# 实验 18 - 休眠、RTC 唤醒实验

## 1. 实验目的

掌握 NRF24LE1 的寄存器维持低功耗模式的使用。

掌握使用 RTC 作为唤醒源时的配置和使用。

## 2. 实验内容

配置 NRF24LE1 的低功耗模式为：寄存器维持低功耗模式，并使用 RTC 作为系统的唤醒源，每秒唤醒一次 NRF24LE1。

NRF24LE1 唤醒后，驱动指示灯 D1 状态翻转，之后，进入低功耗模式，等待下一次 RTC 唤醒。

## 3. 实验原理

### 1. 寄存器

RTC2 包含两个寄存器用来捕获定时器的值，一个由 32.768KHz 时钟的下降沿驱动，另一个由 MCU 的时钟驱动来获得更好的分辨率。两个寄存器均可以由外部事件的结果来更新。当定时器和比较器寄存器预定义周期的值相同时，RTC2 产生一个终端，RTC2 确保唤醒的功能优于中断。

RTC2 由下述寄存器控制。

表 1：RTC2 寄存器

| 地址 | 名称 | 位 | 复位值 | 类型 | 说明 |
|---|---|---|---|---|---|
| 0xB3 | RTC2CON | 4:0 | 0 | RW | RTC2 配置寄存器 |
| | sfrCapture | 4 | 0 | W | 触发信号<br>当 MCU 写 1 到此位，RTC2 将捕获定时器的值，该值将存储在寄存器 RTC2CPT00 和 RTC2CPT01 中。一个由 MCU 时钟控制的附加的计数器，这个计数器包含从上一个正边沿以来到当前，32.768KHz 时钟的周期数(沿检测@MCU 时钟)，计数器的值存储在 RTC2CPT1。 |
| | enableExternalCapture | 3 | 0 | RW | 1：如果一个来自射频的中断请求，定时器的值将被捕获，并存储在 RTC2CPT00 和 RTC2CPT01。一个由 MCU 时钟控制的附加的计数器，这个计数器包含从上一个正边沿以来到当前，32.768KHz 时钟的周期数(沿检测@MCU 时钟)，计数器的值存储在 RTC2CPT1。 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 0：射频捕获禁止。 |
| | compareMode | 2:1 | 00 | RW | 比较模式<br>11：当定时器的值等于 RTC2CMP1 和 RTC2CMP0 串联的值时，RTC2 IRQ 中断置位。RTC2 确保想要的中断起作用，所有的唤醒优先于 RTC2 IRQ 中断。当 RTC2 IRQ 中断置位，定时器复位。<br>10：与上相同，但 RTC2 IRQ 中断不会复位计数器，而是环绕溢出。<br>0x：禁止比较。 |
| | rtc2Enable | 0 | 0 | RW | 1：RTC2 使能，到 RTC2 核的时钟运转。<br>0：RTC2 时钟禁止，到 RTC2 核的时钟禁止，定时器复位。 |
| 0xB4 | RTC2CMP0 | 7:0 | 0xFF | RW | RTC2 比较值寄存器 0<br>包含用以与定时器值比较产生 RTC2 IRQ 中断的比较值的低字节，分辨率是：30.52us。 |
| 0xB5 | RTC2CMP1 | 7:0 | 0xFF | RW | RTC2 比较值寄存器 1<br>包含用以与定时器值比较产生 RTC2 IRQ 中断的比较值的高字节。 |
| 0Xb6 | RTC2CPT00 | 7:0 | 0x00 | R | RTC2 捕获值寄存器 00<br>包含时间捕获时定时器值的低字节，分辨率是：30.52us。 |
| 0xAB | RTC2CPT01 | 7:0 | 0x00 | R | TC2 捕获值寄存器 01<br>包含时间捕获时定时器值的低高字节。 |
| 0xAC | RTC2CPT10 | 7:0 | 0x00 | R | RTC2 捕获值寄存器 10<br>包含从上一个正边沿到当前，32.768KHz 时钟的周期数(沿检测@MCU 时钟)的值，计数器的值被截去一位（最低位），分辨率是：125ns。 |

## 2. RTC 定时时间计算

RTC 定时时间按下式计算：

$$\text{RTC 定时时间}=\frac{[RTC2CMP1:RTC2CMP0]+1}{32768}$$

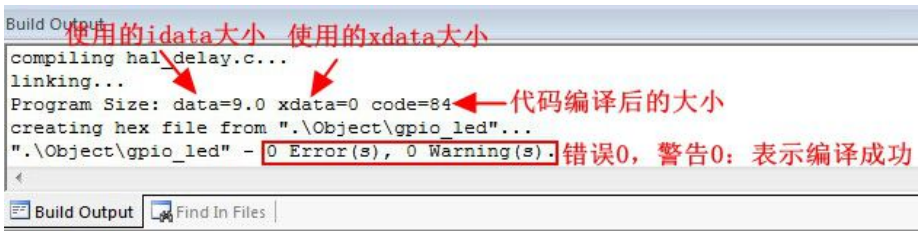
在本例程中的计算：

$$\text{RTC 定时时间}=\frac{\text{SLEEPTIME}}{32768}$$

SLEEPTIME 是程序中休眠时间的宏定义。

# 4. 实验步骤

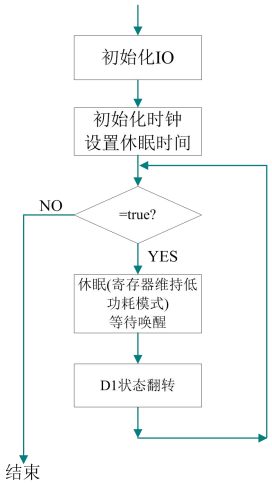
- 在 Keil uVision4 中打开工程“rtc.uvproj”工程；
- 编译工程，注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止；




- 将编译生成的 HEX 文件“rtc.hex” (该文件位于工程目录下的“Object”文件夹中)通过编程器下载到开发板中运行。
- 观察指示灯 D1，其状态每秒翻转一次。

# 5. 实验程序

## 5.1. 程序流程




## 5.2. 程序清单

```
#define   D1      P00   //开发板上的指示灯 D1
#define   SLEEPTIME   32768   //休眠时间：1S

/*******************************************************************************
 *描   述 ：配置 IO P0.0 为输出，驱动 LED。
 *入   参 ：无
 *返回值 ：无
*******************************************************************************/
void IO_Init(void)
```

```
{
  P0DIR &= ~0x01;        //配置 P0.0 为输出
  D1 = 1;    //设置 D4 初始状态为熄灭
}
/*******************************************************************************
 *描   述 : 设置休眠时间，最长时间 2 秒(65536)。
 *入   参 : period:休眠时间，范围 10~65536
 *返回值 : 无
*******************************************************************************/
void set_timer_period(uint16_t period)
{
  hal_rtc_start(false);
  hal_rtc_start(true);
  hal_rtc_set_compare_value(period - 1);
}
/*******************************************************************************
 *描   述 : 时钟和 RTC 唤醒设置
 *入   参 : 无
 *返回值 : 无
*******************************************************************************/
void mcu_init(void)
{
  hal_rtc_start(false); //关闭 32.768KHz 时钟
  hal_clklf_set_source(HAL_CLKLF_RCOSC32K);    //使用 32.768KHz
  hal_rtc_set_compare_mode(HAL_RTC_COMPARE_MODE_0); // 32 KHz 模式 0
  set_timer_period(TAG_TIME);    //设置休眠时间
  hal_clk_set_16m_source(HAL_CLK_XOSC16M); //使用外部 16MHz 晶振
  hal_clk_regret_xosc16m_on(0);    //在寄存器维持低功耗模式下关闭 16MHz 时钟

  hal_rtc_start(true); //启动 32kHz 时钟

  while((CLKLFCTRL&0x80)==0x80);   // 等待时钟启动完成
  while((CLKLFCTRL&0x80)!=0x80);
}

/*******************************************************************************
 *描   述 : 主函数
 *入   参 : 无
 *返回值 : 无
*******************************************************************************/
void main(void)
{
  IO_Init();    //初始化 IO
```

```
    mcu_init();   //初始化时钟、RTC 唤醒、休眠时间

    while(1)
    {
        PWRDWN = 0x04;   //进入寄存器维持低功耗模式(休眠)
        PWRDWN = 0x00;

        D1 = ~D1;   //D1 指示灯状态取反
    }
}
```