计算机科学与技术学院

嵌入式系统实验报告

（六）

姓　　名：　Banban

专　　业：　计 算 机 科 学 与 技 术

班　　级：

学　　号：

指 导 教 师：

2023 年 4 月 13 日

一、任务要求

1、 利用 systick 实现 1us 延时和 1ms 延时，写出 2 个版本的延时函数 delay_us() 和 delay_ms()。

2、 写一个测试程序检验你的延时函数（比如：在数码管上以倒计时方式显示 9—>0，规定时间改变一次）。

二、实验报告要求

1、 任务中自编程序的源代码（加上注释）

2、 能说明软件仿真结果的截图、反映硬件电路连接和硬件验证结果的图片或视频

三、 实验过程

一、 任务一：利用 systick 实现 1us 延时和 1ms 延时，写出 2 个版本的延时函数 delay_us()和 delay_ms()

1. 代码 1

```
// main.c
// ----------------------------------------------------------
#include "stm32f10x.h"

// 1us 延时函数 delay_us()
void delay_us(u8 us) {
    SysTick->LOAD = 72 * us - 1; // 设置 1/72MHz 秒
    SysTick->VAL = 0;      // 清空计数器
    SysTick->CTRL = 0x01; // 开启 SysTick 计时器
    while (!(SysTick->CTRL & 0x10000)); // 等待延时时间到达
    SysTick->CTRL = 0;   // 关闭 SysTick 计时器
}

// 1ms 延时函数 delay_ms()
```

```c
void delay_ms(u8 ms) {
    SysTick->LOAD = 72000 * ms - 1; // 设置 SysTick 重装载值，每个 tick 为 1/72MHz 秒
    SysTick->VAL = 0; // 清空计数器
    SysTick->CTRL = 0x01; // 开启 SysTick 计时器
    while (!(SysTick->CTRL & 0x10000)); // 等待延时时间到达
    SysTick->CTRL = 0; // 关闭 SysTick 计时器
}

void SysTick_Init(void) {
    /* SystemFrequency / 1000    1ms 中断一次
     * SystemFrequency / 100000  10us 中断一次
     * SystemFrequency / 1000000 1us 中断一次
     */
    if (SysTick_Config(SystemCoreClock / 100000)) {   // ST3.5.0 库版本
        while (1);     /* Capture error */
    }
    // 关闭滴答定时器
    SysTick->CTRL &= ~ SysTick_CTRL_ENABLE_Msk;
}

int main(void) {
    SysTick_Init();
    delay_us(1);
    delay_ms(10);
    // ...
}
```

2. 代码 2

```c
// main.c
// -----------------------------------------------------------
#include "stm32f10x.h"

// us 延时函数 delay_us()
void delay_us(unsigned int Delay){
```

```
    uint32_t tickstart = SysTick->VAL;   // 获取当前 tick
    uint32_t tickNum = 0;
    uint32_t tickMax = SysTick->LOAD + 1;
    uint32_t delay_usvalue = (tickMax / 1000) * Delay;  // 计算一共需要延
时的 tick
    while(1){
        uint32_t cur_tick = SysTick->VAL;
        if (cur_tick > tickstart) {      // 进行了一次重载
            tickNum = tickstart + (tickMax - cur_tick);
        } else {             // 未进行过重载
            tickNum = tickstart - cur_tick;
        }
        if (tickNum > delay_usvalue) {     // 达到延时的 tick 数
            return;
        }
    }
}

// ms 延时函数 delay_ms()
void delay_ms(u32 i){
    u32 temp;
    SysTick->LOAD=9000*i;
    SysTick->CTRL=0X01;        // 使能，减到零是无动作，采用外部时钟源
    SysTick->VAL=0;            // 清零计数器
    do{
        temp=SysTick->CTRL;    // 读取当前倒计数值
    } while((temp&0x01)&&(!(temp&(1<<16))));    // 等待时间到达
    SysTick->CTRL=0;    // 关闭计数器
    SysTick->VAL=0;     // 清空计数器
}
int main(void) {
  while (1){
    static u8 start_time = 0;
    static u8 end_time = 0;
    start_time = SysTick->VAL;    // 延时前获取 time
    delay_us(10);                 // 延时
```

```
    end_time = SysTick->VAL;       // 延时后获取 time
    printf("delay time tick is %d\r\n", start_time - end_time);
    delay_ms(1000);
  }
}
```

二、 任务二：写一个测试程序检验你的延时函数（比如：在数码管上以倒计时方式显示 9—>0，规定时间改变一次）

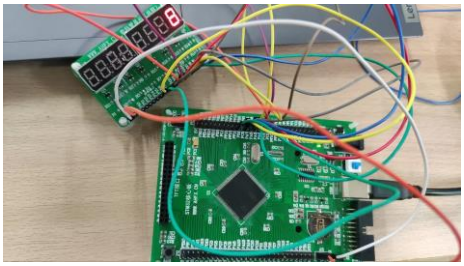
1. 代码

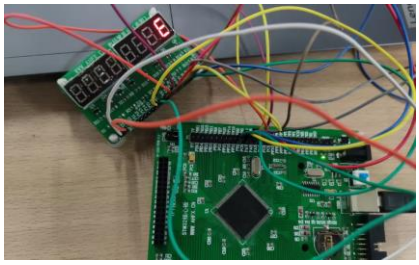
```
// main.c
// -------------------------------------------------------
#include "stm32f10x.h"
//h g f e d c b a
//1 1 0 0 0 0 0 0 --共阳方式显示 0
u8 seg_tab[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80,
0x90};

//0x7f=01111111 此时仅最左边的管子被激活
u8 pos[8]={0x7f,0xBF,0xDF,0xEF,0xf7,0xFB,0xFD,0xFe};

void LED_Init(void) {
     GPIO_InitTypeDef  GPIO_InitStructure;
     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);//使能 PA 口时钟
     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1| GPIO_Pin_2|
GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
     GPIO_Init(GPIOA, &GPIO_InitStructure);
     GPIO_SetBits(GPIOA,GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO
_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7);
     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE,ENABLE);//使能 PE 口时钟
```

```
    GPIO_InitStructure.GPIO_Pin  = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|
GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;//KEY0-KEY2
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; //推挽输出
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOE, &GPIO_InitStructure);//
    GPIO_SetBits(GPIOE,GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO
_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7);
}

// 1us 延时函数 delay_us()
void delay_us(uint32_t us) {
    SysTick->LOAD = 72 * us - 1; // 设置 SysTick 重装载值，每个 tick 为
1/72MHz 秒
    SysTick->VAL = 0; // 清空计数器
    SysTick->CTRL = 0x01; // 开启 SysTick 计时器
    while (!(SysTick->CTRL & 0x10000)); // 等待延时时间到达
    SysTick->CTRL = 0; // 关闭 SysTick 计时器
}

// 1ms 延时函数 delay_ms()
void delay_ms(uint32_t ms) {
    SysTick->LOAD = 72000 * ms - 1; // 设置 SysTick 重装载值，每个 tick 为
1/72MHz 秒
    SysTick->VAL = 0; // 清空计数器
    SysTick->CTRL = 0x01; // 开启 SysTick 计时器
    while (!(SysTick->CTRL & 0x10000)); // 等待延时时间到达
    SysTick->CTRL = 0; // 关闭 SysTick 计时器
}

void SysTick_Init(void) {
    /* SystemFrequency / 1000    1ms 中断一次
     * SystemFrequency / 100000  10us 中断一次
     * SystemFrequency / 1000000 1us 中断一次
     */
    if (SysTick_Config(SystemCoreClock / 100000)) {   // ST3.5.0 库版本
        while (1);     /* Capture error */
```

```
    }
    // 关闭滴答定时器
    SysTick->CTRL &= ~ SysTick_CTRL_ENABLE_Msk;
}

int main(void){
    int i;
    SysTick_Init();  // 延时初始化函数
    LED_Init();      // 数码管初始化函数
    GPIOE->ODR=0xff; // 数码管全熄灭

    while(1){
        for(i=0 ; i<8 ; i++){
            GPIOA->ODR = seg_tab[i]; // 字形控制
            GPIOE->ODR = pos[7];     // 最后一个位置
            delay_ms(100);
            // delay_us(100);
        }
    }
}
```

2. 图片效果





四、总结与分析

在老师的帮助指导下，本次实验我们利用 STM32F103 芯片和数码管编写了一个倒计时程序，每隔 100us 或 100ms 更新一次显示。在程序中，我们使用了 SysTick 定时器来实现精确的延时，同时使用 GPIO 控制数码管的显示。

具体而言，我们将倒计时的数字从 9 一直减少到 0，每隔 100us 进行一次更新，并在数码管上进行显示，比较顺畅，但设置每隔 100ms 时，数码管显示了 8，原因可能是速度太快，前一个数字未熄灭，下一个数字就显示了。通过本次实验，我们深入了解了 STM32F103 芯片的定时器和 GPIO 模块的使用方法，同时掌握了利用这些模块编写实际应用程序的技能。此外，通过对程序的调试和优化，我们也加深了对嵌入式系统设计的理解和掌握程度。