《单片机技术》实验思考

严昕宇 20121802

上海大学 计算机工程与科学学院

1　学习STM32的简单应用系统：参照实验指南，练习两个版本的跑马灯实验（寄存器版本和库函数版本）

1.1　软件代码是如何使能硬件的?

首先确保硬件上的连接。ALIENTEK 战舰 STM32F103 上默认将DS0 接 PB5，DS1 接 PE5。

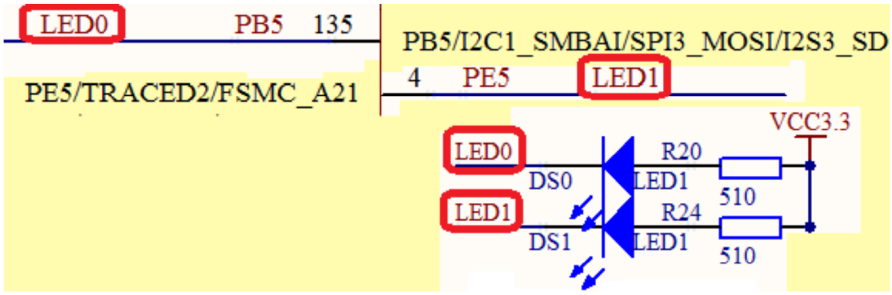

图 6.2.1 LED 与 STM32F1 连接原理图

在软件部分，先配置时钟。APB2ENR 是 APB2 总线上的外设时钟使能寄存器。此处要使能的 PORTB 和 PORTE 的时钟使能位，分别在 bit3 和 bit6，只要将这两位置 1 就可使能 PORTB 和 PORTE 的时钟。

```
RCC->APB2ENR|=1<<3;      //使能 PORTB 时钟
RCC->APB2ENR|=1<<6;      //使能 PORTE 时钟
```

在配置完时钟之后，LED_Init 配置了 PB5 和 PE5 的模式为推挽输出，并且默认输出 1。这样就完成了对这两个 IO 口的初始化，随后可以通过位带操作来实现操作某个 IO 口的 1 个位。

```
GPIOB->CRL&=0XFF0FFFFF;
GPIOB->CRL|=0X00300000;//PB.5  推挽输出
GPIOB->ODR|=1<<5;        //PB.5  输出高
GPIOE->CRL&=0XFF0FFFFF;
GPIOE->CRL|=0X00300000;//PE.5 推挽输出
GPIOE->ODR|=1<<5;        //PE.5 输出高
```

1.2　寄存器版本、库函数版本的区别在哪

寄存器版本和库函数版本的代码如下：

```
// 通过调用库函数来实现IO控制的方法
int main(void)
{

    delay_init();            //初始化延时函数
    LED_Init();              //初始化LED端口
    while(1)
```

```
    {
        GPIO_ResetBits(GPIOB,GPIO_Pin_5);
        //LED0对应引脚GPIOB.5拉低，亮  等同LED0=0;
        GPIO_SetBits(GPIOE,GPIO_Pin_5);
        //LED1对应引脚GPIOE.5拉高，灭 等同LED1=1;
        delay_ms(300);     //延时300ms
        GPIO_SetBits(GPIOB,GPIO_Pin_5);
        //LED0对应引脚GPIOB.5拉高，灭  等同LED0=1;
        GPIO_ResetBits(GPIOE,GPIO_Pin_5);
        //LED1对应引脚GPIOE.5拉低，亮 等同LED1=0;
        delay_ms(300);     //延时300ms
    }
}

// 直接操作寄存器方式实现IO口控制
int main(void)
{

    delay_init();                 //初始化延时函数
    LED_Init();                   //初始化LED端口
    while(1)
    {
        GPIOB->BRR=GPIO_Pin_5;       //LED0亮
        GPIOE->BSRR=GPIO_Pin_5; //LED1灭
        delay_ms(300);
        GPIOB->BSRR=GPIO_Pin_5; //LED0灭
        GPIOE->BRR=GPIO_Pin_5;  //LED1亮
        delay_ms(300);
     }
 }
```

寄存器版本通过控制LED对应GPIO的寄存器，如BRR（端口位清除寄存器）和BSRR（端口位设置寄存器）中的值来改变LED灯的亮暗情况。

库函数版本通过调用库函数来实现对LED的控制。库函数本质是封装了对寄存器的操作，如下所示：

```
void GPIO_ResetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
{
  /* Check the parameters */
  assert_param(IS_GPIO_ALL_PERIPH(GPIOx));
  assert_param(IS_GPIO_PIN(GPIO_Pin));

  GPIOx->BRR = GPIO_Pin;
}
```

1.3 不使用第三方函数，实现跑马灯

此处以实现一个LED灯的交替亮暗为例，跑马灯只需再加一个LED灯交替即可。

相应寄存器的地址是由于stm32f10x.h中宏定义计算得到的，如下所示：

- #define PERIPH_BASE ((uint32_t)0x40000000)
- #define APB1PERIPH_BASE PERIPH_BASE
- #define GPIOB_BASE (APB2PERIPH_BASE + 0x0C00)
- #define GPIOE_BASE (APB2PERIPH_BASE + 0x1800)
- #define GPIOB ((GPIO_TypeDef *) GPIOB_BASE)
- #define GPIOE ((GPIO_TypeDef *) GPIOE_BASE)

GPIOB/GPIOE的地址就是在这个PERIPH_BASE地址的基础上偏移得到得，且与操作手册上的相同。

程序如下：

```
#include "sys.h"
//初始化PB5、PE5为输出.并使能这个口的时钟
void LED_Init(void) {
    volatile unsigned int *RCC_APB2ENR;
    volatile unsigned int *GPIOB_CRL;
    volatile unsigned int *GPIOB_ODR;
    volatile unsigned int *GPIOE_CRL;
    volatile unsigned int *GPIOE_ODR;
    RCC_APB2ENR = (volatile unsigned int *) (0x40021000 + 0x18);
    GPIOB_CRL = (volatile unsigned int *) (0x40010C00 + 0x00);
    GPIOB_ODR = (volatile unsigned int *) (0x40010C00 + 0x0C);
    GPIOE_CRL = (volatile unsigned int *) (0x40011800 + 0x00);
    GPIOE_ODR = (volatile unsigned int *) (0x40011800 + 0x0C);
    //初始化PB5、PE5引脚
    (*RCC_APB2ENR) |= 1 << 3;     //使能GPIOB时钟
    (*GPIOB_CRL) &= 0XFF0FFFFF; //PB5推挽输出
    (*GPIOB_CRL) |= 0X00300000;
    (*GPIOB_ODR) &= 0XFFFFFFDF;   //0<<5，PB5输出低，点亮LED
    (*GPIOE_CRL) &= 0XFF0FFFFF;
    (*GPIOE_CRL) |= 0X00300000;//PE5推挽输出
    (*GPIOE_ODR) |= 1<<5;       //PE5输出高
}

//延时,以us为基本单位
void delay(u32 us_number) {
    for (int i = 0; i++; i < us_number) {
        delay_us();
    }
}
```

```
//微秒延时函数
void delay_us(u32 nus)
{
    u32 temp;
    SysTick->LOAD=nus*fac_us;                //时间加载
    SysTick->VAL=0x00;                       //清空计数器
    SysTick->CTRL=0x01 ;                     //开始倒数
    do
    {
        temp=SysTick->CTRL;
    }while((temp&0x01)&&!(temp&(1<<16)));    //等待时间到达
    SysTick->CTRL=0x00;                      //关闭计数器
    SysTick->VAL =0X00;                      //清空计数器
}

//入口函数
int main(void) {    //函数名不是main也可以的
    LED_Init();     //初始化与LED连接的硬件接口
    while (1) {
        (*GPIOB_ODR) |= 1 << 5;       //PB5输出高，熄灭
        (*GPIOE_ODR) &= 0XFFFFFFDF;   //0<<5，PE5输出低，点亮LED
        delay(500000);     //延时500ms
        (*GPIOB_ODR) &= 0XFFFFFFDF;   //0<<5，PB5输出低，点亮LED
        (*GPIOE_ODR) |= 1 << 5;       //PE5输出高，熄灭
        delay(500000);     //延时500ms
    }
}
```

2　练习一个汇编语言程序（不涉及外设）：用ARM汇编指令编写程序，实现“求两变量的最大公约数”，并调试运行正确

此处使用《九章算术》中的更相减损术，具体步骤如下：

- 第一步：任意给定两个正整数；判断它们是否都是偶数。若是，则用2约简；若不是则执行第二步。
- 第二步：以较大的数减较小的数，接着把所得的差与较小的数比较，并以大数减小数。继续这个操作，直到所得的减数和差相等为止。
- 则第一步中约掉的若干个2的积与第二步中等数的乘积就是所求的最大公约数。
- 其中所说的“等数”，就是公约数。求“等数”的办法是“更相减损”法。

代码如下：

```
x       equ 2002
y       equ 318

        mov r1,#x           ;第一个数x存至r1
        mov r2,#y           ;第二个数y存至r2
;
sub_num
        cmp r1,r2           ;比较r1和r2中的值
        beq exit            ;若两数相等，则退出
        subgt r1, r1, r2    ;若r1>r2，则r1=r1-r2
        sublt r2, r2, r1    ;若r1<r2，则r2=r2-r1
        b sub_num
exit
        mov r0,r1           ;最大公约数在r0
```

作为拓展，我也尝试了使用ARM汇编进行高斯求和，即1+2+……+100的结果，代码如下：

```
        mov R0, #1          ;求和变量
        mov R1, #0          ;累加器

loop
        cmp R0, #100
        bhi stop
        addls R1,R1,R0
        add R0, #1
        b loop
stop
        b stop              ;高斯和在r0中
```