湖南科技大学计算机科学与工程学院

嵌入式系统及应用课程设计报告

专业班级：21 级计科七班

姓　　名：陈琪琪

学　　号：2102010629

指导教师：黄晶

时　　间：2024.3.3-2024.4.26

地　　点：逸夫楼 537

指导教师评语：
成绩：　　等级： 签名： 年　月　日

目录

一、实验题目

使用 STM32 开发板实现倒计时显示，通过按键控制时钟的延迟计时、暂停和复位

二、实验目的

1. 熟悉 STM32 的 GPIO 配置及使用。
2. 学习和掌握 STM32 的定时器中断及其配置方法。
3. 掌握按键扫描与中断处理机制。
4. 实现一个基本的倒计时器功能，包括时间显示、延迟 5 秒计时、暂停和复位功能。

三、总体设计

3.1 基本原理

1. 定时器中断原理

(1) stm32 提供了 8 个定时器，分为：通用是定时器(TIM2~TIM5)，高级定时器(TIM1 和 TIM8)，基本定时器(TIM6 和 TIM7)。三者的区别如下：

a. TIM1 和 TIM8 定时器的功能包括:

- 16 位向上、向下、向上/下自动装载计数器
- 16 位可编程(可以实时修改)预分频器，计数器时钟频率的分频系数为 1～65535 之间的任意数值
- 多达 4 个独立通道： — 输入捕获 — 输出比较 — PWM 生成(边缘或中间对齐模式) — 单脉冲模式输出
- 死区时间可编程的互补输出
- 使用外部信号控制定时器和定时器互联的同步电路
- 允许在指定数目的计数器周期之后更新定时器寄存器的重复计数器
- 刹车输入信号可以将定时器输出信号置于复位状态或者一个已知状态
- 如下事件发生时产生中断/DMA： — 更新：计数器向上溢出/向下溢出，计数器初始化(通过软件或者内部/外部触发) — 触发事件(计数器启动、停止、初始化或者由内部/外部触发计数) — 输入捕获 — 输出比较— 刹车信号输入
- 支持针对定位的增量(正交)编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

b. TIMx 主要功能通用 TIMx (TIM2、TIM3、TIM4 和 TIM5)定时器功能包括:

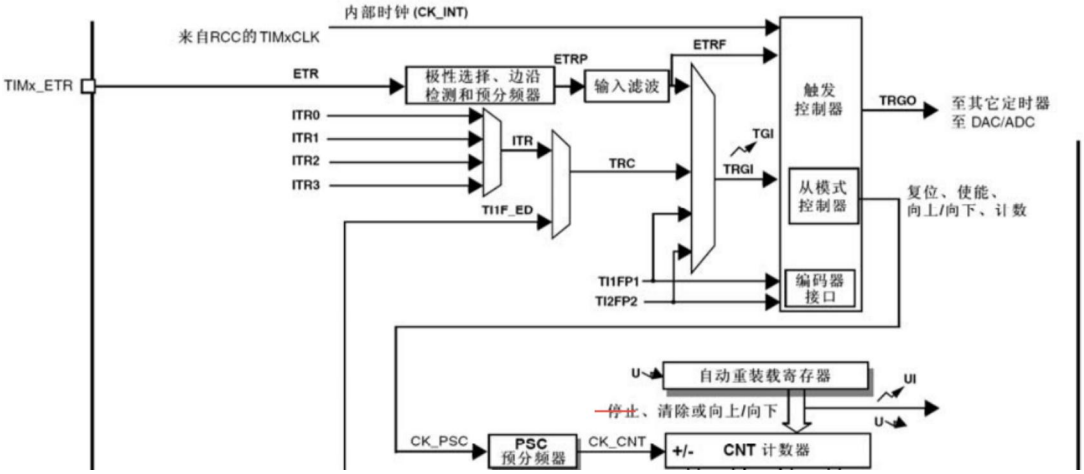
- 16 位向上、向下、向上/向下自动装载计数器
- 16 位可编程(可以实时修改)预分频器，计数器时钟频率的分频系数为 1～65536 之间的任意数值
- 4 个独立通道： — 输入捕获 — 输出比较 — PWM 生成(边缘或中间对齐模式) — 单脉冲模式输出
- 使用外部信号控制定时器和定时器互连的同步电路

- 如下事件发生时产生中断/DMA： — 更新：计数器向上溢出/向下溢出，计数器初始化(通过软件或者内部/外部触发) — 触发事件(计数器启动、停止、初始化或者由内部/外部触发计数) — 输入捕获 — 输出比较
- 支持针对定位的增量(正交)编码器和霍尔传感器电路
- 触发输入作为外部时钟或者按周期的电流管理

c. TIM6 和 TIM7 定时器的主要功能包括:

- 16 位自动重装载累加计数器
- 16 位可编程(可实时修改)预分频器,用于对输入的时钟按系数为 1～65536 之间的任意数值分频
- 触发 DAC 的同步电路　注:此项是 TIM6/7 独有功能。
- 在更新事件(计数器溢出)时产生中断/DMA 请求。

(2) 通用计时器框架图如下：



(3) 计数模式：

a. 向上计数：从 0 计数到预装载值就会产生一个溢出事件，然后继续从 0 开始计数。

b. 向下计数：从预装载值计数到 0 就会产生一个溢出事件，然后继续从预装载值开始计数。

c. 中央对齐：计数器从 0 开始计数到自动装入的值-1，产生一个计数器溢出事件，然后向下计数到 1 并且产生一个计数器溢出事件；然后再从 0 开始重新计数。

(4) 相关寄存器：计数器当前值寄存器 CNT，预分频寄存器 TIMx_PSC，自动重装载寄存器（TIMx_ARR)，控制寄存器 1（TIMx_CR1），DMA 中断使能寄存器（TIMx_DIER）。

2. 按键扫描原理

(1) 键盘扫描

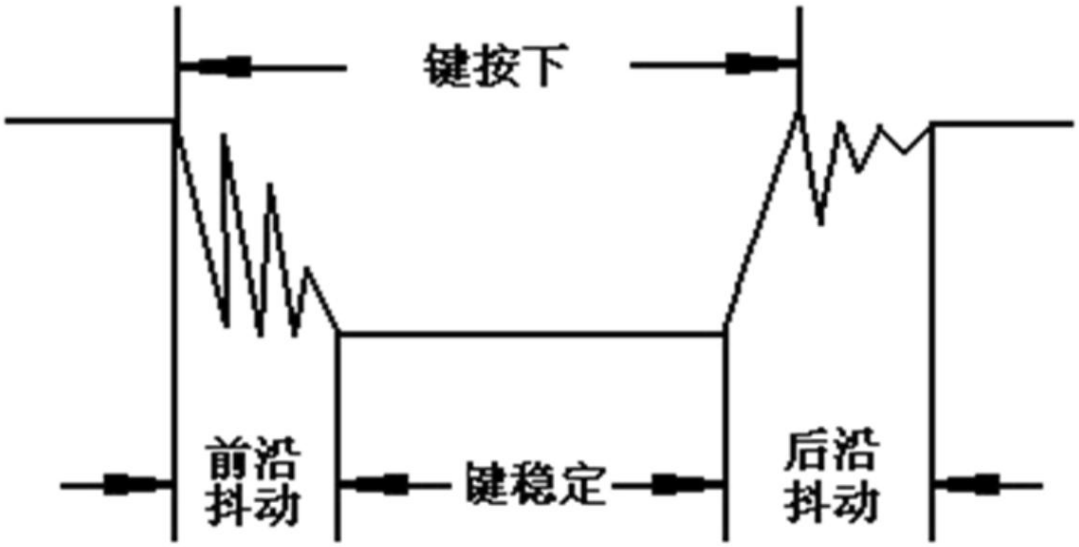
a. 交叉扫描

- 原理：将 I／O 口分成两组，分别作为行和列，形成一个键盘矩阵。扫描每隔一段时间进行一次，如 50ms。扫描时，先将行作为输出，列作为输入。先在第一行输出 L（低电平），其余行输出高电平，读取 N 列的值，如果有 L，则说明在这一列上有按键按下；然后将行变为输入，列变为输出，在该列输出 L，其余列输出 H，读出 M 行的值，哪一行为 L，则说明该行与该列交叉的按键被按下，这样便得到了键的 ID。在扫描过程中，如果有多行或者多列读出来的值为 L，则说明有多个按键按下。
- 特点：键数多，处理复杂。

b. 直接扫描

- 原理：每个端口直接与按键相连依次扫描各端口，直接获取按键值。
- 特点：键数少，处理简单。

(2) 去抖动



a. 按键机械时间—ms 级

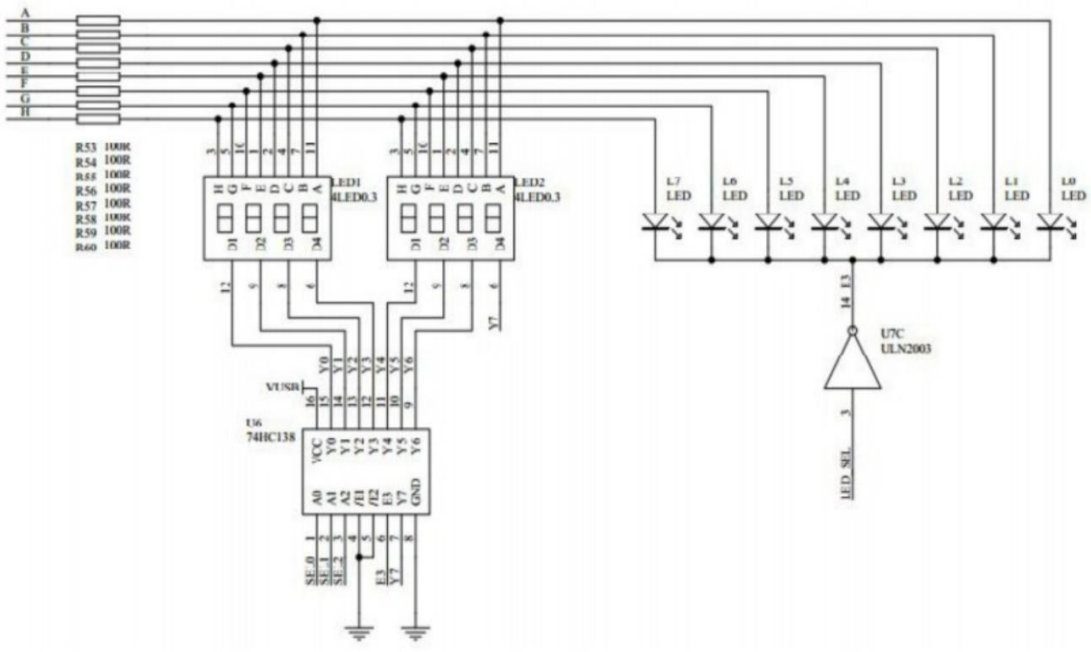
b. 处理时间—us 级

c. 采用软件延时的方法去除抖动

- 检测按键动作，延迟 10ms 再次检测，若仍为动作电平则接受为按键
- 检测到按键释放（端口为高电平）时延迟 10ms 再检测，再对键值进行处理。

3. 数码管显示原理

(1) 原理图



(2) 工作原理

电路中有 L0、L1、L2、L3、L4、L5、L6、L7 共八个发光二极管，当引脚 LED_SEL 输入为 1，对于 A、B、C、D、E、F、G、H 引脚，只要输入为 1，则点亮相连接的发光二极管。

A~H 引脚连接 STM32F103VB 芯片的 PE8~PE15，程序初始化时，对其进行初始设置。引脚 LED_SEL 为 1 时，发光二极管才工作，否则右边的数码管工作。注意，LED_SEL 连接于 PB3，该引脚具有复用功能，在默认状态下，该引脚的 IO 不可用，需对 AFIO_MAPR 寄存器进行设置，设置其为 IO 可用。

(3) 各寄存器的作用

名称	寄存器	意义
端口配置寄存器	GPIOx_CRL GPIOx_CRH	配置 GPIO 工作模式
端口输入数据寄存器	GPIOx_IDR	读取 GPIO 输入状态
端口输出数据寄存器	GPIOx_ODR	控制 GPIO 输出状态
端口位设置/复位寄存器	GPIOx_BSRR	用于位操作 CPIO 的输出状态； 设置端口为 0 或 1
端口位复位寄存器	GPIOx_BRR	用于位操作 GPIO 的输出状态； 设置端口为 0
端口配置锁定寄存器	GPIOx_LCKR	端口锁定后下次系统复位之前 将不能再更改端口位的配置

(4) ODR(端口输出数据寄存器)

LED 0 - 7 的灯分别对应着 ORD 8 - 15 的位置,如果想将某个 LED 点亮，把对应的 ORD 位置为 1 即可。

LED	7	6	5	4	3	2	1	0
ORD	15	14	13	12	11	10	9	8

说明：ORD 端口输入数据寄存器（高 16 位始终为 0，只操作低 16 位的高 8 位）。

- 例如想将 LED 3 点亮，其他灯全灭，则需要将 ORD11 置为 1，其他的置为 0 。此时的 ORD 寄存器的值即为 0X 0000 0800
- 如果想将 LED 5 和 6 点亮，其他灯全灭，则需要将 ORD13 和 14 置为 1，其他的置为 0。此时的 ORD 寄存器的值即为 0X 0000 6000

3.2 功能介绍

1. 倒计时显示：通过数码管显示倒计时的时间，初始化时间为 23：59：59。
2. 启动/暂停：按下 K3 按键可以启动或暂停倒计时器。
3. 复位：按下 K2 按键可以复位时钟到初始值。
4. 延迟倒计时：按下 K1 按键可以延迟倒计时 5 秒，并且 LED 灯上显示延迟时间计数。
5. 按键扫描：通过按键扫描检测用户输入。

3.3 模块介绍

1. GPIO 配置模块：配置 GPIO 口用于数码管显示和按键输入。
2. 定时器模块：配置定时器用于时钟计时，并产生中断。
3. 按键处理模块：实现按键扫描和中断处理。
4. 显示模块：实现数码管显示倒计时初始值。

3.4 设计步骤

1. 初始化系统时钟和定时器。
2. 配置 GPIO 口用于数码管显示和按键输入。
3. 配置定时器用于时钟计时，并使能中断。
4. 实现按键扫描和中断处理函数。
5. 实现数码管显示函数。
6. 在主循环中处理按键输入和更新显示。

四、详细设计

1. 硬件资源
   - STM32 开发板
   - 数码管显示模块
   - 按键输入模块
2. 程序代码

(1) 初始化系统时钟和定时器

```
void TimerxInit(u16 arr, u16 psc) {
    RCC->APB1ENR |= 1<<1;
    TIM3->ARR = arr;
    TIM3->PSC = psc;
    TIM3->DIER |= 1<<0;
    TIM3->CR1 |= 0X01;
```

```
    MY_NVIC_Init(1, 3, TIM3_IRQChannel, 2);
}
```

(2) 配置 GPIO

```
void LED_Init() {
    RCC->APB2ENR |= 1<<0;
    RCC->APB2ENR |= 1<<3;
    RCC->APB2ENR |= 1<<6;
    AFIO->MAPR |= 0x02000000;

    GPIOB->CRL &= 0xFFFF0000;
    GPIOB->CRL |= 0x00003333;
    GPIOB->ODR |= 0x000000FF;

     GPIOE->CRH &= 0X00000000;
     GPIOE->CRH |= 0X33333333;
     GPIOE->ODR |= 0x0000FF00;
}
```

(3) 按键扫描和中断处理

```
u8 KEY_Scan(void) {
    static u8 key_up = 1;
    if(key_up && (KEY1 == 0 || KEY2 == 0 || KEY3 == 0)) {
        delay_ms(10);
        key_up = 0;
        if(KEY1 == 0)
            return 1;
        else if(KEY2 == 0)
            return 2;
         else if(KEY3 == 0)
             return 3;
     } else if(KEY1 == 1 && KEY2 == 1 && KEY3 == 1)
         key_up = 1;
     return 0;
}

void EXTI0_IRQHandler(void) {
    delay_ms(10);
    if(KEY3 == 0) {
        LED7 = !LED7;
    }
```

```
    EXTI->PR = 1<<0;
}
```

(4) 数码管显示

```
void SetLed(u8 w, u8 value) {
    SEL0 = w % 2;
    SEL1 = w / 2 % 2;
    SEL2 = w / 4;
    LedValue(segTable[value]);
}

void DisplayDigitalClock(void) {
    SetLed(0, hour / 10);
    delay_ms(1);
    SetLed(1, hour % 10);
    delay_ms(1);
    SetLed(2, 10);
    delay_ms(1);
    SetLed(3, minute / 10);
    delay_ms(1);
    SetLed(4, minute % 10);
    delay_ms(1);
    SetLed(5, 10);
    delay_ms(1);
    SetLed(6, second / 10);
    delay_ms(1);
    SetLed(7, second % 10);
    delay_ms(1);
}
```

(5) 主程序

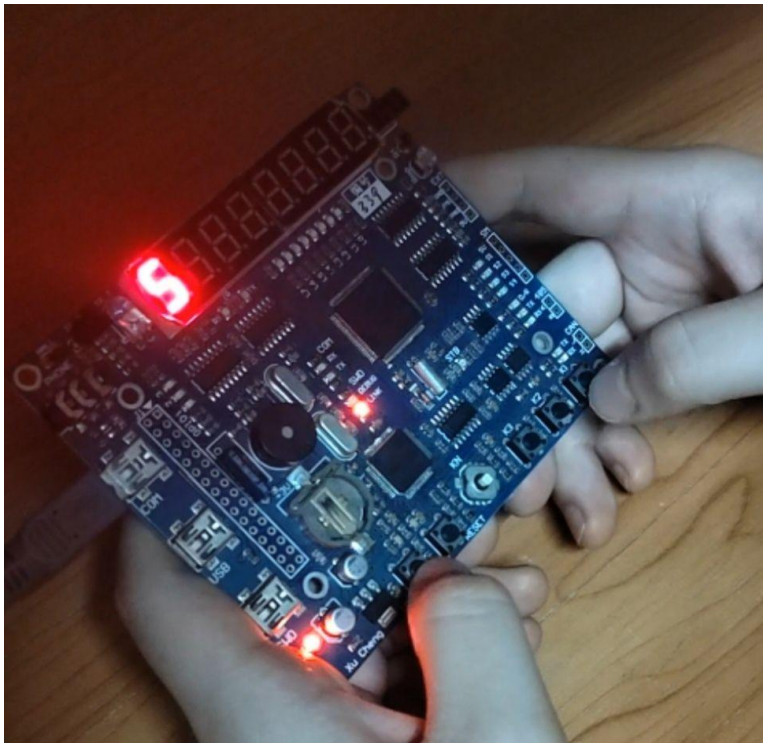
```
int main() {
    u8 t;
    Stm32_Clock_Init(9);
    delay_ms(72);
    TimerxInit(9999, 7199);
    LED_Init();
    LED_SEL = 0;
    KEY_Init();
    bool flag = TRUE;
    while(1) {
        t = KEY_Scan();
        if(t) {
            switch(t) {
```

```
                case 1:
                    TIM3->CR1 |= 0x02;
                    flag = TRUE;
                    SetLed(0, 5);
                    delay_s(10);
                    SetLed(0, 4);
                    delay_s(10);
                    SetLed(0, 3);
                    delay_s(10);
                    SetLed(0, 2);
                    delay_s(10);
                    SetLed(0, 1);
                    delay_s(10);
                    SetLed(0, 0);
                    delay_s(10);
                    break;
                case 2:
                    flag = TRUE;
                    hour = 23;
                    minute = 59;
                    second = 59;
                    break;
                case 3:
                    flag = !flag;
                    TIM3->CR1 |= 0x02;
                    break;
            }
        } else {
            delay_ms(1);
            if (flag) {
                TIM3->CR1 &= 0xFD;
            }
            DisplayDigitalClock();
        }
    }
    return 0;
}
```

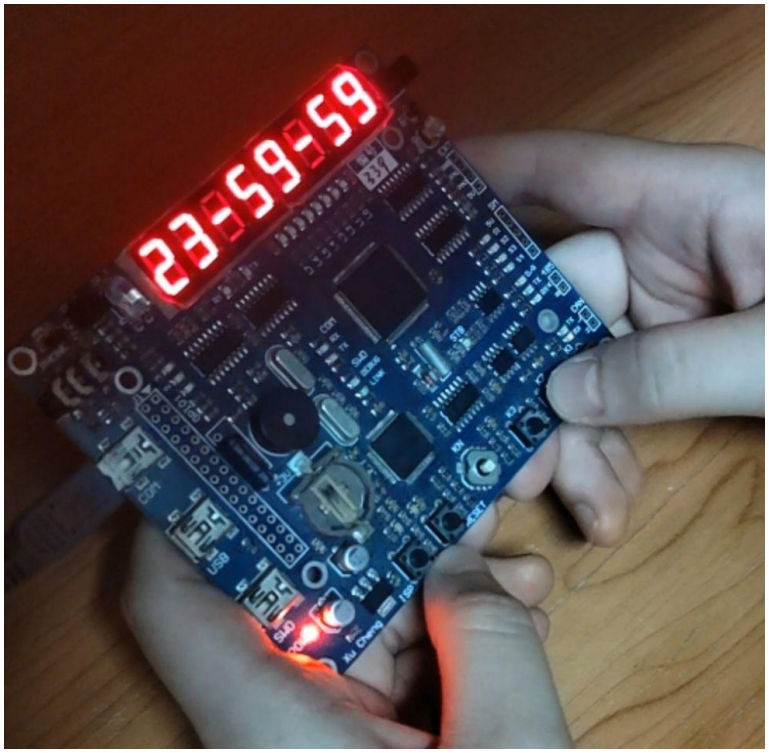
五、实验结果与分析

1. 实验结果

(1) 按下 K1 按键，数码管延迟倒计时 5s，随后继续在原来时间的基础上继续倒计时。



(2) 按下 K2 按键，时间重置为 23:59:59，并重新开始倒计时。



(3) 按下 K3 按键，计时停止，显示冻结在当前时间，再次按下 K3 按键则继续倒计时。

2. 实验分析

(1) 通过定时器中断实现了秒级时间更新。

(2) 通过 GPIO 和按键实现了对时钟的控制。

(3) 数码管正确显示倒计时，功能实现符合预期。

六、小结与心得体会

1. 小结

本实验通过 STM32 开发板实现了一个倒计时器功能，利用定时器和 GPIO 模块实现时钟计时和数码管显示，并通过按键控制时钟的停止计时 5s、复位和暂停。实验过程中，掌握了 STM32 的定时器和 GPIO 配置方法，以及按键扫描和中断处理的实现。通过实验，加深了对嵌入式系统开发的理解，提高了实践能力。同时，实验中遇到的问题和解决方法也增强了对嵌入式系统调试和优化的经验，为以后的项目开发奠定了基础。

2. 心得体会

(1) 在实际开发过程中，定时器和中断的正确配置非常关键，必须仔细调试。

(2) 数码管的显示需要注意刷新频率，以避免闪烁。

(3) 实验过程中遇到的问题和解决方法有助于提高对嵌入式系统开发的理解和实践能力。