评分：____

《DSP 原理及应用》
实验报告

学生专业班级 自卓 2201

学 生 姓 名 杨欣怡 学号 U202215067

自动化学院教学实验中心

目录

实验报告内容

一、定时器和 GPIO 实验

1.1 实验要求

(1) 通过定时器实现延时，编写跑马灯程序；
(2) 对主板上的按键 GPIO12 进行采样，添加软件去抖功能；
(3) 实现按键对跑马灯动作的选择，至少两种跑马灯动作；
(4) 学习 TM1638 驱动程序，完成数码管的显示功能。

1.2 实验目的

(1) 掌握 GPIO 端口的配置和操作方法；
(2) 掌握定时器的配置和操作方法；
(3) 掌握定时器中断的配置；
(4) 学会使用 GPIO 实现 TM1638 的驱动，实现数码管的显示功能。

1.3 实验过程

1）跑马灯程序实现

在 Timer.c 中，通过 InitTimer 函数配置了两个 CPU 定时器，定时器 0 用于控制跑马灯的闪烁频率，定时器 1 用于按键去抖功能。在 cpu_timer0_isr 中断服务程序中，根据 Running 变量的状态，控制 ledPattern 变量的增减，从而实现跑马灯的两种动作模式。HorseRunning 函数根据 ledPattern 的值控制四个 LED 灯的亮灭，实现跑马灯效果。

2）按键采样与去抖

在 Xint1.c 中，通过 Xint1_Init 函数配置了 GPIO12 为外部中断 XINT1 的输入源，并设置了低电平触发模式。在 myXint1_isr 中断服务程序中，判断 KeyDLTime 变量是否大于 20，若大于则切换 Running 变量的值，实现按键对跑马灯动作的选择。KeyDLTime 变量在 cpu_timer1_isr 中断服务程序中递增，起到按键去抖的作用。

3）TM1638 驱动与数码管显示

在 LED_TM1638.c 中，通过 TM1638_Init 函数初始化 TM1638 芯片，配置了相关引脚为 GPIO 模式，并发送初始化命令。LED_Show 函数通过调用 TM1638_WriteLED 函数，将数字显示在数码管上。在 main.c 中，根据 Experiment_N 的值，在主循环中调用 LED_Show 函数，将跑马灯的模式或数码管的显示内容更新。

1.4 实验代码

```
void InitTimer(float CPU0_Period, float CPU1_Period)
{
    // 初始化定时器 0 和定时器 1
    CpuTimer0.RegsAddr = &CpuTimer0Regs;
    CpuTimer0Regs.PRD.all = 0xFFFFFFFF;
```

```
    CpuTimer0Regs.TPR.all = 0;
    CpuTimer0Regs.TPRH.all = 0;
    CpuTimer0Regs.TCR.bit.TSS = 1;
    CpuTimer0Regs.TCR.bit.TRB = 1;
    CpuTimer0.InterruptCount = 0;

    CpuTimer1.RegsAddr = &CpuTimer1Regs;
    CpuTimer1Regs.PRD.all = 0xFFFFFFFF;
    CpuTimer1Regs.TPR.all = 0;
    CpuTimer1Regs.TPRH.all = 0;
    CpuTimer1Regs.TCR.bit.TSS = 1;
    CpuTimer1Regs.TCR.bit.TRB = 1;
    CpuTimer1.InterruptCount = 0;

    EALLOW;
    SysCtrlRegs.PCLKCR3.bit.CPUTIMER0ENCLK = 1; // 启用定时器 0 时钟
    SysCtrlRegs.PCLKCR3.bit.CPUTIMER1ENCLK = 1; // 启用定时器 1 时钟
    EDIS;

    ConfigTimer(CPU0_Period, CPU1_Period); // 配置定时器周期
}
```

二、PWM 和 eCAP 实验

2.1 实验要求

(1) 输出带死区的互补 PWM 波形；
(2) 通过 eCAP 单元捕获其中一路 PWM 波形并获取其周期和占空比。

2.2 实验目的

(1) 掌握 PWM 的周期、死区电平和死区时间的设定方法；
(2) 掌握 PWM 计数周期中断和比较值匹配中断的配置方法。
(3) 掌握 eCAP 单元的操作方法及其中断的配置；
(4) 掌握 eCAP 单元时间测量和占空比测量的方法。

2.3 实验过程

1）PWM 波形输出

在 PWM.c 中，通过 InitPWM1 函数配置了 EPWM1 模块，设置了时基周期 PWM1PRD 为 2000，占空比 PWM1Duty 为 500。在 EPWM1Int_isr 中断服务程序中，根据实验编号 Experiment_N 的值，进行不同的操作。当 Experiment_N 为 4 时，通过 PID 控制算法计算新的占空比 PWM1Duty，并更新 EPWM1 的比较值 A 和 B，从而改变 PWM 波形的占空比。

2）eCAP 捕获 PWM 波形

在 CAP.c 中，通过 InitCAP 函数配置了 eCAP1 模块，设置了相关控制寄存器的值。在 Ecap1Int_isr 中断服务程序中，读取 eCAP1 模块的捕获寄存器 CAP1、CAP2、CAP3、CAP4 的值，计算 PWM 波形的高电

平时间 PWM_HI、低电平时间 PWM_LO 和周期 PWM_PRD。

2.4 实验代码

```
void InitPWM1()
{
    // 初始化 PWM1
    EALLOW;
    SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;
    SysCtrlRegs.PCLKCR1.bit.EPWM1ENCLK = 1; // 启用 EPWM1 时钟
    EDIS;

    EPwm1Regs.TBPRD = PWM1PRD; // 设置时基周期
    EPwm1Regs.TBPHS.half.TBPHS = 0;
    EPwm1Regs.TBCTL.bit.CLKDIV = 0;
    EPwm1Regs.TBCTL.bit.HSPCLKDIV = 0;
    EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;
    EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;
    EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
    EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;

    EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO_PRD;
    EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO_PRD;

    EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
    EPwm1Regs.AQCTLA.bit.CAD = AQ_SET;

    PWM1Duty = 500;
    EPwm1Regs.CMPA.half.CMPA = PWM1Duty;
    EPwm1Regs.CMPB = PWM1Duty;

    EPwm1Regs.ETSEL.bit.INTEN = 1;
    EPwm1Regs.ETSEL.bit.INTSEL = 1;
    EPwm1Regs.ETPS.bit.INTPRD = 1;
    EPwm1Regs.ETCLR.bit.INT = 1;

    EALLOW;
    SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1;
    EDIS;
}
```

三、ADC 实验

3.1 实验要求

通过 ADC 读取点位器(目标板上 RP1)中间抽头上的电压值，并将结果在数码管上显示。

3.2 实验目的

(1) 掌握 ADC 的配置和操作方法；
(2) 掌握使用 PWM 触发 AD 采样的方法。

3.3 实验过程

1）ADC 配置与采样

在 ADC.c 中，通过 InitMyADC 函数配置了 ADC 模块，启用了 ADC 的时钟，调用了设备校准函数，配置了多个模拟输入通道（如 AIO2、AIO4 等）。设置了 ADC 中断，当 ADC 转换完成后触发中断。在 MyAdcInt1_isr 中断服务程序中，读取 ADC 结果寄存器的值，计算平均值 ADC_GD，并将其转换为浮点数 ADC_GDF，再根据参考电压计算实际电压值 Value_ADC。

2）数码管显示

在 main.c 中，根据 Experiment_N 的值，在主循环中调用 LED_Show 函数，将 ADC 采样结果 Value_ADC 显示在数码管上。

3.4 实验代码

```
void InitMyADC()
{
    InitADCIntr();

    EALLOW;
    SysCtrlRegs.PCLKCR0.bit.ADCENCLK = 1; // 启用 ADC 时钟
    (*Device_cal)(); // 调用设备校准函数
    EDIS;

    DELAY_US(ADC_usDELAY);

    EALLOW;
    AdcRegs.ADCCTL1.bit.ADCBGPWD = 1;
    AdcRegs.ADCCTL1.bit.ADCREFPWD = 1;
    AdcRegs.ADCCTL1.bit.ADCPWDN = 1;
    AdcRegs.ADCCTL1.bit.ADCENABLE = 1;
    AdcRegs.ADCCTL1.bit.ADCREFSEL = 0;
    EDIS;
```

```
DELAY_US(ADC_usDELAY);

EALLOW;
AdcRegs.ADCCTL1.bit.INTPULSEPOS = 1;
AdcRegs.INTSEL1N2.bit.INT1E = 1;
AdcRegs.INTSEL1N2.bit.INT1CONT = 0;
AdcRegs.INTSEL1N2.bit.INT1SEL = 0x09;

AdcRegs.ADCSOC0CTL.bit.CHSEL = 0x07;
AdcRegs.ADCSOC2CTL.bit.CHSEL = 0x07;
AdcRegs.ADCSOC4CTL.bit.CHSEL = 0x07;
AdcRegs.ADCSOC6CTL.bit.CHSEL = 0x07;
AdcRegs.ADCSOC8CTL.bit.CHSEL = 0x07;

AdcRegs.ADCSOC0CTL.bit.TRIGSEL = TrigSelNo;
AdcRegs.ADCSOC2CTL.bit.TRIGSEL = TrigSelNo;
AdcRegs.ADCSOC4CTL.bit.TRIGSEL = TrigSelNo;
AdcRegs.ADCSOC6CTL.bit.TRIGSEL = TrigSelNo;
AdcRegs.ADCSOC8CTL.bit.TRIGSEL = TrigSelNo;

AdcRegs.ADCSOC0CTL.bit.ACQPS = ADCSampT;
AdcRegs.ADCSOC2CTL.bit.ACQPS = ADCSampT;
AdcRegs.ADCSOC4CTL.bit.ACQPS = ADCSampT;
AdcRegs.ADCSOC6CTL.bit.ACQPS = ADCSampT;
AdcRegs.ADCSOC8CTL.bit.ACQPS = ADCSampT;

EDIS;

EPwm2Regs.ETSEL.bit.SOCAEN = 1;
EPwm2Regs.ETSEL.bit.SOCASEL = 1;
EPwm2Regs.ETPS.bit.SOCAPRD = 1;
```

```
    EPwm2Regs.ETCLR.bit.SOCA = 1;
}
```

四、综合实验

4.1 实验要求

(1) 通过 ADC 获取目标板上电位器 RP1 中间抽头的电压，并实时显示在左边数码管上；
(2) 将（1）中得到的采样结果转换成 PWM 的占空比输出到目标板上，将 PWM 信号低通滤波得到与 PWM 占空比成正比关系的电压，相当于 DAC。
(3) 通过另一路 ADC 对（2）中 DAC 得到的电压进行采样，并实时显示在右边数码管上。
(4) 编制控制程序，实现 DAC 输出电压对电位器 RP1 中间抽头电位的无差跟踪。

4.2 实验目的

本实验综合所学的 DSP 的知识和前面做过的实验，设计一个涉及到多个外设的嵌入式系统，要求能够熟练配置系统时钟和各个外设。

考察对芯片多个外设的配置的掌握情况，考察学生的综合设计和嵌入式开发的能力。

4.3 实验过程

1）ADC 配置与采样

在 ADC.c 中，通过 InitMyADC 函数配置了 ADC 模块，启用了 ADC 的时钟，调用了设备校准函数，配置了多个模拟输入通道（如 AIO2、AIO4 等）。设置了 ADC 中断，当 ADC 转换完成后触发中断。在 MyAdcInt1_isr 中断服务程序中，读取 ADC 结果寄存器的值，计算平均值 ADC_GD，并将其转换为浮点数 ADC_GDF，再根据参考电压计算实际电压值 Value_ADC。

2）PWM 输出与 DAC 实现

在 PWM.c 中，通过 InitPWM1 函数配置了 EPWM1 模块，将 ADC 采样结果 Value_ADC 转换为 PWM 占空比 PWM1Duty，并输出 PWM 波形。通过低通滤波器将 PWM 信号转换为与占空比成正比的电压，实现 DAC 功能。

3）ADC 采样 DAC 输出电压

在 ADC.c 中，通过 InitMyADC 函数配置了另一路 ADC 通道，对 DAC 输出的电压进行采样，计算实际电压值 Value_PWM。在 main.c 中，将 Value_PWM 显示在右边数码管上。

4）PID 控制实现无差跟踪

在 EPWM1Int_isr 中断服务程序中，通过 PID 控制算法计算新的占空比 PWM1Duty，使 DAC 输出电压 Value_PWM 能够无差跟踪电位器中间抽头的电压 Value_ADC。PID 控制参数 kp、ki、kd 根据实验需要进行调整。

4.4 实验代码

```
void main(void)
{
```

```
    // 初始化系统
    InitSysCtrl();
    DINT;
    IER = 0x0000;
    IFR = 0x0000;
    InitPieCtrl();
    InitPieVectTable();
    EINT;
    MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);
    InitFlash();

// 根据实验编号初始化外设
switch(Experiment_N)
{
case 1: // 定时器和 GPIO 实验
    Running = 1;
    InitTimer(500000, 20000);
    Xint1_Init();
    HorseIO_Init();
    TM1638_Init();
    break;

case 2: // PWM 和 eCAP 实验
    InitPWM1();
    InitCAP();
    TM1638_Init();
    break;

case 3: // ADC 实验
    InitPWM2();
    InitMyADC();
```

```
        TM1638_Init();
        break;

case 4: // 综合实验
        InitPWM1();
        InitPWM2();
        InitMyADC();
        TM1638_Init();
        break;

default:
        break;
}

int value_int;
int HI;
int PRD;

ERTM; // 启动实时多任务
while(1)
{
        switch(Experiment_N)
        {
        case 1: // 定时器和 GPIO 实验
                LED_Show(1, (ledPattern % 10), 0);
                LED_Show(2, (ledPattern / 10), 0);
                break;

        case 2: // PWM 和 eCAP 实验
                HI = PWM_HI * 1000 / PWM_PRD;
                LED_Show(1, (HI % 10), 0);
```

```
        HI /= 10;
        LED_Show(2, (HI % 10), 0);
        HI /= 10;
        LED_Show(3, (HI % 10), 0);
        HI /= 10;
        LED_Show(4, (HI % 10), 1);

        PRD = PWM_PRD;
        LED_Show(5, (PRD % 10), 0);
        PRD /= 10;
        LED_Show(6, (PRD % 10), 0);
        PRD /= 10;
        LED_Show(7, (PRD % 10), 0);
        PRD /= 10;
        LED_Show(8, (PRD % 10), 0);
        break;

    case 3: // ADC 实验
        value_int = Value_ADC * 1000;
        LED_Show(5, (value_int % 10), 0);
        value_int /= 10;
        LED_Show(6, (value_int % 10), 0);
        value_int /= 10;
        LED_Show(7, (value_int % 10), 0);
        value_int /= 10;
        LED_Show(8, (value_int % 10), 1);
        break;

    case 4: //  综合实验
        value_int = Value_ADC * 1000;
        LED_Show(5, (value_int % 10), 0);
```

```
            value_int /= 10;
            LED_Show(6, (value_int % 10), 0);
            value_int /= 10;
            LED_Show(7, (value_int % 10), 0);
            value_int /= 10;
            LED_Show(8, (value_int % 10), 1);

            value_int = Value_PWM * 1000;
            LED_Show(1, (value_int % 10), 0);
            value_int /= 10;
            LED_Show(2, (value_int % 10), 0);
            value_int /= 10;
            LED_Show(3, (value_int % 10), 0);
            value_int /= 10;
            LED_Show(4, (value_int % 10), 1);
            break;

        default:
            break;
        }
        delay(3000);
}
```