

32 位微控制器

HC32F120 系列的实时时钟 RTC

适用对象

系列	产品型号
HC32F120	HC32F120H8TA
	HC32F120F8TA
	HC32F120H6TA
	HC32F120F6TA

目 录

1	摘要	3
2	RTC 简介	3
2.1	主要特性	3
2.2	基本框图	4
3	HC32F120 系列的 RTC.....	5
3.1	复位设置	5
3.2	日历功能	5
3.3	闹钟功能	5
3.4	时钟补偿	6
3.4.1	无补偿 1Hz 输出	6
3.4.2	分布式补偿 1Hz 输出	6
3.5	寄存器介绍.....	7
4	样例代码	8
4.1	代码介绍	8
4.2	工作流程	12
4.3	代码运行	13
5	总结	14
6	版本信息 & 联系方式	15

1 摘要

本篇应用笔记主要介绍 HC32F120 系列的实时时钟（RTC）模块，并简要说明 RTC 的闹钟功能如何实现。

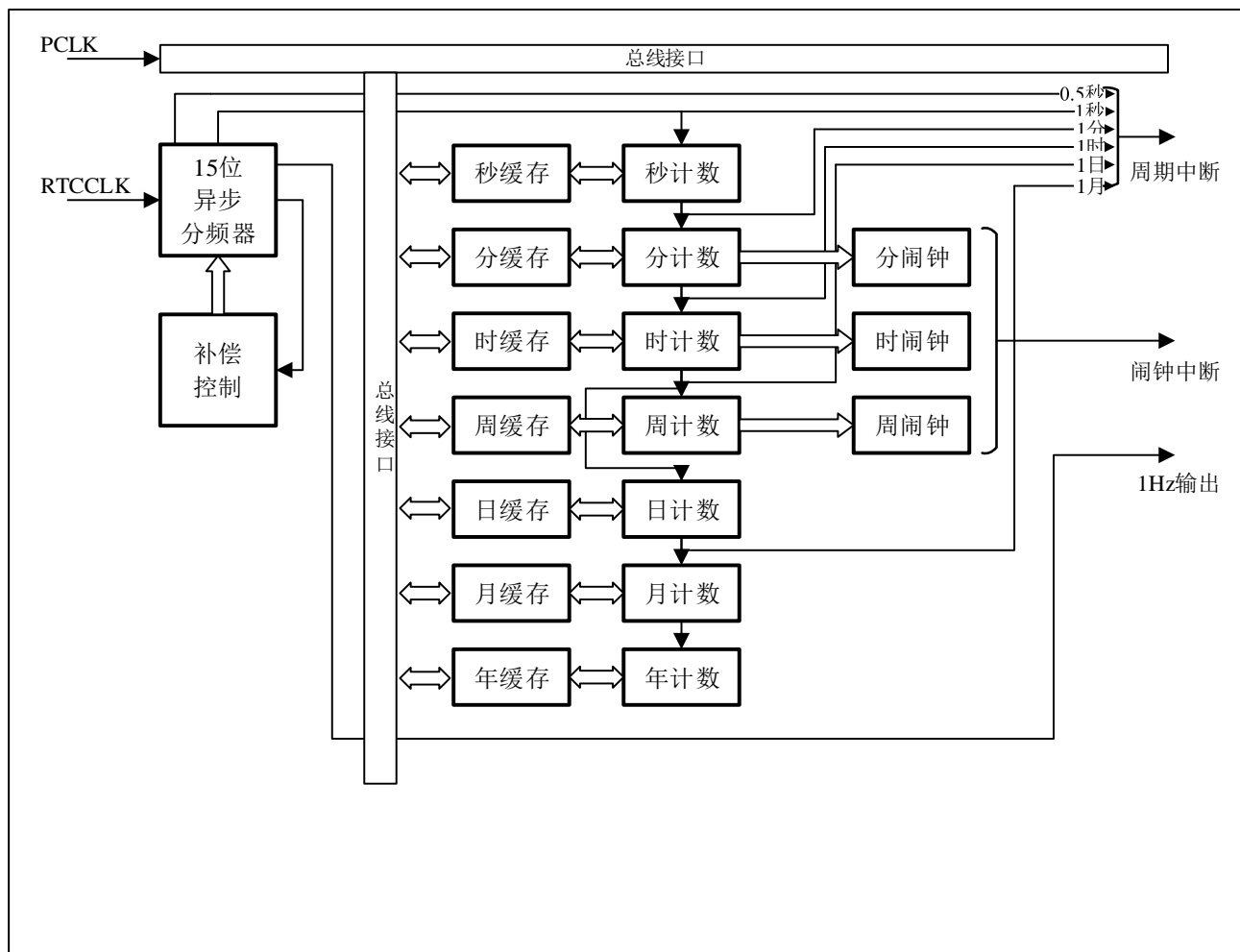
2 RTC 简介

实时时钟（RTC）是一个以 BCD 码格式保存时间信息的计数器，记录从 00 年到 99 年间的具体日历时间；支持 12/24 小时两种时制，根据月份和年份自动计算日数 28、29（闰年）、30 和 31 日。

2.1 主要特性

- BCD 码表示秒、分、时、日、周、月、年时间
- 软件启动或停止
- 12/24 时制可选、闰年自动识别
- 可编程闹钟
- 1Hz 时钟输出
- 时钟误差补偿功能
- 中断
 - 定周期中断
 - 闹钟中断

2.2 基本框图



3 HC32F120 系列的 RTC

3.1 复位设置

RTC 寄存器通过上电复位和设定控制寄存器 RESET 位复位所有寄存器，RTC 启动之后，其他外部各种复位请求都不能复位 RTC，RTC 会一直处于工作状态，可以设定控制寄存器的 START 位为“0”停止 RTC 工作。

上电后，除 RTC_CR0 以外的所有寄存器复位；也可以设定 RTC_CR0.RESET=0，确认 RESET 位为“0”后，设定 RTC_CR0.RESET=1，复位所有寄存器。

3.2 日历功能

在 RTC_CR1.START=0 时可以直接配置日历寄存器，之后设定 RTC_CR1.START=1，开始计数。

在 RTC_CR1.START=1 后，需要先成功切换到读写模式后再对日历寄存器进行操作，具体方式为设定 RTC_CR2.WAIT=1，直到查询 RTC_CR2.WAITF=1 后再读、写日历寄存器，然后设定 RTC_CR2.WAIT=0，直到查询 RTC_CR2.WAITF=0 后退出读写模式。

3.3 闹钟功能

RTC 闹钟功能主要使用周、时、分三个参数作为闹钟数据源，可设置某一时间段在一周的任意一天或几天进行闹钟提示。

闹钟中断 RTC_ALM，在控制寄存器 2 (RTC_CR2) 的 ALMIE=1 并且控制寄存器 2

(RTC_CR2) 的 ALME=1 时，若当前日历时间与分闹钟寄存器 (RTC_ALMMIN)、时闹钟寄存器 (RTC_ALMHOUR)、周闹钟寄存器 (RTC_ALMWEED) 相等时，触发闹钟中断。

3.4 时钟补偿

由于外部副发振晶振在各种温度条件下工作存在误差，在需要得到高精度的计数结果时，需要对误差进行补偿，可补偿范围-275.5ppm~+212.9ppm，最小分辨率 0.96ppm。

RTC 可输出 1Hz 时钟，提供两种精度输出方式：无时钟误差补偿的普通精度 1Hz 输出、每 32 秒内平均补偿的分布式补偿 1Hz 输出。

3.4.1 无补偿 1Hz 输出

当时钟误差补偿功能有效 `RTC_ERRCRH.COMPEN=1` 时，普通精度的 1Hz 输出设定如下：

- 1) 1Hz 输出引脚设定；
- 2) `RTC_CR1.oneHZOE=1`，时钟输出许可；
- 3) 设定 `RTC_CR1.START=1`，计数开始；
- 4) 等待 2 个计数周期以上，1Hz 输出开始。

3.4.2 分布式补偿 1Hz 输出

当时钟误差补偿功能有效 `RTC_ERRCRH.COMPEN=1` 时可选择分布式补偿 1Hz 输出,分布式补偿 1Hz 输出设定如下：

- 1) 1Hz 输出引脚设定；
- 2) `RTC_CR1.oneHZOE=1`，时钟输出许可；
- 3) 时钟误差补偿寄存器 `RTC_ERRCRL.COMP[7:0]`与 `RTC_ERRCRH.COMP[8]`补偿数设定；
- 4) 时钟误差补偿寄存器 `RTC_ERRCRH.COMPEN=1`，误差补偿有效；
- 5) 设定 `RTC_CR1.START=1`，计数开始；
- 6) 等待 2 个计数周期以上，1Hz 输出开始。

3.5 寄存器介绍

实时时钟（RTC）模块的寄存器如下表所示，若需了解具体细节，请参考用户手册：

寄存器简称	寄存器功能
CR0	控制寄存器 0
CR1	控制寄存器 1
CR2	控制寄存器 2
CR3	控制寄存器 3
SEC	秒计数寄存器
MIN	分计数寄存器
HOUR	时计数寄存器
WEEK	周计数寄存器
DAY	日计数寄存器
MON	月计数寄存器
YEAR	年计数寄存器
ALMMIN	分闹钟寄存器
ALMHOUR	时闹钟寄存器
ALMWEEK	周闹钟寄存器
ERRCRH	时钟误差补偿寄存器 H
ERRCRL	时钟误差补偿寄存器 L

4 样例代码

4.1 代码介绍

用户可以根据上述的工作流程编写自己的代码来学习验证该模块，也可以直接通过华大半导体的网站下载到 HC32F120 系列 MCU 的设备驱动库（Device Driver Library, DDL）来体验 RTC 闹钟功能。

以下部分主要基于 DDL 的 RTC 模块闹钟功能样例 rtc_alarm 代码，简要介绍 RTC 的使用方法：

1) 开启 XTAL32 时钟

```
/* Configure XTAL32 */
stcXTAL32Init.u8Xtal32State = CLK_XTAL32_ON;
stcXTAL32Init.u8Xtal32Drv = CLK_XTAL32DRV_HIGH;
stcXTAL32Init.u8Xtal32SupDrv = CLK_XTAL32_SUPDRV_OFF;
stcXTAL32Init.u8Xtal32NF = CLK_XTAL32NF_FULL;
CLK_XTAL32Init(&stcXTAL32Init);
```

2) 配置 LED 和 UART:

```
/* LED Port/Pin initialization */
stcGpioInit.u16PinMode = PIN_MODE_OUT;
GPIO_Init(LED_R_PORT, LED_R_PIN, &stcGpioInit);
LED_R_OFF();

/* Configure UART */
DDL_UartInit();
```

3) 初始化 RTC:

```
/* Enable peripheral clock */
CLK_FcgPeriphClockCmd(CLK_FCG_RTC, Enable);

/* Reset RTC counter */
if (ErrorTimeout == RTC_DeInit())
{
    printf("Reset RTC failed!\r\n");
    return;
}

/* Configuration RTC structure */
stcRtcInit.u8ClockSource = RTC_CLOCK_SOURCE_XTAL32;
stcRtcInit.u8HourFormat = RTC_HOUR_FORMAT_12;
stcRtcInit.u8PeriodInterrupt = RTC_PERIOD_INT_ONE_SECOND;
RTC_Init(&stcRtcInit);
```



```
/* Configuration alarm clock time: Monday to Friday, PM 12:00 */
stcRtcAlarmCfg.u8AlarmHour = 0x12;
stcRtcAlarmCfg.u8AlarmMinute = 0x00;
stcRtcAlarmCfg.u8AlarmWeekday = RTC_ALARM_WEEKDAY_MONDAY |
RTC_ALARM_WEEKDAY_TUESDAY | RTC_ALARM_WEEKDAY_WEDNESDAY |
RTC_ALARM_WEEKDAY_THURSDAY | RTC_ALARM_WEEKDAY_FRIDAY;
stcRtcAlarmCfg.u8AlarmAmPm = RTC_HOUR12_AM;
RTC_SetAlarm(RTC_DATA_FORMAT_BCD, &stcRtcAlarmCfg);
RTC_AlarmCmd(Enable);

/* Update date and time */
RTC_CalendarConfig();

/* Configure interrupt of RTC period */
stcIrqRegister.enIntSrc = INT_RTC_PRD;
stcIrqRegister.enIRQn = Int022_IRQn;
stcIrqRegister.pfnCallback = RtcPeriod_IrqCallback;
u8Ret = INTC_IrqRegistration(&stcIrqRegister);
if (Ok != u8Ret)
{
    // check parameter
    while (1);
}

/* Clear pending */
NVIC_ClearPendingIRQ(stcIrqRegister.enIRQn);
/* Set priority */
NVIC_SetPriority(stcIrqRegister.enIRQn, DDL_IRQ_PRIORITY_DEFAULT);
/* Enable NVIC */
NVIC_EnableIRQ(stcIrqRegister.enIRQn);

/* Configure interrupt of RTC alarm */
stcIrqRegister.enIntSrc = INT_RTC_ALM;
stcIrqRegister.enIRQn = Int023_IRQn;
stcIrqRegister.pfnCallback = RtcAlarm_IrqCallback;
u8Ret = INTC_IrqRegistration(&stcIrqRegister);
if (Ok != u8Ret)
{
    // check parameter
    while (1);
}

/* Clear pending */
NVIC_ClearPendingIRQ(stcIrqRegister.enIRQn);
/* Set priority */
NVIC_SetPriority(stcIrqRegister.enIRQn, DDL_IRQ_PRIORITY_DEFAULT);
/* Enable NVIC */
NVIC_EnableIRQ(stcIrqRegister.enIRQn);

/* Enable period and alarm interrupt */
RTC_IntCmd(RTC_INT_PERIOD, Enable);
RTC_IntCmd(RTC_INT_ALARM, Enable);

/* Startup RTC count */
RTC_Cmd(Enable);
```

4) 每秒通过串口打印时间信息和报警信息:

```
if (1u == u8SecIntFlag)
{
    u8SecIntFlag = 0;
    /* Print alarm information */
    if ((1u == u8AlarmIntFlag) && (u8AlarmCnt > 0))
    {
        /* Alarm LED flicker */
        LED_R_TOGGLE();
        u8AlarmCnt--;
        if (0 == u8AlarmCnt)
        {
            u8AlarmIntFlag = 0u;
            LED_R_OFF();
        }
        /* Get alarm date */
        if (Ok == RTC_GetDate(RTC_DATA_FORMAT_BCD, &stcCurrentDate))
        {
            /* Get alarm time */
            if (Ok == RTC_GetTime(RTC_DATA_FORMAT_BCD, &stcCurrentTime))
            {
                /* Print alarm date and time */
                RTC_DisplayDateTime(RTC_DATA_FORMAT_BCD, "Alarm",
                                    &stcCurrentDate, &stcCurrentTime);
            }
            else
            {
                printf("Get alarm time failed!\r\n");
            }
        }
        else
        {
            printf("Get alarm date failed!\r\n");
        }
    }
    /* Print current date and time */
    else
    {
        /* Get current date */
        if (Ok == RTC_GetDate(RTC_DATA_FORMAT_DEC, &stcCurrentDate))
        {
            /* Get current time */
            if (Ok == RTC_GetTime(RTC_DATA_FORMAT_DEC, &stcCurrentTime))
            {
                /* Print current date and time */
                RTC_DisplayDateTime(RTC_DATA_FORMAT_DEC, "Normal",
                                    &stcCurrentDate, &stcCurrentTime);
            }
            else
            {
                printf("Get time failed!\r\n");
            }
        }
        else
        {
            printf("Get date failed!\r\n");
        }
    }
}
```

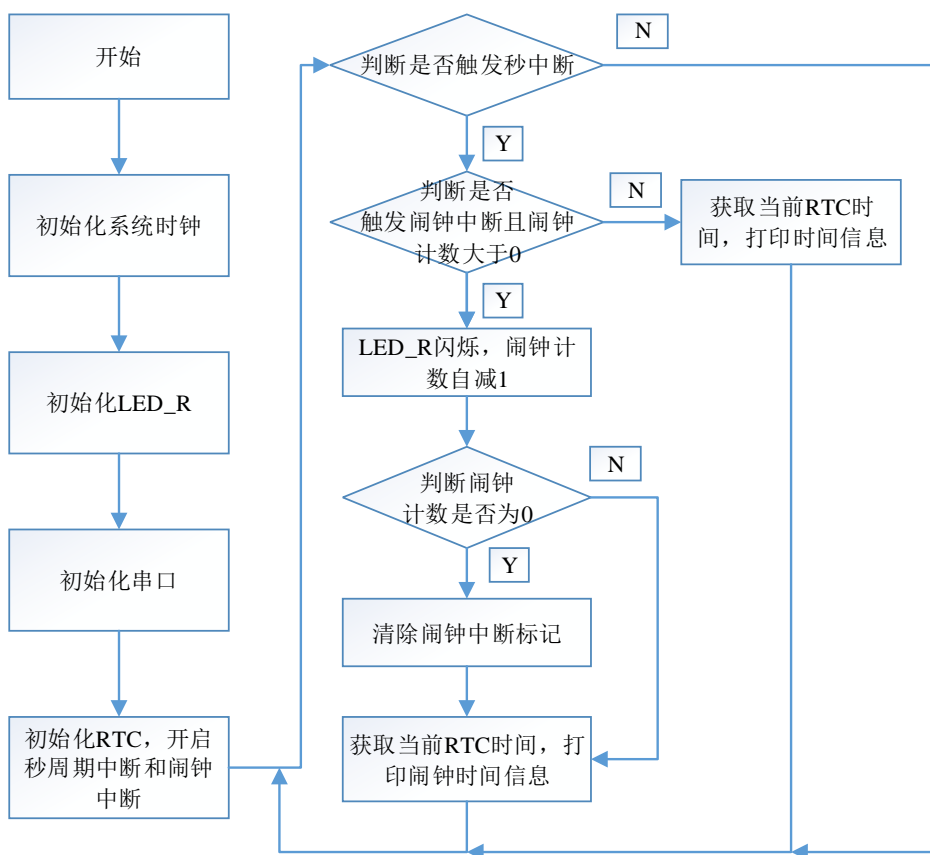
```
}  
}  
}
```

- 5) 每秒周期中断和报警中断处理程序，设置 RTC 为 12 小时时间格式，初始化设置时间为 18 年 10 月 10 日下午 11 点 59 分 55 秒，再设置闹钟时间为周一到周五每天的上午 12 点 0 分，使用 micro USB 连接电脑及目标板 CN1 端口，打开串口调试助手，正常情况下通过串口每秒输出目标板时间信息；到达闹钟时间则通过串口输出闹钟时间信息，并闪烁 LED_R 进行提示，持续 10 秒：

```
/**  
 * @brief RTC period interrupt callback function.  
 * @param None  
 * @retval None  
 */  
void RtcPeriod_IrqCallback(void)  
{  
    u8SecIntFlag = 1u;  
    RTC_ClearIntFlag(RTC_FLAG_PERIOD);  
}  
  
/**  
 * @brief RTC alarm interrupt callback function.  
 * @param None  
 * @retval None  
 */  
void RtcAlarm_IrqCallback(void)  
{  
    u8AlarmCnt = 10u;  
    u8AlarmIntFlag = 1u;  
    RTC_ClearIntFlag(RTC_FLAG_ALARM);  
}
```

4.2 工作流程

样例代码中 RTC 操作流程如下图所示：



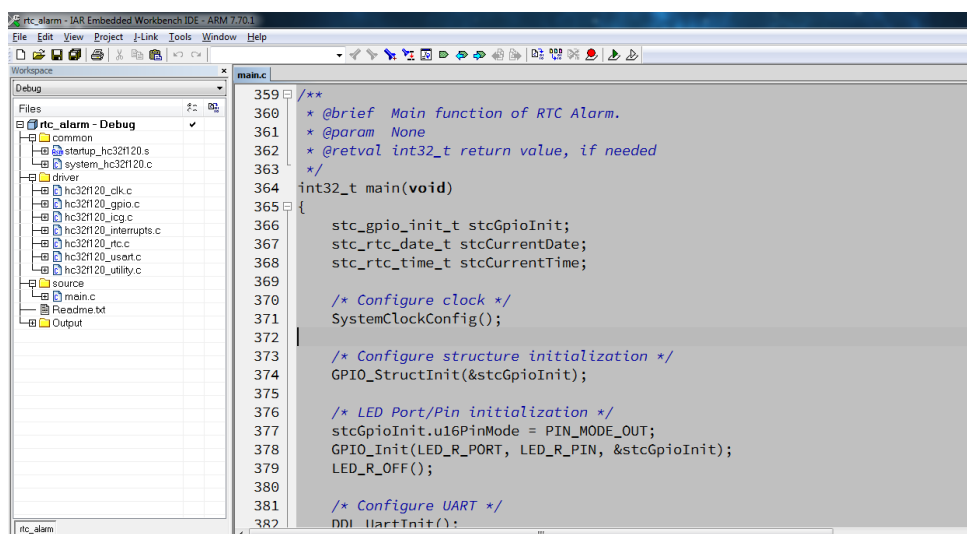
4.3 代码运行



用户可以通过华大半导体的网站下载到 DDL 的样例代码（rtc_alarm、rtc_calendar、rtc_calibration_output、rtc_low_power），并配合评估用板（比如‘STK_HC32F120_LQFP44_080_V11’）运行相关代码学习使用 RTC 模块。

以下部分主要介绍如何在‘STK_HC32F120_LQFP44_080_V11’评估板上，通过 IAR EWARM 编译、运行 rtc_alarm 样例代码并观察结果：

- 确认安装正确的 IAR EWARM v7.7 工具（请从 IAR 官方网站下载相应的安装包，并参考用户手册进行安装）。
- 获取‘STK_HC32F120_LQFP44_080_V11’评估板。
- 从华大半导体网站下载 HC32F120 DDL 代码。
- 下载并运行 rtc\rtc_alarm\中的项目文件：

1) 打开 rtc_alarm\项目，并打开‘main.c’如下视图：



- 2) 点击  重新编译整个项目；
- 3) 点击  将代码下载到评估板上，全速运行；
- 4) 使用 USB 线连接电脑及目标板；
- 5) 观察串口打印的时间信息，并在 5 秒后观察 LED_R 闪烁情况。
 - 波特率：115200
 - 数据位：8

- 校验位: None

- 停止位: 1

6) 在串口调试助手 SecureCRT 上打开 USB 对应串口, 配置端口如下参数:

7) 打开工程并重新编译, 启动 IDE 的下载程序功能, 关闭 IDE 下载界面。

5 总结

以上章节简要介绍 HC32F120 系列的 RTC 寄存器、功能模式、注意事项。演示了如何操作 RTC 的闹钟功能样例代码, 在开发中用户可以根据自己的实际需要使用 RTC 模块。

6 版本信息 & 联系方式

日期	版本	修改记录
2019/10/31	Rev1.0	初版发布



如果您在购买与使用过程中有任何意见或建议，请随时与我们联系。

Email: mcu@hdsc.com.cn

网址: www.hdsc.com.cn

通信地址: 上海市浦东新区中科路 1867 号 A 座 10 层

邮编: 201203

