青风带你玩蓝牙 nRF51822 系列教程

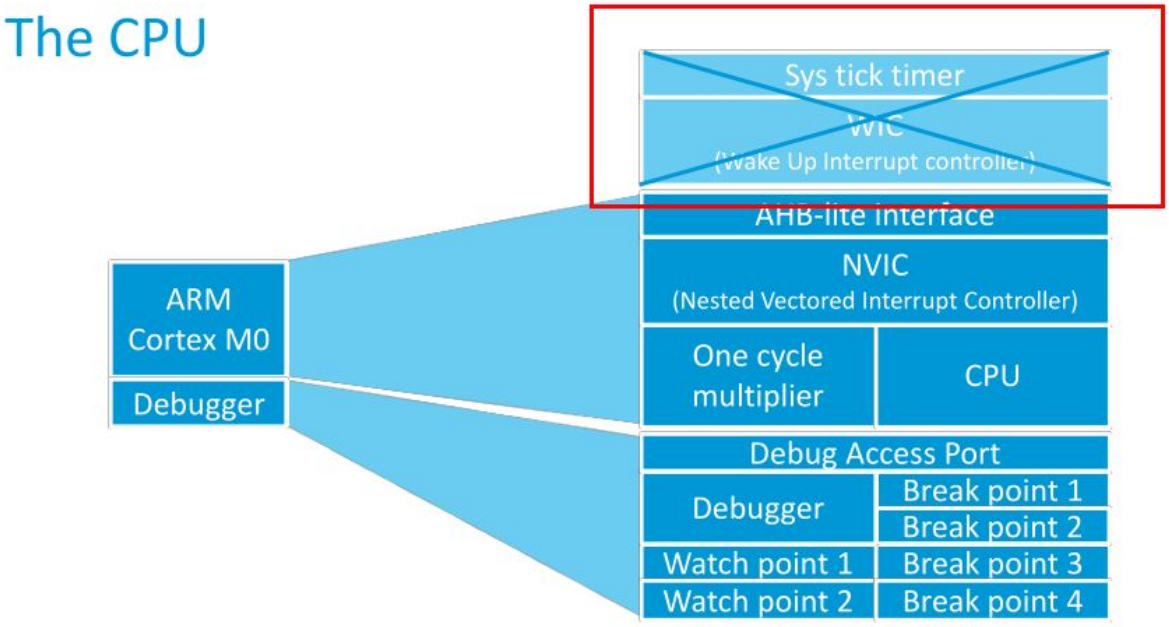
----------作者：青风

出品论坛：　www.qfv8.com　青风电子社区

作者： 青风
出品论坛: www.qfv8.com
淘宝店：http://qfv5.taobao.com
QQ 技术群：346518370
硬件平台：青云 QY-nRF51822 开发板

2.5 rtc 时钟和比较器

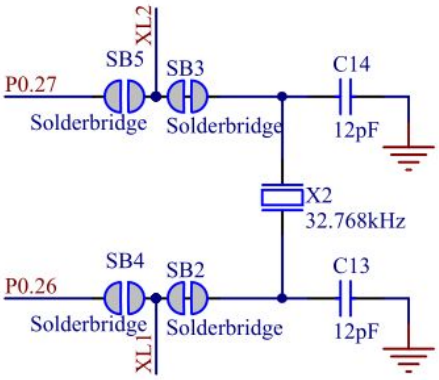
nRF51822 虽然是是 cortex m0 内核，但是为了节省内部资源，芯片把唤醒中断和滴答时钟 systick 给省略掉了，如下图所示。那么为了精确定时，我们就直接采用定时器进行定时了。



在使用 nRF51822 时，你会发现在 startup 文件里有设置 systick 中断声明，但是本人编写了 systick 函数证明 nRF51822 的系统滴答无法开启，这里特此提示一下。

2.5.1 硬件准备：

如下图所示： 青云 QY-nRF51822 开发板上，通过管脚 P0.27 和管脚 P0.26 连接低速外部晶振，晶振大小为 32.768KHZ。图中 SB2 和 SB3,SB4,SB5 分别为外设隔离点，当不需要使用低速晶振的时候，可以把这两个点割开。



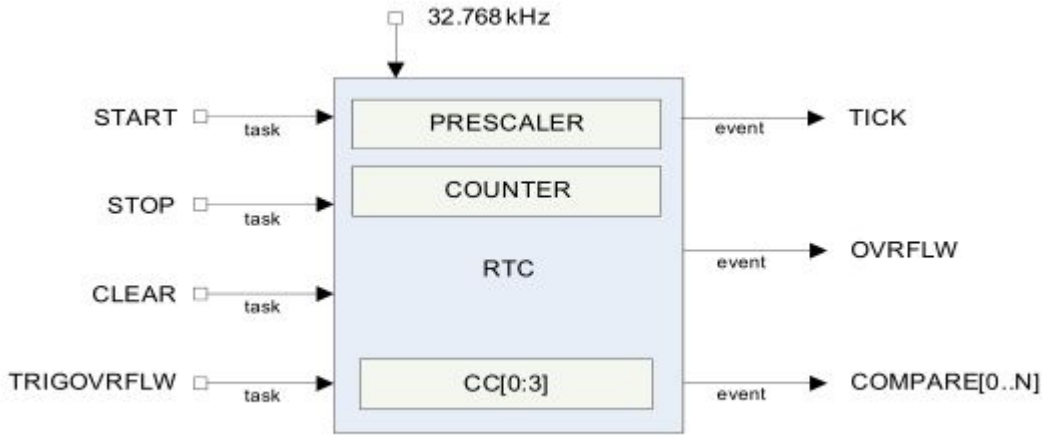
2.5.2 软件准备：

在代码文件中，实验五建立了一个演示历程，我们打开看看需要那些库文件。打开 user 文件夹中的 RTC 工程：

如上图所示：只需要自己编写红色框框里的两个文件就 OK 了,因为采用子函数的方式其中 led.c 在控制 LED 灯的时候已经写好，现在我们就来讨论下如何编写 RTC.C 这个驱动子文件。

RTC.C 文件主要是要起到两个作用：第一：初始化使能 RTC 相关参数。第二：设置 RTC 匹配和比较中断。完成这两个功能够就可以在 main.c 文件中直接调用本驱动了。

下面我们就结合寄存器来详细分析下 RTC 的设置：



上图为 nrf518222 的 RTC 的基本 RTC 时钟结构，RTC 在广义的 MCU 方面的定义就是一个独立的定时器，因此其寄存器的设置应该和 time 定时器的设置相似，这个从 nrf518222 的手册上寄存器定义可以很明显的看出来。回忆一下之前 time 定时器的内容，time 是高速时钟 HFCLK 提供时钟，而 RTC 则是由 LFCLK 提供时钟。因此必须在 16MHZ 时钟断开的时候才能够运行 RTC.

下面我们来配置时钟源，设置代码如下

```
01.    /**  启动 LFCLK 晶振功能*/
02.   void lfclk_config(void)
03.  {
04.        NRF_CLOCK->LFCLKSRC = (CLOCK_LFCLKSRC_SRC_Xtal <<
     CLOCK_LFCLKSRC_SRC_Pos);//设置时钟源
05.        NRF_CLOCK->EVENTS_LFCLKSTARTED    = 0;//关 16M 震荡
06.        NRF_CLOCK->TASKS_LFCLKSTART        = 1;//开 32K 震荡
07.        while (NRF_CLOCK->EVENTS_LFCLKSTARTED == 0)
08.        {
09.              //Do nothing.
10.        }
11.        NRF_CLOCK->EVENTS_LFCLKSTARTED = 0;
12.  }
```

Register	Offset	Description
TASKS		
HFCLKSTART	0x000	Start HFCLK crystal oscillator
HFCLKSTOP	0x004	Stop HFCLK crystal oscillator
LFCLKSTART	0x008	Start LFCLK source
LFCLKSTOP	0x00C	Stop LFCLK source
CAL	0x010	Start calibration of LFCLK RC oscillator
CTSTART	0x014	Start calibration timer
CTSTOP	0x018	Stop calibration timer
EVENTS		
HFCLKSTARTED	0x100	16 MHz oscillator started
LFCLKSTARTED	0x104	32 kHz oscillator started
DONE	0x10C	Calibration of LFCLK RC oscillator complete event
CTTO	0x110	Calibration timer timeout
REGISTERS		
INTENSET	0x304	Interrupt enable set register
INTENCLR	0x308	Interrupt enable clear register
HFCLKSTAT	0x40C	Which HFCLK source is running
LFCLKSTAT	0x418	Which LFCLK source is running
LFCLKSRC	0x518	Clock source for the 32 kHz clock
CTIV	0x538	Calibration timer interval
XTALFREQ	0x550	Crystal frequency

上面红色框框的几个寄存器就是代码里定义的寄存器。

EVENTS_LFCLKSTARTED 高速时钟必须先关掉，因此清零。TASKS_LFCLKSTART 开低速时钟振荡。

时钟源的选择：

```
13. #define CLOCK_LFCLKSRC_SRC_RC (0UL) /*!< 内部 32KiHz RC 时钟. */
14. #define CLOCK_LFCLKSRC_SRC_Xtal (1UL) /*!< 外部 32KiHz 振荡.l. */
15. #define CLOCK_LFCLKSRC_SRC_Synth (2UL) /*!< 内部 32KiHz 从 HFCLK 系统时钟产生. */
```

我们选择外部低速时钟振荡作为时钟源。这里就把低速时钟源设置完成了。接下来进行 RTC 的配置，我们实验的要求是实现 8HZ 的计数频率，也就是计数时间 125ms 控制 led 翻转一次，翻转 24 次接近 3000ms，也就是 3s 的时候进行模拟比较报警，报警控制另一个 led 灯点亮报警。

这个是下面代码设置要用的几个寄存器，解释很清楚，关键是搞清楚怎么用，我们之间对着代码段分析：

```
16.    void rtc_config(void)
17. {
18.        NVIC_EnableIRQ(RTC0_IRQn);                  // 使能 RTC 中断.
19.        NRF_RTC0->PRESCALER        = COUNTER_PRESCALER;        //12 位预分频器的计数器
    频率（ 32768 /（预分频器+1 ） ）
20. //当 RTC 停止时候才能设置
21. // 设置预分频值 prescaler to a TICK of RTC_FREQUENCY.
22.        NRF_RTC0->CC[0]             = COMPARE_COUNTERTIME * RTC_FREQUENCY;   // 设
    置比较寄存器值 Compare0 after approx COMPARE_COUNTERTIME seconds.
23.
```

```
24.       // Enable TICK event and TICK interrupt:启用 Tick 事件和节拍中断
25.       NRF_RTC0->EVTENSET          = RTC_EVTENSET_TICK_Msk;
26.       NRF_RTC0->INTENSET          = RTC_INTENSET_TICK_Msk;
27.
28.       // Enable COMPARE0 event and COMPARE0 interrupt:启用事件比较匹配和比较匹配中断
29.       NRF_RTC0->EVTENSET          = RTC_EVTENSET_COMPARE0_Msk;
30.       NRF_RTC0->INTENSET          = RTC_INTENSET_COMPARE0_Msk;
31. }
32.
```

上面一段代码的编写严格按照了寄存器要求进行，首先是使能 RTC 嵌套中断，这个 NVIC 是 cortex m 系列处理器通用的中断方式。

然后我们设计 RTC 的计数频率，根据手册给出的公式：

$$f_{RTC} = \frac{32,768kHz}{PRESCALER + 1}$$

根据这个公式，下面举两个例子：

1. 设置计数频率为 100 Hz (10 ms 一个计数周期)

代入公式 PRESCALER = round(32.768 kHz / 100 Hz) - 1 = 327

RTC 的时钟频率 fRTC = 99.9 Hz

10009.576 μs 一个计数周期

2. 设置计数频率为 8 Hz (125ms 一个计数周期)

代入公式 PRESCALER = round (32.768 kHz / 8 Hz) － 1 = 4095

RTC 的时钟频率 fRTC = 8 Hz

125 ms 一个计数周期

PRESCALER 寄存器设置 RTC 预分频计数器，这个寄存器是我们需要设置的，在代码中，根据要求的参数值进行设置：

```
NRF_RTC0->PRESCALER         = COUNTER_PRESCALER;
#define   COUNTER_PRESCALER              ((LFCLK_FREQUENCY/RTC_FREQUENCY) - 1)
#define LFCLK_FREQUENCY   (32768UL) //
#define RTC_FREQUENCY                  (8UL)
```

这个参数是更加上面举的例子 2 里

代入公式 PRESCALER = round (32.768 kHz / 8 Hz) － 1 = 4095 来设置这个 **RTC 预分频计数器然后设置比较器中断，根据分析需要翻转**转 24 次才能接近 3000ms，因此比较次数也是 24 次，设置 CC[0] 寄存器的值为 24。

18.2.6 CC[N] (n=0..3)

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RW	-	-	-	-	-	-	-	-	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ID	Field	Value	Description
A			Compare value

设置完成后，使能计数器中断和比较中断：

```
33.     // Enable TICK event and TICK interrupt:启用 Tick 事件和节拍中断
34.     NRF_RTC0->EVTENSET          = RTC_EVTENSET_TICK_Msk;
35.     NRF_RTC0->INTENSET          = RTC_INTENSET_TICK_Msk;
36.
37.     // Enable COMPARE0 event and COMPARE0 interrupt:启用事件比较匹配和比较匹配中断
38.     NRF_RTC0->EVTENSET          = RTC_EVTENSET_COMPARE0_Msk;
39.     NRF_RTC0->INTENSET          = RTC_INTENSET_COMPARE0_Msk;
```

并且设置中断功能，中断做的内容就比较简单了，判断中断发生后翻转 LED 灯。

```
40.   void RTC0_IRQHandler()
41. {
42.     if ((NRF_RTC0->EVENTS_TICK != 0) &&
43.         ((NRF_RTC0->INTENSET & RTC_INTENSET_TICK_Msk) != 0))
44.     {
45.         NRF_RTC0->EVENTS_TICK = 0;
46.         LED1_Toggle();
47.     }
48.
49.     if ((NRF_RTC0->EVENTS_COMPARE[0] != 0) &&
50.         ((NRF_RTC0->INTENSET & RTC_INTENSET_COMPARE0_Msk) != 0))
51.     {
52.         NRF_RTC0->EVENTS_COMPARE[0] = 0;
53.         LED2_Toggle();
54.     }
55. }
56.
57.
```

那么主函数就是十分的简单了，直接调用我们写好的驱动函数，LED 灯指示定时器相应的变化。函数如下所示：

```
58. /******************** (C) COPYRIGHT 2012 青风电子 ********************
59.  * 文件名  ：main
60.  * 描述    ：
61.  * 实验平台：青风 stm32f051 开发板
62.  * 描述    ：RTC
```

```
63.  * 作者      ：青风
64.  * 店铺      ：qfv5.taobao.com
65. ********************************************************************************/
66. #include "nrf51.h"
67. #include   "led.h"
68. #include   "rtc.h"
69.
70. int main(void)
71. {
72.       LED_Init();
73.   LED1_Close();
74.   LED2_Close();
75.       lfclk_config();//启动内部 LFCLK 晶振功能
76.       rtc_config();//配置 RTC
77.
78.       NRF_RTC0->TASKS_START = 1;//开启 rtc
79.
80.       while (1)
81.       {
82.             // Do nothing.
83.       }
84. }
```

实验下载到青云 nRF51822 开发板后的实验现象如下，led1 灯 125ms 翻转，led2 灯 3s 后的报警点亮：