嵌入式考试相关试题详解

（本资料为嘉豪教育旗下 VIP 会员独享资料）

主编：赵泽雨
2021 年 7 月 1 日星期四

目录

（免责条款：限于作者水平和经验，加之时间比较仓促，疏漏或者错误之处在所难免，试题详解不保证绝对正确，仅为同学们提供解题思路，欢迎讨论指正）

一、作业题三道

作业 1:

编写一个流水灯程序。已知, 有 8 个 LED 灯(LED0~7)分别连接到 S5PV210 引脚的 GPA0_0~7。当这些引脚高电平的时候，LED 灯亮，反之则灭；两个按键（KEY0,KEY1）分别连接到引脚的 GPH0_0, GPH0_1。当按下按键 KEY0 时，实现 LED0->LED1->...->LED7 ->LED0 的亮灯循环；当按下按键 KEY1 时，实现 LED0->LED7->...->LED1->LED0 的亮灯循环。现启动汇编代码 start.S 已有，请写出包括引脚初始化在内的其他 C 语言代码。

答:

```
#define GPA0CON   (*(volatile unsigned long *) 0xE0200000)
#define GPA0DAT (*(volatile unsigned long *) 0xE0200004)
#define      GPH0CON     *((volatile unsigned long *)0xE0200C00)
#define      GPH0DAT     *((volatile unsigned long *)0xE0200C04)
//外中断 0-7 的中断控制寄存器
#define      EXT_INT_0_CON        *((volatile unsigned int *)0xE0200E00)
//外中断 0-7 的中断屏蔽寄存器
#define      EXT_INT_0_MASK       *((volatile unsigned int *)0xE0200F00)
//第一组矢量中断选择寄存器
#define VIC0INTSELECT          *((volatile unsigned int *)0xF200000C)
//第一组矢量中断使能寄存器
#define VIC0IRQSTATUS          *((volatile unsigned int *)0xF2000000)
#define VIC0INTENABLE          *((volatile unsigned int *)0xF2000010)
#define VIC0VECTADDR2           *((volatile unsigned int *)0xF2000100)
#define VIC0VECTADDR3           *((volatile unsigned int *)0xF2000104)
#define VIC0ADDRESS                *((volatile unsigned int *)0xF2000F00)
//外部中断 0-7 的中断挂起寄存器，记录是否有中断产生
#define EXT_INT_0_PEND         *((volatile unsigned int *)0xE0200F40)

extern void key_isr(void);

void delay(volatile unsigned int t)
{
      volatile unsigned int t2 = 0xFFFF;
      while (t--)
            for (; t2; t2--);
}

//初始化 LED 控制引脚
void led_init(void)
```

```
{
	GPA0CON &= ~(0xFF << 0);
	GPA0CON |= ((0x01 << 0) | (0x01 << 4) | (0x01 << 8) | (0x01 << 12) | (0x01 <<16 ) | (0x01 << 20) | (0x01 << 24) | (0x01 << 28));
	GPA0DAT |=~ (0xFF << 0);
}

//初始化中断引脚
void key_init(void)
{
	GPH0CON &= ~(0xFF << 0);
	 GPH0CON |= (0xFF << 0);
/* 配置 GPH0_0 和 GPH0_1 为外部中断：key0 和 key1 */
	EXT_INT_0_CON &= ~(0xFF << 8);
/* 清空低八位*/
	EXT_INT_0_CON |= (2 << 0) | (2 << 4);
/* 配置 EXT_INT[0]和 EXT_INT[1]为下降沿触发*/
	EXT_INT_0_MASK &= ~0x3;
 /* 取消屏蔽外部中断 EXT_INT[0]和 EXT_INT[1] */
}

//初始化中断控制器
void int_init(void)
{
	VIC0INTSELECT &= ~0x3;
 /* 选择外部中断 EXT_INT[0]和外部中断 EXT_INT[1]为 IRQ 类型的中断 */
	VIC0ADDRESS = 0;
	VIC0VECTADDR0 = (int)key_isr;
 /* 当 EXT_INT[0]触发中断，即用户按下 key0 时， CPU 就会自动的将 VIC0VECTADDR0 的值赋给 VIC0ADDRESS 并跳转到这个地址去执行*/
	VIC0VECTADDR1 = (int)key_isr;
	VIC0INTENABLE |= 0x3;
 /* 使能外部中断 EXT_INT[0]和 EXT_INT[1] */
}

void key_handle()
{
	volatile unsigned char key_code = VIC0IRQSTATUS & 0x3;
	VIC0ADDRESS = 0x0;
/* 清中断向量寄存器 */
	EXT_INT_0_PEND |= 0x3;
/* 清中断挂起寄存器 */

	if (key_code == 0x01)		/* key0 */
```

```
    {
    for(int i=0;i<8;i++)
        {
                        GPA0DAT ^= 0 << i;          //  点亮 LED0~7
            delay(100);
        }

    }
/* toggle LED1 */

    else if (key_code == 0x02) /* key2 */
    {
    for(int i=8;i>0;i--)
        {
                        GPA0DAT ^= 0 << i;          //  点亮 LED7~0
            delay(100);
        }

    }
 /* toggle LED2 */
}

int main()
{
    led_init();
    key_init();
    int_init();

    while (1);
    return 0;
}
```

作业 2:

编写一个串口数据接收程序。已知 S5PV210 默认选 PCLK 为时钟源，且 PCLK 为 66MHz（即此题不需要编写 PCLK 的初始化程序）。请初始化串口 UART0，即让 UART0 的波特率为 115200，正常模式（非红外模式），以中断或轮询模式发生数据，并选择 PCLK 为串口时钟源，8 位数据位，1 位停止位，无校验位。然后让 UART0 串口不断的接收其他设备发送的数据。当接收到的一个字节的数据为 0x0f 时，引脚 GPH0_0 为高电平，以让 LED0 点亮，同时让引脚 GPH0_1 为低电平；当接收到的一个字节的数据为 0xf0 时，引脚 GPH0_1 为高电平，

以让 LED1 点亮，同时让引脚 GPH0_0 为低电平。现启动汇编代码 start.S 已有，请写出包括初始化函数在内的其他 C 语言代码。

答：

```
// GPIO、UART 寄存器地址
#define GPA0CON             *((volatile unsigned int *)0xE0200000)
#define GPA0DAT             *((volatile unsigned int *)0xE0200004)

#define ULCON0          *((volatile unsigned int *)0xE2900000)
#define UCON0           *((volatile unsigned int *)0xE2900004)
#define UFCON0          *((volatile unsigned int *)0xE2900008)
#define UTRSTAT0            *((volatile unsigned int *)0xE2900010)
#define UTXH0           *((volatile unsigned int *)0xE2900020)
#define URXH0           *((volatile unsigned int *)0xE2900024)
#define UBRDIV0         *((volatile unsigned int *)0xE2900028)
#define UDIVSLOT0           *((volatile unsigned int *)0xE290002C)

#define   GPA0_0_out   (1<<(0*4))
#define   GPA0_1_out   (1<<(1*4))

#define   GPA0_0_MASK     (0xF<<(0*4))
#define   GPA0_1_MASK     (0xF<<(1*4))


// 初始化 UART0(COM1)
void uart0_init(void)
{
    // 配置 GPA0_0 为 UART_0_RXD，GPA0_1 为 UART_0_TXD

    GPA0CON &= ~0xFF;
    GPA0CON |= 0x22; // bit[3:0]=0b0010,bit[7:4]=0b0010

    /* 配置数据传输格式
     * data:8bits    bit[1:0]=0x3
     * stop:1bit      bit[2]=0
     * parity:no      bit[5:3]=0b0xx
     * mode:normal bit[6]=0
     */
     ULCON0 = (0x3<<0) | (0<<2) | (0<<3) | (0<<6);

    /* 配置 UCON0
       * Receive Mode: bit[1:0]=01
       * TransmitMode: bit[3:2]=01
       * Rx Error Status Interrupt Enable: bit[6]=1
       * Clock Selection: bit[10]=0
```

```
        */
        UCON0 = (1<<0) | (1<<2) | (1<<6) | (0<<10);

        // 不使用 FIFO
        UFCON0 = 0;

      /* 计算波特率(参考手册上面的公式)，设置为 115200
          * PCLK=66MHz
          * DIV_VAL = (66000000/(115200 x 16))-1 = 35.8 - 1 = 34.8
          * UBRDIV0 = 34(DIV_VAL 的整数部分)
         * (num of 1's in UDIVSLOTn)/16 = 0.8
         * (num of 1's in UDIVSLOTn) = 12
          * UDIVSLOT0 = 0xDDDD (参考手册查表)
         */
          UBRDIV0 = 34;
          UDIVSLOT0 = 0XDDDD;
        return;
 }
// 发送数据
 void putc(unsigned char c)
 {
      // 查询状态寄存器，等待发送缓存为空
      while (! (UTRSTAT0 & (1<<2)));
      UTXH0 = c; // 写入发送寄存器
      return;
}

// 接收数据
unsigned char getc(void)
{
      // 查询状态寄存器，等待接收缓存有数据
      while (!(UTRSTAT0 & (1<<0)));
      return (URXH0);
}

int main(void)
{
      char c;
       while (1)
      {
          uart0_init();              // 初始化 uart0
          c = getc();     // 从串口终端获取一个字符
          putc(c);        // 回显
          GPA0CON &= ~(GPA0_0_MASK | GPA0_1_MASK);
```

```
                                                    // 清 bit[3:0]和 bit[7:4]
        GPA0CON |= (GPA0_0_out | GPA0_1_out);
                                                    // 配置 GPA0_0 和 GPA0_1 为输
出引脚
        GPC0DAT &= ~(0x3);   // 向 bit[4:3]写入 0 熄灭 LED1、LED2

        if (c == 0X0F)
            GPA0DAT |= 1 << 0;      // GPA0_0 高电平，GPA0_1 低电平
        else if (c ==   0XF0)
            GPA0DAT |= 2 << 0);     // GPA0_0 低电平，GPA0_1 高电平

    }
    return 0;
}
```

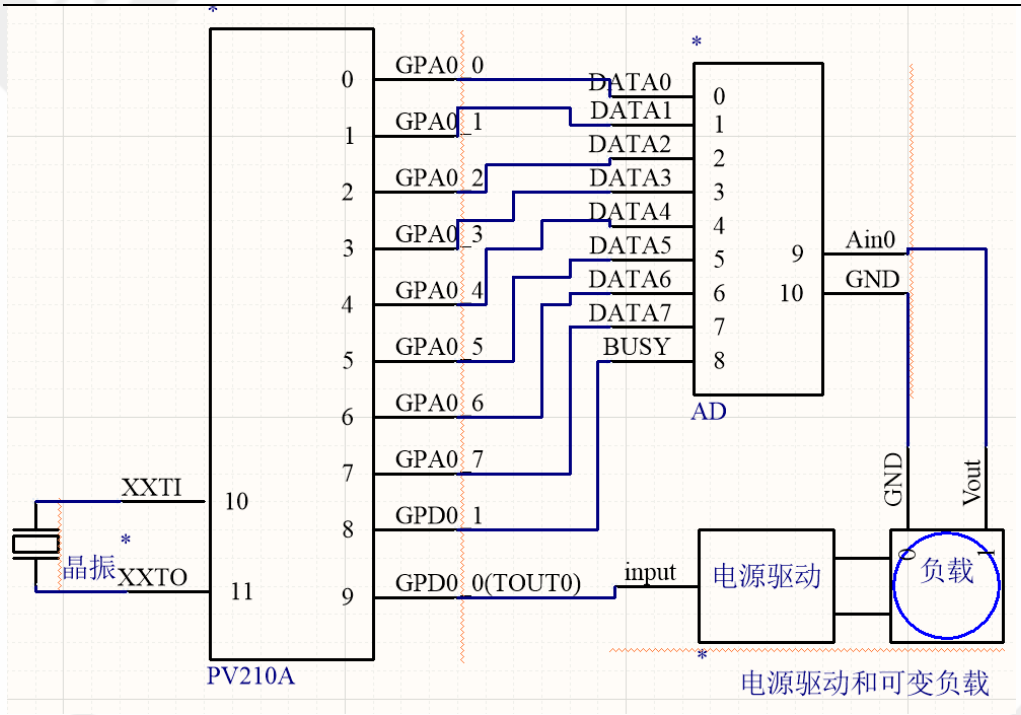
作业 3:

编写一个稳压电源的控制程序。以下是稳压电源的示意图。高速 AD 在电压采集的时候，BUSY 引脚为低电平，当模数转换完毕时，BUSY 引脚为高电平，要求 S5PV210 采用**查询 BUSY 引脚**的方式里判断其是否转换完毕，然后再读取 AD 数据。电源驱动和可变负载部分，由于负载上的电压不稳定，所以需要 S5PV210 输出**频率为 100KHz** 的 PWM 波来稳定输出电压，即根据其占空比稳定负载上的电压值。现要求 S5PV210 以 10us 的间隔进行 AD 的电压采集，并根据读取数据设定 PWM 的占空比，读取数据和占空比之间的换算方式如下：

占空比=（读取数据）/256

即当读取的数据为 0x80 时，占空比设定 50%。

已知 S5PV210 默认选 PCLK_PSYS 为定时器 0 的时钟源，且已知 PCLK_PSYS 为 66.7MHz（即此题不需要编写 PCLK 的初始化程序）。**现启动汇编代码 start.S 已有**，请写出包括**定时器初始化函数、中断初始化函数、中断服务函数等**在内的其他 **C 语言代码，并给与代码注释**。其中占空比的比例值、PWM 波的频率值 100KHz 等设置上允许有误差。




```
external void IRQ_handle(void);
/**
∵需要 10μs 的 AD 转换时间
∴AD 转换器频率 A/D converter freq. = 66.7MHz/(132.4+1) = 0.5MHz
∴预分频值 presaler value = 132 = 0x84
   转换时间 Conversion time = 1/(0.5MHz / 5cycles) = 1/100kHz = 10us
*/
//adc 初始化
void adc_init(void)
{
      TSADCCON0 |= (1 << 16);                  //设置精度为 12bit
      TSADCCON0 |= (1 << 14);          //使能 presaler
      TSADCCON0 &= ~(0xFF << 6);       //设置时钟
      TSADCCON0 |= (0x84 << 6);
      TSADCCON0 &= ~(1 << 2);        //standby 为 normal 模式
      TSADCCON0 &= ~(1 << 1);          //read_start disable
      ADCMUX &= ~(0xF << 0);           //设置模拟输入通道选择为 Ain 0
}

void io_init(void) //  初始化引脚
{
      GPD0CON &= ~0xffffffff; //全部清零
      GPD0CON |= (0x02 << 0) ; //初始化 GPD0，让 GPD0_0 为 Tout，GPD0_1 为输入
      GPD0DAT = ~ 0xFF; //  初始化 GPD0DAT
}
```

```
// 定时器 timer0 初始化
void timer0_init(void)
{
TCNTB0 = 333; // 10us 触发一次中断
TCMPB0 = 166; // 初始化 pwm 的占比为 50%
TCFG0 |= 1; // prescaler valve = 1;
TCFG1 = 0x00; // 选择 1/1 分频
TCON |= ( 1 << 1); // 手动更新
TCON = 0x09 ;// 自动加载，清“手动更新”位，启动定时器 0

}


// 使能 Timer0 中断
void init_irq(void)
{
	TINT_CASTAT |= 1;
}

//清中断
void clear_irq(void)
{
	TINT_CSTAT |= (1 << 5);
}

// 初始化中断控制器
void init_int(void)
{
	// 选择中断类型位 IRQ
	VIC0INTSELECT |= ~(1 << 21); // timer0 中断为 IRQ
	//清 VIC0ADDRESS
	VIC0ADDRESS = 0x0;
	//设置 TIMER0 中断对应中断服务程序的入口地址
	VIC0VECTADDR21 = (int) IRQ_handle;
	//使能 timer0 中断
	VIC0INTENABLE |= (1 << 21);
}

//清除需要处理的中断的中断处理函数的地址
void clear_vectaddr(void)
{
	VIC0ADDRESS = 0x0;
}
```

```
// 读中断状态
unsigned long get_irqstatus(void)
{
    return VIC0IRQSTATUS;
}

// 中断服务程序
void irq_handler()
{
    volatile unsigned char state = ((get_irqstatus() & (1 << 21)) >> 21) & 0x1;
    clear_vectaddr(); // 清中断向量寄存器
    clear_irq() // 清 Timer 0 中断
    while(!(GPD0DAT & 0x2)); //查看 AD 是否已经转换好
    if(status == 0x1)
    {
        unsigned long *n = GPA0DAT; // 获取 GPA0DAT 的引脚值
        int num = 0;
        num = sscanf(n, "%x", &n); //把 16 进制转为 10 进制
        TCMPB0 = 333 * (num / 256);
    }
}


// 主程序
void main()
{
    adc_init();//adc 初始化
    io_init (); //初始化引脚
    timer0_init(); //时钟初始化

    init_irq(); // 使能 Timer0 中断
    init_int(); //初始化中断控制器、使能中断
}
```

二、电子专业学习通综合设计题目

综合设计 1

1. 利用定时器 0 设计一个 24 小时制实时时钟，并将时间（时分秒）通过 UART0 显示于 PC 端。同时可从外部调节时间，调节方式为：串口接收先导信号 0xFF，随后接收 4 个时间信号的 ASCII 码以设置时、分，如依次接收 0xFF、0x31、0x31、0x32、0x32，表示将时间设置为 11:22。

请编写定时器 0 的初始化函数 timer0_init()和主函数 main()。

寄存器宏定义已完成，系统时钟已初始化，UART0 初始化及数据发送函数 void putc(char c)和接收函数 char getc()已定义，时钟源为 PCLK=66MHz。

```
unsigned int myTime;

// 初始化 timer0
 void timer0_init(void)
{
      TCNTB0 = 62500;       /* 1 秒钟触发一次中断 */
      TCMPB0 = 31250;            /* PWM 占空比=50% */
   TCFG0 |= 65;                  /* timer 0 Prescaler value = 65 */
   TCFG1   = 0x04;         /* 选择 16 分频 */
   TCON   |= (1<<1);     /* 手动更新 */
   TCON    = 0x09;         /* 自动加载，清“手动更新”位，启动定时器 0 */
}

// 初始化中断控制器
void init_int(void)
{
      // 选择中断类型为 IRQ
      VIC0INTSELECT |= ~(1<<21); // TIMER0 中断为 IRQ
      // 清 VIC0ADDRESS
      VIC0ADDRESS = 0x0;
      // 设置 TIMER0 中断对应的中断服务程序的入口地址
      VIC0VECTADDR21 = (int)IRQ_handle;
      // 使能 TIMER0 中断
      VIC0INTENABLE |= (1<<21);
}

// 初始化中断服务程序
void irq_handler()
```

```
{
    volatile unsigned char status = ((VIC0IRQSTATUS & (1<<21))>>21)&0x1;    // TIMER0's
int status
    VIC0ADDRESS = 0x0;          /* 清中断向量寄存器 */
    TINT_CSTAT |= (1 << 5);                                  /* 清 timer0 中断 */
    if (status == 0x1)
    {
        myTime++;
    }
}

int main(void)
{
    int a,b,c,d,e = 0;
    int hour,min,sec = 0;
    int nowTime = 0;
    uart0_init();                   /* 初始化 UART0 */
    timer0_init();              /* 初始化 Timer0 */

    TINT_CSTAT |= 0x1;                  /* 使能 TIMER0 中断 */
    init_int();                     /* 初始化中断控制器、使能中断 */

    while (1){
        hour = myTime/3600;
        min = myTime %3600/60;
        sec = myTime %60;
        nowTime = hour*10000+min*100+sec;
        putc(nowTime);
        UTXH0 &= ~0xFFFFFFFF;          //清零 UTXH0 寄存器，准备接收数据
        a = getc();
        if(a == 0xff)
        {
            UTXH0 &= ~0xFFFFFFFF;
            b = getc();
            if(b != 0x0)
            {
                UTXH0 &= ~0xFFFFFFFF;
                c = getc();
                if(c != 0x0)
                {
                    UTXH0 &= ~0xFFFFFFFF;
                    d = getc();
                    if(d != 0x0)
                    {
```

```
                        UTXH0 &= ~0xFFFFFFFF;
                        e = getc();
                        if(e != 0x0)
                        {
                            UTXH0 &= ~0xFFFFFFFF;
                            nowTime =
((char)b-'0')*100000+((char)c-'0')*10000+((char)d-'0')*1000+((char)e-'0')*100;
                            myTime = nowTime;
                        }
                    }
                }
            }

        }

    }
}
```

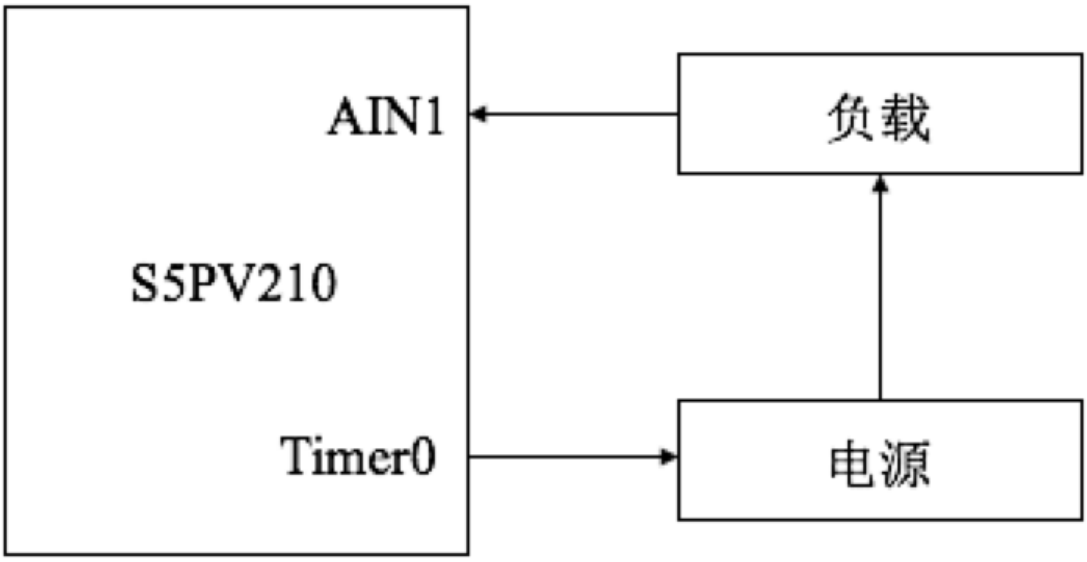
综合设计 2

2. 一个稳压电源的系统框图如下图所示。从负载采样的电压值由 AIN1 输入 S5PV210 转换为 10 位数字结果 V，转换结果通过以下关系式设定定时器 0 的 PWM 占空比以调节电压源的输出电压

duty = V/1024

要求电压采样周期为 1s，PWM 频率为 1kHz。

请编写程序实现稳压功能。其中，寄存器宏定义已完成，系统时钟已初始化，时钟源为 PCLK=66MHz。




```
external void IRQ_handle(void);
/**
∵设置 5μs 的 AD 转换时间
∴AD 转换器频率 A/D converter freq. = 66MHz/(65+1) = 1MHz
∴预分频值 presaler value = 65 = 0x41
   转换时间 Conversion time = 1/(1MHz / 5cycles) = 1/200kHz = 5us
*/
//adc 初始化
void adc_init(void)
{
      TSADCCON0 |= (0 << 16);                  //设置精度为 10bit
      TSADCCON0 |= (1 << 14);         //使能 presaler
      TSADCCON0 &= ~(0xFF << 6);       //设置时钟
      TSADCCON0 |= (0x41 << 6);
      TSADCCON0 &= ~(1 << 2);       //standby 为 normal 模式
      TSADCCON0 &= ~(1 << 1);         //read_start disable
      ADCMUX &= ~(0xF << 0);
      ADCMUX |= (0x1 << 0); //设置模拟输入通道选择为 Ain 1
}

void io_init(void) //  初始化引脚
{
      GPD0CON &= ~(0xff << 0); //清零
      GPD0CON |= (0x02 << 0) ; //初始化 GPD0，让 GPD0_0 为 Tout_0
      GPD0DAT = ~ 0xFF; //  初始化 GPD0DAT
}

//  定时器 timer0 初始化，用作产生 PWM 波
```

```
void timer0_init(void)
{
TCNTB0 =1000; // PWM 频率为 1kHz
TCMPB0 = 500; //  初始化 pwm 的占比为  50%
TCFG0 |= 65; // prescaler valve = 65;
TCFG1 = 0x00;//  选择 1/1 分频
TCON |= ( 1 << 1); //  手动更新
TCON = 0x09; //  自动加载，清“手动更新”位，启动定时器 0

}

//  定时器 timer1 初始化，用作限制采样周期
void timer1_init(void)
{
TCNTB1 = 62500; // 1s 触发一次中断
TCMPB1 = 31250; //  初始化 pwm 的占比为  50%
TCFG0 |= 65; // prescaler valve = 65;
TCFG1 |= (0x04 << 4);//  选择 1/16 分频
TCON |= ( 1 << 9); //  手动更新
TCON |= (0x09 << 8); //  自动加载，清“手动更新”位，启动定时器 1

}


//  使能 Timer1 中断
void init_irq(void)
{
    TINT_CASTAT |= 1;
}

//清中断
void clear_irq(void)
{
    TINT_CSTAT |= (1 << 5);
}

//  初始化中断控制器
void init_int(void)
{
    //  选择中断类型位 IRQ
    VIC0INTSELECT |= ~(1 << 22); // timer1  中断为 IRQ
    //清  VIC0ADDRESS
    VIC0ADDRESS = 0x0;
    //设置 TIMER0  中断对应中断服务程序的入口地址
```

```
    VIC0VECTADDR22 = (int) IRQ_handle;
    //使能 timer0 中断
    VIC0INTENABLE |= (1 << 22);
}

//清除需要处理的中断的中断处理函数的地址
void clear_vectaddr(void)
{
    VIC0ADDRESS = 0x0;
}

// 读中断状态
unsigned long get_irqstatus(void)
{
    return VIC0IRQSTATUS;
}

// 中断服务程序
void irq_handler()
{
    volatile unsigned char state = ((get_irqstatus() & (1 << 22)) >> 22) & 0x1;
    clear_vectaddr(); // 清中断向量寄存器
    clear_irq() // 清 Timer 1 中断
    while (!(_REG_TSADCCON0 & (1 << 15))) //读取状态位，查看 AD 是否已经转换好，如果 bit15 位 0 的话，一直读取
    if(status == 0x1)
    {
        unsigned int val;
        val = TSDATX0; //从 TSDATX0 寄存器中读取 AD 转换结果
        unsigned long *n = valx & (0xFFF << 0)); // 获取 TSDATX0[9:0]的值
        int num = 0;
        num = sscanf(n, "%x", &n); //把 16 进制转为 10 进制
        TCMPB0 = 1000 * (num / 1024);
    }
}


// 主程序
void main()
{
    adc_init();//adc 初始化
    io_init (); //初始化引脚
    timer0_init(); //时钟初始化
    timer1_init(); //时钟初始化
```

```
    init_irq(); //  使能 Timer1 中断
    init_int(); //初始化中断控制器、使能中断
}
```