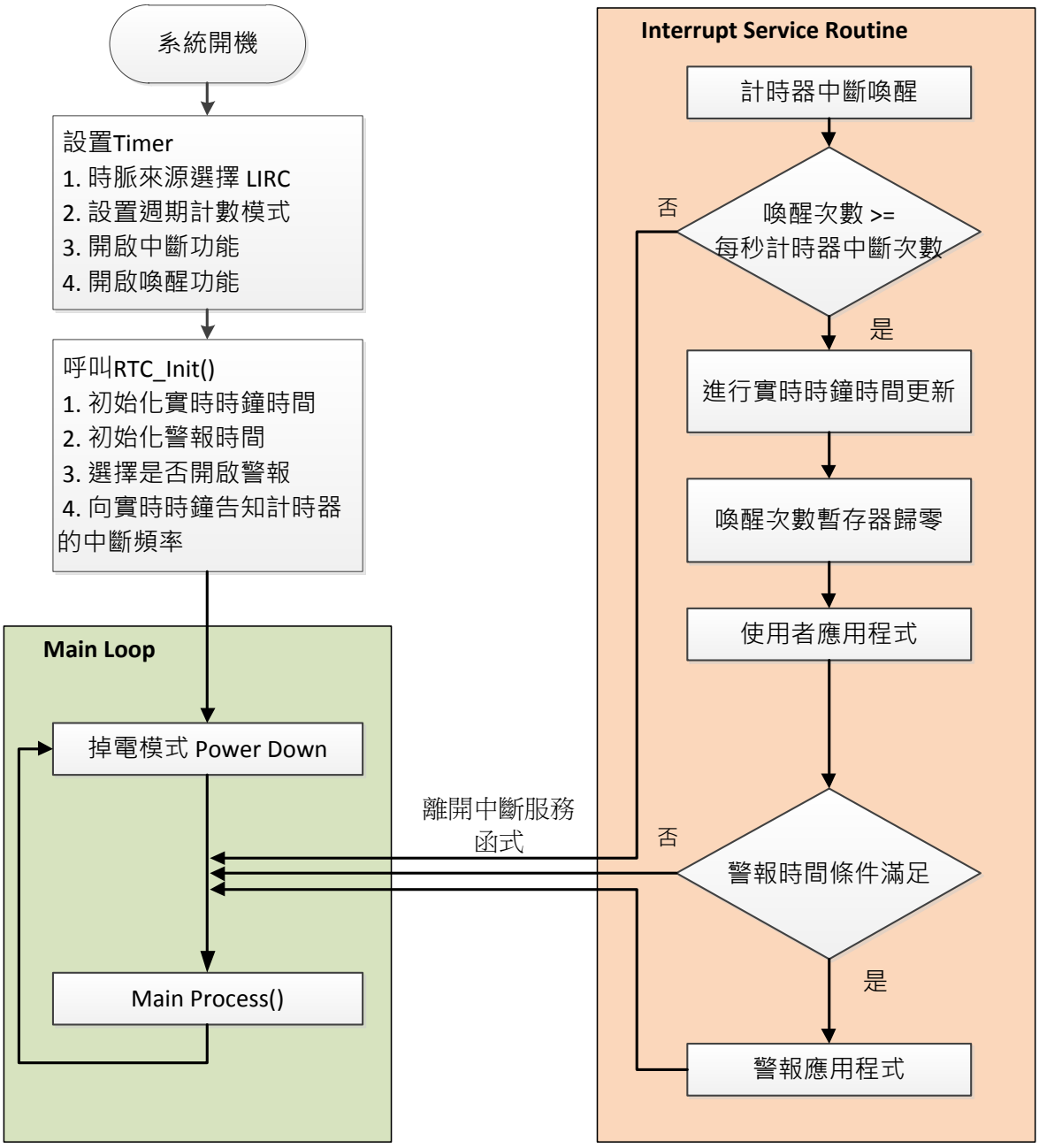


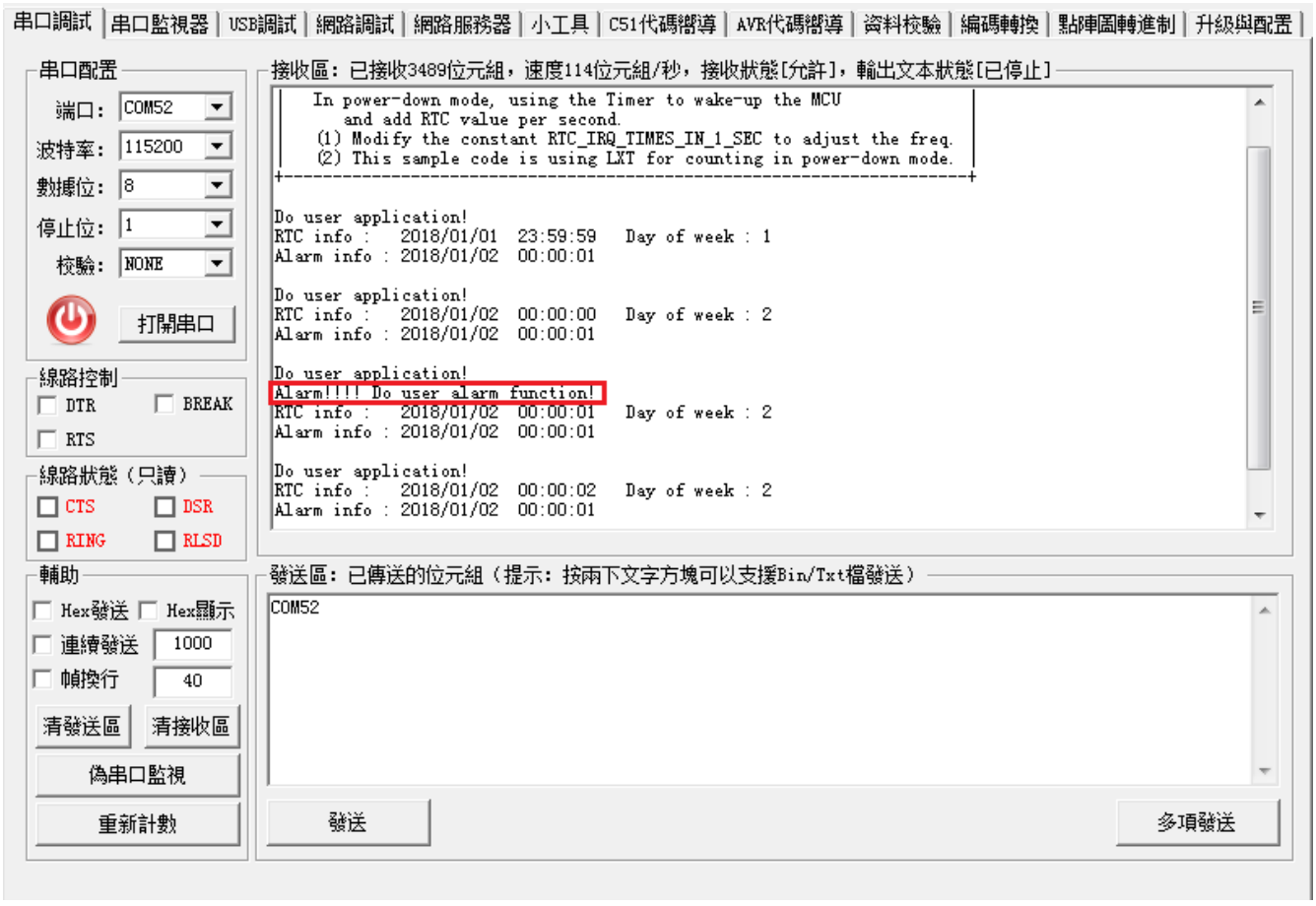
Figure 1-6 系統軟體流程圖

1.7 執行結果

實時時鐘時間：2018/01/01 23:59:58

實時時鐘警報時間：2018/01/02 00:00:01

當實時時鐘的警報被觸發時，會執行警報副程式，此外天數增加的同時星期也會隨之改變。



2 代碼介紹

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 軟體與硬體環境

● 軟體環境

- ■ BSP 版本
  - ◆ M031_Series_BSP_CMSIS_V3.01.000
- ■ IDE 版本
  - ◆ Keil uVersion 5.26

● 硬體環境

- ■ 電路元件
  - ◆ NuMaker-M031
  - ◆ Nu-Bridge
- ■ 接線圖

使用者可以將 M031 上的 UART_RX0(PB.12), UART_TX0(PB.13)接到 Nu-Bridge 上打印 UART 的除錯訊息。使用者需要先打開 PC 上的裝置管理員，查看“NuBridge Virtual Com Port (COMX)”，並打開終端機，將 Baudrate 設定為 115200 接收對應的 Com Port 訊息。

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

4 目錄資訊

- 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 如何執行範例程式

1. 根據目錄資訊章節進入 ExampleCode/SWRTC_V1.00/KEIL 資料夾，雙擊 SW_RTC.uvproj 打開專案。
2. 進入編譯模式介面
   a. 編譯
   b. 下載代碼至記憶體
   c. 進入 / 離開除錯模式
3. 進入除錯模式介面
   a. 執行代碼

6 修訂紀錄

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.