青风带你学 stm32f030 系列教程

---------- 库函数操作版本

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：241364123
硬件平台：QF-STM32F030 开发板

2.4 实时时钟 RTC

区别于 51 单片机没有内部 RTC, stm32f030 系列 CORTEX M0 中内置一个独立的 BCD 定时器/计数器做为 RTC 实时时钟。

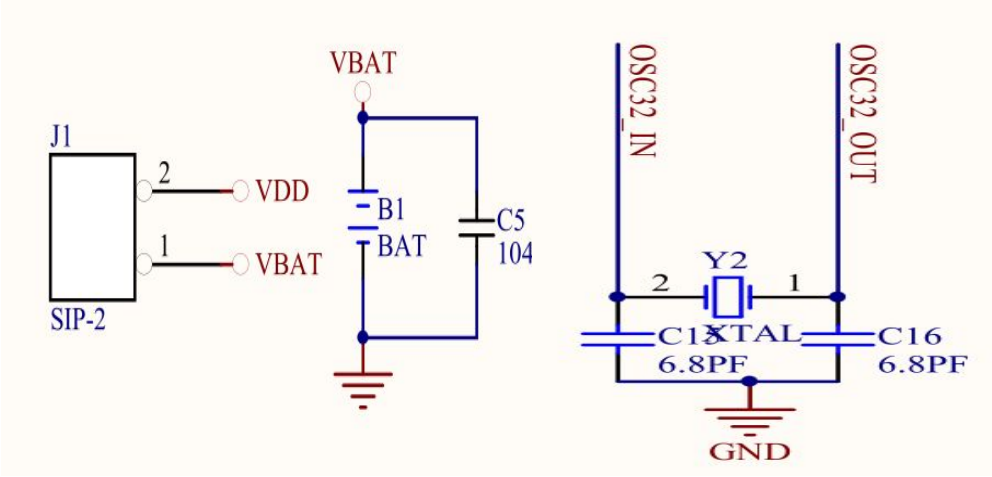
实际上，RTC 就只一个定时器而已，但这个定时器/计数器，不管系统是否上电，都要为系统完成当前时间显示的功能。也就是说，掉电之后时钟信息会被保存下来。比如，当前时间是下午 2:20 分，那么，STM32 断电，1 个小时后再开机，当前时间应该自动显示 3:20 分。

如何完成这个过程呢？STM32 这样处理：

在 STM32 的 CPU 有一个 RTC 的引脚，用于连接电池。当系统掉电，自动由电池给 STM32 的 RTC 供电，完成时间计数的继续。并且在相应软件配置下，可提供时钟日历的功能。同时 RTC 里还包含了一个报警模块，可以定时报警。

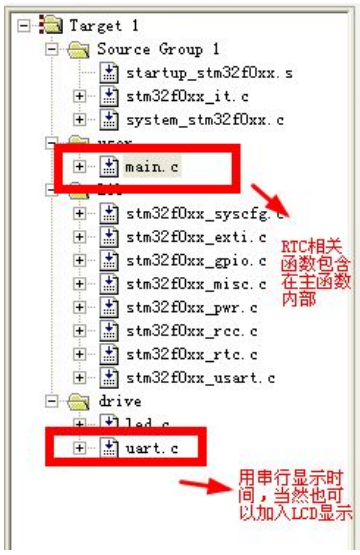
2.4.1 硬件准备：

掉电的情况下才有纽扣电池供电，并且 PC14，PC15 外接 32.768 的低速晶振，如下面电路图所示：



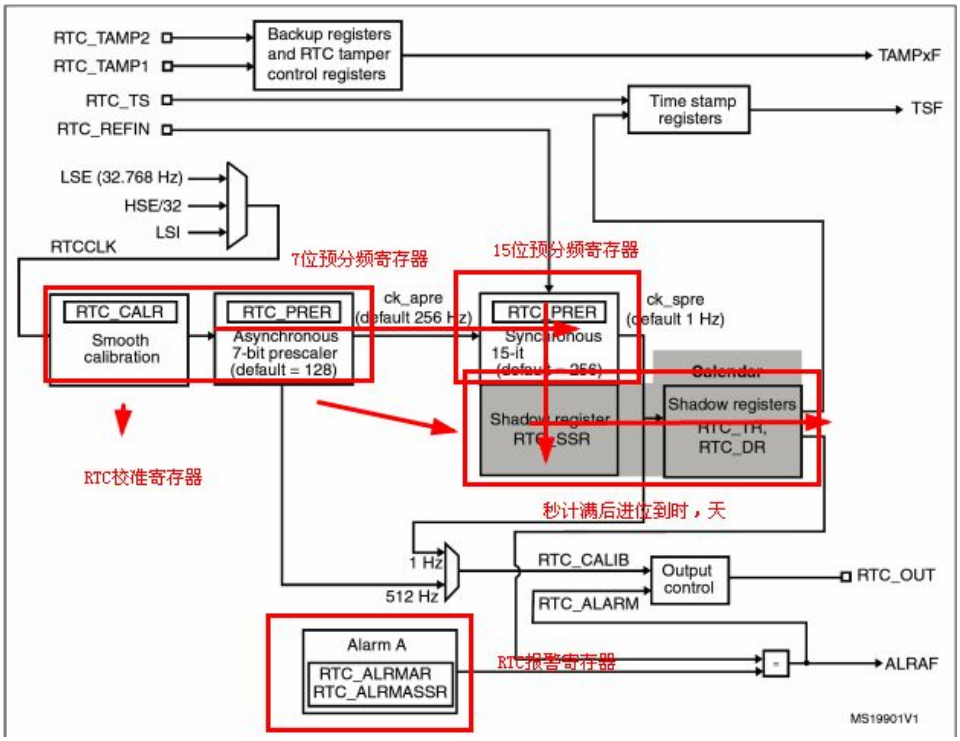
2.4.2 软件准备：

打开 keil 编译环境，设置系统工程树如下图所示：



如上图所示，我们需要 RTC 的所有配置都写到了 main 主函数内部，然后调用串口中断进行实时时钟的显示，而库函数调用 stm32f0xx_rtc.c 和 stm32f0xx_rcc.c,中断函数在 stm32f0xx_it.c 函数内。

我们要配置好一个 RTC 模块，首先要搞清楚 RTC 的整个系统结构以及他的工作方式，RTC 内部结构如下图所示,红色箭头表示箭头走向：



如上图所示 RTC 的时钟控制器从以下 3 种时钟中选择 RTC 时钟源(RTCCLK)：

- LSE 振荡器作为 RTC 时钟；
- LSI 振荡器作为 RTC 时钟；
- HSE 振荡器作为 RTC 时钟。

RTC 的时钟经过校准后输入到预分频器，为最大限度地减少功耗，预分频器可分割成 2 个可编程预分频器。一个是 7 位预分频器，一个是 15 位预分频器。我们的要求是通过这两个预分频器之后能够输出 1HZ 的时钟信号，因为 1HZ 正好是计数 1s，秒计数器累加一次。

所以程序员设置 RTC 运转的主要工作就分为了两个部分：

1. 选择好 RTC 时钟的振荡源。
2. 设置两个预分频器的值，是的对应振荡源通过预分频器后能够输出 1HZ 的时钟信号。

RTC 振荡源的设置我们可以参考参考手册关于 RTC 时钟的设置：

7.2.9 RTC 时钟

通过设置备份域控制寄存器 (RCC_BDCR) 里的 RTCSEL[1:0] 位，RTCCLK 时钟源可以由 HSE/32、LSE 或 LSI 时钟提供。除非备份域复位，此选择不能被改变。系统必须按 PCLK 的频率须快于或等于 RTCCLK 的频率的方式配置才能正确操作 RTC。

并且在每种时钟状态下的供电方式说明：

- 若 LSE 被选为 RTC 时钟：
  - 只要维持 VBAT 正常供电，即使 VDD 掉电，RTC 仍会继续工作。
- 若 LSI 被选为 RTC 时钟：
  - 当 VDD 掉电时，RTC 处于不定的状态。
- 若 HSE/32 被选为 RTC 时钟：
  - 当 VDD 掉电或内部电压调压器 (1.8V 域的供电切断) 掉电时，RTC 处于不定的状态。

那么我们在代码中设置如下：

```
01.  #if defined (RTC_CLOCK_SOURCE_LSI)  /* 当使用 LSI 作为 RTC 时钟源*/
02. /* The RTC Clock may varies due to LSI frequency dispersion. */
03.   /* 使能 LSI 振荡 */
04.   RCC_LSICmd(ENABLE);
05.
06.   /* 等待到 LSI 预备*/
07.   while(RCC_GetFlagStatus(RCC_FLAG_LSIRDY) == RESET)
08.   {
09.   }
10.
11.   /* 把 RTC 时钟源配置为 LSI */
12.   RCC_RTCCLKConfig(RCC_RTCCLKSource_LSI);
```

设置 LSE 和 LSI 的方法类似，我们在文件里定义使用 LSE。设置好时钟源后，我们在来设置预分频计数器的值。参考手册有如下说明：

f_{ck_apre} 满足：

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

The ck_apre 时钟为二进制 RTC_SSR 亚秒递减计数器提供时钟。当值为 0 时，RTC_SSR 的值将被重置为 PREDIV_S 里的值。

f_{ck_spre} 满足

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$

也就是说，分频器计数满后，正好需要花的时间为 1s，那么 7 位同步预分频的全

满为 1111111，转换为 10 进制为：127, 15 位异步预分频的全满，转换为 10 进制为：32767，举例在选择 32.768 kHz LSE 的时候，根据公式，可以达到最多分频 1/127s ,那么要得到 1S，为了降低功耗，首先是把 7 位预分频用满 127，那么根据公式在 LSE 的时候 32768/127=255,因此 15 位异步预分频系数为 256。

那么代码可以 RTC 的配置代码按照下面方式进行设定：

```
13. void RTC_Config(void)
14. {
15.     /* 使能 PWR 时钟 */
16.   RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR, ENABLE);
17.
18.   /* 允许访问 RTC */
19.   PWR_BackupAccessCmd(ENABLE);
20.
21. #if defined (RTC_CLOCK_SOURCE_LSI)  /* 当使用 LSI 作为 RTC 时钟源*/
22.   /* 使能 LSI 振荡 */
23.   RCC_LSICmd(ENABLE);
24.
25.   /* 等待到 LSI 预备*/
26.   while(RCC_GetFlagStatus(RCC_FLAG_LSIRDY) == RESET)
27.   {
28.   }
29.
30.   /* 把 RTC 时钟源配置为 LSI */
31.   RCC_RTCCLKConfig(RCC_RTCCLKSource_LSI);
32.     /* 定褊同步分频值和异步分频值 */
33.   SynchPrediv = 0x18F;//400-1=399
34.   AsynchPrediv = 0x63;//100-1=99 40KHZ/400*100 =1HZ
35.
36. #elif defined (RTC_CLOCK_SOURCE_LSE) /* 当使用 LSE 最为 RTC 时钟源 */
37.   /*使能 LSE 振荡 */
38.   RCC_LSEConfig(RCC_LSE_ON);
39.
40.   /*等待 LSE 预备 */
41.   while(RCC_GetFlagStatus(RCC_FLAG_LSERDY) == RESET)
42.   {
43.   }
44.
45.   /* 把 RTC 时钟源配置为使用 LSE */
46.   RCC_RTCCLKConfig(RCC_RTCCLKSource_LSE);
47.       /* 定褊同步分频值和异步分频值 */
48.   SynchPrediv = 0xFF;?/256-1=255
49.   AsynchPrediv = 0x7F;?/128-1=127
50.
51. #else
```

```
52.    #error Please select the RTC Clock source inside the main.c file
53. #endif /* RTC_CLOCK_SOURCE_LSI */
54.
55.    /* 使能 RTC 时钟 */
56.    RCC_RTCCLKCmd(ENABLE);
57.
58.    /* 等待 RTC APB 寄存器同步 */
59.    RTC_WaitForSynchro();
60. }
61.
```

然后设置初始时钟和报警时钟，我们采用串口输入方式实现：

```
62.   printf("    Please Set Hours:\n\r");
63.    while (tmp_hh == 0xFF)
64.    {
65.      tmp_hh = USART_Scanf(23);
66.      RTC_TimeStructure.RTC_Hours = tmp_hh;
67.    }
68.
```

配置成功后就可以显示当前时钟了：

```
1.    printf("\n\r>> !! RTC Set Time success. !! <<\n\r");
2.        RTC_TimeShow();
3.        /* Indicator for the RTC configuration */
4.        RTC_WriteBackupRegister(RTC_BKP_DR0, BKP_VALUE);
```

而时钟报警的设置方式可以入下设置，设置报警时钟和相关寄存器：

```
1.    RTC_AlarmCmd(RTC_Alarm_A, DISABLE);
2.
3.     printf("\n\r==============Alarm A
      Settings====================================\n\r");
4.     RTC_AlarmStructure.RTC_AlarmTime.RTC_H12 = RTC_H12_AM;
5.     printf("    Please Set Alarm Hours:\n\r");//输入报警的小时
6.     while (tmp_hh == 0xFF)
7.     {
8.       tmp_hh = USART_Scanf(23);
9.       RTC_AlarmStructure.RTC_AlarmTime.RTC_Hours = tmp_hh;
10.    }
11.    printf("    %0.2d\n\r", tmp_hh);
12.    printf("    Please Set Alarm Minutes:\n\r");//输入报警分钟
13.    while (tmp_mm == 0xFF)
14.    {
15.      tmp_mm = USART_Scanf(59);
16.      RTC_AlarmStructure.RTC_AlarmTime.RTC_Minutes = tmp_mm;
17.    }
18.    printf("    %0.2d\n\r", tmp_mm);
19.    printf("    Please Set Alarm Seconds:\n\r");//输入报警秒
```

```
20.    while (tmp_ss == 0xFF)
21.    {
22.      tmp_ss = USART_Scanf(59);
23.      RTC_AlarmStructure.RTC_AlarmTime.RTC_Seconds = tmp_ss;
24.    }
25.    printf("    %0.2d", tmp_ss);
26.    /* 设置报警 A */
27.    RTC_AlarmStructure.RTC_AlarmDateWeekDay = 0x31;
28.    RTC_AlarmStructure.RTC_AlarmDateWeekDaySel = RTC_AlarmDateWeekDaySel_Date;
29.    RTC_AlarmStructure.RTC_AlarmMask = RTC_AlarmMask_DateWeekDay;
30.    /* 配置 RTC 报警 A 寄存器 */
31.    RTC_SetAlarm(RTC_Format_BIN, RTC_Alarm_A, &RTC_AlarmStructure);
32.    printf("\n\r>> !! RTC Set Alarm success. !! <<\n\r");
33.    RTC_AlarmShow();
34.    /* 时能 RTC 报警中断   */
35.    RTC_ITConfig(RTC_IT_ALRA, ENABLE);
36.    /* 使能报警   A */
37.    RTC_AlarmCmd(RTC_Alarm_A, ENABLE);
```

我们需要使用到 **RTC** 中断，这个需要我们设置好：

```
38.    /* RTC Alarm A Interrupt Configuration */
39.    /* EXTI configuration *********************************************************/
40.    EXTI_ClearITPendingBit(EXTI_Line17);
41.    EXTI_InitStructure.EXTI_Line = EXTI_Line17;
42.    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
43.    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;
44.    EXTI_InitStructure.EXTI_LineCmd = ENABLE;
45.    EXTI_Init(&EXTI_InitStructure);
46.
47.    /* Enable the RTC Alarm Interrupt */
48.    NVIC_InitStructure.NVIC_IRQChannel = RTC_IRQn;
49.    NVIC_InitStructure.NVIC_IRQChannelPriority = 0;
50.    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
51.    NVIC_Init(&NVIC_InitStructure);
```

主函数主要就是调用上面设置的几个函数实现 **RTC** 初始化和配置报警寄存器，由于代码比较长，这里就不贴出来了。

2.4.3 实验现象：

下载工程项目后，打开串口调整助手，然后按下复位键，按照串口提示输入当前时钟和报警时钟，注意输入时保证两位数输入，比如 **1** 用 **01,2** 用 **02** 形式输入，下图是实现效果：