厦门大学

XIAMEN UNIVERSITY

嵌入式系统开发

实验四 PWM 控制实验

姓 名	
学 号	
学 院	信息学院
专 业	计算机科学与技术专业
年 级	2022
日 期	2024/12/6

目录

1 实验目的

（1） 本实验通过配置 PWM 相关寄存器和步进电机，能够实现使用 PWM 模块控制电机的运行。完成本实验后，实验者应能掌握 PWM 模块和步进电机的工作原理。

2 实验方案设计

2.1 基础实验

（1）实验要求

根据实验箱提供的电机特性设置 PWM 模块相关参数，实现电机的转动。

（2）实验步骤

步骤一：新建工程文件，在 Soft_Drive 文件夹中添加 PWM.c 文件用放置电机驱动程序，将 PC6、 PC7、PC8、PC9 配置为 PWM 的 4 个输出管脚，设定好电机的转速与转向，实现电机的运动。

步骤二：连接步进电机的驱动线，下载运行程序验证实验。

2.2 提高实验

（1）实验要求

根据实验箱提供的电机特性设置 PWM 模块相关参数，使得电机的位置与转速能通过改变 PWM 的脉冲个数与频率而被控制，实现电机的加速减速，并在 OLED 上显示电机转动信息，当拨码开关 4 拨为 RESET 时通过按动按键开关设置电机速度、转向，当拨码开关 4 拨为 SET 时启动电机。

（2）实验步骤

步骤一：新建工程文件，在 PWM.c 文件用、PWM 波输出的占空比来改变电机的转动方向，改变 PWM 的频率来改变电机的转速，编写按键中断函数，当按键被按下时改变电机转速、转向。

步骤二：添加 OLED 驱动文件，在屏幕上显示电机速度、正反转信息。

2.3 拓展实验

（1）实验要求

在实验 2.2 的基础上，一个 Led 灯用来指示是否反转，一个 Led 灯的亮度用来表示电机的速度大小，电机转速越大，Led 越亮。

（2）实验步骤

步骤一：新建工程文件，在 PWM.c 文件用、PWM 波输出的占空比来改变电机的转动方向，改变 PWM 的频率来改变电机的转速，编写按键中断函数，当按键被按下时改变电机转速、转向。

步骤二：添加 OLED 驱动文件，在屏幕上显示电机速度、正反转信息。

步骤三：添加一个定时器，用来控制 Led 灯的 pwm。

3 实验过程与结果

3.1 基础实验

先初始化

```
void system_init(void)
{
    systick_config();///系统时钟初始化
    gd_XII_systeminit();////xll 外设初始化
    dipinit();////8 个拨码开关初始化
```

```
    motor_gpio_config();//Configure PC6 PC7 PC8 PC9

    timer_config();////PWM 配置
    //timer3_config();///定时器配置

    OLED_Gpio_Init();//oled 引脚配置
    OLED_Init();//oled 初始化
   timer4_config();

   //按键中断初始化
    gd_eval_key_init(USER1_KEY,KEY_MODE_EXTI);
  gd_eval_key_init(USER2_KEY,KEY_MODE_EXTI);
}
```

接下来 pwm.c 里有时间配置函数

```
void timer_config(void)
{


    /* ----------------------------------------------------------------------
    TIMER1 configuration: generate 3 PWM signals with 3 different duty cycles:
    TIMER1CLK = SystemCoreClock / 200 = 1MHz

    TIMER1 channel1 duty cycle = (4000/ 16000)* 100  = 25%
    TIMER1 channel2 duty cycle = (8000/ 16000)* 100  = 50%
    TIMER1 channel3 duty cycle = (12000/ 16000)* 100 = 75%
    ---------------------------------------------------------------------- */
    timer_oc_parameter_struct timer_ocintpara;
    timer_parameter_struct timer_initpara;
     timer_break_parameter_struct timer_bdtrpara;

    rcu_periph_clock_enable(RCU_TIMER7);
    rcu_timer_clock_prescaler_config(RCU_TIMER_PSC_MUL4);

    timer_deinit(TIMER7);

    /* TIMER1 configuration */
    timer_initpara.prescaler         = 199;
    timer_initpara.alignedmode       = TIMER_COUNTER_EDGE;
    timer_initpara.counterdirection  = TIMER_COUNTER_UP;
    timer_initpara.period            = 15999;
    timer_initpara.clockdivision     = TIMER_CKDIV_DIV1;
    timer_initpara.repetitioncounter = 0;
    timer_init(TIMER7,&timer_initpara);
```

```
    /* CH1,CH2 and CH3 configuration in PWM mode1 */
    timer_ocintpara.ocpolarity  = TIMER_OC_POLARITY_HIGH;
    timer_ocintpara.outputstate = TIMER_CCX_ENABLE;

    timer_ocintpara.ocnpolarity = TIMER_OCN_POLARITY_HIGH;
    timer_ocintpara.outputnstate = TIMER_CCXN_ENABLE;
    timer_ocintpara.ocidlestate  = TIMER_OC_IDLE_STATE_LOW;
    timer_ocintpara.ocnidlestate = TIMER_OCN_IDLE_STATE_LOW;

    timer_channel_output_config(TIMER7,TIMER_CH_0,&timer_ocintpara);
    timer_channel_output_config(TIMER7,TIMER_CH_1,&timer_ocintpara);
    timer_channel_output_config(TIMER7,TIMER_CH_2,&timer_ocintpara);
    timer_channel_output_config(TIMER7,TIMER_CH_3,&timer_ocintpara);



    // CH0:
    timer_channel_output_pulse_value_config(TIMER7, TIMER_CH_0, 15999);
     timer_channel_output_mode_config(TIMER7, TIMER_CH_0, TIMER_OC_MODE_LOW);
    timer_channel_output_shadow_config(TIMER7, TIMER_CH_0,
TIMER_OC_SHADOW_DISABLE);

    // CH1:
  timer_channel_output_pulse_value_config(TIMER7, TIMER_CH_1, 15999);
     timer_channel_output_mode_config(TIMER7, TIMER_CH_1, TIMER_OC_MODE_LOW);
    timer_channel_output_shadow_config(TIMER7, TIMER_CH_1,
TIMER_OC_SHADOW_DISABLE);

    // CH2:
   timer_channel_output_pulse_value_config(TIMER7, TIMER_CH_2, 15999);
     timer_channel_output_mode_config(TIMER7, TIMER_CH_2, TIMER_OC_MODE_LOW);
    timer_channel_output_shadow_config(TIMER7, TIMER_CH_2,
TIMER_OC_SHADOW_DISABLE);

    // CH3:
 timer_channel_output_pulse_value_config(TIMER7, TIMER_CH_3, 15999) ;
    timer_channel_output_mode_config(TIMER7, TIMER_CH_3, TIMER_OC_MODE_LOW);
    timer_channel_output_shadow_config(TIMER7, TIMER_CH_3,
TIMER_OC_SHADOW_DISABLE);
  //    ñ
    timer_bdtrpara.runoffstate     = TIMER_ROS_STATE_DISABLE;
    timer_bdtrpara.ideloffstate    = TIMER_IOS_STATE_DISABLE;
    timer_bdtrpara.deadtime        = 100;
    timer_bdtrpara.breakpolarity   = TIMER_BREAK_POLARITY_LOW;
```

```
    timer_bdtrpara.outputautostate = TIMER_OUTAUTO_ENABLE;
    timer_bdtrpara.protectmode     = TIMER_CCHP_PROT_OFF;
    timer_bdtrpara.breakstate      = TIMER_BREAK_DISABLE;
    timer_break_config(TIMER7, &timer_bdtrpara);

    timer_primary_output_config(TIMER7,ENABLE);
    /* auto-reload preload enable */
    timer_auto_reload_shadow_enable(TIMER7);
    /* auto-reload preload enable */
    timer_enable(TIMER7);
}

void timer3_config(void)
{
    timer_parameter_struct timer_initpara;

    // TIMER3
    rcu_periph_clock_enable(RCU_TIMER3);
    timer_deinit(TIMER3);


    timer_initpara.prescaler         = 199;          //分频，f=200M/200=1MHz
    timer_initpara.alignedmode       = TIMER_COUNTER_EDGE;
    timer_initpara.counterdirection  = TIMER_COUNTER_UP;
    timer_initpara.period            = 20000/3;         // 周期设置
    timer_initpara.clockdivision     = TIMER_CKDIV_DIV1;
    timer_initpara.repetitioncounter = 0;
    timer_init(TIMER3, &timer_initpara);


    timer_interrupt_enable(TIMER3, TIMER_INT_UP);
    nvic_irq_enable(TIMER3_IRQn, 3, 3);

    timer_enable(TIMER3);
}
```

还有定时器中断函数

```
void TIMER3_IRQHandler(void)
{
    if (timer_interrupt_flag_get(TIMER3, TIMER_INT_UP)) {

        timer_interrupt_flag_clear(TIMER3, TIMER_INT_UP);
        switch (step_index) {
            case 0:  //   ㄏ  :CH0   ˇPWM
                timer_channel_output_mode_config(TIMER7, TIMER_CH_0,
TIMER_OC_MODE_PWM1);
```

```
                timer_channel_output_mode_config(TIMER7, TIMER_CH_1,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_2,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_3,
TIMER_OC_MODE_HIGH);
                break;
            case 1:  //  ل   :CH1   ʿPWM
                timer_channel_output_mode_config(TIMER7, TIMER_CH_0,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_1,
TIMER_OC_MODE_PWM1);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_2,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_3,
TIMER_OC_MODE_HIGH);
                break;
            case 2:  //      :CH2   ʿPWM
                timer_channel_output_mode_config(TIMER7, TIMER_CH_0,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_1,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_2,
TIMER_OC_MODE_PWM1);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_3,
TIMER_OC_MODE_HIGH);
                break;
            case 3:  //       :CH3   ʿPWM
                timer_channel_output_mode_config(TIMER7, TIMER_CH_0,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_1,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_2,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_3,
TIMER_OC_MODE_PWM1);
                break;
        }
      //更新拍状态
        step_index = (step_index + 1) % 4;

    }
  }
```

main 主函数里直接 while(1)死循环即可。于是电机就可以转动了。

3.2 提高实验

首先，在实验 1 的基础上，加按键中断改变速度和旋转方向

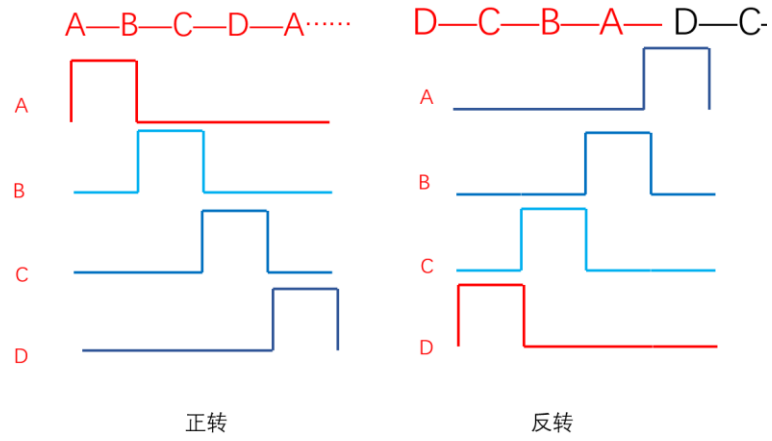
```
void EXTI2_IRQHandler(void)
{
    if (RESET != exti_interrupt_flag_get(USER1_KEY_EXTI_LINE))
    {
        if(motor_speed++ >= 10)//motor_speed 快变慢
            motor_speed = 3;
        exti_interrupt_flag_clear(USER1_KEY_EXTI_LINE);
        update_pwm_speed(motor_speed);
    }
}

//
void EXTI3_IRQHandler(void)
{
    if (RESET != exti_interrupt_flag_get(USER2_KEY_EXTI_LINE))
    {
        if(mode_select++ >= 2)
        {
            mode_select = 0;
        }
        setMode(mode_select);
        exti_interrupt_flag_clear(USER2_KEY_EXTI_LINE);
    }
}
```

调整速度的函数，改变 timer_initpara.period。

```
void update_pwm_speed(uint8_t speed)
{
    timer_parameter_struct timer_initpara;

    // TIMER3
    rcu_periph_clock_enable(RCU_TIMER3);
    timer_deinit(TIMER3);
```

```
    timer_initpara.prescaler         = 199;          //分频，f=200M/200=1MHz
    timer_initpara.alignedmode       = TIMER_COUNTER_EDGE;
    timer_initpara.counterdirection  = TIMER_COUNTER_UP;
    timer_initpara.period            = 20000/speed;         // 周期设置
    timer_initpara.clockdivision     = TIMER_CKDIV_DIV1;
    timer_initpara.repetitioncounter = 0;
    timer_init(TIMER3, &timer_initpara);


    timer_interrupt_enable(TIMER3, TIMER_INT_UP);
    nvic_irq_enable(TIMER3_IRQn, 3, 3);

    timer_enable(TIMER3);
}
```

改变转速的思路如下：
正转反转原理图：



正转　　反转

在 switch 里，正转是 0→1→2→3→0，反转是 3→2→1→0→3。

也就是，正转的更新拍状态是 `step_index = (step_index + 1) % 4;`
反转的更新拍状态是 `step_index = (step_index - 1) % 4;`

这里，按 c 语言的编程习惯，所以把反转公式改成 `step_index = (step_index + 3) % 4;`

归总为 `step_index = (step_index + mode) % 4;`

正转时 `mode` 为 1，反转时为 3

所以代码如下：

```
uint8_t mode=1;
void setMode(uint8_t newmode){
  //0->1
  //1->3
```

```
  mode=(newmode*2+1);
}
void TIMER3_IRQHandler(void)
{
    if (timer_interrupt_flag_get(TIMER3, TIMER_INT_UP)) {

        timer_interrupt_flag_clear(TIMER3, TIMER_INT_UP);
        switch (step_index) {
            case 0:  //   ʜ  :CH0   ᶜPWM
                timer_channel_output_mode_config(TIMER7, TIMER_CH_0,
TIMER_OC_MODE_PWM1);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_1,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_2,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_3,
TIMER_OC_MODE_HIGH);
                break;
            case 1:  //  ل   :CH1   ᶜPWM
                timer_channel_output_mode_config(TIMER7, TIMER_CH_0,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_1,
TIMER_OC_MODE_PWM1);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_2,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_3,
TIMER_OC_MODE_HIGH);
                break;
            case 2:  //      :CH2   ᶜPWM
                timer_channel_output_mode_config(TIMER7, TIMER_CH_0,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_1,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_2,
TIMER_OC_MODE_PWM1);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_3,
TIMER_OC_MODE_HIGH);
                break;
            case 3:  //       :CH3   ᶜPWM
                timer_channel_output_mode_config(TIMER7, TIMER_CH_0,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_1,
TIMER_OC_MODE_HIGH);
```

```
                timer_channel_output_mode_config(TIMER7, TIMER_CH_2,
TIMER_OC_MODE_HIGH);
                timer_channel_output_mode_config(TIMER7, TIMER_CH_3,
TIMER_OC_MODE_PWM1);
                break;
        }
       //更新拍状态
        step_index = (step_index + mode) % 4;
    }
  }
```

显示 OLED 的代码：

```
void test2(void)
{
        //light_delay();
    if(gpio_input_bit_get(DIP4_GPIO_PORT,DIP4_PIN)==RESET) {
        char str[4];
        speed = motor_speed;
        sprintf(str,"%.2f",speed);
        OLED_ShowString(0,0,"speed:");
        OLED_ShowString(50,0,str);
        sprintf(speed_str,"%.2f",speed);

        if(mode_select == reverse)
        {
            OLED_ShowString(0,2," Reverse");
            for(int i=0;i<15;i++)
            {
                        dir[i]=pos[i];
                    }
        }
        else
        {
            OLED_ShowString(0,2,"Positive");
            for(int i=0;i<15;i++)
            {
                        dir[i]=rev[i];
                    }
        }
    }
}
```

3.3 拓展实验

在实验 2 的基础上加指示灯。对于改变方向灯的指示，

只需要在 void EXTI3_IRQHandler(void)里加上

```
if(mode_select==1)
            gd_eval_led_on(LED2);
else
            gd_eval_led_off(LED2);
```

对于速度灯和亮度，新开一个定时器，自己写一个函数模拟 pwm 占空比。

```
void timer4_config(void)
{
    timer_parameter_struct timer_initpara;

    // TIMER4
    rcu_periph_clock_enable(RCU_TIMER4);
    timer_deinit(TIMER4);

    timer_initpara.prescaler         = 199;          // ??,f=200M/200=1MHz
    timer_initpara.alignedmode       = TIMER_COUNTER_EDGE;
    timer_initpara.counterdirection  = TIMER_COUNTER_UP;
    timer_initpara.period            = 1000;         // ????,1kHz
    timer_initpara.clockdivision     = TIMER_CKDIV_DIV1;
    timer_initpara.repetitioncounter = 0;
    timer_init(TIMER4, &timer_initpara);

    timer_interrupt_enable(TIMER4, TIMER_INT_UP);
    nvic_irq_enable(TIMER4_IRQn, 3, 3);

    timer_enable(TIMER4);
}
void TIMER4_IRQHandler(void)
{
    if (timer_interrupt_flag_get(TIMER4, TIMER_INT_UP)) {
        timer_interrupt_flag_clear(TIMER4, TIMER_INT_UP);
        light_delay();
    }
}
```

```
void light_delay(){
    light_time++;
    if(light_time<speed)
        gd_eval_led_on(LED1);
    else
    {
        gd_eval_led_off(LED1);
        if(light_time==14)
            light_time=0;
    }
}
```

这样灯亮的时间占总时间的 speed/14×100%
速度越大，灯越亮

4 实验分析

在本次实验中，我通过配置 PWM（脉冲宽度调制）相关寄存器和步进电机，实现了使用 PWM 模块控制电机的运行。

PWM 是一种通过改变脉冲宽度来控制输出电压或电流的技术。在本实验中，我通过调整 PWM 的占空比（即脉冲的高电平时间与周期的比例）来控制电机的转速，以及通过改变 PWM 的频率来控制电机的位置和转速。

首先配置了 PWM 模块的相关参数，使得电机能够转动。通过设置不同的占空比和频率，能够控制电机的转速和转向。

在基础实验的基础上，我增加了按键中断功能，允许用户通过按键来动态调整电机的速度和转向。此外，我还通过 OLED 显示屏实时显示电机的转速和转向状态。

在提高实验的基础上，我增加了 LED 灯来指示电机的转向和速度。一个 LED 灯用

于指示电机是否反转，另一个 LED 灯的亮度用来表示电机的速度大小。

代码分析

定时器配置：配置了多个定时器，包括系统时钟初始化、外设初始化、PWM 配置等。这些配置确保了 PWM 信号的准确输出。

中断服务程序：编写了中断服务程序来处理按键中断，允许用户通过按键来改变电机的速度和转向。

OLED 显示：添加了 OLED 驱动文件，并在屏幕上显示电机的速度和正反转信息，提高了实验的交互性。

LED 指示：通过控制 LED 灯的 PWM 来模拟电机的速度，使得 LED 灯的亮度与电机速度成正比。

5 实验总结

通过本次实验，我成功实现了使用 PWM 模块控制步进电机的运行。我不仅掌握了 PWM 模块和步进电机的工作原理，还学会了如何通过编程来动态调整电机的速度和转向。

本次实验比较容易，加深了我对 PWM 控制技术的理解。在理解电机转动原理的基础上，我想出了巧妙的改变电机转向的方法，也想出了改变灯亮度的方法，很有成就感。