

零死角玩转STM32

与野火同行 乐意惬意无边



原创教程，完全开源。



由浅入深，结合实操。



通俗易懂，详尽解读。



配套板子，全面玩转。



强强联合，不断更新。



野火团队 Wild Fire Team

0、友情提示

《零死角玩转 STM32》系列教程由初级篇、中级篇、高级篇、系统篇、四个部分组成，根据野火 STM32 开发板旧版教程升级而来，且经过重新深入编写，重新排版，更适合初学者，步步为营，从入门到精通，从裸奔到系统，让您零死角玩转 STM32。M3 的世界，与野火同行，乐意惬意无边。

另外，野火团队历时一年精心打造的《STM32 库开发实战指南》将于今年 10 月份由机械工业出版社出版，该书的排版更适于纸质书本阅读以及更有利于查阅资料。内容上会给你带来更多的惊喜。是一本学习 STM32 必备的工具书。敬请期待！



4、RTC（万年历）

4.1 实验描述及工程文件清单

实验描述	<p>利用 STM32 的 RTC 实现一个简易的电子时钟。在超级终端中显示时间值。</p> <p>显示格式为 Time: XX:XX:XX(时：分：秒)，当时间计数为：23：59：59 时将刷新为：00：00：00。</p>
硬件连接	VBAT 引脚需外接电池。
用到的库文件	<p>startup/start_stm32f10x_hd.c</p> <p>CMSIS/core_cm3.c</p> <p>CMSIS/system_stm32f10x.c</p> <p>FWlib/stm32f10x_gpio.c</p> <p>FWlib/stm32f10x_rcc.c</p> <p>FWlib/stm32f10x_usart.c</p> <p>FWlib/stm32f10x_pwr.c</p> <p>FWlib/stm32f10x_bkp.c</p> <p>FWlib/stm32f10x_rtc.c</p> <p>FWlib/stm32f10x_misc.c</p>
用户编写的文件	<p>USER/main.c</p> <p>USER/stm32f10x_it.c</p> <p>USER/usart.c</p> <p>USER/rtc.c</p>



4.2 RTC（实时时钟）简介

实时时钟是一个独立的定时器。RTC 模块拥有一组连续计数的计数器，在相应软件配置下，可提供时钟日历的功能。修改计数器的值可以重新设置系统当前的时间和日期。

RTC 模块和时钟配置系统(RCC_BDCR 寄存器)是在后备区域，即在系统复位或从待机模式唤醒后 RTC 的设置和时间维持不变。

系统复位后，禁止访问后备寄存器和 RTC，防止对后备区域(BKP)的意外写操作。执行以下操作使能对后备寄存器和 RTC 的访问：

- 设置寄存器 RCC_APB1ENR 的 PWREN 和 BKPEN 位来使能电源和后备接口时钟。
- 设置寄存器 PWR_CR 的 DBP 位使能对后备寄存器和 RTC 的访问。

当我们需要在掉电之后，又需要 RTC 时钟正常运行的话，单片机的 VBAT 脚需外接 3.3V 的锂电池。当我们重新上电的时候，主电源给 VBAT 供电，当系统掉电之后 VBAT 给 RTC 时钟工作，RTC 中的数据都会保持在后备寄存器当中。

野火 STM32 开发板的 VBAT 引脚接了 3.3V 的锂电。

4.3 代码分析

首先添加需要的库文件：

```
FWlib/stm32f10x_gpio.c  
FWlib/stm32f10x_rcc.c  
FWlib/stm32f10x_usart.c  
FWlib/stm32f10x_pwr.c  
FWlib/stm32f10x_bkp.c  
FWlib/stm32f10x_rtc.c  
FWlib/stm32f10x_misc.c
```



在 `stm32f10x_conf.g` 中将相应库文件的头文件的注释去掉，这样才能够真正使用这些库，否则将会编译错误。

```
1.  /* Uncomment the line below to enable peripheral header file inclusion */
2.  /* #include "stm32f10x_adc.h" */
3.  #include "stm32f10x_bkp.h"
4.  /* #include "stm32f10x_can.h" */
5.  /* #include "stm32f10x_crc.h" */
6.  /* #include "stm32f10x_dac.h" */
7.  /* #include "stm32f10x_dbgmcu.h" */
8.  /* #include "stm32f10x_dma.h" */
9.  /* #include "stm32f10x_exti.h" */
10. /*#include "stm32f10x_flash.h"*/
11. /* #include "stm32f10x_fsmc.h" */
12. #include "stm32f10x_gpio.h"
13. /* #include "stm32f10x_i2c.h" */
14. #include "stm32f10x_iwdg.h"
15. #include "stm32f10x_pwr.h"
16. #include "stm32f10x_rcc.h"
17. #include "stm32f10x_rtc.h"
18. /* #include "stm32f10x_sdio.h" */
19. /* #include "stm32f10x_spi.h" */
20. /* #include "stm32f10x_tim.h" */
21. #include "stm32f10x_usart.h"
22. /* #include "stm32f10x_wwdg.h" */
23. #include "misc.h" /* High level functions for NVIC and SysTick (add-
    on to CMSIS functions) */
```

好嘞，配置好库的环境之后，我们就从 `main` 函数开始分析。`main` 函数有点长，大家给点耐心，好好分析下，也不难。

```
1.  /**
2.   * @brief Main program.
3.   * @param None
4.   * @retval : None
5.   */
6.
7.  int main(void)
8.  {
9.      /* config the sysclock to 72M */
10.     SystemInit();
11.
12.     /* USART1 config */
13.     USART1_Config();
14. }
```





```
15.      /* 配置 RTC 秒中断优先级 */
16.      NVIC_Configuration();
17.
18.      printf( "\r\n This is a RTC demo..... \r\n" );
19.
20.      if (BKP_ReadBackupRegister(BKP_DR1) != 0xA5A5)
21.      {
22.          /* Backup data register value is not correct or not yet programmed (when
23.             the first time the program is executed) */
24.          printf("\r\nThis is a RTC demo!\r\n");
25.          printf("\r\n\n RTC not yet configured...");
26.
27.          /* RTC Configuration */
28.          RTC_Configuration();
29.
30.          printf("\r\n RTC configured...");
31.
32.          /* Adjust time by values entered by the user on the hyperterminal */
33.          Time_Adjust();
34.
35.          BKP_WriteBackupRegister(BKP_DR1, 0xA5A5);
36.      }
37.      else
38.      {
39.          /* Check if the Power On Reset flag is set */
40.          if (RCC_GetFlagStatus(RCC_FLAG_PORRST) != RESET)
41.          {
42.              printf("\r\n\n Power On Reset occurred...");
43.          }
44.          /* Check if the Pin Reset flag is set */
45.          else if (RCC_GetFlagStatus(RCC_FLAG_PINRST) != RESET)
46.          {
47.              printf("\r\n\n External Reset occurred...");
48.          }
49.
50.          printf("\r\n No need to configure RTC...");
51.          /* Wait for RTC registers synchronization */
52.          RTC_WaitForSynchro();
53.
54.          /* Enable the RTC Second */
55.          RTC_ITConfig(RTC_IT_SEC, ENABLE);
56.          /* Wait until last write operation on RTC registers has finished */
57.          RTC_WaitForLastTask();
58.      }
59.
60. #ifdef RTCClockOutput_Enable
61.      /* Enable PWR and BKP clocks */
62.      RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR | RCC_APB1Periph_BKP, ENABLE);
63.
64.      /* Allow access to BKP Domain */
65.      PWR_BackupAccessCmd(ENABLE);
66.
67.      /* Disable the Tamper Pin */
68.      BKP_TamperPinCmd(DISABLE); /* To output RTCCLK/64 on Tamper pin, the tamper
69.                                     functionality must be disabled */
70.
71.      /* Enable RTC Clock Output on Tamper Pin */
72.      BKP_RTCOutputConfig(BKP_RTCOutputSource_CalibClock);
73. #endif
74.
75.      /* Clear reset flags */
76.      RCC_ClearFlag();
77.
78.      /* Display time in infinite loop */
79.      Time_Show();
80.      while (1)
81.      {
82.      }
83.  }
84. }
85.
```



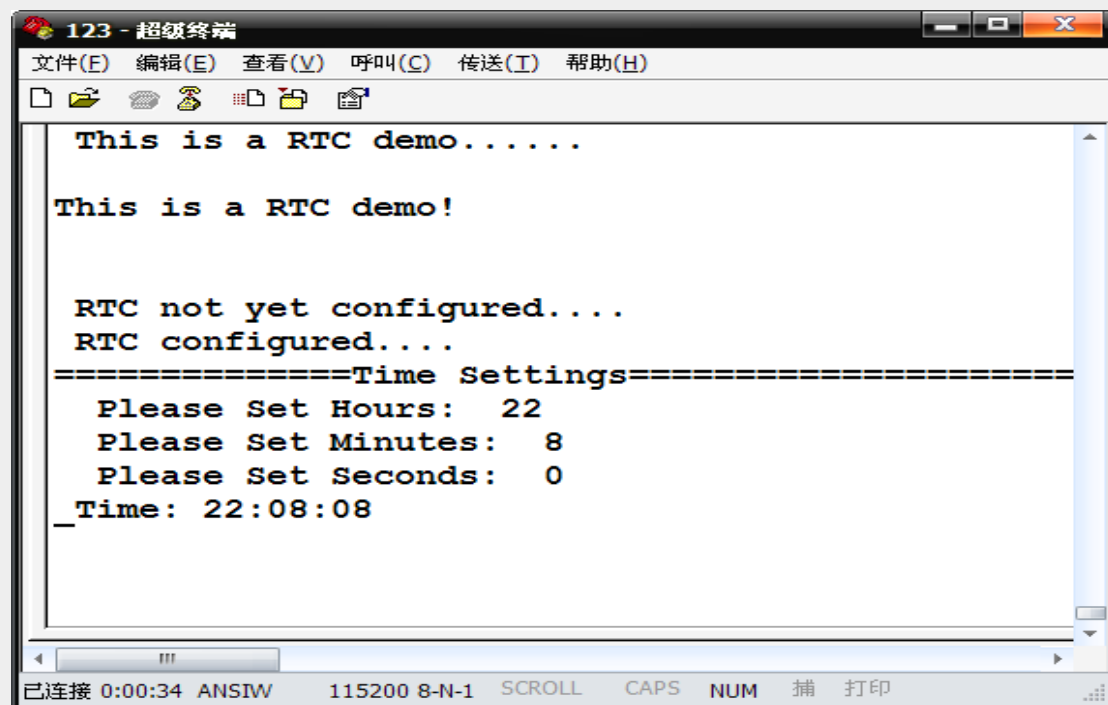
在 `main` 函数开始首先调用库函数 `SystemInit()`; 将我的系统时钟初始化为 72M。

因为我们在实验中需要用到串口，所以我们调用 `USART1_Config()`; 函数将串口配置好。`SystemInit()`; 和 `USART1_Config()`; 这两个函数已在前面相关的教程中讲解过，这里不再详述。

`NVIC_Configuration()`; 函数用于配置 RTC（实时时钟）的中断优先级，我们将它的主优先级设置为 1，次优先级为 0。这里只用到了 RTC 一个中断，所以 RTC 的主和次优先级不必太关心。

接下来的代码部分就是真正跟 RTC 有关的啦：

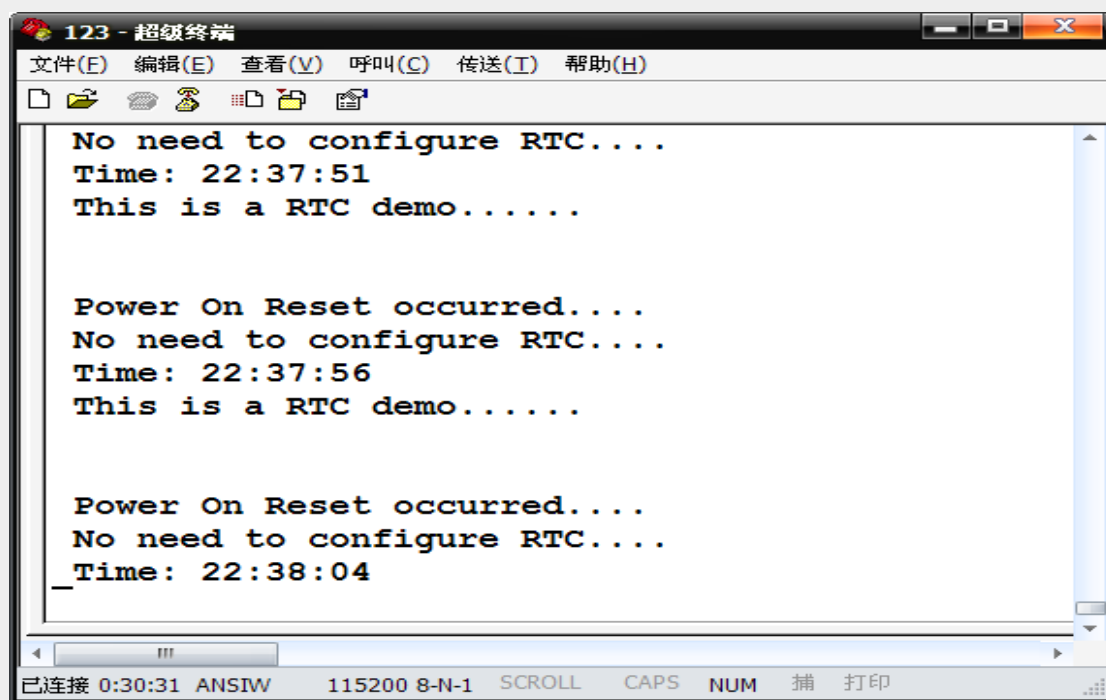
(1) `if()` 部分首先读取 RTC 备份寄存器里面的值，看看备份寄存器里面的值是否正确（如果 RTC 曾经被设置过的话，备份寄存器里面的值为 0XA5A5）或判断这是不是第一次对 RTC 编程。如果这两种情况有任何一种发生的话，则调用 `RTC_Configuration()`;（在 `rtc.c` 中实现）函数来初始化 RTC，并往电脑的超级终端打印出相应的调试信息。初始化好 RTC 之后，调用函数 `Time_Adjust()`;（在 `rtc.c` 中实现）让用户键入（通过超级终端输入）时间值，如下截图所示：



```
123 - 超级终端
文件(E) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
This is a RTC demo.....
This is a RTC demo!
RTC not yet configured....
RTC configured....
====Time Settings====
Please Set Hours: 22
Please Set Minutes: 8
Please Set Seconds: 0
Time: 22:08:08
已连接 0:00:34 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印
```

当我们输入时间值后，RTC 时钟就运行起来了。我这里显示的时间是与我电脑上的时间一样的。设置好时间后，我们把 0XA5A5 这个值写入 RTC 的备份寄存器，这样当我们下一次上电时就不用重新输入 RTC 里面的时间值了。

(2) 如果 RTC 值曾经被设置过，则进入 `else()` 部分。`else` 部分检测是上电复位 还是按键复位，根据不同的复位情况在超级终端中打印出不同的调试信息，但这两种复位都不需要重新设置 RTC 里面的时间值。当检测到系统上电复位时，打印出如下信息：



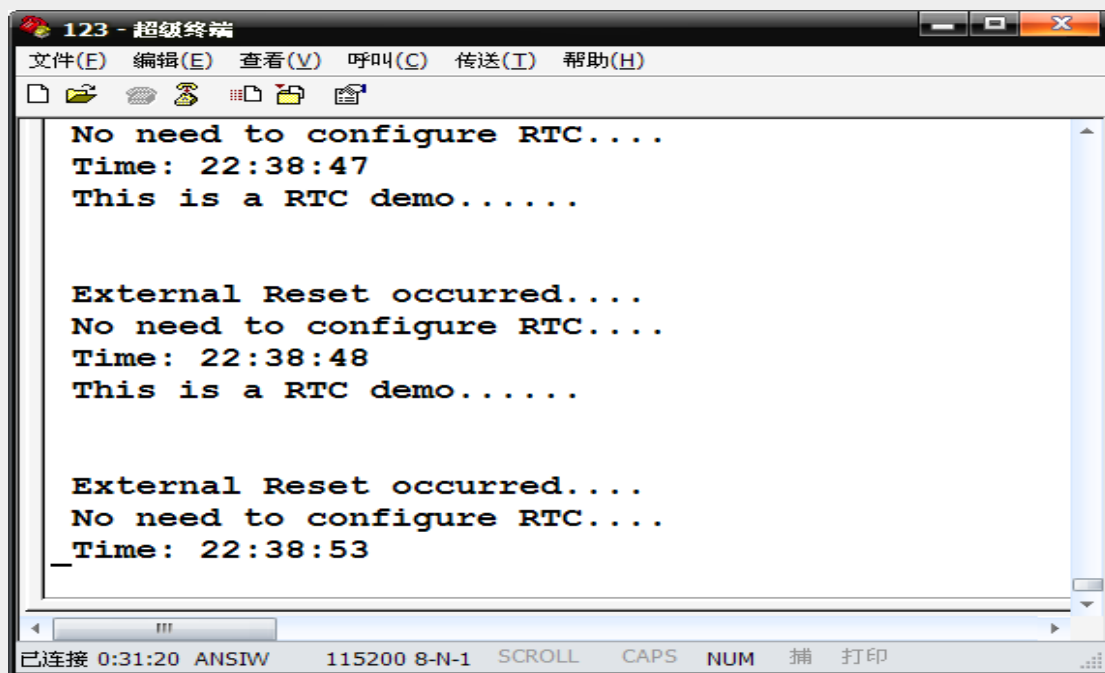
```
123 - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
No need to configure RTC....
Time: 22:37:51
This is a RTC demo.....

Power On Reset occurred....
No need to configure RTC....
Time: 22:37:56
This is a RTC demo.....

Power On Reset occurred....
No need to configure RTC....
Time: 22:38:04
_

已连接 0:30:31 ANSIV 115200 8-N-1 SCROLL CAPS NUM 捕 打印
```

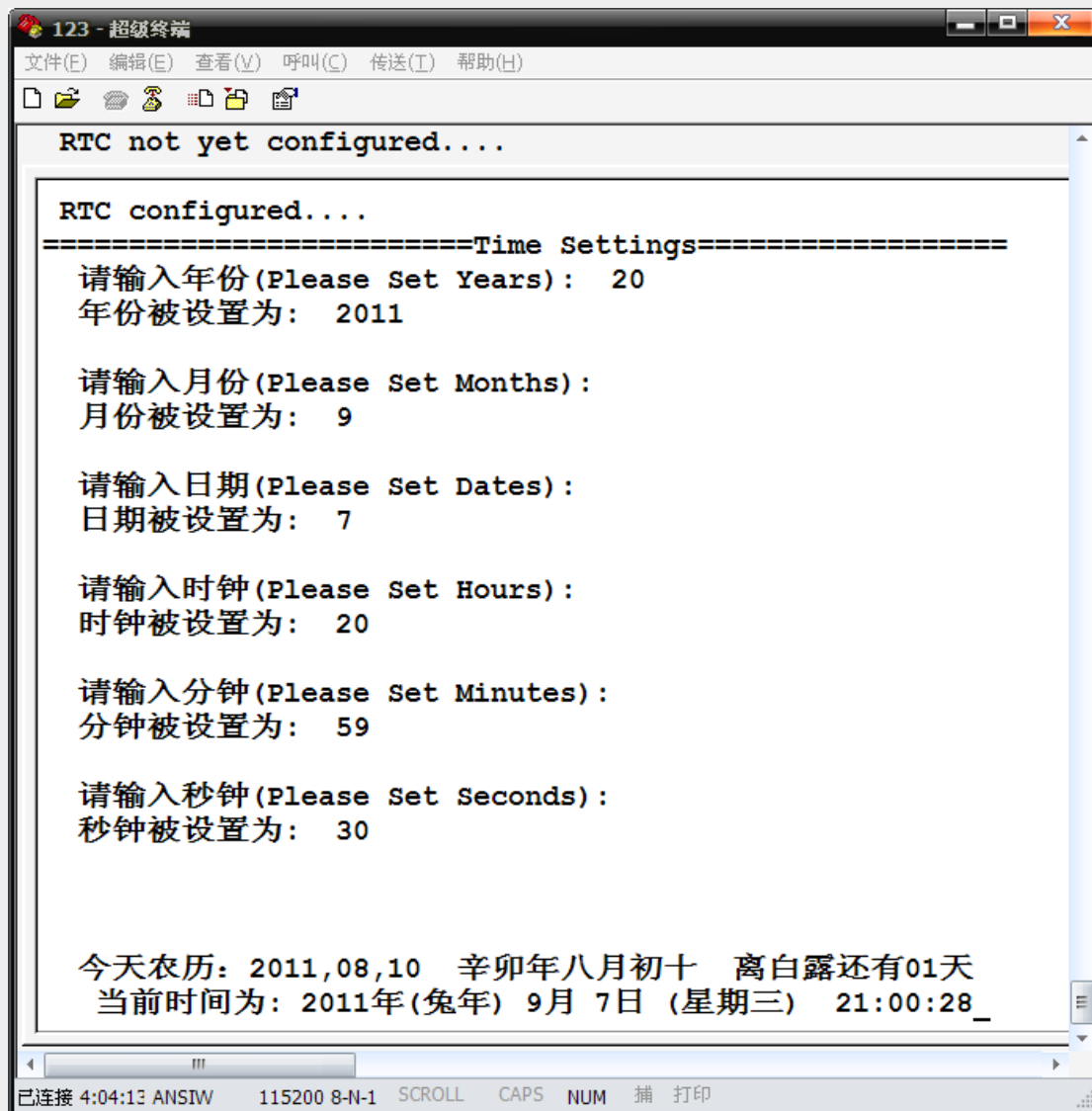
当检测到系统按键复位时，打印出如下信息：



```
123 - 超级终端
文件(E) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
No need to configure RTC....
Time: 22:38:47
This is a RTC demo.....

External Reset occurred....
No need to configure RTC....
Time: 22:38:48
This is a RTC demo.....

External Reset occurred....
No need to configure RTC....
Time: 22:38:53
_
已连接 0:31:20 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印
```



```
123 - 超级终端
文件(E) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
RTC not yet configured....

RTC configured....
=====Time Settings=====
请输入年份(Please Set Years): 20
年份被设置为: 2011

请输入月份(Please Set Months):
月份被设置为: 9

请输入日期(Please Set Dates):
日期被设置为: 7

请输入时钟(Please Set Hours):
时钟被设置为: 20

请输入分钟(Please Set Minutes):
分钟被设置为: 59

请输入秒钟(Please Set Seconds):
秒钟被设置为: 30

今天农历: 2011,08,10 辛卯年八月初十 离白露还有01天
当前时间为: 2011年(兔年) 9月 7日 (星期三) 21:00:28_
已连接 4:04:13 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印
```

(3) 条件编译选项部分问我们是否需要 `output RTCCLK/64 on`

`Tamperpin`, (在 `rtc.c` 中实现) 因为 RTC 可以在 `PC13` 这个引脚输出时钟信号, 这个时钟信号可以作为其他外设的时钟。野火 STM32 开发板中没用到这个时钟信号, 所以我们没定义 `RTCClockOutput_Enable` 这个宏。假如用户需要用到这个时钟信号的话, 只需在头文件中定义 `RTCClockOutput_Enable` 这个宏即可。

(4) 一切就绪之后, 我们调用 `Time_Show();` 函数将我们的时间显示在电脑的超级终端上。`Time_Show();` 在 `rtc.c` 中实现:

```
1.  /*
2.  * 函数名: Time_Show
3.  * 描述   : 在超级终端中显示当前时间值
4.  * 输入   : 无
5.  * 输出   : 无
6.  * 调用   : 外部调用
7.  */
8.  void Time_Show(void)
9.  {
10.     printf("\n\r");
11.
12.     /* Infinite loop */
13.     while (1)
14.     {
15.         /* If 1s has passed */
16.         if (TimeDisplay == 1)
17.         {
18.             /* Display current time */
19.             Time_Display(RTC_GetCounter());
20.             TimeDisplay = 0;
21.         }
22.     }
23. }
```

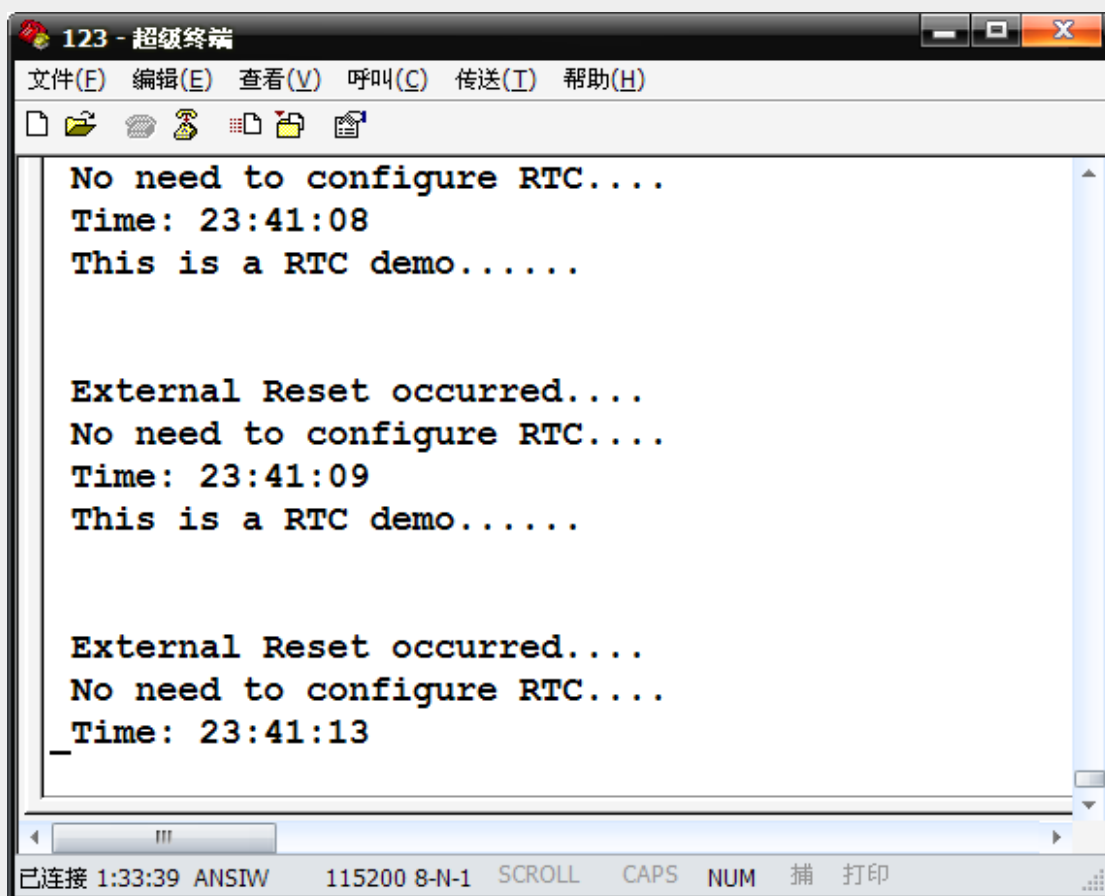
其中 `TimeDisplay` 是 RTC 秒中断标志, 当 RTC 秒中断一次的话, RTC 时间计数器就实现秒加 1, 函数 `Time_Display(RTC_GetCounter());` 就将这些时间值转换成 HH:MM:SS 的格式显示出来, 时间更新显示完之后 `TimeDisplay` 清 0。 `TimeDisplay` 在 RTC 秒中断服务程序中置位:

```
1.  /**
2.  * @brief This function handles RTC global interrupt request.
3.  * @param None
4.  * @retval : None
5.  */
6.  void RTC_IRQHandler(void)
7.  {
8.     if (RTC_GetITStatus(RTC_IT_SEC) != RESET)
9.     {
10.        /* Clear the RTC Second interrupt */
11.        RTC_ClearITPendingBit(RTC_IT_SEC);
12.
13.        /* Toggle GPIO_LED pin 6 each 1s */
```



```
14. //GPIO_WriteBit(GPIO_LED, GPIO_Pin_6, (BitAction)(1 -
    GPIO_ReadOutputDataBit(GPIO_LED, GPIO_Pin_6)));
15.
16. /* Enable time update */
17. TimeDisplay = 1;
18.
19. /* Wait until last write operation on RTC registers has finished */
20. RTC_WaitForLastTask();
21. /* Reset RTC Counter when Time is 23:59:59 */
22. if (RTC_GetCounter() == 0x00015180)
23. {
24.     RTC_SetCounter(0x0);
25.     /* Wait until last write operation on RTC registers has finished */
26.     RTC_WaitForLastTask();
27. }
28. }
29. }
```

现在我电脑的时间是 23: 41: 13。我们把它写到 RTC 的寄存器中，然后在超级终端上显示出来，效果图如下：



过瘾哩，以前我们想实现个时钟的时候还需借助时钟芯片，如 DS1302 或 DS12C887，而现在我们用一个定时器就搞定。不过我们接下来来点更过瘾的，只用 RTC 这个定时器 实现我们的超级日历，日历包括如下功能：

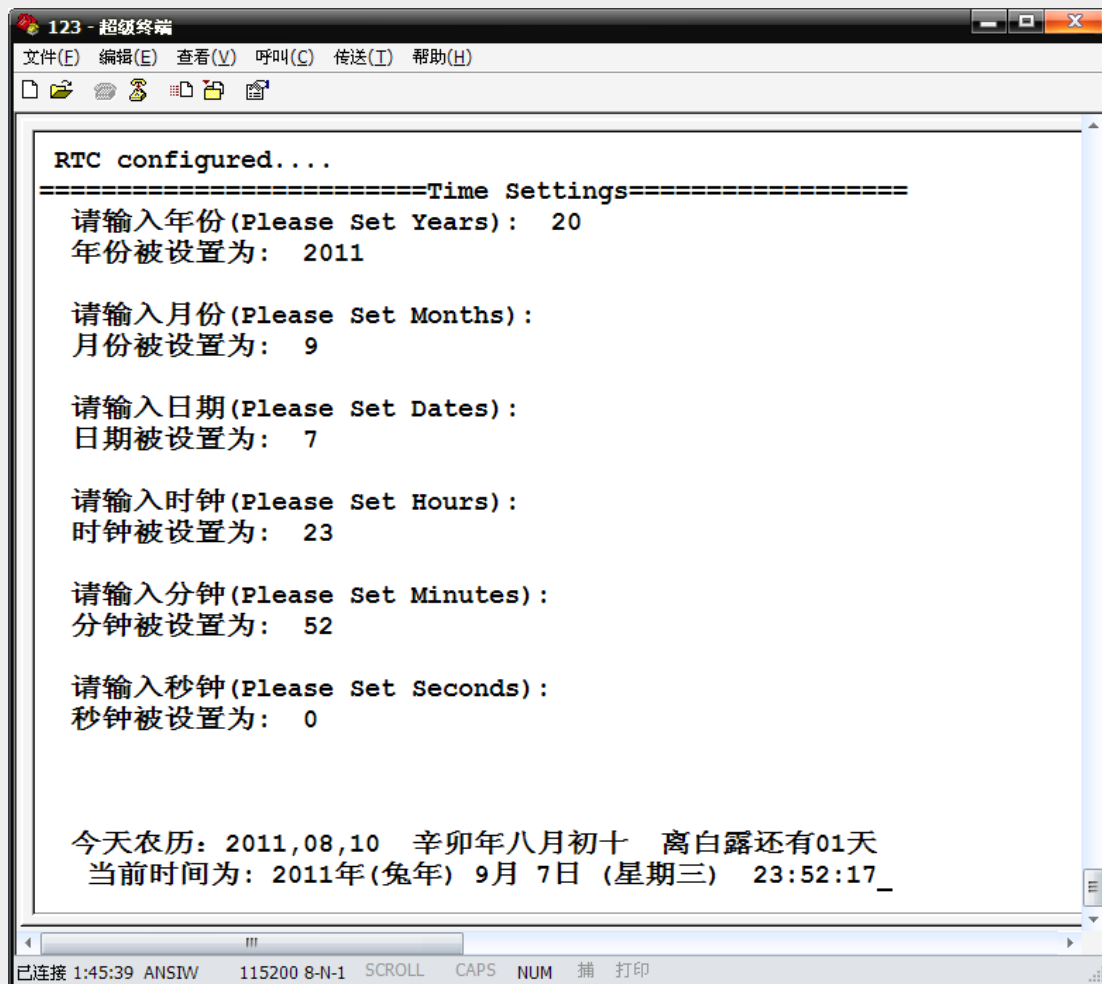
时钟： 如 23: 40: 50

阴历： 如 辛卯年八月初十

阳历： 如 2011-9-7

年份： 如 兔年

24 节气：如 夏至



这个日历以 1970 年为计时元年，用 32bit 的时间寄存器可以运行到 2100 年左右。之所以以 1970 年为计时元年是这份代码是从 LINUX 里面移植过来的，而 LINUX 的诞辰就是 1970 年，我想这样做应该为为了纪念 LINUX (纯属个人观点，没有考证)。不过计时起始元年可调，可调到随便纪念谁:))。

这里就暂且给出个效果图先，分析源码的任务就给大家了。里面涉及到些算法和结构体的知识，大家就好好琢磨吧。源码在野火 STM32 开发板的光盘目录下《11-野火 M3-Calendar》。

