|  |  |
| --- | --- |
| 001 | jsjlogo |

**合肥工业大学**

**计算机与信息学院**

**课程设计**

**课 程：嵌入式系统课程设计**

**专业班级：**

**学 号：**

**姓 名：**

一、设计题目及要求：

【课题1】 单个按键控制LED灯

设计要求：

根据单个按键输入情况控制LED灯的不同显示效果。K1连续按下偶数次时，四个LED灯（LED1～LED4）

按1秒（定时器中断实现）的间隔同时闪烁对应的次数，然后保持LED1和LED2常亮，LED3和LED4熄灭；K1

连续按下奇数次时，四个灯按0.5秒（定时器中断实现）的间隔同时闪烁对应的次数，然后保持LED1和LED2

熄灭，LED3和LED4常亮。K2按下时，四个灯按1秒（定时器中断实现）的间隔逐一点亮（同一个时刻只有

一个灯亮），并循环显示。K3按下时，四个灯按0.5秒（定时器中断实现）的间隔逐一点亮（同一个时刻

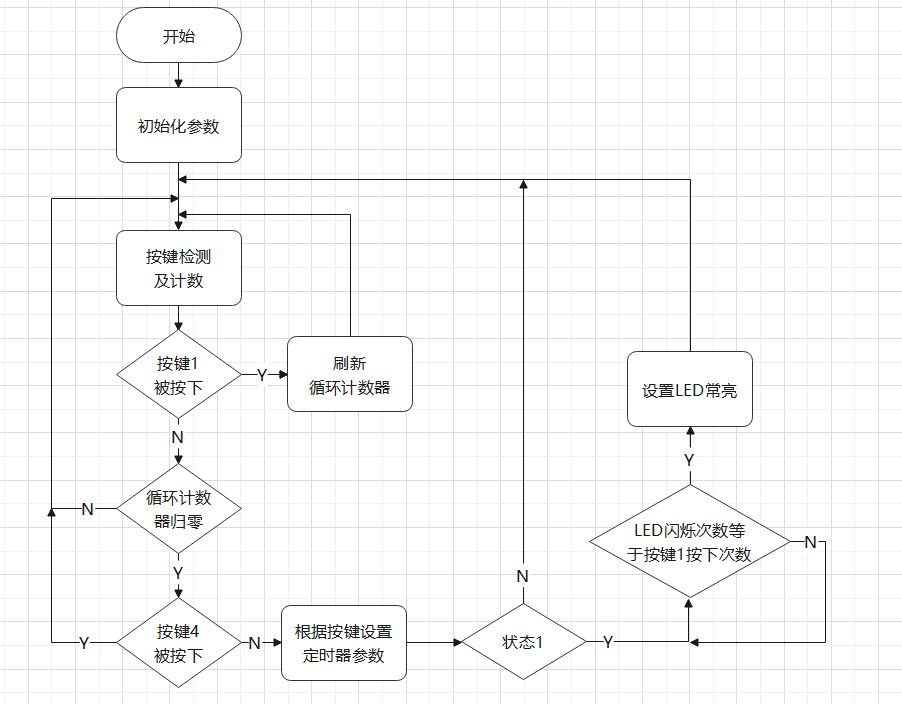
只有一个灯亮），并循环显示。K4按下时，所有灯熄灭，系统进入初始状态。

 说明：程序需要区分连续按键与间隔时间按键。连续按键的间隔时间自行确定，合理即可。

# 设计思想：

通过软件程序判断的方法进行按键的选择和判断以此来实现对LED灯的控制，K1~K4按键对应不同的功能。开机后程序以一定时间（2500000次循环）为周期进行按键检测，在K1被按下后且延时一定时间后判断K1依然处于被按下状态后（实现按键防抖）刷新此周期计数器并将按键计数器加一，重复此过程直到周期计数器归零前未检测到K1再次被按下，结束该周期进入状态1；K2和K3是切换到状态3和状态4；而K4本质上是初始化所有参数。  
 按键的汇总功能在main.c当中，通过接受不同的按键信息并进行判断实现按键的计数和进入不同的状态，而主要的亮灯功能的实现在timer.c当中，通过计时器中断进行当前的亮灯策略。当处于状态1且按键计数为偶数时，定时器中断周期为0.5s，每中断一次对LED灯寄存器中的参数进行一次取反操作，在判断LED灯闪烁次数与按键按下次数相同后退出此次循环停止定时器中断并保持LED1和LED2常亮；当处于状态1且按键计数为奇数时，定时器中断周期为0.25s，每中断一次对LED灯寄存器中的参数进行一次取反操作，在判断LED灯闪烁次数与按键按下次数相同后退出此次循环停止定时器中断并保持LED3和LED4常亮；当处于状态2时，定时器中断周期为0. 5s，利用循环将8个定时器中断周期设为一个流水灯周期，在一个流水灯周期内根据当前的定时器中断周期标志点亮或熄灭对应的LED灯；当处于状态3时，定时器中断周期为0.25s，流水灯周期同状态2；当处于状态4时，不执行定时器中断。  
 其中中断功能的实现使我们能够进行不同亮灯状态间的转换，是我们程序要求能够实现的基础。

# 功能流程图：



# 结果讨论：

8月我参加了学校安排的课题实训。我参加的课题实训是嵌入式方向的，开课的内容主要是嵌入式开发方面的，C语言，数据结构，LINUX基本操作，网络编程，操作系统和汇编语言，这些都是嵌入式不可缺少的。每个专业技能都决定了你以后找工作。

我们班分了很多小组，每组都有小组长，小组长直接对我们负责。我教课老师很认真，都是从很基础的开始讲；同学们听课也很认真，由于代码操作比较多，键盘的声音源源不断的在响。好久没有感觉到这种浓重的学习气氛了，而在这紧张而充实的学习氛围中我获益匪浅，感受颇多。

通过这几个月的实训我学到了很多，同时也让我们意识到我们要学的更多。从程序的设计到编写、画图、调试、修改、完善、到最后的实训报告，每个人都付出了时间和精力去做好自己的任务帮助队友，所以团队的合作和队友之间的相互协助非常重要。通过实训让我们更深一步的对自己的专业，对自己的能力，对自己所学的有正确的认识，并且能在以后的学习工作中不断提高和完善自己。

特别是讲C语言和进程通信中老师给了我们每个小组的几个小项目，让我学到了更多项目经验，这在学校之前是没有的。这次实训也碰到了些问题，比如项目文档的写法，代码的规范，流程图的画法，这些都有讲究，在老师和同学的帮助下，我不断的提高了在这些方面的学习。

生活上老师对我们关怀备至，让我们没有感觉在外面的那种不适感，同学们也互帮互助，所以生活上大致没什么问题，这让我们更好的去学专业。很感谢学校有这样的实训安排。总的来说，这次实训还是比较成功的，无论是个人动手能力还是团队协作能力都得到了很好的锻炼。在实训过程中老师也交给我们很多简单实用的技巧。我们也学到了，做好做对一个项目固然重要，同时美观程度也是必不可少的，无论是从参观者的角度还是自己看的角度都很重要。

附录--实验代码：**（**完整的源程序**）**

**①main.c**

#include "stdio.h"

#define GPNCON (\*(volatile unsigned long \*)0x7F008830)

#define GPNDAT (\*(volatile unsigned long \*)0x7F008834)

#define GPKCON (\*((volatile unsigned long \*)0x7F008800))

#define GPKDAT (\*((volatile unsigned long \*)0x7F008808))

#define GPFCON (\*((volatile unsigned long \*)0x7F0080A0))

#define GPFDAT (\*((volatile unsigned long \*)0x7F0080A4))

#define TCON ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x08)) )

#define PWMTIMER\_BASE (0x7F006000)

void timer\_init(unsigned long utimer,unsigned long uprescaler,unsigned long udivider,unsigned long utcntb,unsigned long utcmpb);

int i = 0; //控制三短一长

int j = 0;

int ishalf = 0;

int flag1 = 0;

int main()

{

//配置GPK为输出，同时熄灭所有灯

GPKCON=0x11110000;

GPKDAT=0x000000f0;

// 配置GPN为输入功能

GPNCON = 0;

while(1){

int dat=0;

int count=0;

int flag=0;

int times=2500000;

int time1=50000;

while (1){

dat=GPNDAT;

if(!(dat & (1<<0))){//K1连续按下，开始计数

count++;

times=2500000;

time1=50000;

while(time1--);

while(!(dat & (1<<0))){

dat=GPNDAT;

}

}

if(times<0)break;

if(!(dat & (1<<1))){//K2按下

flag1 = 2;

timer\_init(0,65,4,31250,0);

while(!(dat & (1<<1))){

dat=GPNDAT;

}

}

if(!(dat & (1<<2))){//K3按下

flag1 = 3;

timer\_init(0,65,4,15625,0);

while(!(dat & (1<<2))){

dat=GPNDAT;

}

}

if((!(dat & (1<<3)))){//key4被按下，复位

dat=GPNDAT;

flag=1;

flag1=0;

i = 0;

j = 0;

GPKDAT=0x000000f0;

break;

}

times--;

}

if(flag)continue;

if(count%2&&count>0){//按下奇数次

flag1=1;

ishalf = 0;

timer\_init(0,65,4,15625,0);//计时器工作

}

if(!(count%2)&&count>0){ //按下偶数次

flag1=1;

ishalf = 1;

timer\_init(0,65,4,31250,0);

}

int num = 0;

while(dat&(1<<3))

{//key4被按下，循环退出

dat=GPNDAT;

if(GPKDAT & (1<<4))

{

num++;

while (GPKDAT & (1 << 4)) {}

}

if(num==(count))

{

if(count%2&&count>0) GPKDAT=0x000000c0;;

if(!(count%2)&&count>0) GPKDAT=0x00000030;;

break;

}

//灯亮了指定次数后，退出

}

TCON &= ~(1<<0);

}

return 0;

}

②timer.c

#include "stdio.h"

#define GPKCON0 (\*((volatile unsigned long \*)0x7F008800))

#define GPKDAT (\*((volatile unsigned long \*)0x7F008808))

#define GPFCON (\*((volatile unsigned long \*)0x7F0080A0))

#define GPFDAT (\*((volatile unsigned long \*)0x7F0080A4))

#define EINT0CON0 (\*((volatile unsigned long \*)0x7F008900))

#define EINT0MASK (\*((volatile unsigned long \*)0x7F008920))

#define EINT0PEND (\*((volatile unsigned long \*)0x7F008924))

#define PRIORITY (\*((volatile unsigned long \*)0x7F008280))

#define SERVICE (\*((volatile unsigned long \*)0x7F008284))

#define SERVICEPEND (\*((volatile unsigned long \*)0x7F008288))

#define VIC0IRQSTATUS (\*((volatile unsigned long \*)0x71200000))

#define VIC0FIQSTATUS (\*((volatile unsigned long \*)0x71200004))

#define VIC0RAWINTR (\*((volatile unsigned long \*)0x71200008))

#define VIC0INTSELECT (\*((volatile unsigned long \*)0x7120000c))

#define VIC0INTENABLE (\*((volatile unsigned long \*)0x71200010))

#define VIC0INTENCLEAR (\*((volatile unsigned long \*)0x71200014))

#define VIC0PROTECTION (\*((volatile unsigned long \*)0x71200020))

#define VIC0SWPRIORITYMASK (\*((volatile unsigned long \*)0x71200024))

#define VIC0PRIORITYDAISY (\*((volatile unsigned long \*)0x71200028))

#define VIC0ADDRESS (\*((volatile unsigned long \*)0x71200f00))

#define PWMTIMER\_BASE (0x7F006000)

#define TCFG0 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x00)) )

#define TCFG1 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x04)) )

#define TCON ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x08)) )

#define TCNTB0 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x0C)) )

#define TCMPB0 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x10)) )

#define TCNTO0 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x14)) )

#define TCNTB1 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x18)) )

#define TCMPB1 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x1C)) )

#define TCNTO1 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x20)) )

#define TCNTB2 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x24)) )

#define TCMPB2 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x28)) )

#define TCNTO2 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x2C)) )

#define TCNTB3 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x30)) )

#define TCMPB3 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x34)) )

#define TCNTO3 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x38)) )

#define TCNTB4 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x3C)) )

#define TCNTO4 ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x40)) )

#define TINT\_CSTAT ( \*((volatile unsigned long \*)(PWMTIMER\_BASE+0x44)) )

typedef void (isr) (void);

extern void asm\_timer\_irq();

extern int ishalf;

extern int num;

extern int flag1;

extern int i;

extern int j;

void irq\_init(void)

{

/\* 在中断控制器里使能timer0中断 \*/

VIC0INTENABLE |= (1<<23);

VIC0INTSELECT =0;

isr\*\* isr\_array = (isr\*\*)(0x7120015C);

isr\_array[0] = (isr\*)asm\_timer\_irq;

//配置GPF14为输出

GPFCON |= 1<<28;

GPFCON &= ~(1<<29);

}

// timer0中断的中断处理函数

void do\_irq()

{

unsigned long uTmp;

//led亮/不亮

if(flag1 == 1){//key1被按下

GPKDAT ^= 0x000000f0;

}

if (flag1 == 2) //key2 被按下

{

i++;

if(i == 0)

GPKDAT = 0x000000e0;

else if(i == 2)

GPKDAT = 0x000000d0;

else if(i == 4)

GPKDAT = 0x000000b0;

else if(i == 6)

GPKDAT = 0x00000070;

else if(i == 1 || i == 3 || i == 5 || i == 7)

GPKDAT = 0x000000f0;

else if(i >= 8)

i = -1;

}

if (flag1 == 3) //key3被按下

{

j++;

if(j == 0)

GPKDAT = 0x000000e0;

else if(j == 2)

GPKDAT = 0x000000d0;

else if(j == 4)

GPKDAT = 0x000000b0;

else if(j == 6)

GPKDAT = 0x00000070;

else if(j == 1 || j == 3 || j == 5 || j == 7)

GPKDAT = 0x000000f0;

else if(j >= 8)

j = -1;

}

//清timer0的中断状态寄存器

uTmp = TINT\_CSTAT;

TINT\_CSTAT = uTmp;

VIC0ADDRESS=0x0;

}

// 初始化timer

void timer\_init(unsigned long utimer,unsigned long uprescaler,unsigned long udivider,unsigned long utcntb,unsigned long utcmpb)

{

unsigned long temp0;

// 定时器的输入时钟 = PCLK / ( {prescaler value + 1} ) / {divider value} = PCLK/(65+1)/16=62500hz

//设置预分频系数为66

temp0 = TCFG0;

temp0 = (temp0 & (~(0xff00ff))) | (uprescaler<<0);

TCFG0 = temp0;

// 16分频

temp0 = TCFG1;

temp0 = (temp0 & (~(0xf<<4\*utimer))& (~(1<<20))) |(udivider<<4\*utimer);

TCFG1 = temp0;

// 1s = 62500hz

TCNTB0 = utcntb;

TCMPB0 = utcmpb;

// 手动更新

TCON |= 1<<1;

// 清手动更新位

TCON &= ~(1<<1);

// 自动加载和启动timer0

TCON |= (1<<0)|(1<<3);

// 使能timer0中断

temp0 = TINT\_CSTAT;

temp0 = (temp0 & (~(1<<utimer)))|(1<<(utimer));

TINT\_CSTAT = temp0;

}