```c
/*
 * main.c
 */
#include "DSP28x_Project.h"
#include "LED_TM1638.h"


// 函数的声明
void HorseRunning(int16 no);
interrupt void myXint1_isr(void);
void HorseIO_Init();
interrupt void cpu_timer0_isr(void);
void DelaymS(int tm);
void HorseRunning(int no);
void Horse2(int no);
//

#define Led0Blink() GpioDataRegs.GPACLEAR.bit.GPIO0 = 1
#define Led1Blink() GpioDataRegs.GPACLEAR.bit.GPIO1 = 1
#define Led2Blink() GpioDataRegs.GPACLEAR.bit.GPIO2 = 1
#define Led3Blink() GpioDataRegs.GPACLEAR.bit.GPIO3 = 1
#define Led0Blank() GpioDataRegs.GPASET.bit.GPIO0 = 1
#define Led1Blank() GpioDataRegs.GPASET.bit.GPIO1 = 1
#define Led2Blank() GpioDataRegs.GPASET.bit.GPIO2 = 1
#define Led3Blank() GpioDataRegs.GPASET.bit.GPIO3 = 1

//变量的定义
int hourH = 0, hourL = 0;
int minH = 0, minL = 0;
int secH = 0, secL = 0, TenmS = 0;
int Running = 0;    //0=等待 1= 计时 2= 停止
int NewLedEn = 0;   //0=  显示更新允许  1= 已经更新
int KeyDLTime = 0;  //按键去抖动用
int LedFlashCtr;    //用显示更新和跑马灯控制
int zoumadeng; //走马灯


void main(void)
{
    int j;
    InitSysCtrl();      //初始化系统时钟，选择内部晶振1，10MHZ，12倍频，2分频，初始化外设时钟，低速外设,4分频
    DINT;               //关总中断
    IER = 0x0000;       //关CPU中断使能
    IFR = 0x0000;       //清CPU中断标志
    InitPieCtrl();      //关pie中断
    InitPieVectTable(); //清中断向量表

    EALLOW;             /**配置中断向量表*****/
    PieVectTable.TINT0 = &cpu_timer0_isr;
    PieVectTable.XINT1 = &myXint1_isr;
    EDIS;

    //  MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);
    InitFlash();

    InitCpuTimers();    // 初始化定时器

    /*********自己添加部分**********/
```

```
59      ConfigCpuTimer(&CpuTimer0, 60, 10000); //初始化时钟
60      EALLOW;
61      CpuTimer0Regs.TCR.bit.TSS = 0; //重载
62      CpuTimer0Regs.TCR.bit.TRB = 1; //启动
63      CpuTimer0.InterruptCount = 0;
64      EDIS;
65
66
67      HorseIO_Init();
68      Xint1_Init();
69      TM1638_Init();      //初始化LED
70
71
72      //自己添加 实验1
73      PieCtrlRegs.PIECTRL.bit.ENPIE = 1;
74      PieCtrlRegs.PIEIER1.bit.INTx7 = 1; //TINT0 CPU 定时器0
75      PieCtrlRegs.PIEIER1.bit.INTx4 = 1; //TINT1 XINT1
76
77
78      IER |= M_INT1;  //使能 CPU 中断
79
80      EINT;
81      ERTM;
82      zoumadeng = 0;
83      while(1){
84        DelaymS(20);
85        j++;
86        j = j & 0xf;
87  //      HorseRunning(j);
88        /*实验1*/
89        if (NewLedEn == 0){
90            LED_Show(1,(TenmS % 10),0);
91            LED_Show(2,(TenmS / 10),0);
92            LED_Show(3,secL,1);
93            LED_Show(4,secH,0);
94            LED_Show(5,minL,1);
95            LED_Show(6,minH,0);
96            LED_Show(7,hourL,1);
97            LED_Show(8,hourH,0);
98            NewLedEn = 1;
99        }
100       }
101 }
102
103
104
105
106
107 void  Xint1_Init()
108 {
109     EALLOW;
110     //自己添加部分
111     GpioCtrlRegs.GPAMUX1.bit.GPIO12 = 0;
112     GpioCtrlRegs.GPADIR.bit.GPIO12 = 0;
113     GpioCtrlRegs.GPAPUD.bit.GPIO12 = 0;
114     GpioIntRegs.GPIOXINT1SEL.bit.GPIOSEL = 12;
115     XIntruptRegs.XINT1CR.bit.POLARITY = 0; // 00 or 10下降沿 01 上升沿 11都可以
116     XIntruptRegs.XINT1CR.bit.ENABLE = 1; //使能
117     //
118     EDIS;
```

```
}


interrupt void myXint1_isr(void)
{
    //自己添加
    //Running  0=等待 1= 计时 2= 停止
    if((Running == 0)&&(KeyDLTime > 20)){
        EALLOW;
        CpuTimer0Regs.TCR.bit.TSS = 0;//停止计时器
        CpuTimer0Regs.TCR.bit.TRB = 1; // Reload Period
        CpuTimer0Regs.TCR.bit.TSS = 0; // Start Timer
        EDIS;
        zoumadeng = 1;
        Running = 1;
        KeyDLTime = 0;
    }
    else if((Running == 1)&&(KeyDLTime > 20)){
        zoumadeng = 0;
        Running = 2;
        KeyDLTime = 0;
    }
    else if((Running == 2)&&(KeyDLTime > 20)) {
        zoumadeng = 0;
        Running = 0;
        hourH=0;hourL=0;minH=0;minL=0;
        secH=0;secL=0;TenmS = 0;
        KeyDLTime = 0;
    }
    //
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;// Clear PIEACK
}


void HorseIO_Init()
{
  EALLOW;
    GpioDataRegs.GPASET.bit.GPIO0 = 1;
    GpioDataRegs.GPASET.bit.GPIO1 = 1;
    GpioDataRegs.GPASET.bit.GPIO2 = 1;
    GpioDataRegs.GPASET.bit.GPIO3 = 1;
    GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 0;
    GpioCtrlRegs.GPADIR.bit.GPIO0 = 1;
    GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 0;
    GpioCtrlRegs.GPADIR.bit.GPIO1 = 1;
    GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 0;
    GpioCtrlRegs.GPADIR.bit.GPIO2 = 1;
    GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 0;
    GpioCtrlRegs.GPADIR.bit.GPIO3 = 1;
    EDIS;

}

interrupt void cpu_timer0_isr(void) {
    //自己添加
    KeyDLTime++;
    LedFlashCtr++;
    if((LedFlashCtr & 0xf)==0)
        NewLedEn = 0;
    if(Running == 1){
```

```
179         TenmS++;
180         if(TenmS == 100){ TenmS = 0; secL++; }
181         if(secL==10){ secL=0; secH++; }
182         if(secH==6){ secH=0; minL++; }
183         if(minL==10){ minL=0; minH++; }
184         if(minH==6){ minH=0; hourL++; }
185         if(hourL==4 && hourH==2){ hourL=0; hourH=0; }
186         else if(hourL==10){ hourL=0; hourH++; }
187         HorseRunning((LedFlashCtr &0xf0)>>4);
188     }
189     else{
190         Horse2((LedFlashCtr &0xf0)>>4);
191     }
192     //
193     PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
194 }
195
196
197 void DelaymS(int tm)
198 {
199   int i;
200   unsigned int j;
201     for(i = 0;i < tm ;i++){
202       j =  60000;
203       while(j != 0)j--;
204       }
205 }
206
207 void HorseRunning(int no)
208 {
209     if(no & 0x1)Led0Blink();
210     else Led0Blank();
211     if(no & 0x2)Led1Blink();
212     else Led1Blank();
213     if(no & 0x4)Led2Blink();
214     else Led2Blank();
215     if(no & 0x8)Led3Blink();
216     else Led3Blank();
217 }
218 void Horse2(int no){
219     if(no & 0xf){
220             Led0Blink();
221             Led1Blink();
222             Led3Blink();
223             Led2Blink();
224         }
225         else {
226             Led0Blank();
227             Led1Blank();
228             Led3Blank();
229             Led2Blank();
230         }
231
232 }
233
234
235
```

```c
/*
 * main2.c
 *
 *  Created on: 2023年4月5日
 *      Author: Asus
 */

#include "DSP28x_Project.h"
#include "LED_TM1638.h"

void HorseRunning(int16 no);

//变量的定义
int Running = 0;    //0=等待 1= 计时 2= 停止
int NewLedEn = 0;   //0=  显示更新允许  1= 已经更新
int KeyDLTime = 0;  //按键去抖动用
int LedFlashCtr;    //用显示更新和跑马灯控制

int li1, li2, li3, li4;
int PWM_HI, PWM_LO, PWM_PRD;
int PWMTimes, PWMPRD;
int Tirdir, PWMDuty;


// 函数的声明
interrupt void cpu_timer0_isr(void);


void InitCAP(void);
void InitCAPGpio(void);
void InitPWM4 (void);
interrupt void Ecap1Int_isr(void);
interrupt void EPWM4Int_isr(void);
//

#define Led0Blink() GpioDataRegs.GPACLEAR.bit.GPIO0 = 1
#define Led1Blink() GpioDataRegs.GPACLEAR.bit.GPIO1 = 1
#define Led2Blink() GpioDataRegs.GPACLEAR.bit.GPIO2 = 1
#define Led3Blink() GpioDataRegs.GPACLEAR.bit.GPIO3 = 1
#define Led0Blank() GpioDataRegs.GPASET.bit.GPIO0 = 1
#define Led1Blank() GpioDataRegs.GPASET.bit.GPIO1 = 1
#define Led2Blank() GpioDataRegs.GPASET.bit.GPIO2 = 1
#define Led3Blank() GpioDataRegs.GPASET.bit.GPIO3 = 1


void  Xint1_Init()
{
    EALLOW;
    //自己添加部分
    GpioCtrlRegs.GPAMUX1.bit.GPIO12 = 0;
    GpioCtrlRegs.GPADIR.bit.GPIO12 = 0;
    GpioCtrlRegs.GPAPUD.bit.GPIO12 = 0;
    GpioIntRegs.GPIOXINT1SEL.bit.GPIOSEL = 12;
    XIntruptRegs.XINT1CR.bit.POLARITY = 0; // 00 or 10下降沿 01 上升沿 11都可以
    XIntruptRegs.XINT1CR.bit.ENABLE = 1; //使能
    //
    EDIS;
}

```

```
60
61 void HorseIO_Init()
62 {
63     EALLOW;
64     GpioDataRegs.GPASET.bit.GPIO0 = 1;
65     GpioDataRegs.GPASET.bit.GPIO1 = 1;
66     GpioDataRegs.GPASET.bit.GPIO2 = 1;
67     GpioDataRegs.GPASET.bit.GPIO3 = 1;
68     GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 0;
69     GpioCtrlRegs.GPADIR.bit.GPIO0 = 1;
70     GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 0;
71     GpioCtrlRegs.GPADIR.bit.GPIO1 = 1;
72     GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 0;
73     GpioCtrlRegs.GPADIR.bit.GPIO2 = 1;
74     GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 0;
75     GpioCtrlRegs.GPADIR.bit.GPIO3 = 1;
76     EDIS;
77
78 }
79
80 interrupt void cpu_timer0_isr(void) {
81     //自己添加
82
83 }
84
85
86 void DelaymS(int tm)
87 {
88   int i;
89   unsigned int j;
90     for(i = 0;i < tm ;i++){
91       j =  60000;
92       while(j != 0)j--;
93       }
94 }
95
96
97 void InitCAP(void)
98 {
99     //实验2
100     ECap1Regs.ECEINT.all = 0x0000; // Disable all capture interrupts
101     ECap1Regs.ECCLR.all = 0xFFFF; // Clear all CAP interrupt flags
102     ECap1Regs.ECCTL1.bit.CAP1POL = 0;
103     ECap1Regs.ECCTL1.bit.CAP2POL = 1;
104     ECap1Regs.ECCTL1.bit.CAP3POL = 0;
105     ECap1Regs.ECCTL1.bit.CAP4POL = 1;
106     ECap1Regs.ECCTL1.bit.CTRRST1 = 0;
107     ECap1Regs.ECCTL1.bit.CTRRST2 = 0;
108     ECap1Regs.ECCTL1.bit.CTRRST3 = 0;
109     ECap1Regs.ECCTL1.bit.CTRRST4 = 0;
110     ECap1Regs.ECCTL1.bit.CAPLDEN = 1;
111     ECap1Regs.ECCTL1.bit.PRESCALE = 0;
112     ECap1Regs.ECCTL2.bit.CAP_APWM = 0;
113     ECap1Regs.ECCTL2.bit.CONT_ONESHT = 0;
114     ECap1Regs.ECCTL2.bit.SYNCO_SEL = 2;
115     ECap1Regs.ECCTL2.bit.SYNCI_EN = 0;
116     ECap1Regs.ECCTL2.bit.TSCTRSTOP = 1; //允许 TSCTR
117     ECap1Regs.ECEINT.bit.CEVT4 = 1; // CEVT4
118 }
119
```

```
120 void InitCAPGpio(void)
121 {
122     //实验2
123     EALLOW;
124     GpioCtrlRegs.GPAPUD.bit.GPIO5 = 0; // Enable pull up on GPIO5 (ECAP1)
125     GpioCtrlRegs.GPAMUX1.bit.GPIO5 = 3; // Configure GPIO5 as ECAP1
126     EDIS;
127 }
128
129 interrupt void Ecap1Int_isr(void)
130 {
131     //实验2
132     li1=ECap1Regs.CAP1;
133     li2=ECap1Regs.CAP2;
134     li3=ECap1Regs.CAP3;
135     li4=ECap1Regs.CAP4;
136     PWM_HI=((li2 - li1)+(li4 - li3)) >> 1;
137     PWM_LO=li3-li2;
138     PWM_PRD=((li3 - li1)+(li4 - li2)) >> 1;
139     ECap1Regs.ECCLR.bit.CEVT4 = 1;
140     ECap1Regs.ECCLR.bit.INT = 1;
141     PieCtrlRegs.PIEACK.all = PIEACK_GROUP4;
142 }
143
144 void InitPWM4 (void){
145     //实验2
146     int PWMPRD, DeadTime;
147     EALLOW;
148     SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;
149     GpioCtrlRegs.GPAPUD.bit.GPIO6 = 1; // 上拉禁能
150     GpioCtrlRegs.GPAMUX1.bit.GPIO6 = 1; //
151     EDIS;
152     PWMPRD = 3000; // 10 kHz
153     DeadTime = 120;
154     EPwm4Regs.TBPRD = PWMPRD; //时基寄存器
155     EPwm4Regs.TBPHS.half.TBPHS = 0; //设置相位寄存器为 0
156     EPwm4Regs.TBCTL.bit.CLKDIV = 0;//时基时钟预分频位 =0 默认为1
157     EPwm4Regs.TBCTL.bit.HSPCLKDIV = 0; //高速时基预分频位
158     EPwm4Regs.TBCTL.bit.CTRMODE = 2; //计数模式 =0递增 =1 递减 = 2先递增再递减 =3 停止
    （默认）
159     EPwm4Regs.TBCTL.bit.PHSEN = 0;// 主模式
160     EPwm4Regs.TBCTL.bit.PRDLD = 0;//
161     EPwm4Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;
162     EPwm4Regs.CMPCTL.bit.SHDWAMODE = 0; //映射模式
163     EPwm4Regs.CMPCTL.bit.SHDWBMODE = 0;
164     EPwm4Regs.CMPCTL.bit.LOADAMODE = 2; //CTR=0 or CTR=PRD装载
165     EPwm4Regs.CMPCTL.bit.LOADBMODE = 2;
166     EPwm4Regs.AQCTLA.bit.CAU = 1;//递增强制变低
167     EPwm4Regs.AQCTLA.bit.CAD = 2;//递减 强制变高
168     EPwm4Regs.AQCTLA.bit.CBU = 0;//无动作
169     EPwm4Regs.AQCTLA.bit.CBD = 0;
170     EPwm4Regs.DBCTL.bit.OUT_MODE = 3; //死区使能
171     EPwm4Regs.DBCTL.bit.POLSEL = 2; //高电平互补有效
172     EPwm4Regs.DBFED = DeadTime;
173     EPwm4Regs.DBRED = DeadTime;
174
175     EPwm4Regs.CMPA.half.CMPA = PWMPRD/2;
176     EPwm4Regs.CMPB = PWMPRD/2;//
177
178     // Enable CNT_zero interrupt using EPWM 4 Time base
```

```
179     EPwm4Regs.ETSEL.bit.INTEN = 1; //使能 EPWM 4 中断
180     EPwm4Regs.ETSEL.bit.INTSEL = 1; //CNT_zero 时中断
181
182     EPwm4Regs.ETPS.bit.INTPRD = 1; //在第一个事件处触发中断
183     EPwm4Regs.ETCLR.bit.INT = 1; //清中断标志
184     EALLOW;
185     SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1; //同步
186     EDIS;
187 }
188
189 interrupt void EPWM4Int_isr(void){
190     //实验2
191     EPwm4Regs.ETCLR.bit.INT = 1;
192     PieCtrlRegs.PIEACK.all = PIEACK_GROUP3;
193     EINT;
194 }
195 void HorseRunning(int no)
196 {
197     if(no & 0x1)Led0Blink();
198     else Led0Blank();
199     if(no & 0x2)Led1Blink();
200     else Led1Blank();
201     if(no & 0x4)Led2Blink();
202     else Led2Blank();
203     if(no & 0x8)Led3Blink();
204     else Led3Blank();
205 }
206
207
208 void main(void)
209 {
210     int j;
211     InitSysCtrl();       //初始化系统时钟，选择内部晶振1，10MHZ，12倍频，2分频，初始化外设
    时钟，低速外设,4分频
212     DINT;                //关总中断
213     IER = 0x0000;        //关CPU中断使能
214     IFR = 0x0000;        //清CPU中断标志
215     InitPieCtrl();       //关pie中断
216     InitPieVectTable(); //清中断向量表
217
218     EALLOW;              /**配置中断向量表*****/
219 //    PieVectTable.TINT0 = &cpu_timer0_isr;
220 //    PieVectTable.XINT1 = &myXint1_isr;
221     PieVectTable.ECAP1_INT = &Ecap1Int_isr;
222     PieVectTable.EPWM4_INT = &EPWM4Int_isr;
223 //    PieVectTable.ADCINT1 = &MyAdcInt1_isr;
224     EDIS;
225
226     InitFlash();
227
228
229     HorseIO_Init();
230 //    Xint1_Init();
231     TM1638_Init();       //初始化LED
232
233     //实验2
234     InitCAP();
235     InitCAPGpio();
236     InitPWM4 ();
237
```

```
238     PieCtrlRegs.PIECTRL.bit.ENPIE = 1;
239
240     //实验2
241     PieCtrlRegs.PIEIER3.bit.INTx4 = 1;       // EPWM4
242     PieCtrlRegs.PIEIER4.bit.INTx1 = 1;       // ECAP1
243
244 //    IER |= M_INT1;  //使能 CPU 中断
245     IER |= M_INT3;      //使能 CPU 中断
246     IER |= M_INT4;     //使能 CPU 中断
247
248     EINT;
249     ERTM;
250
251     while(1){
252       DelaymS(20);
253       j++;
254       j = j & 0xf;
255       HorseRunning(j);
256
257       LED_Show(1,(PWM_HI % 10),0);
258       LED_Show(2,((PWM_HI / 10)% 10),0);
259       LED_Show(3,((PWM_HI / 100)% 10),0);
260       LED_Show(4,((PWM_HI / 1000)% 10),0);
261       LED_Show(1+4,(PWM_PRD % 10),1);
262       LED_Show(2+4,((PWM_PRD / 10)% 10),0);
263       LED_Show(3+4,((PWM_PRD / 100)% 10),0);
264       LED_Show(4+4,((PWM_PRD / 1000)% 10),0);
265     }
266 }
267
268
269
270
271
```

```c
/*
 * main3.c
 *
 *  Created on: 2023年4月6日
 *      Author: Asus
 */

#include "DSP28x_Project.h"
#include "LED_TM1638.h"

float ADCA7;
float ADCB7;

void HorseRunning(int16 no);

// 函数的声明
interrupt void myXint1_isr(void);
interrupt void cpu_timer0_isr(void);
interrupt void Ecap1Int_isr(void);
interrupt void EPWM1Int_isr(void);
interrupt void MyAdcInt1_isr(void);
//
#define ADC_usDELAY 1000L
#define DELAY_US(A) DSP28x_usDelay(((((long double)A * 1000.0L) / (long double)CPU_RATE) - 9.0L) / 5.0L)
#define Led0Blink() GpioDataRegs.GPACLEAR.bit.GPIO0 = 1
#define Led1Blink() GpioDataRegs.GPACLEAR.bit.GPIO1 = 1
#define Led2Blink() GpioDataRegs.GPACLEAR.bit.GPIO2 = 1
#define Led3Blink() GpioDataRegs.GPACLEAR.bit.GPIO3 = 1
#define Led0Blank() GpioDataRegs.GPASET.bit.GPIO0 = 1
#define Led1Blank() GpioDataRegs.GPASET.bit.GPIO1 = 1
#define Led2Blank() GpioDataRegs.GPASET.bit.GPIO2 = 1
#define Led3Blank() GpioDataRegs.GPASET.bit.GPIO3 = 1

void Xint1_Init()
{
    EALLOW;
    // 自己添加部分
    //
    EDIS;
}

void HorseIO_Init()
{
    EALLOW;
    GpioDataRegs.GPASET.bit.GPIO0 = 1;
    GpioDataRegs.GPASET.bit.GPIO1 = 1;
    GpioDataRegs.GPASET.bit.GPIO2 = 1;
    GpioDataRegs.GPASET.bit.GPIO3 = 1;
    GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 0;
    GpioCtrlRegs.GPADIR.bit.GPIO0 = 1;
    GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 0;
    GpioCtrlRegs.GPADIR.bit.GPIO1 = 1;
    GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 0;
    GpioCtrlRegs.GPADIR.bit.GPIO2 = 1;
    GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 0;
    GpioCtrlRegs.GPADIR.bit.GPIO3 = 1;
    EDIS;
}
```

```
59
60 void DelaymS(int tm)
61 {
62     int i;
63     unsigned int j;
64     for (i = 0; i < tm; i++)
65     {
66         j = 60000;
67         while (j != 0)
68             j--;
69     }
70 }
71
72 void InitCAP(void)
73 {
74     // 实验2
75     ECap1Regs.ECEINT.all = 0x0000; // Disable all capture interrupts
76     ECap1Regs.ECCLR.all = 0xFFFF;  // Clear all CAP interrupt flags
77     ECap1Regs.ECCTL1.bit.CAP1POL = 0;
78     ECap1Regs.ECCTL1.bit.CAP2POL = 1;
79     ECap1Regs.ECCTL1.bit.CAP3POL = 0;
80     ECap1Regs.ECCTL1.bit.CAP4POL = 1;
81     ECap1Regs.ECCTL1.bit.CTRRST1 = 0;
82     ECap1Regs.ECCTL1.bit.CTRRST2 = 0;
83     ECap1Regs.ECCTL1.bit.CTRRST3 = 0;
84     ECap1Regs.ECCTL1.bit.CTRRST4 = 0;
85     ECap1Regs.ECCTL1.bit.CAPLDEN = 1;
86     ECap1Regs.ECCTL1.bit.PRESCALE = 0;
87     ECap1Regs.ECCTL2.bit.CAP_APWM = 0;
88     ECap1Regs.ECCTL2.bit.CONT_ONESHT = 0;
89     ECap1Regs.ECCTL2.bit.SYNCO_SEL = 2;
90     ECap1Regs.ECCTL2.bit.SYNCI_EN = 0;
91     ECap1Regs.ECCTL2.bit.TSCTRSTOP = 1; // 允许 TSCTR
92     ECap1Regs.ECEINT.bit.CEVT4 = 1;     // CEVT4
93 }
94
95 void InitCAPGpio(void)
96 {
97     // 实验2
98     EALLOW;
99     EDIS;
100 }
101
102 void InitPWM1(void)
103 {
104     // 实验2
105     int PWMPRD, DeadTime;-
106     EALLOW;
107     SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;
108     GpioCtrlRegs.GPAPUD.bit.GPIO6 = 1;  // 上拉禁能
109     GpioCtrlRegs.GPAMUX1.bit.GPIO6 = 1; //
110     EDIS;
111     PWMPRD = 3000; // 10 kHz
112     DeadTime = 120;
113     EPwm1Regs.TBPRD = PWMPRD;       // 时基寄存器
114     EPwm1Regs.TBPHS.half.TBPHS = 0; // 设置相位寄存器为 0
115
116     EPwm1Regs.TBCTL.bit.CLKDIV = 0;    // 时基时钟预分频位 =0 默认为1
117     EPwm1Regs.TBCTL.bit.HSPCLKDIV = 0; // 高速时基预分频位
```

```
118     EPwm1Regs.TBCTL.bit.CTRMODE = 2;    // 计数模式 =0递增 =1 递减 = 2先递增再递减 =3 停止（默认）
119     EPwm1Regs.TBCTL.bit.PHSEN = 0;      // 主模式
120     EPwm1Regs.TBCTL.bit.PRDLD = 0;      //
121     EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;
122
123     EPwm1Regs.CMPCTL.bit.SHDWAMODE = 0; // 映射模式，使能CMPA寄存器
124     EPwm1Regs.CMPCTL.bit.SHDWBMODE = 0;
125     EPwm1Regs.CMPCTL.bit.LOADAMODE = 2; // CTR=0 or CTR=PRD装载
126     EPwm1Regs.CMPCTL.bit.LOADBMODE = 2;
127
128     EPwm1Regs.AQCTLA.bit.CAU = 1; // 递增强制变低
129     EPwm1Regs.AQCTLA.bit.CAD = 2; // 递减 强制变高
130     EPwm1Regs.AQCTLA.bit.CBU = 0; // 无动作
131     EPwm1Regs.AQCTLA.bit.CBD = 0;
132
133     EPwm1Regs.DBCTL.bit.OUT_MODE = 3; // 死区使能，延迟的输入信号有IN_MODE 决定
134     EPwm1Regs.DBCTL.bit.IN_MODE = 0;  // 为0表示EPWMxA 是下降与上升延迟的源—参见AQCTLA的设置
135     EPwm1Regs.DBCTL.bit.POLSEL = 2;   // 高电平互补有效， EPWMxB反相
136
137     EPwm1Regs.DBFED = DeadTime; // 死区发生上升延迟
138     EPwm1Regs.DBRED = DeadTime; // 死区发生下降延迟
139
140     EPwm1Regs.CMPA.half.CMPA = PWMPRD / 2;
141     EPwm1Regs.CMPB = PWMPRD / 2; //
142
143     // Enable CNT_zero interrupt using EPWM 1 Time base
144     EPwm1Regs.ETSEL.bit.INTEN = 1;  // 使能 EPWM x 中断
145     EPwm1Regs.ETSEL.bit.INTSEL = 1; // CNT_zero  时中断
146
147     EPwm1Regs.ETPS.bit.INTPRD = 1; // 在第一个事件处触发中断
148     EPwm1Regs.ETCLR.bit.INT = 1;   // 清中断标志
149     EALLOW;
150     SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1; // 同步
151     EDIS;
152 }
153
154 void HorseRunning(int no)
155 {
156     if (no & 0x1)
157         Led0Blink();
158     else
159         Led0Blank();
160     if (no & 0x2)
161         Led1Blink();
162     else
163         Led1Blank();
164     if (no & 0x4)
165         Led2Blink();
166     else
167         Led2Blank();
168     if (no & 0x8)
169         Led3Blink();
170     else
171         Led3Blank();
172 }
173
174 void InitADC()
175 {
```

```
176     int ADCSampT;
177     ADCSampT = 15;
178     EALLOW;
179     SysCtrlRegs.PCLKCR0.bit.ADCENCLK = 1;
180     (*Device_cal)();
181     EDIS;
182     DELAY_US(ADC_usDELAY);
183     EALLOW;
184     AdcRegs.ADCCTL1.bit.ADCBGPWD = 1;
185     AdcRegs.ADCCTL1.bit.ADCREFPWD = 1;
186     AdcRegs.ADCCTL1.bit.ADCPWDN = 1;
187     AdcRegs.ADCCTL1.bit.ADCENABLE = 1;
188     AdcRegs.ADCCTL1.bit.ADCREFSEL = 0;
189     EDIS;
190     DELAY_US(ADC_usDELAY);
191     AdcOffsetSelfCal();
192     DELAY_US(ADC_usDELAY);
193     EALLOW;
194     AdcRegs.ADCCTL1.bit.INTPULSEPOS = 1;
195
196     AdcRegs.INTSEL1N2.bit.INT1E = 1;
197     AdcRegs.INTSEL1N2.bit.INT1CONT = 0;
198     AdcRegs.INTSEL1N2.bit.INT1SEL = 7;
199     AdcRegs.ADCSAMPLEMODE.all = 0x0f;
200     AdcRegs.ADCSOC0CTL.bit.CHSEL = 0x07;
201     AdcRegs.ADCSOC2CTL.bit.CHSEL = 0x07;
202     AdcRegs.ADCSOC4CTL.bit.CHSEL = 0x07;
203     AdcRegs.ADCSOC6CTL.bit.CHSEL = 0x07;
204
205     AdcRegs.ADCSOC0CTL.bit.TRIGSEL = 5;
206     AdcRegs.ADCSOC2CTL.bit.TRIGSEL = 5;
207     AdcRegs.ADCSOC4CTL.bit.TRIGSEL = 5;
208     AdcRegs.ADCSOC6CTL.bit.TRIGSEL = 5;
209
210     AdcRegs.ADCSOC0CTL.bit.ACQPS = ADCSampT;
211     AdcRegs.ADCSOC2CTL.bit.ACQPS = ADCSampT;
212     AdcRegs.ADCSOC4CTL.bit.ACQPS = ADCSampT;
213     AdcRegs.ADCSOC6CTL.bit.ACQPS = ADCSampT;
214
215     AdcRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;
216     EDIS;
217     EPwm1Regs.ETSEL.bit.SOCAEN = 1;
218     EPwm1Regs.ETSEL.bit.SOCASEL = 1;
219     EPwm1Regs.ETPS.bit.SOCAPRD = 1;
220     EPwm1Regs.ETCLR.bit.SOCA = 1;
221 }
222
223 void main(void)
224 {
225     int j;
226     // 实验3
227     int voltage1;
228     int voltage2;
229
230     InitSysCtrl();       // 初始化系统时钟，选择内部晶振1，10MHZ，12倍频，2分频，初始化外
    设时钟，低速外设,4分频
231     DINT;                // 关总中断
232     IER = 0x0000;        // 关CPU中断使能
233     IFR = 0x0000;        // 清CPU中断标志
234     InitPieCtrl();       // 关pie中断
```

```
235     InitPieVectTable(); // 清中断向量表
236
237     EALLOW; /**配置中断向量表*****/
238     PieVectTable.EPWM1_INT = &EPWM1Int_isr;
239     PieVectTable.ADCINT1 = &MyAdcInt1_isr;
240     EDIS;
241
242     InitFlash();
243
244     InitCpuTimers(); // 初始化定时器
245
246     HorseIO_Init();
247     TM1638_Init(); // 初始化LED
248
249     // 实验3
250     InitPWM1();
251     InitADC();
252
253     PieCtrlRegs.PIECTRL.bit.ENPIE = 1;
254
255     // 实验2
256     PieCtrlRegs.PIEIER1.bit.INTx1 = 1; // ADC
257     PieCtrlRegs.PIEIER3.bit.INTx1 = 1;
258
259     IER |= M_INT3; // 使能 CPU 中断
260     IER |= M_INT1; // 使能 CPU 中断
261
262     EINT;
263     ERTM;
264
265     while (1)
266     {
267         DelaymS(20);
268         j++;
269         j = j & 0xf;
270         HorseRunning(j);
271
272         voltage1 = (int)(ADCA7 / 4 / 4096 * 3.3 * 1000);
273         voltage2 = (int)(ADCB7 / 4 / 4096 * 3.3 * 1000);
274
275         LED_Show(1, (voltage1 % 10), 0);
276         LED_Show(2, (voltage1 % 100 / 10), 0);
277         LED_Show(3, (voltage1 % 1000 / 100), 0);
278         LED_Show(4, (voltage1 / 1000), 1);
279         LED_Show(5, voltage2 % 10, 0);
280         LED_Show(6, (voltage2 % 100 / 10), 0);
281         LED_Show(7, (voltage2 % 1000 / 100), 0);
282         LED_Show(8, (voltage2 / 1000), 1);
283     }
284 }
285
286 interrupt void myXint1_isr(void)
287 {
288     PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
289 }
290
291 interrupt void cpu_timer0_isr(void)
292 {
293
294     PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
```

```
}

interrupt void Ecap1Int_isr(void)
{

    ECap1Regs.ECCLR.bit.INT = 1;
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP4;
}
interrupt void EPWM1Int_isr(void)
{
    // 实验2
    EPwm1Regs.ETCLR.bit.INT = 1;
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP3;
    EINT;
}

interrupt void MyAdcInt1_isr(void)
{
    ADCA7 = AdcResult.ADCRESULT0 + AdcResult.ADCRESULT2;
    ADCA7 += AdcResult.ADCRESULT4 + AdcResult.ADCRESULT6;

    ADCB7 = AdcResult.ADCRESULT1 + AdcResult.ADCRESULT3;
    ADCB7 += AdcResult.ADCRESULT5 + AdcResult.ADCRESULT7;

    EPwm1Regs.ETCLR.bit.SOCA = 1;
    AdcRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
}

```

```c
/*
 * main4.c
 *
 *  Created on: 2023年4月7日
 *      Author: Asus
 */
#include "DSP28x_Project.h"
#include "LED_TM1638.h"
void DelaymS(int tm);
void HorseRunning(int no);
void InitPWM1();
interrupt void myXint1_isr(void);
interrupt void cpu_timer0_isr(void);
interrupt void Ecap1Int_isr(void);
interrupt void EPWM1Int_isr(void); // EPWM1
interrupt void MyAdcInt1_isr(void); // ADCINT1
void InitADC();
void HorseIO_Init();

#define ADC_usDELAY 1000L
#define DELAY_US(A) DSP28x_usDelay(((((long double)A * 1000.0L) / (long double)CPU_RATE) - 9.0L) / 5.0L)
#define Led0Blink() GpioDataRegs.GPACLEAR.bit.GPIO0 = 1
#define Led1Blink() GpioDataRegs.GPACLEAR.bit.GPIO1 = 1
#define Led2Blink() GpioDataRegs.GPACLEAR.bit.GPIO2 = 1
#define Led3Blink() GpioDataRegs.GPACLEAR.bit.GPIO3 = 1
#define Led0Blank() GpioDataRegs.GPASET.bit.GPIO0 = 1
#define Led1Blank() GpioDataRegs.GPASET.bit.GPIO1 = 1
#define Led2Blank() GpioDataRegs.GPASET.bit.GPIO2 = 1
#define Led3Blank() GpioDataRegs.GPASET.bit.GPIO3 = 1

float ADCA7;
float ADCB7;
signed int count = 0;
int V_mubiao = 0;
int V_epwm = 0;
int e_last = 0;
int e_sum = 0;
float kp = 2;
float ki = 0.01;
float kd = 0.2;
float delta = 0;
float delta_e = 0;

void main(void)
{
    int j;
    InitSysCtrl(); // 初始化系统时钟，选择内部晶振1，10MHZ，12倍频，2分频，初始化外设时钟，低速外设,4分频

    DINT;               // 关总中断
    IER = 0x0000;       // 关CPU中断使能
    IFR = 0x0000;       // 清CPU中断标志
    InitPieCtrl();      // 关pie中断
    InitPieVectTable(); // 清中断向量表

    EALLOW; /**配置中断向量表*****/
    PieVectTable.EPWM1_INT = &EPWM1Int_isr;
    PieVectTable.ADCINT1 = &MyAdcInt1_isr;
```

```
58      EDIS;
59
60      //  MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);
61      InitFlash();
62      InitCpuTimers(); // 初始化定时器
63      // ConfigCpuTimer(&CpuTimer0,60,10000);
64      HorseIO_Init();
65      InitPWM1();
66      InitADC();
67      TM1638_Init(); // 初始化LED
68
69      PieCtrlRegs.PIECTRL.bit.ENPIE = 1;
70
71      PieCtrlRegs.PIEIER1.bit.INTx1 = 1;
72      PieCtrlRegs.PIEIER3.bit.INTx1 = 1;
73      IER |= M_INT3;
74      IER |= M_INT1;
75
76      EINT;
77      ERTM;
78
79      while (1)
80      {
81          DelaymS(10);
82          j++;
83          j = j & 0xf;
84          HorseRunning(j);
85
86          LED_Show(1, (V_epwm % 10), 0);
87          LED_Show(2, (V_epwm % 100 / 10), 0);
88          LED_Show(3, (V_epwm % 1000 / 100), 0);
89          LED_Show(4, (V_epwm / 1000), 1);
90          LED_Show(5, V_mubiao % 10, 0);
91          LED_Show(6, (V_mubiao % 100 / 10), 0);
92          LED_Show(7, (V_mubiao % 1000 / 100), 0);
93          LED_Show(8, (V_mubiao / 1000), 1);
94      }
95  }
96
97  void InitADC()
98  {
99      int ADCSampT;
100     ADCSampT = 15;
101     EALLOW;
102     SysCtrlRegs.PCLKCR0.bit.ADCENCLK = 1;
103     (*Device_cal)();
104     EDIS;
105     DELAY_US(ADC_usDELAY);
106
107     EALLOW;
108     AdcRegs.ADCCTL1.bit.ADCBGPWD = 1;
109     AdcRegs.ADCCTL1.bit.ADCREFPWD = 1;
110     AdcRegs.ADCCTL1.bit.ADCPWDN = 1;
111     AdcRegs.ADCCTL1.bit.ADCENABLE = 1;
112     AdcRegs.ADCCTL1.bit.ADCREFSEL = 0;
113     EDIS;
114
115     DELAY_US(ADC_usDELAY);
116     AdcOffsetSelfCal();
117     DELAY_US(ADC_usDELAY);
```

```
118
119     EALLOW;
120     AdcRegs.ADCCTL1.bit.INTPULSEPOS = 1;
121     AdcRegs.INTSEL1N2.bit.INT1E = 1;
122     AdcRegs.INTSEL1N2.bit.INT1CONT = 0;
123     AdcRegs.INTSEL1N2.bit.INT1SEL = 7;
124     AdcRegs.ADCSAMPLEMODE.all = 0x0f;
125     AdcRegs.ADCSOC0CTL.bit.CHSEL = 0x07;
126     AdcRegs.ADCSOC2CTL.bit.CHSEL = 0x07;
127     AdcRegs.ADCSOC4CTL.bit.CHSEL = 0x07;
128     AdcRegs.ADCSOC6CTL.bit.CHSEL = 0x07;
129
130     AdcRegs.ADCSOC0CTL.bit.TRIGSEL = 5;
131     AdcRegs.ADCSOC2CTL.bit.TRIGSEL = 5;
132     AdcRegs.ADCSOC4CTL.bit.TRIGSEL = 5;
133     AdcRegs.ADCSOC6CTL.bit.TRIGSEL = 5;
134
135     AdcRegs.ADCSOC0CTL.bit.ACQPS = ADCSampT;
136     AdcRegs.ADCSOC2CTL.bit.ACQPS = ADCSampT;
137     AdcRegs.ADCSOC4CTL.bit.ACQPS = ADCSampT;
138     AdcRegs.ADCSOC6CTL.bit.ACQPS = ADCSampT;
139
140     AdcRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;
141     EDIS;
142
143     EALLOW;
144     EPwm1Regs.ETSEL.bit.SOCAEN = 1;
145     EPwm1Regs.ETSEL.bit.SOCASEL = 1;
146     EPwm1Regs.ETPS.bit.SOCAPRD = 1;
147     EPwm1Regs.ETCLR.bit.SOCA = 1;
148     EDIS;
149 }
150
151 void Xint1_Init()
152 {
153     EALLOW;
154     EDIS;
155 }
156
157 void IniteCAP1Config()
158 {
159     EALLOW;
160     EDIS;
161 }
162
163 void InitPWM1()
164 {
165     int PWMPRD, DeadTime;
166     EALLOW;
167     SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;
168     EDIS;
169     EALLOW;
170     GpioCtrlRegs.GPAPUD.bit.GPIO0 = 1;
171     GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1;
172     EDIS;
173
174     PWMPRD = 3300;
175     DeadTime = 0;
176
177     EALLOW;
```

```
178     EPwm1Regs.TBPRD = PWMPRD;
179     EPwm1Regs.TBPHS.half.TBPHS = 0;
180     EPwm1Regs.TBCTL.bit.CLKDIV = 0;
181     EPwm1Regs.TBCTL.bit.HSPCLKDIV = 0;
182     EPwm1Regs.TBCTL.bit.CTRMODE = 2;
183     EPwm1Regs.TBCTL.bit.PHSEN = 0;
184     EPwm1Regs.TBCTL.bit.PRDLD = 0;
185     EPwm1Regs.TBCTL.bit.SYNCOSEL = 11;
186     EPwm1Regs.CMPCTL.bit.SHDWAMODE = 0;
187     EPwm1Regs.CMPCTL.bit.SHDWBMODE = 0;
188     EPwm1Regs.CMPCTL.bit.LOADAMODE = 2;
189     EPwm1Regs.CMPCTL.bit.LOADBMODE = 2;
190     EPwm1Regs.AQCTLA.bit.CAU = 1;
191     EPwm1Regs.AQCTLA.bit.CAD = 2;
192     EPwm1Regs.AQCTLA.bit.CBU = 0;
193     EPwm1Regs.AQCTLA.bit.CBD = 0;
194     EPwm1Regs.DBCTL.bit.OUT_MODE = 0;
195     EPwm1Regs.DBCTL.bit.IN_MODE = 0;
196     EPwm1Regs.DBCTL.bit.POLSEL = 2;
197     EPwm1Regs.DBFED = DeadTime;
198     EPwm1Regs.DBRED = DeadTime;
199     EPwm1Regs.CMPA.half.CMPA = PWMPRD / 2;
200     EPwm1Regs.CMPB = PWMPRD / 2;
201     EPwm1Regs.ETSEL.bit.INTEN = 1;
202     EPwm1Regs.ETSEL.bit.INTSEL = 1;
203     EPwm1Regs.ETPS.bit.INTPRD = 1;
204     EPwm1Regs.ETCLR.bit.INT = 1;
205     EDIS;
206     EALLOW;
207     SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1;
208     EDIS;
209 }
210 void HorseIO_Init()
211 {
212     EALLOW;
213     GpioDataRegs.GPASET.bit.GPIO0 = 1;
214     GpioDataRegs.GPASET.bit.GPIO1 = 1;
215     GpioDataRegs.GPASET.bit.GPIO2 = 1;
216     GpioDataRegs.GPASET.bit.GPIO3 = 1;
217     GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 0;
218     GpioCtrlRegs.GPADIR.bit.GPIO0 = 1;
219     GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 0;
220     GpioCtrlRegs.GPADIR.bit.GPIO1 = 1;
221     GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 0;
222     GpioCtrlRegs.GPADIR.bit.GPIO2 = 1;
223     GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 0;
224     GpioCtrlRegs.GPADIR.bit.GPIO3 = 1;
225     EDIS;
226 }
227
228 interrupt void EPWM1Int_isr(void)
229 {
230     int e;
231 //    float kp = 2;
232 //    float ki = 0.1;
233 //    float kd = 0.2;
234     e = (V_mubiao - V_epwm);
235     delta_e = e - e_last;
236     delta = V_mubiao  + e * kp + e_sum * ki + delta_e * kd;
237     if (delta > 3300)
```

```
238     {
239         EALLOW;
240         EPwm1Regs.CMPA.half.CMPA = 3100;
241         EDIS;
242         e_sum = 0;
243     }
244     else if (delta <= 3300 && delta >=0){
245         EALLOW;
246         EPwm1Regs.CMPA.half.CMPA = (int)(delta);
247         EDIS;
248         e_sum = e_sum + e;
249     }
250     else
251     {
252         EALLOW;
253         EPwm1Regs.CMPA.half.CMPA = 200;
254         EDIS;
255         e_sum = 0;
256     }
257     if(e_sum > 1000){
258         e_sum  = 1000;
259     }
260     if (e_sum < -1000){
261         e_sum = -1000;
262     }
263     e_last = e;
264     EPwm1Regs.ETCLR.bit.INT = 1;
265     PieCtrlRegs.PIEACK.all = PIEACK_GROUP3;
266 }
267 interrupt void MyAdcInt1_isr(void)
268 {
269     ADCA7 = AdcResult.ADCRESULT0 + AdcResult.ADCRESULT2;
270     ADCA7 += AdcResult.ADCRESULT4 + AdcResult.ADCRESULT6;
271 
272     ADCB7 = AdcResult.ADCRESULT1 + AdcResult.ADCRESULT3;
273     ADCB7 += AdcResult.ADCRESULT5 + AdcResult.ADCRESULT7;
274 
275     V_mubiao = (int)((ADCB7 / 4 / 4096) * 3.3 * 1000);
276     V_epwm = (int)((ADCA7 / 4 / 4096) * 3.3 * 1000);
277 
278     AdcRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;
279     EPwm1Regs.ETCLR.bit.SOCA = 1;
280     PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
281 }
282 
283 
284 void DelaymS(int tm)
285 {
286     int i;
287     unsigned int j;
288     for (i = 0; i < tm; i++)
289     {
290         j = 60000;
291         while (j != 0)
292             j--;
293     }
294 }
295 
296 void HorseRunning(int no)
297 {
```

```
    if (no & 0x1)
        Led0Blink();
    else
        Led0Blank();
    if (no & 0x2)
        Led1Blink();
    else
        Led1Blank();
    if (no & 0x4)
        Led2Blink();
    else
        Led2Blank();
    if (no & 0x8)
        Led3Blink();
    else
        Led3Blank();
}

interrupt void myXint1_isr(void)
{
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
}

interrupt void cpu_timer0_isr(void)
{

    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
}

interrupt void Ecap1Int_isr(void)
{

    ECap1Regs.ECCLR.bit.INT = 1;
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP4;
}


```