STM32F407_简易电子时钟(标准库实现)

STM32F407简易电子时钟项目

引言:

在现代生活中，时间的重要性不言而喻。为了方便我们掌握时间，各种各样的电子时钟应运而生。而今天，我将为大家介绍一个基于STM32F407的简易电子时钟项目。通过这个项目，我们可以学习到STM32极为重要的一个外设**RTC 实时时钟**

什么是RTC

RTC (Real Time Clock)：RTC实质是一个掉电后还继续运行的定时器,从定时器的角度来看,相对于通用定时器TIM外设,它的功能十分简单,只有计时功能(也可以触发中断)。 但其高级指出也就在于 掉电之后还可以正常运行。

两个 32 位寄存器包含二进码十进数格式 (BCD) 的秒、分钟、小时（ 12 或 24 小时制）、星期几、日期、月份和年份。此外，还可提供二进制格式的亚秒值。系统可以自动将月份的天数补偿为 28、29（闰年）、30 和 31 天。

上电复位后，所有RTC寄存器都会受到保护，以防止可能的非正常写访问。

无论器件状态如何（运行模式、低功耗模式或处于复位状态），只要电源电压保持在工作范围内，RTC使不会停止工作。

- **本质：计数器**
- **RTC中断是外部中断（EXTI）**
- **当VDD掉电的时候，Vbat可以通过电源--->实时计时**

一，项目概述:

本项目旨在利用STM32F407开发板和相关的硬件模块，设计并制作一个简易的电子时钟。该时钟具备显示当前时间、时间校准模式等功能。

- **KEY0**:进入时间校准模式
- **KEY1**:增加数码管数值
- **KEY2**:保存数据并退出
- **KEY_UP**:位选数码管

按下按键KEY0，进入时间校准模式，数码管上显示四个零，
按下按键KEY1(进入时间校准模式时默认位选最右边数码管),
最右边数码管数值加一，按下KEY_UP键，位选右边第二位数码管，按下KEY2按键，将数值保存并退出。

二、硬件准备：

1. STM32F407开发板：作为整个项目的核心控制器，用于控制各个硬件模块的工作。
2. 数码管：用于显示当前时间。
3. 按键模块：用于设置闹钟和切换显示模式。

三，软件设计

我们的程序主要在`TIMER.c`和`main.c`中编写设计，其他的模块不做讲解，项目文件地址放在文章最后，有需要的博友请自行下载阅读

TIMER模块

在定时器中断模块中初始化通用定时器3（TIM3）的中断处理函数，以及在定时器溢出时获取RTC时间并刷新数码管显示。

首先，定义了一个名为`RTC_TimeTypeDef`的结构体变量`RTC_TimeStruct`

```
RTC_TimeTypeDef RTC_TimeStruct;
```

定义名为`TIM3_init`的函数初始化通用定时器3（TIM3）的中断处理函数，以及在定时器溢出时获取RTC时间并刷新数码管显示

```
void TIM3_init(u16 arr,u16 psc)
{
	TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStructure;
	NVIC_InitTypeDef NVIC_InitStructure;

	RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3,ENABLE);

	TIM_TimeBaseInitStructure.TIM_ClockDivision = TIM_CKD_DIV1;
	TIM_TimeBaseInitStructure.TIM_CounterMode = TIM_CounterMode_Up;//
	TIM_TimeBaseInitStructure.TIM_Period = arr;
	TIM_TimeBaseInitStructure.TIM_Prescaler = psc;
	TIM_TimeBaseInit(TIM3,&TIM_TimeBaseInitStructure);//

	TIM_ITConfig(TIM3,TIM_IT_Update,ENABLE);
	TIM_Cmd(TIM3,ENABLE);

	NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn;
	NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
	NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x01;
	NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x01;
	NVIC_Init(&NVIC_InitStructure);
}
```

之后在封装一个定时器中断处理函数，用于处理TIM3定时器的更新中断。当定时器溢出时，该函数会被调用。

函数首先通过 `TIM_GetITStatus` 函数检查TIM3定时器是否发生了更新中断。如果发生了中断，函数会执行以下操作：

1. 调用 `RTC_GetTime` 函数获取当前的时间信息，并将其存储在 `RTC_TimeStruct` 结构体中。这里使用了二进制格式（`RTC_Format_BIN`）来获取时间。
2. 根据获取到的秒数和分钟数，计算出对应的数码管显示数字，分别赋值给变量 `t_ge`、`t_shi`、`t_bai` 和 `t_qian`。
3. 调用 `seg_cnt` 函数开始刷新数码管显示。

最后，函数使用 `TIM_ClearITPendingBit` 函数清除TIM3定时器的更新中断标志位，以确保下一次中断发生时能够正确处理。

```
void TIM3_IRQHandler(void)
{
        if(TIM_GetITStatus(TIM3,TIM_IT_Update)==SET)
        {

                RTC_GetTime(RTC_Format_BIN,&RTC_TimeStruct);  //获取RTC时间
                t_ge  = seg_8[RTC_TimeStruct.RTC_Seconds % 10];
                t_shi = seg_8[RTC_TimeStruct.RTC_Seconds / 10];
                t_bai = seg_8[RTC_TimeStruct.RTC_Minutes % 10];
                t_qian = seg_8[RTC_TimeStruct.RTC_Minutes / 10];
                seg_cnt();    //开始刷新数码管
        }
        TIM_ClearITPendingBit(TIM3,TIM_IT_Update);  //清除中断标志位
}
```

main主程序

首先包含所有模块的头文件

```
#include "SEG_8.h"  // 包含数码管显示驱动头文件
#include "sys.h"    // 包含系统相关头文件
#include "delay.h"   // 包含延时函数头文件
#include "timer.h"   // 包含定时器头文件
#include "rtc.h"     // 包含实时时钟头文件
#include "key.h"     // 包含按键输入头文件
```

main（）函数中：

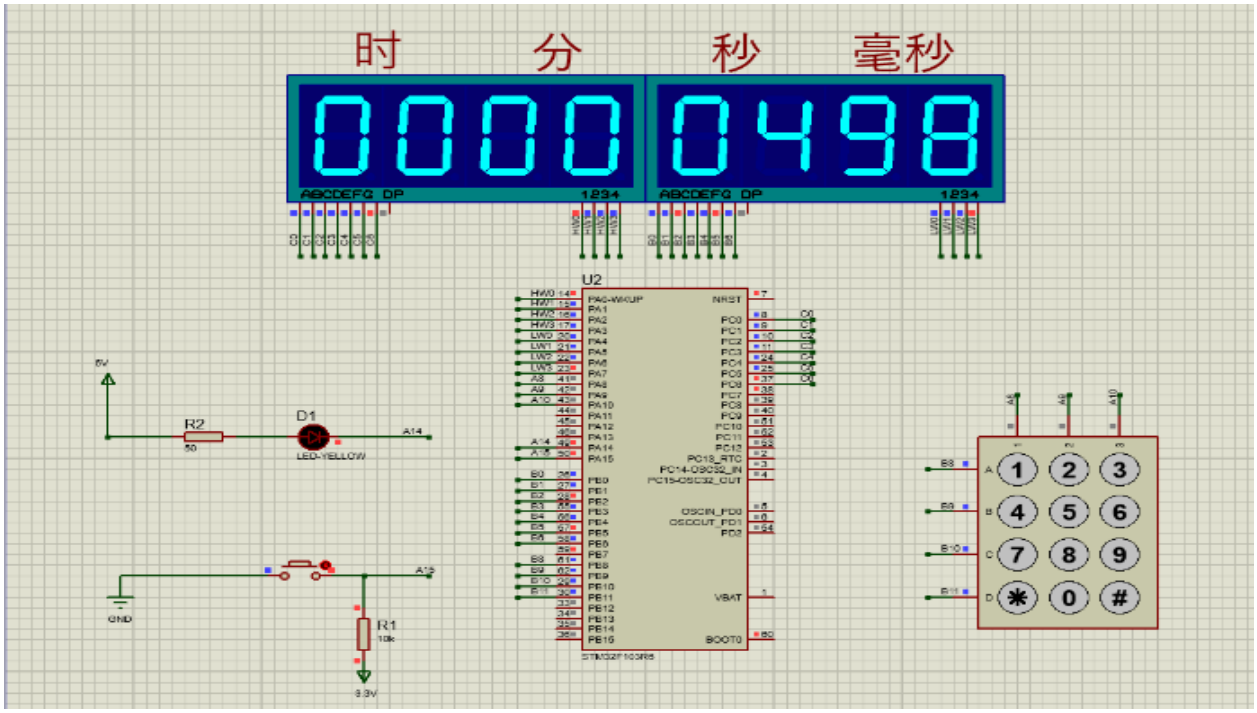
1. 初始化延时函数
2. 初始化数码管显示模块。
3. 初始化按键输入模块。
4. 初始化定时器，设置定时周期为2ms，在中断中刷新数码管。
5. 初始化实时时钟模块。

6. 进入一个无限循环，不断检测按键输入，如果按下某个键，则调用时间校准函数。同时，为了避免过于频繁的按键检测，每次检测后会延时5ms。

```
int main(void)
{
    delay_init(168);  // 初始化延时函数，
    SEG_Init();       // 初始化数码管显示模块
    KEY_Init();       // 初始化按键输入模块
    TIM3_init(20-1,8400-1);  // 初始化定时器，设置定时周期为2ms，在中断中刷新数码管
    My_RTC_Init();          // 初始化实时时钟模块

    while(1)
    {
        if(KEY_SCAN() == 1)  // 检测按键输入，如果按下某个键
            KeyDeal();       // 调用时间校准函数
        delay_ms(5);         // 延时5ms，避免过于频繁的按键检测
    }
}
```

proteus仿真图



STM32F407 简易电子时钟完整项目文件：

> https://github.com/1589326497/STM32F407-Simple-electronic-clock