**LCM12864移植贪吃蛇EVE模式**

**第一步**：硬件电路连接：LCM12864\_date->GPIOA[7:0] LCM12864\_contr->GPIOB[4:0]

风扇驱动控制线->PC4 外部中断按键->PC13

**第二步**：问题的罗列

首先得实现LCM12864的字符显示来输出提示语句，直接调用字符库即可。

其次得实现LCM12864的像素点显示来输出贪吃蛇图像，其中包含以下问题：

①LCM12864扩展指令集的应用；

②自定义输出函数时输出像素点的重新定位；

③其显示为16位显示单独更改某个像素点要读其他的像素取原来的显示状态；

④完成硬件移植的时候，控制台程序和硬件编写方式不兼容问题，硬件存储空间问题。

**第四步**：问题解决

LCM12864的字符显示较简单可以参照之前PWMinterruptLCM的工程。

实现LCM12864的像素点显示的步骤如下：子函数传递用户定义的位置以及置一或者置零，打开扩展指令集，根据用户定义地址计算LCM12864的地址实现输出的重定位，根据置零置一与八位相关的初始状态计算出下一次显示状态，在新的LCM12864地址写入相应数据，打开绘图显示，回到基本指令集。

读取其显示状态略为麻烦：因为其设计到GPIO管脚的输入与输出功能切换，此问题可用自定义初始化函数，在对应步骤之前调用子函数改变其输入输出功能。但是其读取的数据存在小问题，因为LCM12864的显示是16位显示，而读取的时候只能通过PA[7:0]八位数据总线读取，其输入寄存器是32位，较为复杂。所以现在采用另一方法，显示器初始化后清除屏幕，自定义初始值为0的二位数组存储LCM12864的上一显示状态，每次计算新的显示数据时访问此数组对应位置即可，输出完了再更新此数组。

硬件移植的不兼容问题：随机函数不需要再种下随机数种子（有待商榷），结构体的写法有差异得分开参数传递，参数传递不能直接传递结构体只能分为多个参数传递，有些关键字得转变。

[注]LCM12864显示分上下两半屏幕：上0-31行0-7大列，下0-31行8-15大列，每一列16位，即16个小列。

**第五步**：添加设计代码

①参数以及函数定义

#include "main.h"

#define u8 unsigned char

#define u16 unsigned int

int last\_num[32][16];

void delay(void);

void lcdwritecmd(unsigned char cmd);

void lcdwritedata(unsigned char dat);

void lcdinit(void);

void clear(void);

void drowpoint(char x,char y,char z);

void lcdsetcursor(unsigned char x, unsigned char y);

void lcdshowstr(unsigned char x, unsigned char y, unsigned char \*str);

const int flg[4][2]={{1,0},{0,1},{0,-1},{-1,0}};//下右左上坐标变化规律

int n=30,m=30; //n和m是蛇的活动范围28行76列+边框=可视化窗口大小

struct xcw{int x,y;}tang[5][15],foodd[2];//结构体tan存储每条蛇的坐标 结构体foodd记录两个食物坐标

int tot[5]; //tot[0123]是四条蛇的长度

int f[5]; //存储蛇的运行状态0123下右左上 搭配flg使用

int score; //得分

int food; //食物个数

int ans\_len=1e9;

int vis[15][15];//记录食物存在点的矩阵

int mods(); //对应模式小蛇初始化

void draw(); //显示边框

void rand\_food(); //没有食物时随机投喂

int change(int t); //没有按键按下蛇的运行

int check(int x,int y); //检验某一个点

void machine(int t,int x,int y); //机器人寻路

void result(int t); //积分够了-胜利

int ads(int x); //取绝对值

int other(int x); //03交换、12交换

int get\_food(int x,int y,int xxx,int yyy); //辅助机器人找寻食物

②识别中断以及释放中断使能标志位

ITStatus EXTI\_GetITStatus(uint32\_t EXTI\_Line)

{

ITStatus bitstatus = RESET; //chushi wei zhuangtai 0

uint32\_t enablestatus = 0; //chushi shineng zhuangtai 0

/\* Check the parameters \*/

assert\_param(IS\_GET\_EXTI\_LINE(EXTI\_Line));

enablestatus = EXTI->IMR1 & EXTI\_Line;

if (((EXTI->FPR1 & EXTI\_Line) != (uint32\_t)RESET) && (enablestatus != (uint32\_t)RESET))

bitstatus = SET;

else

bitstatus = RESET;

return bitstatus;

}

void EXTI\_ClearITPendingBit(uint32\_t EXTI\_Line)

{

/\* Check the parameters \*/

assert\_param(IS\_EXTI\_LINE(EXTI\_Line));

EXTI->FPR1 = EXTI\_Line;

}

③主函数循环开始之前的初始化

lcdinit();

clear();

draw();

HAL\_Delay(1000);

score=0;

food=0;

mods(); //对应模式小蛇初始化

④主函数循环扫描

HAL\_Delay(100);

rand\_food(); //没有食物时随机投喂

change(2),change(3);

⑤和LCM两种显示模式相关的子函数

void delay ()

{

for(int i=0;i<99;i++)

for(int j=0;j<99;j++)

{}

}

void lcdwritecmd(unsigned char cmd)

{

delay(); // 使能 写 指令

GPIOB->ODR=0x18; //RST=1\PSB=1\E=0\RW=0\RS=0

GPIOA->ODR=cmd;

GPIOB->ODR=0x1C; //RST=1\PSB=1\E=1\RW=0\RS=0

delay();

GPIOB->ODR=0x18; //RST=1\PSB=1\E=0\RW=0\RS=0

}

void lcdwritedata(unsigned char dat)

{

delay(); // 使能 写 数据

GPIOB->ODR=0x19; //RST=1\PSB=1\E=0\RW=0\RS=1

GPIOA->ODR=dat;

GPIOB->ODR=0x1D; //RST=1\PSB=1\E=1\RW=0\RS=1

delay();

GPIOB->ODR=0x19; //RST=1\PSB=1\E=0\RW=0\RS=1

}

void lcdinit()

{

lcdwritecmd(0x34); //E 使能 RS1数据0指令 RW1读0写

lcdwritecmd(0x30);

lcdwritecmd(0x0C);

lcdwritecmd(0x01);

}

void clear(void)

{

for(int i=0;i<=32;i++) // 64行 不用一个一个像素清除了

for(int j=0;j<=18;j++) // 18大列

{

lcdwritecmd(0x34); //打开扩展指令集

lcdwritecmd(0x80 + i - 1);

lcdwritecmd(0x80 + j);

lcdwritedata(0x00);

lcdwritedata(0x00);

lcdwritecmd(0x36); //打开绘图显示

lcdwritecmd(0x30); //回到基本指令集

}

}

void drowpoint(char x,char y,char z) //像素输出

{

char i = 0;

char j = 0;

int num = 0x8000; //16位输出高位在左

lcdwritecmd(0x34); //打开扩展指令集

if( y <= 32 )

{

i += y; // i=y -1 i的范围32以下

j = (x - 1) / 16; //大列 j的范围16以下

x = ( x%16>0? (x%16):16 ) - 1 ;

num = num >> x;

}

else

{

i = y - 32; //i=y-32 -1

j = ( (x - 1) / 16 ) + 8;

x = ( x%16>0? (x%16):16 ) - 1 ;

num = num >> x;

}

if(z>0)

num = last\_num[i][j] | num;

else

{

num=~num;

num=last\_num[i][j] & num;

}

lcdwritecmd(0x80 + i - 1);

lcdwritecmd(0x80 + j);

lcdwritedata(num / 256);

lcdwritedata(num % 256);

lcdwritecmd(0x36); //打开绘图显示

lcdwritecmd(0x30); //回到基本指令集

last\_num[i][j] = num;

}

void lcdsetcursor(unsigned char x, unsigned char y) //字符定位

{

unsigned char address;

if(y==0)

address=0x80+x;

else if(y==1)

address=0x90+x;

else if(y==2)

address=0x88+x;

else

address=0x98+x;

lcdwritecmd(address|0x80);

}

void lcdshowstr(unsigned char x, unsigned char y, unsigned char \*str) //字符输出

{

lcdsetcursor(x,y);

while((\*str)!='\0')

{

lcdwritedata(\*str);

str++;

}

}

⑥与贪吃蛇运行相关的函数

int mods() //对应模式小蛇初始化

{

tang[2][1].x=n;

tang[2][1].y=m-3;

tang[2][2].x=n;

tang[2][1].y=m-2;

tang[2][tot[2]=3].x=n;

tang[2][tot[2]=3].y=m-1;

f[2]=2; //电脑1初始方向向左

for(int i=1;i<=tot[2];i++)

drowpoint(tang[2][i].x,tang[2][i].y,1);

tang[3][1].x=2;

tang[3][1].y=5;

tang[3][2].x=2;

tang[3][2].y=4;

tang[3][tot[3]=3].x=2;

tang[3][tot[3]=3].y=3;

f[3]=2; //电脑1初始方向向右

for(int i=1;i<=tot[3];i++)

drowpoint(tang[3][i].x,tang[3][i].y,1);

}

void draw() //显示边框 【注】1行1格 2列1格

{

for(int i=1;i<=n+1;i++)

{

drowpoint(i,1,1); //\*\*\*\*

drowpoint(i,m+1,1);

}

for(int i=1;i<=m+1;i++)

{

drowpoint(1,i,1);

drowpoint(n+1,i,1);

}

}

void rand\_food() //没有食物时随机投喂

{

if(!vis[foodd[0].x][foodd[0].y]) //如果0号食物不见了

{

int x=rand()%(n+1),y=(rand()%m/2)\*2; //随机产生xy坐标y为偶数

while(!check(x,y))

x=rand()%(n+1),y=(rand()%m/2)\*2; //检验随机投喂的点，是否需要重新投喂

drowpoint(x,y,1);

food++;

vis[x][y]=1;

foodd[0].x=x;

foodd[0].y=y;

}

if(!vis[foodd[1].x][foodd[1].y]) //第二个食物--同上

{

int x=rand()%n,y=rand()%m;

while(!check(x,y))

x=rand()%n,y=(rand()%m);

drowpoint(x,y,1);

food++;

vis[x][y]=1;

foodd[1].x=x;

foodd[1].y=y;

}

}

int change(int t) //没有按键按下蛇的运行--第t条蛇

{

int x=tang[t][1].x,y=tang[t][1].y; //x和y是蛇头位置的xy

if(t==2||t==3) //2号和3号蛇是电脑

machine(t,x,y); //机器人寻路

x+=flg[f[t]][0],y+=flg[f[t]][1]; //蛇头(x,y)按规律变化

for(int i=tot[t];i;i--) //蛇的身子位置数据库刷新

{

tang[t][i+1]=tang[t][i];

drowpoint(tang[t][i].x,tang[t][i].y,1); //身子显示刷新，配合蛇头颜色

}

drowpoint(tang[t][tot[t]+1].x,tang[t][tot[t]+1].y,0); //原来的蛇尾消失

if(vis[x][y]) //如果吃掉一个食物的话

{

vis[x][y]=0,score+=(t==0||t==1),food--;

if(++tot[t]>=10) //有一方吃够分

result(t);

drowpoint(tang[t][tot[t]].x,tang[t][tot[t]].y,1); //尾巴处+1

}

// fnd(t,x,y); //一方撞墙撞蛇则失败

tang[t][1].x=x; //把蛇头位置反馈到全局变量中

tang[t][1].y=y;

drowpoint(tang[t][1].x,tang[t][1].y,1); //显示蛇头

}

int check(int x,int y) //检验某一个点

{

if(x<1||x>n||y<2||y>m) //边界外返回0

return 0;

int t=1;

for(int i=1;i<=tot[2];i++)

if(x==tang[2][i].x&&y==tang[2][i].y)

{t=0;break;}

for(int i=1;i<=tot[3];i++)

if(x==tang[3][i].x&&y==tang[3][i].y)

{t=0;break;}

return t; //都不在这些点返回1

}

void machine(int t,int x,int y) //机器人寻路

{

int tt=0;

int foodid,minn=1e9,newf=f[t];

if((get\_food(x,y,foodd[0].x,foodd[0].y)<=get\_food(x,y,foodd[1].x,foodd[1].y)&&vis[foodd[0].x][foodd[0].y])||!vis[foodd[1].x][foodd[1].y])

foodid=0;

else

foodid=1;

for(int i=0;i<4;i++)

if(f[t]^other(i))

{

if(check(x+flg[i][0],y+flg[i][1]))

{

int now=get\_food(x+flg[i][0],y+flg[i][1],foodd[foodid].x,foodd[foodid].y);

if(now<minn)

newf=i,tt=1,minn=now;

if(!tt)

newf=i;

}

}

f[t]=newf;

}

void result(int t) //积分够了--胜利

{

if(t==2)

lcdshowstr(1,1,"Robote1 Wine (Enough)");

else

lcdshowstr(1,1,"Robote2 Wine (Enough)");

while(1);

}

int ads(int x) //取绝对值

{

return x<0?-x:x; //x是否<0,如果x<0的话就执行:前面的语句，如果x不小于0的话就执行:后面的语句

}

int other(int x) //03交换、12交换

{

if (x==0) return 3;

else if(x==1) return 2;

else if(x==2) return 1;

else return 0;

}

int get\_food(int x,int y,int xxx,int yyy) //辅助机器人找寻食物

{

return ads(x-xxx)+ads(y-yyy);

}

**第六步**：结果展示



**第七步**：总结展望

功能上增加PVE、EVE模式。结构上实现读取（结构代码时序复杂，占用内存空间小）