#include<reg52.h>

#include "mydefine.h"

#define step 15

unsigned char TH\_H,TL\_H,TH\_L,TL\_L;//

unsigned char keynum,pwmnum=1,change\_flag=0;

unsigned int temp,count\_pwm1=1500,count\_pwm2=1500,count\_pwm3=1500,count\_pwm4=1500,count\_pwm5=1500,count\_pwm6=1500;

unsigned char temphh[7],temphl[7],templh[7],templl[7];//temphh,temphl,templh,templl;

unsigned char KeyScan(void);

void keypro();

void circle(unsigned char);

void delay(unsigned char);

void iruse();

void ir\_scan();

/\*------------------------------------------------

定时器初始化子程序

------------------------------------------------\*/

void Init\_Timer(void)

{

TMOD = 0x12; //使用模式1，16位定时器，使用"|"符号可以在使用多个定时器时不受影响 TOMD = TOMD | 0X01

TH1=0x00; //给定初值，这里使用定时器最大值从0开始计数一直到65535溢出

TL1=0x00;

TH0=0;

TL0=0;

EA=1;

ET0=0;

ET1=1;

EX0=0;

IT0=1;

TR0=0;

TR1=1; //定时器开关打开

}

/\*------------------------------------------------

数据处理

------------------------------------------------\*/

void DataPro(unsigned int temp)

{

TH\_H=(65536-temp)/256;

TL\_H=(65536-temp)%256;

TH\_L=(46536+temp)/256;

TL\_L=(46536+temp)%256;

}

/\*------------------------------------------------

主程序

------------------------------------------------\*/

main()

{

unsigned char i;

temp=count\_pwm1;

Init\_Timer();

DataPro(temp);

for(i=1;i<7;i++) circle(i);

while(1)

{

//iruse();

keynum= KeyScan();

if(keynum==1)

{

temphh[pwmnum]=TH\_H;temphl[pwmnum]=TH\_L;templh[pwmnum]=TL\_H;templl[pwmnum]=TL\_L;

pwmnum++;

change\_flag=1;

if(pwmnum==7) pwmnum=1;

}

if(keynum==2)

{

temphh[pwmnum]=TH\_H;temphl[pwmnum]=TH\_L;templh[pwmnum]=TL\_H;templl[pwmnum]=TL\_L;

pwmnum--;

change\_flag=1;

if(pwmnum==0) pwmnum=6;

}

circle(pwmnum);

}

}

/\*------------------------------------------------

定时器中断子程序

------------------------------------------------\*/

void Timer1\_isr(void) interrupt 3

{

switch(pwmnum)

{

case 1:{

if(OUT1)

{

TH1=TH\_L; //重新赋值

TL1=TL\_L;

}

else

{

TH1=TH\_H; //重新赋值

TL1=TL\_H;

}

OUT1=!OUT1;

};break;

case 2:

{

if(OUT2)

{

TH1=TH\_L; //重新赋值

TL1=TL\_L;

}

else

{

TH1=TH\_H; //重新赋值

TL1=TL\_H;

}

OUT2=!OUT2;

}break;

case 3:

{

if(OUT3)

{

TH1=TH\_L; //重新赋值

TL1=TL\_L;

}

else

{

TH1=TH\_H; //重新赋值

TL1=TL\_H;

}

OUT3=!OUT3;

}break;

case 4:

{

if(OUT4)

{

TH1=TH\_L; //重新赋值

TL1=TL\_L;

}

else

{

TH1=TH\_H; //重新赋值

TL1=TL\_H;

}

OUT4=!OUT4;

}break;

case 5:

{

if(OUT5)

{

TH1=TH\_L; //重新赋值

TL1=TL\_L;

}

else

{

TH1=TH\_H; //重新赋值

TL1=TL\_H;

}

OUT5=!OUT5;

}break;

case 6:

{

if(OUT6)

{

TH1=TH\_L; //重新赋值

TL1=TL\_L;

}

else

{

TH1=TH\_H; //重新赋值

TL1=TL\_H;

}

OUT6=!OUT6;

}break;

}

keynum=0;

}

/\*------------------------------------------------

按键扫描函数，返回扫描键值

------------------------------------------------\*/

unsigned char KeyScan()

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(!KEY1) //如果检测到低电平，说明按键按下

{

delay(10); //延时去抖，一般10-20ms

if(!KEY1) //再次确认按键是否按下，没有按下则退出

{

while(!KEY1);//如果确认按下按键等待按键释放，没有则退出

{

return 1;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

else if(!KEY2) //如果检测到低电平，说明按键按下

{

delay(10); //延时去抖，一般10-20ms

if(!KEY2) //再次确认按键是否按下，没有按下则退出

{

while(!KEY2);//如果确认按下按键等待按键释放，没有则退出

{

return 2;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

else if(!KEY3) //如果检测到低电平，说明按键按下

{

delay(10); //延时去抖，一般10-20ms

if(!KEY3) //再次确认按键是否按下，没有按下则退出

{

// while(!KEY3);//如果确认按下按键等待按键释放，没有则退出

{

return 3;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

else if(!KEY4) //如果检测到低电平，说明按键按下

{

delay(10); //延时去抖，一般10-20ms

if(!KEY4) //再次确认按键是否按下，没有按下则退出

{

// while(!KEY4);//如果确认按下按键等待按键释放，没有则退出

{

return 4;

}

}

}

else return 0;

}

void circle(unsigned char pwmnum)

{

switch(pwmnum)

{

case 1:

{

P0=0xfe;

if(keynum==3)

{

if(count\_pwm1<CountMax\_PWM1)

count\_pwm1+=step;

}

else if(keynum==4)

{

if(count\_pwm1>CountMin\_PWM1)

count\_pwm1-=step;

}

if(change\_flag)

{TH\_H=temphh[1];TH\_L=temphl[1];TL\_H=templh[1];TL\_L=templl[1];}

else

{

DataPro(count\_pwm1);

}

change\_flag=0;

};break;

case 2:

{

P0=0xfd;

if(keynum==3)

{

if(count\_pwm2<CountMax\_PWM2)

count\_pwm2+=step;

// DataPro(count\_pwm3);

}

else if(keynum==4)

{

if(count\_pwm2>CountMin\_PWM2)

count\_pwm2-=step;

// DataPro(count\_pwm3);

}

if(change\_flag)

{TH\_H=temphh[2];TH\_L=temphl[2];TL\_H=templh[2];TL\_L=templl[2];}

else

{

DataPro(count\_pwm2);

}

change\_flag=0;

};break;

case 3:

{

P0=0xfb;

if(keynum==3)

{

if(count\_pwm3<CountMax\_PWM3)

count\_pwm3+=step;

// DataPro(count\_pwm3);

}

else if(keynum==4)

{

if(count\_pwm3>CountMin\_PWM3)

count\_pwm3-=step;

// DataPro(count\_pwm3);

}

if(change\_flag)

{TH\_H=temphh[3];TH\_L=temphl[3];TL\_H=templh[3];TL\_L=templl[3];}

else

{

DataPro(count\_pwm3);

}

change\_flag=0;

}break;

case 4:

{

P0=0xf7;

if(keynum==3)

{

if(count\_pwm4<CountMax\_PWM4)

count\_pwm4+=step;

// DataPro(count\_pwm4);

}

else if(keynum==4)

{

if(count\_pwm4>CountMin\_PWM4)

count\_pwm4-=step;

// DataPro(count\_pwm4);

}

if(change\_flag)

{TH\_H=temphh[4];TH\_L=temphl[4];TL\_H=templh[4];TL\_L=templl[4];}

else

{

DataPro(count\_pwm4);

}

change\_flag=0;

};break;

case 5:

{

P0=0xef;

if(keynum==3)

{

if(count\_pwm5<CountMax\_PWM5)

count\_pwm5+=step;

// DataPro(count\_pwm5);

}

else if(keynum==4)

{

if(count\_pwm5>CountMin\_PWM5)

count\_pwm5-=step;

// DataPro(count\_pwm5);

}

if(change\_flag)

{TH\_H=temphh[5];TH\_L=temphl[5];TL\_H=templh[5];TL\_L=templl[5];}

else

{

DataPro(count\_pwm5);

}

change\_flag=0;

};break;

case 6:

{

P0=0xdf;

if(keynum==3)

{

if(count\_pwm6<CountMax\_PWM6)

count\_pwm6+=step;

// DataPro(count\_pwm6);

}

else if(keynum==4)

{

if(count\_pwm6>CountMin\_PWM6)

count\_pwm6-=step;

// DataPro(count\_pwm6);

}

if(change\_flag)

{TH\_H=temphh[6];TH\_L=temphl[6];TL\_H=templh[6];TL\_L=templl[6];}

else

{

DataPro(count\_pwm6);

}

change\_flag=0;

};break;

}

keynum=0;

}

/\*

void ir\_time() interrupt 1

{

irtime++; //用于计数2个下降沿之间的时间

}

void ir\_recieve() interrupt 0 //外部中断0服务函数

{

static unsigned char i; //接收红外信号处理

static bit startflag; //是否开始处理标志位

if(startflag)

{

if(irtime<63&&irtime>=33)//引导码 TC9012的头码，9ms+4.5ms

i=0;

irdata[i]=irtime;//存储每个电平的持续时间，用于以后判断是0还是1

irtime=0;

i++;

if(i==33)

{

irrecieve\_ok=1;

i=0;

}

}

else

{

irtime=0;

startflag=1;

}

}

/\*

void init\_inter()//定时器0初始化

{

TMOD=0x12;//定时器0工作方式2，TH0是重装值，TL0是初值

TH0=0x00; //重载值

TL0=0x00;

EA=1;

IT0 = 1;

EX0 = 1;

ET0=1;

TR0=1;

}

\*/

/\*

void ir\_jiema()//红外码值处理函数

{

unsigned char i, j, k;

unsigned char cord,value;

k=1;

for(i=0;i<4;i++) //处理4个字节

{

for(j=1;j<=8;j++) //处理1个字节8位

{

cord=irdata[k];

if(cord>7)//大于某值为1，这个和晶振有绝对关系，这里使用12M计算，此值可以有一定误差

value|=0x80;

if(j<8)

{

value>>=1;

}

k++;

}

IRcord[i]=value;

value=0;

}

irjiema\_ok=1;//处理完毕标志位置1

}

void ir\_scan()

{

switch(IRcord[2])

{

case 0x15:{ keynum=1;}break;

case 0x07:{keynum=2;}break;

case 0x40:{ keynum=3;}break;

case 0x44:{ keynum=4;}break;

//default : keynum=0;

}

}

\*/

void delay(unsigned char x)

{

unsigned char i,j;

for(i=100;i>0;i--)

for(j=x;j>0;j--);

}

/\*

void iruse()

{

if(irrecieve\_ok)

{

irrecieve\_ok=0;

ir\_jiema();

}

if(irjiema\_ok)

{

irjiema\_ok=0;

ir\_scan();

}

}

\*/