BAT32G137（库函数版本）

Rev 1.0

修订历史

版本	日期	修订人	修订内容
Rev1.1	22.9.22	缪勤文 张刚	

目 录

1. 前言

实时时钟具有以下功能：1、年、月、日、小时分钟和秒计数器，最长计数到 99 年；2、固定周期中断功能（周期有 0.5s、1s、1min、1hour、1day、1month）；3、闹钟中断功能（闹钟：星期、小时、分钟）；4、1Hz 引脚输出功能

2. 实时时钟结构

对于 RTC 实时时钟来说：

只有选择 fmx/fhoco 的分周时钟为（32.768KHZ）或者副系统时钟 Fsub=32.768KHz 作为实时时钟的运行时钟，才可以进行年月日小时分钟和秒计数功能；

选择低速内部振荡器时钟 FIL =15kHz 时，只能使用固定周期中断功能；

3. 中微 RTC 实时时钟应用库简介

中微 BAT32G137 系类软件 RTC 应用库是一个便于移植的标准库代码风格，用户只需要对软件接口相关参数进行简单配置、以及封装接口函数调用即可实现所需功能，节约时间，提高开发效率。

使用方式：

需要将应用层 rtc_demo.c rtc_demo.h 驱动层 rtc.c rtc.h、gpio.c gpio.h、isr.c isr.h 加入到工程中去。

3.1.应用例程使用

包括 RTC 初始化，以及 RTC 相关接口

3.1.1. RTC 初始化

```
1. void Rtc_Init()
2. {
3.  RTC_InitTypeDef RTC_InitStructure = {0};
4.     GPIO_InitTypeDef GPIO_InitStruct ={0};
5.
```

```
6.  /*RTC1HZ OUTPUT CONFIG*/
7.  GPIO_InitStruct.GPIO_Pin    = GPIO_Pin_0;
8.  GPIO_InitStruct.GPIO_Mode   = GPIO_Mode_OUT;
9.  GPIO_InitStruct.GPIO_Level   = GPIO_Level_LOW;
10. GPIO_InitStruct.GPIO_Ctrl  = GPIO_Control_DIG;
11. GPIO_Init(GPIO_PORT3,&GPIO_InitStruct);
12.
13. RTC_InitStructure.RTC_Clk = RTC_FSUB; //选择副系统时钟
14. RTC_InitStructure.RTC_Time.RTC_Seconds = 20;
15. RTC_InitStructure.RTC_Time.RTC_Minutes = 30;
16. RTC_InitStructure.RTC_Time.RTC_Hours = 8;          //上午8点30分20
    秒
17. RTC_InitStructure.RTC_Time.RTC_HourFormat = RTC_HourFormat_12;//RTC
    _HourFormat_24
18.
19. RTC_InitStructure.RTC_Date.RTC_Day = 22;// 22
20. RTC_InitStructure.RTC_Date.RTC_WeekDay = SUNDAY;//周日
21. RTC_InitStructure.RTC_Date.RTC_Month = 5;   //5月
22. RTC_InitStructure.RTC_Date.RTC_Year = 22; //2022年
23. RTC_InitStructure.RTC_Period = One_Second; //RTC 计数每隔1s产生固定
    周期中断
24.
25. RTC_InitStructure.RTC_Alarm_Onoff = RTC_Alarm_Off;//闹钟中断关闭
26. RTC_InitStructure.RTC_Alarm.Alarm_Minute = 35; //闹钟分钟计数 35分
27. RTC_InitStructure.RTC_Alarm.Alarm_Hour  = 11;//闹钟小时计数 上午11点
    0x11
28. RTC_InitStructure.RTC_Alarm.Alarm_Week  = ALARM_WEEK(SUNDAY)|ALARM_
    WEEK(SATURDAY);//闹钟星期计数
29. RTC_InitStructure.RTC_1HZ_Output = ENABLE; //RTC1HZ diable output
30. RTC_Init(&RTC_InitStructure);
31.
32. ISR_Register(RTC_IRQn,rtc_interrupt);      //RTC 中断服务路径注册
33.
34. RTC_Start();
35. RTC_Set_AlarmOn();
36.}
```

- 配置 RTC 1Hz 引脚输出的 GPIO
- 配置 RTC 运行时钟选择，使用副系统时钟或者 32.768KHz 的频率，才能进行年月日小时分钟和秒计数功能
- 配置时间，日期
- 配置计数每秒产生中断，（可以修改半秒、一分钟、一小时、一个月）
- 配置闹钟中断是否打开
- 使能 RTC 1Hz 功能

➢ 中断服务函数注册

注意：1、在配置时候，使用 10 进制进行配置；2、配置时间格式，可以选择 12 小时格式或者 24 小时格式，选择 12 小时格式之后，需要指定是上午还是下午。3、闹钟中断和固定周期中断使用的是同一中断号，通过 RTCC1 中断状态标志位（bit4 bit3）来区分是闹钟一致中断，还是固定周期中断。

3.1.2. 设置 RTC 时间

设置时分秒：

```
1. /**
2.   * @brief  Set the RTC current time.
3.   * @param  RTC_TimeStruct: pointer to a RTC_TimeTypeDef structure t
   hat contains
4.   *                        the time configuration information for th
   e RTC.
5.   * @retval None
6.   */
7. void RTC_SetTime(RTC_TimeTypeDef* RTC_TimeStruct)
8.
```

设置时分秒和 am 还是 pm 计数；具体的时间参数填入 RTC_TimeStruct 结构体中。

设置日期

```
1. /**
2.   * @brief  Set the RTC current date.
3.   * @param  RTC_DateStruct: pointer to a RTC_DateTypeDef structure t
   hat contains
4.   *                         the date configuration information for t
   he RTC.
5.   * @retval None
6.   */
7. void RTC_SetDate(RTC_DateTypeDef* RTC_DateStruct)
```

设置年月日以及星期几

3.1.3. 获取 RTC 时间

获取当前具体的日期以及具体时间，精确到秒

```
1. /**
2.   * @brief  Get the RTC current date and time.
3.   * @param  RTC_DateStruct: pointer to a RTC_DateTypeDef structure t
   hat contains
4.   *                        the date configuration information for t
   he RTC.
5.   * @retval None
6.   */
7.
8. void RTC_Get_CounterValue(RTC_CounterTypeDef *counter_val)
```

3.1.4. 设置 RTC 闹钟时间

设置闹钟具体的事件包括：分钟、小时、星期几

```
1. /**
2.   * @brief  Set the specified RTC Alarm.
3.   * @note
4.   * @param  RTC_AlarmStruct: pointer to a RTC_AlarmTypeDef structure
    that
5.   *                        contains the alarm configuration parame
   ters.
6.   * @retval None
7.   */
8. void RTC_SetAlarm( RTC_Alarm_t * RTC_AlarmStruct)
```

对于闹钟设置星期几来说；其寄存器如下

这是设定闹钟星期的寄存器。

通过8位存储器操作指令设定ALARMWW寄存器。在产生复位信号后，此寄存器的值变为“00H”。

图7-16 闹钟星期寄存器（ALARMWW）的格式

地址：0x40044F5CH　复位后：00H　R/W

符号	7	6	5	4	3	2	1	0
ALARMWW	0	WW6	WW5	WW4	WW3	WW2	WW1	WW0

Bit6~bit0 都置为 1 表示，一周闹钟都生效；

因此在配置时：配置参数 `ALARM_WEEK(SUNDAY)` 可以同时”|”或上多个参数

```
RTC_InitStructure.RTC_Alarm.Alarm_Week  = ALARM_WEEK(SUNDAY)|ALARM_WEEK
(SATURDAY);//闹钟星期计数
```

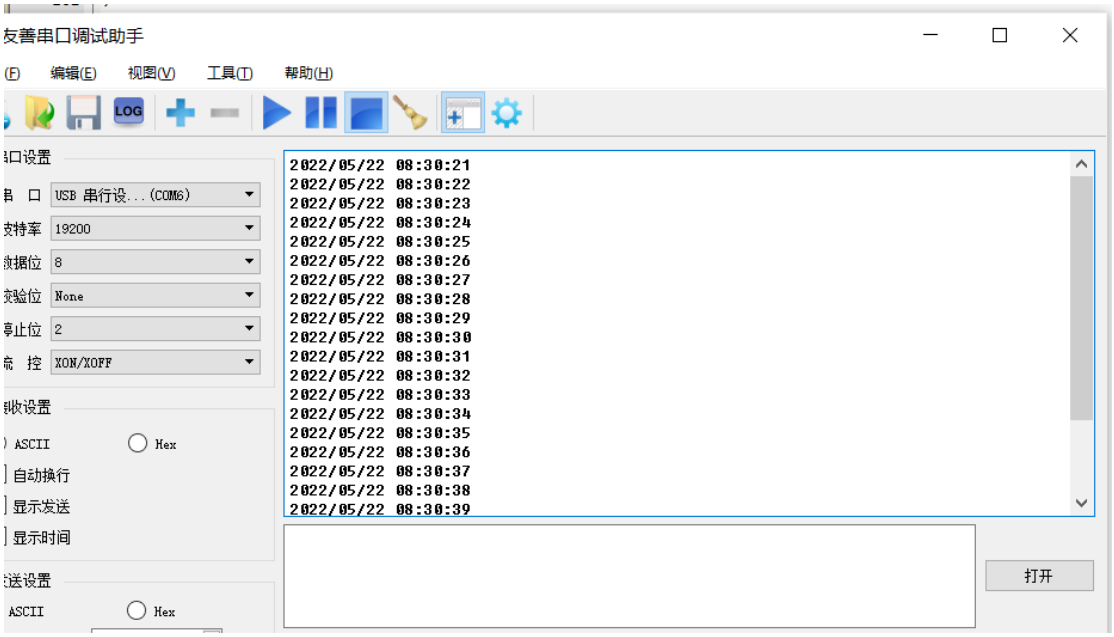
3.1.5. 获取 RTC 闹钟事件

```
1. /**
2.    * @brief  Get the specified RTC Alarm Value.
3.    * @note
4.    * @param  alarm_val: pointer to a RTC_Alarm_t structure that
5.    *                          contains the alarm configuration parame
   ters.
6.    * @retval None
7.    */
8. void RTC_Get_AlarmValue(RTC_Alarm_t * alarm_val)
```

获取具体设定的闹钟事件，结果存放在指向 alarm_val 的地址中

4. 示例演示

在例程使用演示了产生固定时间中断，每一秒产生一次计数中断



图一 RTC 固定周期中断