

**单片机实验报告**

**系 别**  电气信息系

**专 业**  计算机科学与技术

**班 级**  17gb 计算3班

**姓 名**  唐可寅

**学 号**  201710253110

**实验一 LED实验**

**一、实验目的**

**1、**了解I/O访问方式。

**2、**熟悉简单的程序设计。

**3、**学习单片机IO口配置与驱动，实现指示灯LED1闪烁。

**二、实验设备**

计算机，STC单片机下载线，单片机教学系统。

**三、实验原理**

1、LED，英文全称是Light Emitting Diode，翻译成中文是发光二极管，最常用的电子器件之一。

2、本实验是由单片机通过I/O接口对发光二极管进行控制，由数据总线P2口的八位控制8个LED发光二极管。其中8个LED发光二极管为LED1-LED8。

**四、实验步骤**

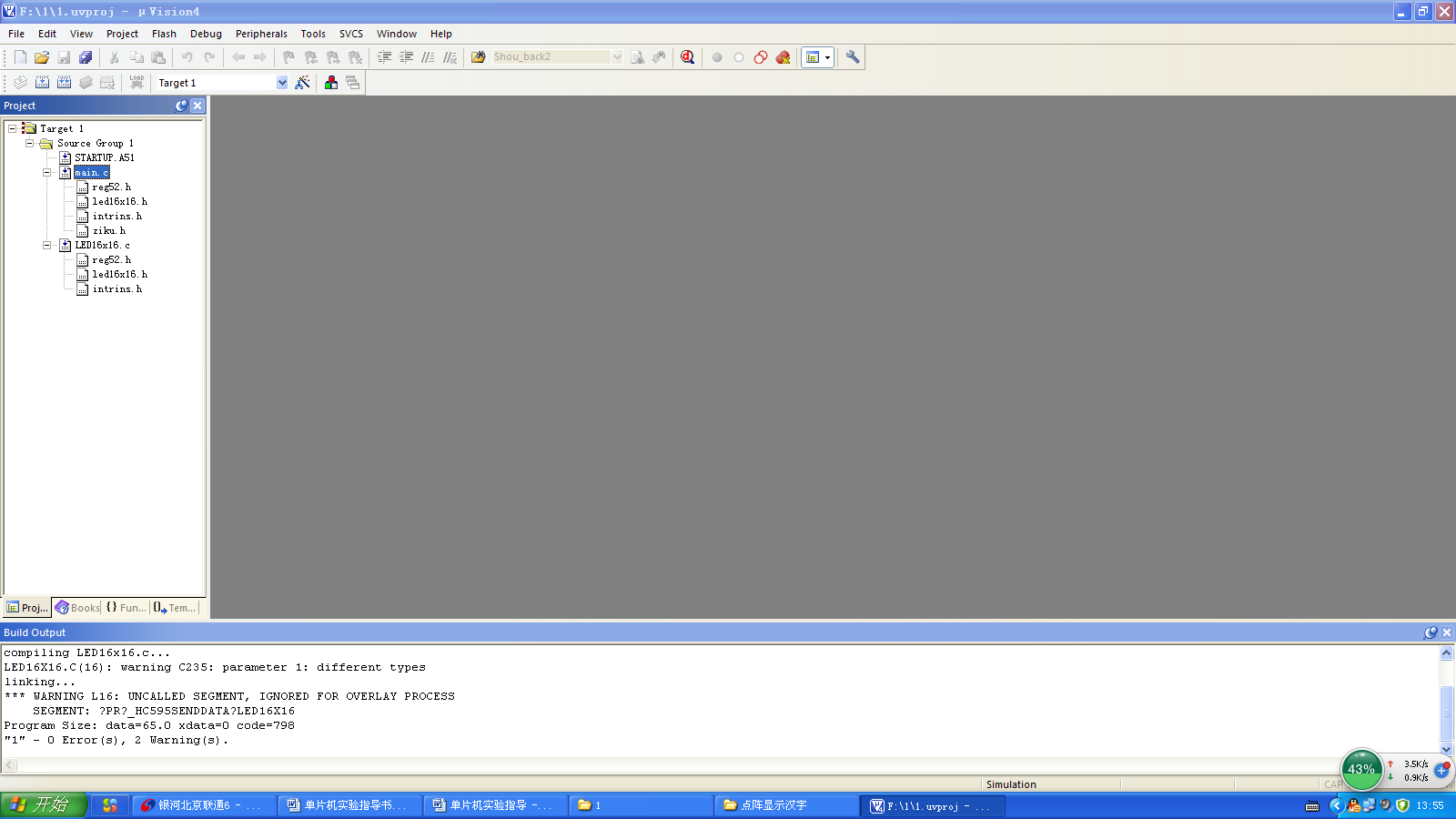
1、在可写入的F盘，新建一个工作文件夹，例如：文件夹名为“1”；

2、双击“”，进入“Keil uVision4”软件环境；

3、选择菜单栏中的“Project”项目中“New uVision Project…”选项，命名建立一个\*.pjt工程项目，弹出“Select a CPU Data Base File”窗口中，选择目标芯片系列：Generic CPU Data Base，点击“OK”；在弹出窗口“Data Base”项目中，选择“Atmel”，再选择“AT89C51”芯片，点击“OK”，弹出“Copy ‘STARTUP.A51’to Project Folder and Add File to Project”窗口中，选择“是”，项目建立完成。

4、将主窗口左侧工程管理窗口中“Target 1”的“+”点开，选择菜单栏中的“File”项目中“New”，根据类型输入源程序后另存为\*.h或 \*.c到\*.pjt所在的文件夹根目录下，注意后缀名.h及.c需要手动输入。项目中所需文件建立完成。

5、选择主窗口左侧工程管理窗口中“Target 1”的“+”点开，鼠标右键“Source Group”，选择菜单中“Add Files to Group ‘Soucre Group 1’ …”，弹出窗口中选择刚才保存的\*.c文件（注意\*.h不能添加），点击“Add”，然后点击“Close”，关闭该窗口。项目组建完成。

6、点击快捷工具栏中“”（Rebuild），对照报错窗口中的提示改错，当报错窗口中的错误（Errors）、警告（Warnings）均无误后，出现提示信息：0 Error（s）, 0 Warning（s），项目生成结果文件\*.hex（注意：点击“Project”中“Options for Target”，出现窗口点击“output”，勾选“Create HEX File”）。

7、下载程序结果，运用STC-ISP软件独立下载\*.hex结果文件到单片机教学系统。

8、运行程序结果，在教学系统对应的显示模块中出现相应的现象。

**五、写出实验源程序**

#include<reg51.h>

#include <intrins.h>

void delayms(unsigned char ms) {

unsigned char i;

while (ms--) {

for (i = 0; i < 120; i++);

}

}

void main() {

unsigned char LED;

LED = 0xfe;

P2 = LED;

while (1) {

delayms(250);

LED = \_crol\_(LED, 1);

P2 = LED;

}

}

**六、针对本实验写出实验心得**

通过本次实验，使我学会了Keil uVision4软件编程的基本步骤。通过I/O接口对发光二极管进行控制。这次试验使我明白单片机是一门应用性和实践性跟强的学科，要多动手，多做实验。

**实验二 数码管实验**

**一、实验目的**

**1、**了解数码管的接口定义及使用。

**2、**熟悉单片机控制数码管的程序设计。

**3、**掌握7段数码管的连接方式和动态显示法。

**二、实验设备**

计算机，STC单片机下载线，单片机教学系统。

**三、实验原理**

1、[led](http://baike.baidu.com/view/52538.htm" \t "_blank)数码管（LED Segment Displays）由多个[发光二极管](http://baike.baidu.com/view/84213.htm" \t "_blank)封装在一起组成“8”字型的器件，引线已在内部连接完成，只需引出它们的各个笔划，公共电极。[数码管](http://baike.baidu.com/view/556862.htm" \t "_blank)实际上是由七个发光管组成8字形构成的，加上小数点就是8个。这些段分别由字母a,b,c,d,e,f,g,dp来表示。

2、本实验是由单片机通过I/O接口对数码管进行控制，由数据总线P2口控制数码管显示，数据总线P1口接收矩阵键盘信息配合显示。

**四、实验步骤**

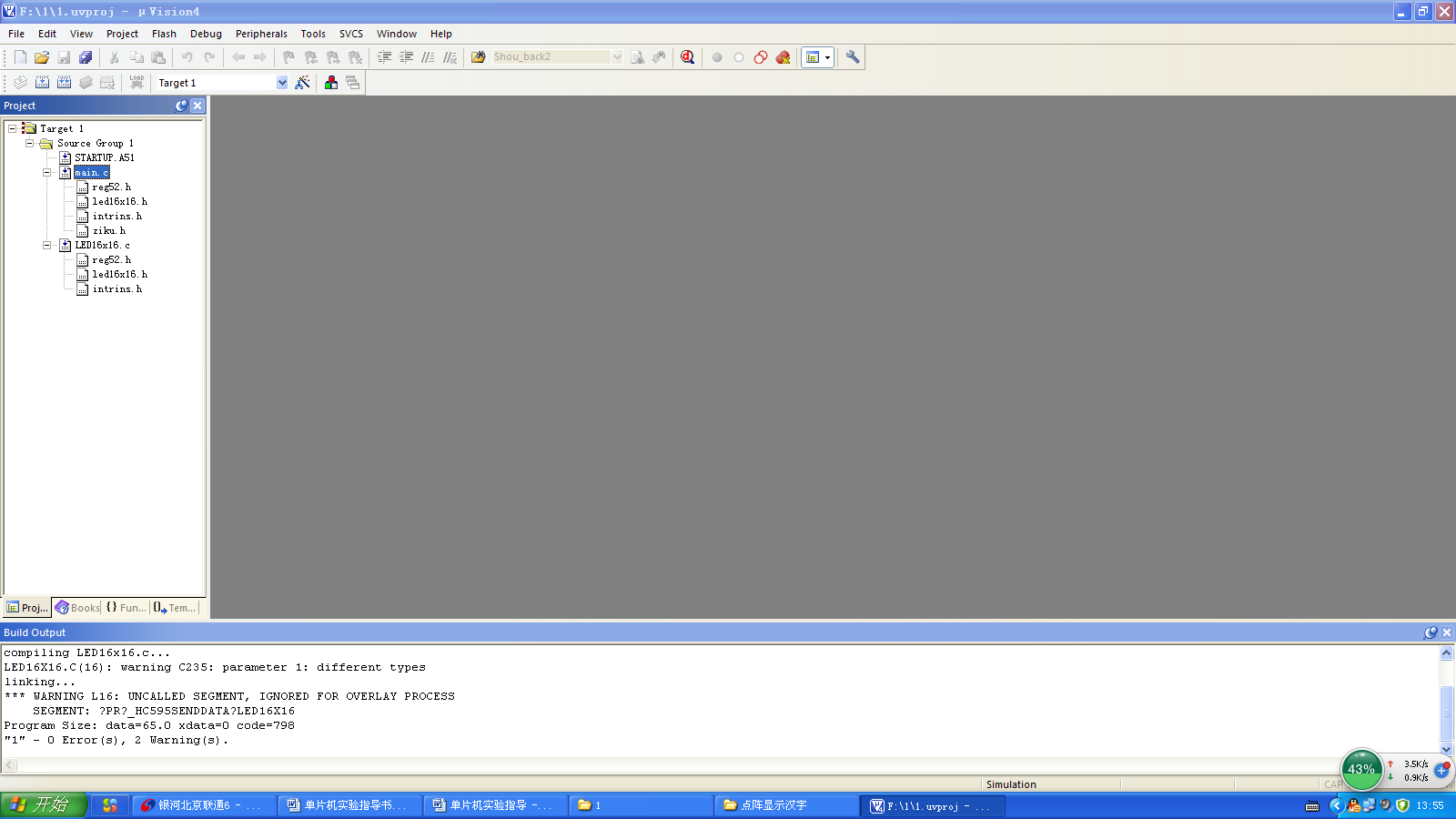
1、在可写入的F盘，新建一个工作文件夹，例如：文件夹名为“1”；

2、双击“”，进入“Keil uVision4”软件环境；

3、选择菜单栏中的“Project”项目中“New uVision Project…”选项，命名建立一个\*.pjt工程项目，弹出“Select a CPU Data Base File”窗口中，选择目标芯片系列：Generic CPU Data Base，点击“OK”；在弹出窗口“Data Base”项目中，选择“Atmel”，再选择“AT89C51”芯片，点击“OK”，弹出“Copy ‘STARTUP.A51’to Project Folder and Add File to Project”窗口中，选择“是”，项目建立完成。

4、将主窗口左侧工程管理窗口中“Target 1”的“+”点开，选择菜单栏中的“File”项目中“New”，根据类型输入源程序后另存为\*.h或 \*.c到\*.pjt所在的文件夹根目录下，注意后缀名.h及.c需要手动输入。项目中所需文件建立完成。

5、选择主窗口左侧工程管理窗口中“Target 1”的“+”点开，鼠标右键“Source Group”，选择菜单中“Add Files to Group ‘Soucre Group 1’ …”，弹出窗口中选择刚才保存的\*.c文件（注意\*.h不能添加），点击“Add”，然后点击“Close”，关闭该窗口。项目组建完成。

6、点击快捷工具栏中“”（Rebuild），对照报错窗口中的提示改错，当报错窗口中的错误（Errors）、警告（Warnings）均无误后，出现提示信息：0 Error（s）, 0 Warning（s），项目生成结果文件\*.hex（注意：点击“Project”中“Options for Target”，出现窗口点击“output”，勾选“Create HEX File”）。

7、下载程序结果，运用STC-ISP软件独立下载\*.hex结果文件到单片机教学系统。

8、运行程序结果，在教学系统对应的显示模块中出现相应的现象。

**五、写出实验源程序**

#include <intrins.h>

#include <reg52.h>

#define AT24C02 0xa0

#define NOP() \_nop\_()

sbit MOSIO = P2 ^0;

sbit R\_CLK = P2 ^1;

sbit S\_CLK = P2 ^2;

void delay(unsigned int i);

void HC595SendData(unsigned char SendVal, unsigned char Wei);

void Led\_Show(unsigned char Wei);

void SetLedNum(unsigned long int Numcode);

void system\_Ini();

void keyscan(void);

void SengUart(unsigned char SenData);

void SendString(unsigned char \*str);

void SengNum(unsigned int num);

unsigned char code

Disp\_Tab[] ={

0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90,0x88,0x83,0xC6,0xA1,0x86,0xbf,0xc7,0x8c,0xc1, 0xff, 0xf7 };

unsigned char code

LED7Code[] ={

~0x3F,~0x06,~0x5B,~0x4F,~0x66,~0x6D,~0x7D,~0x07,~0x7F,~0x6F,~0x77,~0x7C,~0x39,~0x5E,~0x79,~0x71};

unsigned char code

Nuntable[]="0123456789abcdef";

unsigned char NumBuffer[8];

unsigned int LedNum = 0;

unsigned int time = 0;

unsigned char P0flg;

unsigned char temp;

unsigned char key;

unsigned char ReData;

unsigned char pDat[8];

void main() {

unsigned long int Num = 0;

P0 = 0xff;

P1 = 0xff;

P2 = 0xff;

system\_Ini();

P0flg = 0;

NumBuffer[6] = pDat[5];

while (1) {

if (LedNum == 0) {

Num++;

Num %= 10000;

SetLedNum(Num);

}

keyscan();

}

}

void system\_Ini() {

TMOD = 0x21;

TH0 = (65536 - 30000) >> 8;

TL0 = (65536 - 30000) & 0xff;

ET0 = 1;

TR0 = 1;

SCON = 0x50;

TMOD |= 0x20;

PCON |= 0x80;

TH1 = 0xF3;

TL1 = 0xF3;

TR1 = 1;

ES = 1;

IT0 = 1;

EX0 = 1;

EA = 1;

}

void SetLedNum(unsigned long int Numcode) {

unsigned char i;

for (i = 0; i < 6; i++) {

NumBuffer[i] = Numcode % 10;

Numcode /= 10;

}

}

void Led\_Show(unsigned char Wei) {

unsigned char HC595SendVal;

HC595SendVal = ~Disp\_Tab[NumBuffer[Wei]];

HC595SendData(HC595SendVal, Wei);

}

void delay(unsigned int i) {

unsigned int j;

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

for (j = 300; j > 0; j--);

}

void HC595SendData(unsigned char SendVal, unsigned char Wei) {

unsigned char i;

for (i = 0; i < 16; i++) {

if (i < 8) {

if ((SendVal << i) & 0x80) MOSIO = 1;

else MOSIO = 0;

} else {

MOSIO = ((~(1 << Wei) >> (i - 8)) & 0x01);

}

S\_CLK = 0;

NOP();

NOP();

S\_CLK = 1;

}

R\_CLK = 0;

NOP();

NOP();

R\_CLK = 1;

}

void SengUart(unsigned char SenData) {

SBUF = SenData;

while (TI == 0);

TI = 0;

}

void SendString(unsigned char \*str) {

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

SengUart(\*str);

str++;

}

}

void SengNum(unsigned int num) {

unsigned char buffer[10];

unsigned char \*Buf = buffer + 8;

do {

\*Buf = Nuntable[num % 10];

Buf--;

num /= 10;

} while (num != 0);

buffer[9] = 0;

Buf++;

SendString(Buf);

}

void keyscan(void) {

temp = 0;

P1 = 0xF0;

delay(1);

temp = P1;

temp = temp & 0xF0;

temp = ~((temp >> 4) | 0xF0);

if (temp == 1)

key = 0;

else if (temp == 2)

key = 1;

else if (temp == 4)

key = 2;

else if (temp == 8)

key = 3;

else

key = 16;

P1 = 0x0F;

delay(1);

temp = P1;

temp = temp & 0x0F;

temp = ~(temp | 0xF0);

if (temp == 1)

key = key + 0;

else if (temp == 2)

key = key + 4;

else if (temp == 4)

key = key + 8;

else if (temp == 8)

key = key + 12;

else

key = 16;

if (key < 16) {

NumBuffer[7] = key;

SendString("get the key number: ");

SengNum((unsigned int) key);

SendString("\r\n");

if (key == 0) {

SendString("write num 5 to 24c02 !\r\n");

} else if (key == 1) {

SendString("read num from 24c02 : ");

NumBuffer[6] = pDat[0];

SengNum((unsigned int) pDat[0]);

SendString("\r\n");

}

}

}

void counter(void) interrupt 0 {

EX0 = 0;

EX0 = 1;

}

void T1zd(void) interrupt 1 {

TH0 = (65536 - 3000) >> 8;

TL0 = (65536 - 3000) & 0xff;

time++;

if (time == 10) {

P0flg++;

P0flg %= 16;

time = 0;

if (P0flg < 8)

P0 = ~(0x01 << P0flg);

else

P0 = ~(0x80 >> (P0flg - 8));

}

LedNum++;

LedNum %= 8;

Led\_Show(LedNum);

}

void ser\_int(void) interrupt 4 using 1 {

if (RI == 1) {

RI = 0;

ReData = SBUF;

}

}

**六、针对本实验写出实验心得**

通过本次实验使我明确了研究目标。总结这次实验，只有实际动手操作才有可能出现理想的结果。自己不动手永远不知道这其中的原理所以这次实验使我受益匪浅。

**实验三 点阵实验**

**一、实验目的**

**1、**了解点阵的接口定义及使用。

**2、**熟悉单片机控制点阵的程序设计。

**3、**了解点阵式LED显示原理。

**二、实验设备**

计算机，STC单片机下载线，单片机教学系统。

**三、实验原理**

1、LED点阵屏通过[LED](http://baike.baidu.com/view/52538.htm" \t "_blank)([发光二极管](http://baike.baidu.com/subview/84213/84213.htm" \t "_blank)）组成，以灯珠亮灭来显示文字、图片、动画、视频等，是各部分组件都模块化的[显示器件](http://baike.baidu.com/view/3844065.htm" \t "_blank)，通常由显示模块、[控制系统](http://baike.baidu.com/view/57978.htm" \t "_blank)及[电源系统](http://baike.baidu.com/view/3317818.htm" \t "_blank)组成。

2、8\*8[点阵](http://baike.baidu.com/view/545607.htm" \t "_blank)，它共由64个[发光二极管](http://baike.baidu.com/subview/84213/84213.htm" \t "_blank)组成，且每个发光二极管是放置在行线和列线的交叉点上，当对应的某一行置1[电平](http://baike.baidu.com/view/314180.htm" \t "_blank)，某一列置0电平，则相应的二极管就亮。一般我们使用点阵显示汉字是用的16\*16的点阵宋体[字库](http://baike.baidu.com/view/1127103.htm" \t "_blank)，所谓16\*16，是每一个汉字在纵、横各16点的区域内显示的。也就是说用四个8\*8点阵组合成一个16\*16的点阵。

3、本实验是由单片机通过I/O接口对16\*16点阵进行控制，显示汉字。

**四、实验步骤**

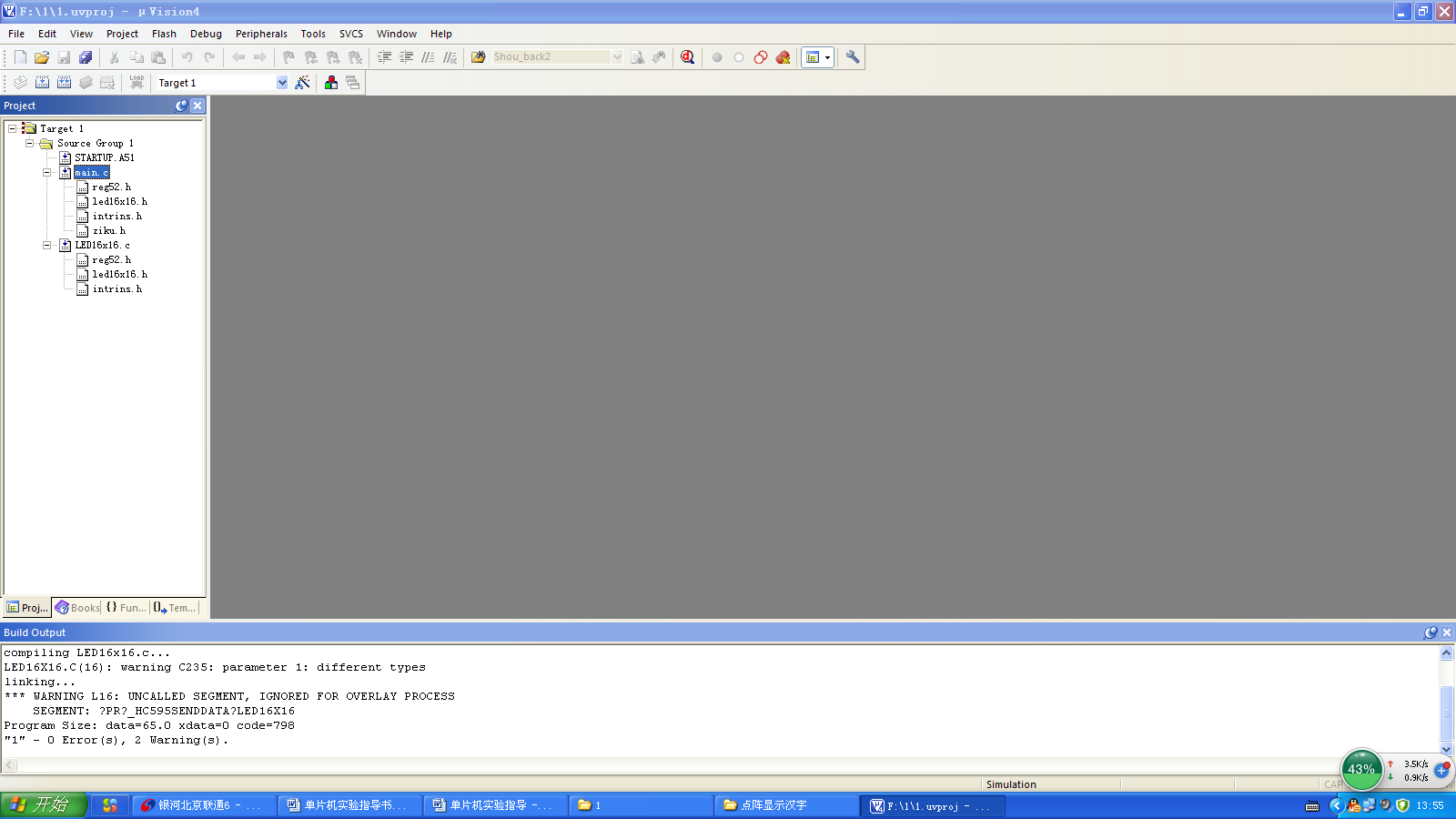
1、在可写入的F盘，新建一个工作文件夹，例如：文件夹名为“1”；

2、双击“”，进入“Keil uVision4”软件环境；

3、选择菜单栏中的“Project”项目中“New uVision Project…”选项，命名建立一个\*.pjt工程项目，弹出“Select a CPU Data Base File”窗口中，选择目标芯片系列：Generic CPU Data Base，点击“OK”；在弹出窗口“Data Base”项目中，选择“Atmel”，再选择“AT89C51”芯片，点击“OK”，弹出“Copy ‘STARTUP.A51’to Project Folder and Add File to Project”窗口中，选择“是”，项目建立完成。

4、将主窗口左侧工程管理窗口中“Target 1”的“+”点开，选择菜单栏中的“File”项目中“New”，根据类型输入源程序后另存为\*.h或 \*.c到\*.pjt所在的文件夹根目录下，注意后缀名.h及.c需要手动输入。项目中所需文件建立完成。

5、选择主窗口左侧工程管理窗口中“Target 1”的“+”点开，鼠标右键“Source Group”，选择菜单中“Add Files to Group ‘Soucre Group 1’ …”，弹出窗口中选择刚才保存的\*.c文件（注意\*.h不能添加），点击“Add”，然后点击“Close”，关闭该窗口。项目组建完成。

6、点击快捷工具栏中“”（Rebuild），对照报错窗口中的提示改错，当报错窗口中的错误（Errors）、警告（Warnings）均无误后，出现提示信息：0 Error（s）, 0 Warning（s），项目生成结果文件\*.hex（注意：点击“Project”中“Options for Target”，出现窗口点击“output”，勾选“Create HEX File”）。

7、下载程序结果，运用STC-ISP软件独立下载\*.hex结果文件到单片机教学系统。

8、运行程序结果，在教学系统对应的显示模块中出现相应的现象。

**五、写出实验源程序**

**实验（一）**

//.h文件

#ifndef \_\_LED16X16\_H\_

#define \_\_LED16X16\_H\_

#include <intrins.h>

#include<reg52.h>

#define COW1 (0<<4)

#define COW2 (1<<4)

#define COW3 (2<<4)

#define COW4 (3<<4)

#define COW5 (4<<4)

#define COW6 (5<<4)

#define COW7 (6<<4)

#define COW8 (7<<4)

#define COW9 (8<<4)

#define COW10 (9<<4)

#define COW11 (10<<4)

#define COW12 (11<<4)

#define COW13 (12<<4)

#define COW14 (13<<4)

#define COW15 (14<<4)

#define COW16 (15<<4)

#define DATALONG 16

#define ADDRESS P2

#define NOP()

sbit ADDA = P2^4;

sbit ADDB = P2^5;

sbit ADDC = P2^6;

sbit ADDD = P2^7;

sbit SHCP = P2^3;

sbit STCP = P2^2;

sbit DAIN = P2^1;

extern unsigned char mScanTable[16];

void HC595SendData(unsigned char SendVal);

void DISPLAY(unsigned int SendVal,unsigned char wei);

void sysShow();

void DrawDot(unsigned int \* Chat);

#endif

//.h文件

#ifndef \_\_ZIKU\_H\_

#define \_\_ZIKU\_H\_

unsigned char code ziku[]={

0x20,0x02,0x70,0x0A,0x1E,0x12,0x10,0x12,0x10,0x02,0xFF,0x7F,0x10,0x02,0x10,0x22,0x50,0x22,0x30,0x12,0x18,0x0C,0x16,0x44,0x10,0x4A,0x10,0x51,0xD4,0x60,0x08,0x40,0x00,0x10,0x80,0x3F,0x7E,0x08,0x44,0x08,0x88,0x04,0xFE,0x7F,0x42,0x40,0x41,0x20,0xFE,0x1F,0x20,0x00,0xE0,0x0F,0x50,0x08,0x88,0x04,0x04,0x03,0xC2,0x0C,0x38,0x70,0x08,0x08,0x10,0x04,0x20,0x02,0xFC,0x1F,0x84,0x10,0x84,0x10,0xFC,0x1F,0x84,0x10,0x84,0x10,0xFC,0x1F,0x80,0x00,0x80,0x00,0xFF,0x7F,0x80,0x00,0x80,0x00,0x80,0x00,0x00,0x02,0x08,0x02,0x08,0x02,0x08,0x02,0x08,0x02,0xF8,0x3F,0x08,0x00,0x08,0x00,0x08,0x00,0xF8,0x07,0x08,0x04,0x08,0x04,0x08,0x04,0x04,0x04,0x04,0x04,0x02,0x04,0x08,0x00,0x88,0x0F,0x88,0x08,0x88,0x08,0xBF,0x08,0x88,0x08,0x8C,0x08,0x9C,0x08,0xAA,0x08,0xAA,0x08,0x89,0x08,0x88,0x48,0x88,0x48,0x48,0x48,0x48,0x70,0x28,0x00,0x00,0x00,0x08,0x00,0x08,0x00,0x08,0x00,0x08,0x00,0x08,0x00,0x08,0x00,0x08,0x00,0x08,0x00,0x08,0x00,0x00,0x00,0x00,0x00,0x08,0x00,0x08,0x00,0x00,0x00,0x00,0x00,};

#endif

//.c文件

#include "reg52.h"

#include "LED16x16.h"

unsigned char mScanTable[16] = {

COW1, COW2, COW3, COW4,

COW5, COW6, COW7, COW8,

COW9, COW10, COW11, COW12,

COW13, COW14, COW15, COW16

};

unsigned int Buffer[16];

void HC595SendData(unsigned int SendVal) {

unsigned char i;

for (i = 0; i < DATALONG; i++) {

if ((SendVal << i) & 0x8000)

DAIN = 0;

else

DAIN = 1;

SHCP = 0;

NOP();

NOP();

SHCP = 1;

}

STCP = 0;

NOP();

NOP();

}

void DISPLAY(unsigned int SendVal, unsigned char wei) {

unsigned char i;

for (i = 0; i < DATALONG; i++) {

if ((SendVal << i) & 0x8000)

DAIN = 0;

else

DAIN = 1;

SHCP = 0;

NOP();

NOP();

SHCP = 1;

}

STCP = 0;

NOP();

NOP();

ADDRESS = (ADDRESS & 0x0f) | mScanTable[wei];

STCP = 1;

}

unsigned char mCow = 0;

void DrawDot(unsigned int \*Chat) {

unsigned char i = 0;

for (i = 0; i < 16; i++) {

Buffer[i] = \*Chat++;

}

}

void sysShow() {

DISPLAY(Buffer[mCow], mCow);

mCow++;

mCow &= 0x0f;

DISPLAY(Buffer[mCow], mCow);

mCow++;

mCow &= 0x0f;

}

//.c文件

#include "reg52.h"

#include "led16x16.h"

#include "ziku.h"

void delays(unsigned int time) {

unsigned int i, j;

for (i = 0; i < time; i++)

for (j - 0; j < 1000; j++);

}

void system\_Ini() {

TMOD = 0x21;

TH0 = (65536 - 30000) >> 8; //12.000

TL0 = (65536 - 30000) & 0xff;

ET0 = 1;

TR0 = 1;

SCON = 0x50;

TMOD |= 0x20;

PCON |= 0x80;

TH1 = 0xF3;

TL1 = 0xF3;

TR1 = 1;

ES = 1;

IT0 = 0;

IT0 = 1;

EX0 = 1;

EA = 1;

}

void main(void) {

unsigned char i = 0;

unsigned char \*CH = ziku;

system\_Ini();

while (1) {

CH = ziku;

for (i = 0; i < 80; i++) {

DrawDot((unsigned int \*) CH);

delays(500);

CH += 2;

if (!(i & 0x0f))

delays(3900);

}

}

}

**实验（二）**

//.h文件

#ifndef \_\_BUTTON\_DRIVE\_H\_

#define \_\_BUTTON\_DRIVE\_H\_

void game\_button() {

switch (basic\_button()) {

case 3:

if (s\_box.y != 0) {

EA = 0;

if (s\_box.shape == 3 & check\_cover(s\_box.x, s\_box.y, box\_read\_data(s\_box.mode, 0))) {

s\_box.shape = 0;

box\_load();

box\_to\_Box\_Ram(s\_box.x, s\_box.y, box\_read\_data(s\_box.mode, 3));

Box\_Ram\_to\_Ram();

} else {

if (check\_cover(s\_box.x, s\_box.y, box\_read\_data(s\_box.mode, s\_box.shape + 1))) {

s\_box.shape++;

box\_load();

box\_to\_Box\_Ram(s\_box.x, s\_box.y, box\_read\_data(s\_box.mode, s\_box.shape - 1));

Box\_Ram\_to\_Ram();

}

}

EA = 1;

}

break;

case 1:

if (s\_box.y != 0) {

EA = 0;

while (check\_cover(s\_box.x, s\_box.y + 1, s\_box.box)) {

s\_box.y++;

box\_to\_Box\_Ram(s\_box.x, s\_box.y - 1, s\_box.box);

Box\_Ram\_to\_Ram();

}

destroy\_row();

box\_build();

box\_load();

game\_over\_flag = check\_game\_over();

next\_box();

box\_to\_Box\_Ram(s\_box.x, s\_box.y, s\_box.box);

Box\_Ram\_to\_Ram();

EA = 1;

}

break;

case 4:

if (s\_box.y != 0) {

EA = 0;

if (s\_box.x != 0 & check\_cover(s\_box.x - 1, s\_box.y, s\_box.box)) {

s\_box.x--;

box\_to\_Box\_Ram(s\_box.x + 1, s\_box.y, s\_box.box);

Box\_Ram\_to\_Ram();

}

EA = 1;

}

break;

case 5:

if (s\_box.y != 0) {

EA = 0;

if (check\_cover(s\_box.x + 1, s\_box.y, s\_box.box)) {

s\_box.x++;

box\_to\_Box\_Ram(s\_box.x - 1, s\_box.y, s\_box.box);

Box\_Ram\_to\_Ram();

}

EA = 1;

}

break;

case 2:

EA = 0;

pause\_game\_flag = 1;

break;

default:;

}

}

unsigned char basic\_button() {

unsigned char tpflag = 0;

if (down == 0) {

if (down\_reg < button\_delay) {

down\_reg++;

} else {

down\_reg = 0;

tpflag = 1;

}

} else {

down\_reg = button\_delay;

}

if (up == 0) {

if (up\_reg < button\_delay) {

up\_reg++;

} else {

up\_reg = 0;

tpflag = 2;

}

} else {

up\_reg = button\_delay;

}

if (button\_a == 0) {

if (button\_a\_reg < button\_delay) {

button\_a\_reg++;

} else {

button\_a\_reg = 0;

tpflag = 3;

}

} else {

button\_a\_reg = button\_delay;

}

if (left == 0) {

if (left\_reg < button\_delay) {

left\_reg++;

} else {

left\_reg = 0;

tpflag = 4;

}

} else {

left\_reg = button\_delay;

}

if (right == 0) {

if (right\_reg < button\_delay) {

right\_reg++;

} else {

right\_reg = 0;

tpflag = 5;

}

} else {

right\_reg = button\_delay;

}

return (tpflag);

}

#endif

//.h文件

#ifndef DISPLAY\_DRIVER\_H

#define DISPLAY\_DRIVER\_H

#include <intrins.h>

#include<reg52.h>

#define COW1 (0<<4)

#define COW2 (1<<4)

#define COW3 (2<<4)

#define COW4 (3<<4)

#define COW5 (4<<4)

#define COW6 (5<<4)

#define COW7 (6<<4)

#define COW8 (7<<4)

#define COW9 (8<<4)

#define COW10 (9<<4)

#define COW11 (10<<4)

#define COW12 (11<<4)

#define COW13 (12<<4)

#define COW14 (13<<4)

#define COW15 (14<<4)

#define COW16 (15<<4)

#define DATALONG 16

#define ADDRESS P2

#define NOP()

sbit ADDA = P2 ^4;

sbit ADDB = P2 ^5;

sbit ADDC = P2 ^6;

sbit ADDD = P2 ^7;

sbit SHCP = P2 ^3;

sbit STCP = P2 ^2;

sbit DAIN = P2 ^1;

unsigned char Ram[] =

{

0x7F, 0x02, 0x1F, 0x10, 0x1F, 0x10, 0x1F, 0x10, 0xFE, 0x00, 0xF0, 0x10, 0xF0, 0x10, 0xF0, 0x10,

0x1F, 0x08, 0x0F, 0x08, 0x16, 0x21, 0x0E, 0x70, 0xF0, 0x00, 0xF0, 0x20, 0x40, 0x80, 0x70, 0x0E,

};

void delay(unsigned char temp) {

unsigned char tp = temp;

while (tp--);

}

unsigned char mScanTable[16] = {

COW1, COW2, COW3, COW4,

COW5, COW6, COW7, COW8,

COW9, COW10, COW11, COW12,

COW13, COW14, COW15, COW16

};

unsigned int Buffer[16];

void HC595SendData(unsigned int SendVal) {

unsigned char i;

for (i = 0; i < DATALONG; i++) {

if ((SendVal << i) & 0x8000)

DAIN = 0;

else

DAIN = 1;

SHCP = 0;

NOP();

NOP();

SHCP = 1;

}

STCP = 0;

NOP();

NOP();

}

void DISPLAY(unsigned char \*SendVal, unsigned char wei) {

unsigned char i;

unsigned int date;

date = ((unsigned int) (\*SendVal)) | ((\*(SendVal + 8)) << 8);

for (i = 0; i < DATALONG; i++) {

if ((date >> i) & 0x0001)

DAIN = 0;

else

DAIN = 1;

SHCP = 0;

NOP();

NOP();

SHCP = 1;

}

STCP = 0;

NOP();

NOP();

ADDRESS = (ADDRESS & 0x0f) | mScanTable[wei];

STCP = 1;

}

unsigned char mCow = 0;

void DrawDot(unsigned int \*Chat) {

unsigned char i = 0;

for (i = 0; i < 16; i++) {

Ram[i] = \*Chat++;

}

}

void sysShow() {

unsigned char i = 0;

unsigned char \*CH = Ram;

for (i = 0; i < 16; i++) {

if (i < 8)

DISPLAY((CH) + mCow, mCow);

else

DISPLAY((CH + 8) + mCow, mCow);

mCow++;

mCow &= 0x0f;

}

}

void display() {

sysShow();

}

#endif

//.h文件

#ifndef \_\_TETRIS\_H\_\_

#define \_\_TETRIS\_H\_\_

void box\_build();

unsigned int box\_read\_data(unsigned char tpmode, unsigned char tpshape);

void box\_load();

void box\_to\_Box\_Ram(unsigned char tpx, unsigned char tpy, unsigned int tpbox);

void Box\_Ram\_to\_Ram();

void game\_execute();

void time0\_initialize();

bit check\_cover(unsigned char tpx, unsigned char tpy, unsigned int tpbox);

void destroy\_row();

void next\_box();

void Tetris\_main();

void game\_over\_show();

void game\_initialize();

void game\_start\_show();

bit check\_game\_over();

void check\_pause\_game();

#endif

//.h文件

unsigned int code

Box\_Ram\_data[]={0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0xffff,0x0000,0x0000};

unsigned int code

game\_data[]={0x64DB,0x8AAA,0x8AAA,0x8AAB,0xEEAA,0xAAAA,0xEAAB,0x0000};

unsigned int code

over\_data[]={0x6566,0x9549,0x9549,0x956F,0x954A,0x9549,0x6268,0x0000};

unsigned int code

score\_data[]={0xC000,0x8000,0x9BB7,0xD2A5,0x52A7,0x52A4,0xDBA7,0x0000};

unsigned int code

tetris\_data[]={0xE000,0x4008,0x4000,0x5A6B,0x574A,0x5A4B,0x5249,0x5B4B};

unsigned long code

num\_data[]={0xF99999F0,0x11111110,0xF11F88F0,0xF11F11F0,0x999F1110,0xF88F11F0,0xF88F99F0,0xF1111110,0xF99F99F0,0xF99F11F0,};

unsigned int idata

Box\_Ram[19];

unsigned char box\_down\_reg;

unsigned char time0\_reg;

unsigned char next\_mode;

unsigned char next\_shape;

bit game\_over\_flag;

bit pause\_game\_flag;

struct {

unsigned char mode;

unsigned char shape;

unsigned char x;

unsigned char y;

unsigned int box;

} s\_box;

void box\_build();

unsigned int box\_read\_data(unsigned char tpmode, unsigned char tpshape);

void box\_load();

void box\_to\_Box\_Ram(unsigned char tpx, unsigned char tpy, unsigned int tpbox);

void Box\_Ram\_to\_Ram();

void game\_execute();

void time0\_initialize();

bit check\_cover(unsigned char tpx, unsigned char tpy, unsigned int tpbox);

void destroy\_row();

void next\_box();

void Tetris\_main();

void game\_over\_show();

void game\_initialize();

void game\_start\_show();

bit check\_game\_over();

void check\_pause\_game();

#ifndef \_\_TETRIS\_DEFINE\_H\_\_

#define \_\_TETRIS\_DEFINE\_H\_\_

#define button\_delay 600

sbit button\_a = P0 ^7;

sbit up = P3 ^4;

sbit down = P3 ^5;

sbit left = P3 ^6;

sbit right = P3 ^7;

unsigned int up\_reg = button\_delay;

unsigned int down\_reg = button\_delay;

unsigned int left\_reg = button\_delay;

unsigned int right\_reg = button\_delay;

unsigned int button\_a\_reg = button\_delay;

void game\_button();

unsigned char basic\_button();

#endif

//.c文件

#include <reg52.h>

#include <stdlib.h>

#include "Tetris\_define.h"

#include "Tetris.h"

#include "display\_drive.h"

#include "button\_drive.h"

void box\_build() {

s\_box.mode = next\_mode;

s\_box.shape = next\_shape;

s\_box.x = 3;

s\_box.y = 0;

next\_mode = rand() % 7;

next\_shape = (rand() / 10) % 4;

}

unsigned int box\_read\_data(unsigned char tpmode, unsigned char tpshape) {

unsigned int tpbox;

switch (tpmode) {

case 0:

switch (tpshape) {

case 0:

tpbox = 0xf000;break;

case 1:

tpbox = 0x8888;break;

case 2:

tpbox = 0xf000;break;

case 3:

tpbox = 0x8888;break;

default:;

}

break;

case 1:

switch (tpshape) {

case 0:

tpbox = 0xe800;break;

case 1:

tpbox = 0xc440;break;

case 2:

tpbox = 0x2e00;break;

case 3:

tpbox = 0x88c0;break;

default:;

}

break;

case 2:

switch (tpshape) {

case 0:

tpbox = 0xe200;break;

case 1:

tpbox = 0x44c0;break;

case 2:

tpbox = 0x8e00;break;

case 3:

tpbox = 0xc880;break;

default:;

}

break;

case 3:

switch (tpshape) {

case 0:

tpbox = 0xcc00;break;

case 1:

tpbox = 0xcc00;break;

case 2:

tpbox = 0xcc00;break;

case 3:

tpbox = 0xcc00;break;

default:;

}

break;

case 4:

switch (tpshape) {

case 0:

tpbox = 0xc600;break;

case 1:

tpbox = 0x4c80;break;

case 2:

tpbox = 0xc600;break;

case 3:

tpbox = 0x4c80;break;

default:;

}

break;

case 5:

switch (tpshape) {

case 0:

tpbox = 0x6c00;break;

case 1:

tpbox = 0x8c40;break;

case 2:

tpbox = 0x6c00;break;

case 3:

tpbox = 0x8c40;break;

default:;

}

break;

case 6:

switch (tpshape) {

case 0:

tpbox = 0x4e00;break;

case 1:

tpbox = 0x8c80;break;

case 2:

tpbox = 0xe400;break;

case 3:

tpbox = 0x4c40;break;

default:;

}

break;

default:;

}

return (tpbox);

}

void box\_load() {

s\_box.box = box\_read\_data(s\_box.mode, s\_box.shape);

}

void box\_to\_Box\_Ram(unsigned char tpx, unsigned char tpy, unsigned int tpbox) {

unsigned char i;

unsigned int temp;

temp = tpbox;

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

Box\_Ram[3 - i + tpy] = Box\_Ram[3 - i + tpy] & (~((temp & 0x000f) << (12 - tpx)));

temp = temp >> 4;

}

temp = s\_box.box;

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

Box\_Ram[3 - i + s\_box.y] = ((temp & 0x000f) << (12 - s\_box.x)) | Box\_Ram[3 - i + s\_box.y];

temp = temp >> 4;

}

}

void Box\_Ram\_to\_Ram() {

unsigned char i;

for (i = 0; i < 8; i++) {

Ram[i] = (Box\_Ram[i] >> 8) & 0x00ff;

Ram[i + 8] = Box\_Ram[i] & 0x00ff;

Ram[i + 16] = (Box\_Ram[i + 8] >> 8) & 0x00ff;

Ram[i + 24] = Box\_Ram[i + 8] & 0x00ff;

}

}

void game\_execute() {

if (box\_down\_reg < 20) {

box\_down\_reg++;

} else {

box\_down\_reg = 0;

if (check\_cover(s\_box.x, s\_box.y + 1, s\_box.box)) {

s\_box.y++;

box\_to\_Box\_Ram(s\_box.x, s\_box.y - 1, s\_box.box);

Box\_Ram\_to\_Ram();

} else {

destroy\_row();

box\_build();

box\_load();

game\_over\_flag = check\_game\_over();

next\_box();

box\_to\_Box\_Ram(s\_box.x, s\_box.y, s\_box.box);

Box\_Ram\_to\_Ram();

}

}

}

void time0\_initialize() {

TMOD = 0x03;

TR0 = 1;

ET0 = 1;

EA = 1;

srand(32667);

}

void timer0() interrupt 1{

TH0 = 0;

TL0 = 0;

if (time0\_reg < 10) {

time0\_reg++;

} else {

time0\_reg = 0;

if (!game\_over\_flag)

game\_execute();

display();

}

}

bit check\_cover(unsigned char tpx, unsigned char tpy, unsigned int tpbox) {

unsigned char i;

bit tpflag = 1;

unsigned int temp;

temp = s\_box.box;

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

Box\_Ram[3 - i + s\_box.y] = Box\_Ram[3 - i + s\_box.y] & (~((temp & 0x000f) << (12 - s\_box.x)));

temp = temp >> 4;

}

temp = tpbox;

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

if ((((temp & 0x000f) << (12 - tpx)) & Box\_Ram[3 - i + tpy]) != 0x0000) {

tpflag = 0;

}

temp = temp >> 4;

}

temp = s\_box.box;

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

Box\_Ram[3 - i + s\_box.y] = ((temp & 0x000f) << (12 - s\_box.x)) | Box\_Ram[3 - i + s\_box.y];

temp = temp >> 4;

}

return (tpflag);

}

void destroy\_row() {

unsigned char i, j = 0;

unsigned char tpflag[4] = {0, 0, 0, 0};

for (i = 0; i < 16; i++) {

if ((Box\_Ram[i] & 0xffc0) == 0xffc0) {

tpflag[j] = i + 1;

j++;

if (j == 4) {

break;

}

}

}

for (j = 0; j < 4; j++) {

if (tpflag[j] != 0) {

for (i = tpflag[j] - 1; i > 0; i--) {

Box\_Ram[i] = (Box\_Ram[i - 1] & 0xffc0) | (Box\_Ram[i] & 0x003f);

Box\_Ram[0] = 0x0000 | (Box\_Ram[0] & 0x003f);

}

}

}

}

void next\_box() {

unsigned char i;

unsigned int temp;

temp = box\_read\_data(next\_mode, next\_shape);

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

Box\_Ram[3 - i] = (temp & 0x000f) | (Box\_Ram[3 - i] & 0xfff0);

temp = temp >> 4;

}

}

void Tetris\_main() {

unsigned char i;

for (i = 0; i < 19; i++) {

Box\_Ram[i] = Box\_Ram\_data[i];

};

game\_over\_flag = 0;

box\_build();

box\_load();

next\_box();

box\_to\_Box\_Ram(s\_box.x, s\_box.y, s\_box.box);

Box\_Ram\_to\_Ram();

time0\_initialize();

while (!game\_over\_flag) {

game\_button();

check\_pause\_game();

}

game\_over\_show();

}

void game\_over\_show() {

unsigned char i;

bit tpflag = 1;

for (i = 0; i < 8; i++) {

Box\_Ram[i] = game\_data[i];

Box\_Ram[i + 8] = over\_data[i];

}

Box\_Ram\_to\_Ram();

while (1) {

}

}

void game\_initialize() {

box\_down\_reg = 0;

time0\_reg = 0;

next\_mode = 6;

next\_shape = 2;

game\_over\_flag = 0;

pause\_game\_flag = 0;

}

void game\_start\_show() {

unsigned char i;

bit tpflag = 1;

game\_initialize();

for (i = 0; i < 16; i++) {

Box\_Ram[i] = 0x0000;

};

for (i = 0; i < 8; i++) {

Box\_Ram[i + 3] = tetris\_data[i];

}

Box\_Ram\_to\_Ram();

while (tpflag) {

display();

switch (basic\_button()) {

case 3:

tpflag = 0;

break;

default:;

}

}

Tetris\_main();

}

bit check\_game\_over() {

unsigned char i;

bit tpflag = 0;

unsigned int temp;

temp = s\_box.box;

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

if ((((temp & 0x000f) << (12 - s\_box.x)) & Box\_Ram[3 - i + s\_box.y]) != 0x0000) {

tpflag = 1;

}

temp = temp >> 4;

}

return (tpflag);

}

void check\_pause\_game() {

if (pause\_game\_flag) {

while (basic\_button() != 2) {

display();

}

}

pause\_game\_flag = 0;

EA = 1;

}

//.c文件

#include "reg51.h"

#include "Tetris.h"

void main() {

time0\_initialize();

game\_initialize();

Tetris\_main();

while (1) { }

}

**六、针对本实验写出实验心得**

科学的魅力无穷大。现在只用写一个程序烧入单片机就可以实现各种功能。程序一直在增加，实现的功能也在变换着，越来越觉得单片机实验的功能，在生活中也随处可见。

**实验四 12864液晶屏实验**

**一、实验目的**

**1、**了解12864液晶屏的接口定义及使用。

**2、**熟悉单片机控制12864液晶屏的程序设计。

**3、**熟悉液晶显示的操作。

**二、实验设备**

计算机，STC单片机下载线，单片机教学系统。

**三、实验原理**

1、LCD是英文Liquid Crystal Display 的简写，即为液晶显示。

2、带中文字库的12864液晶屏是一种具有4位/8位并行、2线或3线串行多种接口方式，内部含有国标一级、二级简体中文字库的点阵图形液晶显示模块；其显示分辨率为128×64, 内置8192个16\*16点汉字，和128个16\*8点ASCII字符集.利用该模块灵活的接口方式和简单、方便的操作指令，可构成全中文人机交互图形界面。可以显示8×4行16×16点阵的汉字，也可完成图形显示。低电压低功耗是其又一显著特点。

3、本实验是由单片机通过I/O接口对12864液晶屏进行显示控制，在可显示范围按要求显示。

**四、实验步骤**

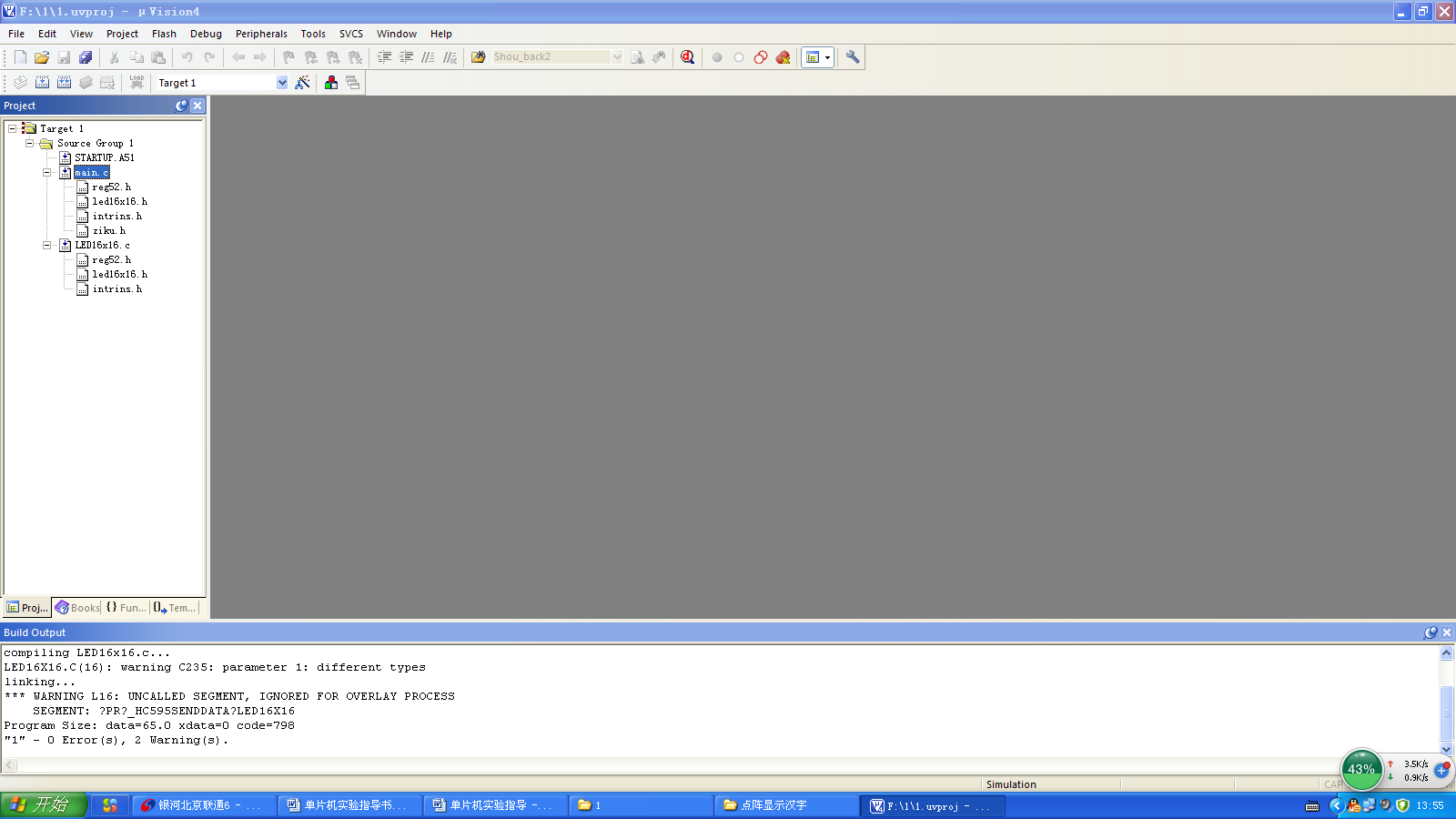
1、在可写入的F盘，新建一个工作文件夹，例如：文件夹名为“1”；

2、双击“”，进入“Keil uVision4”软件环境；

3、选择菜单栏中的“Project”项目中“New uVision Project…”选项，命名建立一个\*.pjt工程项目，弹出“Select a CPU Data Base File”窗口中，选择目标芯片系列：Generic CPU Data Base，点击“OK”；在弹出窗口“Data Base”项目中，选择“Atmel”，再选择“AT89C51”芯片，点击“OK”，弹出“Copy ‘STARTUP.A51’to Project Folder and Add File to Project”窗口中，选择“是”，项目建立完成。

4、将主窗口左侧工程管理窗口中“Target 1”的“+”点开，选择菜单栏中的“File”项目中“New”，根据类型输入源程序后另存为\*.h或 \*.c到\*.pjt所在的文件夹根目录下，注意后缀名.h及.c需要手动输入。项目中所需文件建立完成。

5、选择主窗口左侧工程管理窗口中“Target 1”的“+”点开，鼠标右键“Source Group”，选择菜单中“Add Files to Group ‘Soucre Group 1’ …”，弹出窗口中选择刚才保存的\*.c文件（注意\*.h不能添加），点击“Add”，然后点击“Close”，关闭该窗口。项目组建完成。

6、点击快捷工具栏中“”（Rebuild），对照报错窗口中的提示改错，当报错窗口中的错误（Errors）、警告（Warnings）均无误后，出现提示信息：0 Error（s）, 0 Warning（s），项目生成结果文件\*.hex（注意：点击“Project”中“Options for Target”，出现窗口点击“output”，勾选“Create HEX File”）。

7、下载程序结果，运用STC-ISP软件独立下载\*.hex结果文件到单片机教学系统。

8、运行程序结果，在教学系统对应的显示模块中出现相应的现象。

**五、写出实验源程序**

**实验（一）**

//.h文件

#ifndef lcd12864H

#define lcd12864H

sbit RS = P0 ^7;

sbit RW = P0 ^6;

sbit E = P0 ^5;

sbit PSB = P0 ^4;

sbit RET = P0 ^3;

#define LcdData P2

unsigned char Check\_Busy(void);

void Lcd\_WriteData(unsigned char);

unsigned char Lcd\_ReadData(void);

void Lcd\_WriteCmd(unsigned char);

void Lcd\_PutPixel(unsigned char, unsigned char, unsigned char);

unsigned char Lcd\_ReadPixel(unsigned char, unsigned char);

void Lcd\_HoriLine(unsigned char, unsigned char, unsigned char Length, unsigned char Color);

void Lcd\_VertLine(unsigned char x, unsigned char y, unsigned char Length, unsigned char Color);

void Lcd\_Line(unsigned char x1, unsigned char y1, unsigned char x2, unsigned char y2, unsigned char Color);

void Lcd\_Rectangle(unsigned char x0, unsigned char y0, unsigned char x1, unsigned char y1, unsigned char Color);

void Lcd\_Circle(unsigned char x, unsigned char y, unsigned char r, unsigned char Color);

void Lcd\_Clear(unsigned char);

void Lcd\_WriteStr(unsigned char, unsigned char, unsigned char \*);

void Lcd\_Reset(void);

#endif

//.h文件

#ifndef KEYH

#define KEYH

unsigned char OSScanKey(void);

unsigned char OSReadKey(void);

#endif

//.c文件

#include "Reg52.h"

#include "intrins.h"

#include "Lcd12864.h"

unsigned char Lcd\_CheckBusy(void) {

unsigned char Busy;

LcdData = 0xff;

RS = 0;

RW = 1;

E = 1;

\_nop\_();

Busy = LcdData & 0x80;

E = 0;

return Busy;

}

void Lcd\_WriteData(unsigned char Data) {

while (Lcd\_CheckBusy());

RS = 1;

RW = 0;

E = 0;

nop();

nop();

LcdData = Data;

E = 1;

\_nop\_();

\_nop\_();

E = 0;

}

unsigned char Lcd\_ReadData(void) {

unsigned char Temp;

while (Lcd\_CheckBusy());

LcdData = 0xff;

RS = 1;

RW = 1;

E = 1;

nop();

Temp = LcdData;

E = 0;

return Temp;

}

void Lcd\_WriteCmd(unsigned char CmdCode) {

while (Lcd\_CheckBusy());

RS = 0;

RW = 0;

E = 0;

\_nop\_();

\_nop\_();

LcdData = CmdCode;

\_nop\_();

\_nop\_();

E = 1;

\_nop\_();

\_nop\_();

E = 0;

}

void Lcd\_WriteStr(unsigned char x, unsigned char y, unsigned char \*Str) {

if ((y > 3) || (x > 7))

return;

EA = 0;

switch (y) {

case 0:

Lcd\_WriteCmd(0x80 + x);

break;

case 1:

Lcd\_WriteCmd(0x90 + x);

break;

case 2:

Lcd\_WriteCmd(0x88 + x);

break;

case 3:

Lcd\_WriteCmd(0x98 + x);

break;

}

while (\*Str > 0) {

Lcd\_WriteData(\*Str);

Str++;

}

EA = 1;

}

code unsigned int LcdMaskTab[] =

{0x0001, 0x0002, 0x0004, 0x0008, 0x0010, 0x0020, 0x0040, 0x0080, 0x0100, 0x0200, 0x0400, 0x0800, 0x1000, 0x2000,

0x4000, 0x8000};

void Lcd\_PutPixel(unsigned char x, unsigned char y, unsigned char Color) {

unsigned char z, w;

unsigned int Temp;

if (x >= 128 || y >= 64)

return;

Color = Color % 2;

w = 15 - x % 16;

x = x / 16;

if (y < 32)

z = 0x80;

else

z = 0x88;

y = y % 32;

EA = 0;

Lcd\_WriteCmd(0x36);

Lcd\_WriteCmd(y + 0x80);

Lcd\_WriteCmd(x + z);

Temp = Lcd\_ReadData();

Temp = (unsigned int) Lcd\_ReadData() << 8;

Temp |= (unsigned int) Lcd\_ReadData();

EA = 1;

if (Color == 1)

Temp |= LcdMaskTab[w];

Else

Temp &= ~LcdMaskTab[w];

EA = 0;

Lcd\_WriteCmd(y + 0x80);

Lcd\_WriteCmd(x + z);

Lcd\_WriteData(Temp >> 8);

Lcd\_WriteData(Temp & 0x00ff);

Lcd\_WriteCmd(0x30);

EA = 1;

}

unsigned char Lcd\_ReadPixel(unsigned char x, unsigned char y) {

unsigned char z, w;

unsigned int Temp;

if (x >= 128 || y >= 64)

return 0;

w = 15 - x % 16;

x = x / 16;

if (y < 32)

z = 0x80;

else

z = 0x88;

y = y % 32;

EA = 0;

Lcd\_WriteCmd(0x36);

Lcd\_WriteCmd(y + 0x80);

Lcd\_WriteCmd(x + z);

Temp = Lcd\_ReadData();

Temp = (unsigned int) Lcd\_ReadData() << 8;

Temp |= (unsigned int) Lcd\_ReadData();

EA = 1;

if ((Temp && LcdMaskTab[w]) == 0)

return 0;

else

return 1;

}

void Lcd\_HoriLine(unsigned char x, unsigned char y, unsigned char Length, unsigned char Color) {

unsigned char i;

if (Length == 0)

return;

for (i = 0; i < Length; i++) {

Lcd\_PutPixel(x + i, y, Color);

}

}

void Lcd\_VertLine(unsigned char x, unsigned char y, unsigned char Length, unsigned char Color) {

unsigned char i;

if (Length == 0)

return;

for (i = 0; i < Length; i++) {

Lcd\_PutPixel(x, y + i, Color);

}

}

void Lcd\_Line(unsigned char x1, unsigned char y1, unsigned char x2, unsigned char y2, unsigned char Color) {

unsigned int x, y;

unsigned int d\_x, d\_y;

int err = 0;

unsigned char temp = 0;

if (y2 < y1) {

x = x1;

y = y1;

x1 = x2;

y1 = y2;

x2 = x;

y2 = y;

}

d\_y = y2 - y1;

if (d\_y == 0) {

if (x1 > x2) {

x = x1;

x1 = x2;

x2 = x;

}

for (x = x1; x <= x2; x++)

Lcd\_PutPixel(x, y1, Color);

} else {

if (x2 >= x1) {

temp = 1;

d\_x = x2 - x1;

} else

d\_x = x1 - x2;

x = x1;

y = y1;

Lcd\_PutPixel(x, y, 1);

if (temp && (d\_y <= d\_x))

while (x != x2) {

if (err < 0) {

x = x + 1;

err = err + (y2 - y);

} else {

x = x + 1;

y = y + 1;

err = err + (y2 - y) - (x2 - x);

}

Lcd\_PutPixel(x, y, Color);

}

else if (temp && (d\_y > d\_x))

while (y != y2) {

d\_x = x2 - x;

d\_y = y2 - y;

if (err < 0) {

x = x + 1;

y = y + 1;

err = err + d\_y - d\_x;

} else {

y = y + 1;

err = err - d\_x;

}

Lcd\_PutPixel(x, y, Color);

}

else if (!temp && (d\_y <= d\_x))

while (x != x2) {

d\_x = x - x2;

d\_y = y2 - y;

if (err < 0) {

x = x - 1;

err = err + d\_y;

} else {

x = x - 1;

y = y + 1;

err = err + d\_y - d\_x;

}

Lcd\_PutPixel(x, y, Color);

}

else if (!temp && (d\_y > d\_x))

while (y != y2) {

d\_x = x - x2;

d\_y = y2 - y;

if (err < 0) {

x = x - 1;

y = y + 1;

err = err + d\_y - d\_x;

} else {

y = y + 1;

err = err - d\_x;

}

Lcd\_PutPixel(x, y, Color);

}

}

}

void Lcd\_Rectangle(unsigned char x0, unsigned char y0, unsigned char x1, unsigned char y1, unsigned char Color) {

unsigned char Temp;

if (x0 > x1) {

Temp = x0;

x0 = x1;

x1 = Temp;

}

if (y0 > y1) {

Temp = y0;

y0 = y1;

y1 = Temp;

}

Lcd\_VertLine(x0, y0, y1 - y0 + 1, Color);

Lcd\_VertLine(x1, y0, y1 - y0 + 1, Color);

Lcd\_HoriLine(x0, y0, x1 - x0 + 1, Color);

Lcd\_HoriLine(x0, y1, x1 - x0 + 1, Color);

}

void CircleDot(unsigned char x, unsigned char y, char xx, char yy, unsigned char Color) {

Lcd\_PutPixel((x + yy), (y + xx), Color);

Lcd\_PutPixel((x + xx), (y + yy), Color);

Lcd\_PutPixel((x - xx), (y + yy), Color);

Lcd\_PutPixel((x - yy), (y + xx), Color);

Lcd\_PutPixel((x - yy), (y - xx), Color);

Lcd\_PutPixel((x - xx), (y - yy), Color);

Lcd\_PutPixel((x + xx), (y - yy), Color);

Lcd\_PutPixel((x + yy), (y - xx), Color);

}

void Lcd\_Circle(unsigned char x, unsigned char y, unsigned char r, unsigned char Color) {

unsigned char xx, yy;

char deltax, deltay, d;

xx = 0;

yy = r;

deltax = 3;

deltay = 2 - r - r;

d = 1 - r;

CircleDot(x, y, xx, yy, Color);

while (xx < yy) {

if (d < 0) {

d += deltax;

deltax += 2;

xx++;

} else {

d += deltax + deltay;

deltax += 2;

deltay += 2;

xx++;

yy--;

}

CircleDot(x, y, xx, yy, Color);

}

}

void Lcd\_Clear(unsigned char Mode) {

unsigned char x, y, ii;

unsigned char Temp;

if (Mode % 2 == 0)

Temp = 0x00;

else

Temp = 0xff;

Lcd\_WriteCmd(0x36);

for (ii = 0; ii < 9; ii += 8)

for (y = 0; y < 0x20; y++)

for (x = 0; x < 8; x++) {

EA = 0;

Lcd\_WriteCmd(y + 0x80);

Lcd\_WriteCmd(x + 0x80 + ii);

Lcd\_WriteData(Temp);

Lcd\_WriteData(Temp);

EA = 1;

}

Lcd\_WriteCmd(0x30);

}

void Lcd\_Reset() {

PSB = 1;

Lcd\_WriteCmd(0x30);

Lcd\_WriteCmd(0x0c);

Lcd\_WriteCmd(0x01);

Lcd\_WriteCmd(0x06);

}

//.c文件

#include "REG52.H"

#include "Key.h"

#define OS\_LONG\_KEY\_EN 1

#define KEY P1

void delays(unsigned int i) {

unsigned int j;

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

for (j = 300; j > 0; j--);

}

unsigned char OSScanKey(void) {

unsigned char Temp;

unsigned char i, key;

KEY = 0xF0;

delays(1);

Temp = KEY;

Temp = Temp & 0xF0;

Temp = ~((Temp >> 4) | 0xF0);

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

if ((Temp & (1 << i)) != 0)

break;

}

if (i < 4) {

key = i;

} else

return 0;

KEY = 0x0F;

delays(1);

Temp = KEY;

Temp = Temp & 0x0F;

Temp = ~(Temp | 0xF0);

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

if ((Temp & (1 << i)) != 0)

break;

}

if (i < 4) {

key = key + i \* 4;

return key + 1;

} else

return 0;

}

unsigned char OSReadKey(void) {

static unsigned char KeyEventCnt = 0;

static unsigned char KeySampleCnt = 0;

static unsigned char KeyBuffer = 0;

#define SHORT\_ON\_DITHERING\_COUNTER 3

#define SHORT\_OFF\_DITHERING\_COUNTER 3

#if OS\_LONG\_KEY\_EN > 0

static unsigned int LongKeySampleCnt = 0;

#define LONG\_ON\_DITHERING\_COUNTER 250

#define LONG\_OFF\_DITHERING\_COUNTER 3

#endif

unsigned char KeyTemp;

KeyTemp = OSScanKey();

switch (KeyEventCnt) {

case 0:

if (KeyTemp != 0) {

KeySampleCnt = 0;

KeyBuffer = KeyTemp;

KeyEventCnt = 1;

}

return 0;

break;

#if OS\_LONG\_KEY\_EN > 0

case 1:

if (KeyTemp != KeyBuffer) {

KeyEventCnt = 0;

return 0;//is dithering,return 0

} else {

if (++KeySampleCnt > SHORT\_ON\_DITHERING\_COUNTER) {

KeySampleCnt = 0;

KeyEventCnt = 2;

LongKeySampleCnt = 0;

return ((KeyBuffer - 1) << 2) + 1;//sure that key on,return (KeyBuffer-1)<<2+1

} else

return 0;//not sure that key on,return 0

}

break;

case 2:

if (++LongKeySampleCnt > LONG\_ON\_DITHERING\_COUNTER) {

KeySampleCnt = 0;

KeyEventCnt = 3;

return ((KeyBuffer - 1) << 2) + 2; //sure that key long on,return (KeyBuffer-1)<<2+2

} else {

if (KeyTemp != KeyBuffer) {

if (++KeySampleCnt > SHORT\_OFF\_DITHERING\_COUNTER) {

KeyEventCnt = 0;

return ((KeyBuffer - 1) << 2) + 3;

} else

return 0;

} else {

KeySampleCnt = 0;

return 0;

}

}

break;

case 3:

if (KeyTemp != KeyBuffer) {

if (++KeySampleCnt > LONG\_OFF\_DITHERING\_COUNTER) {

KeyEventCnt = 0;

return ((KeyBuffer - 1) << 2) + 4;

} else

return 0;

} else {

KeySampleCnt = 0;

return 0;

}

break;

else{

case 1:

if (KeyTemp != KeyBuffer) {

KeyEventCnt = 0;

return 0;

} else {

if (++KeySampleCnt >= SHORT\_ON\_DITHERING\_COUNTER) {

KeySampleCnt = 0;

KeyEventCnt = 2;

return ((KeyBuffer - 1) << 2) + 1;

} else

return 0;//not sure that key on,return 0

}

break;

case 2:

if (KeyTemp != KeyBuffer) {

if (++KeySampleCnt >= SHORT\_OFF\_DITHERING\_COUNTER) {

KeyEventCnt = 0;

return ((KeyBuffer - 1) << 2) + 3;

} else

return 0;

} else {

KeySampleCnt = 0;

return 0;

}

break;

#endif

default:

break;

}

return 0;

}

}

//.c文件

#include "reg52.h"

#include "Lcd12864.h"

#include "Key.h"

#define uchar unsigned char

#define uint unsigned int

static unsigned long Seed = 1;

#define A 48271L

#define M 2147483647L

#define Q (M / A)

#define R (M % A)

double Random(void) {

long TmpSeed;

TmpSeed = A \* (Seed % Q) - R \* (Seed / Q);

if (TmpSeed >= 0)

Seed = TmpSeed;

else

Seed = TmpSeed + M;

return (double) Seed / M;

}

void InitRandom(unsigned long InitVal) {

Seed = InitVal;

}

void Delay(unsigned int t) {

unsigned int i, j;

for (i = 0; i < t; i++)

for (j = 0; j < 10; j++);

}

#define LGOU 0

#define RGOU 1

#define RZ 2

#define LZ 3

#define T 4

#define GUN 5

#define BOX 6

unsigned int idata

num[19+2]={0xfff,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0xfff};

unsigned char code

Block[28][2]={{0x88,0xc0},{0xe8,0x00},{0x62,0x20},{0x02,0xe0},{0x22,0x60},{0x08,0xe0},{0xc8,0x80},{0xe2,0x00},{0x8c,0x40},{0x6c,0x00},{0x8c,0x40},{0x6c,0x00},{0x4c,0x80},{0xc6,0x00},{0x4c,0x80},{0xc6,0x00},{0x04,0xe0},{0x8c,0x80},{0xe4,0x00},{0x26,0x20},{0x44,0x44},{0x0f,0x00},{0x44,0x44},{0x0f,0x00},{0x06,0x60},{0x06,0x60},{0x06,0x60},{0x06,0x60}};

struct Jimu {

unsigned int dat;

char x;

unsigned char y;

unsigned char type;

unsigned char change;

} Sign[3];

unsigned char SysFlag = 0;

#define NEWSIGNFLAG 0

#define DEADFLAG 1

#define PAUSEFLAG 2

unsigned char Score = 0;

unsigned char Level = 0;

unsigned char DelayCnt = 5;

unsigned char KeyBuffer = 0;

#define RESEVER 1

#define CHANGE 2

#define DOWN 3

#define LEFT 4

#define RIGHT 5

#define PAUSE 6

void InitCpu(void) {

TMOD = 0x0;

TH0 = 0;

TL0 = 0;

TR0 = 1;

ET0 = 1;

EA = 1;

}

void Timer0Int(void) interrupt 1 {

switch (OSReadKey()) {

case 9:

KeyBuffer = PAUSE;break;

case 13:

KeyBuffer = CHANGE;break;

case 17:

KeyBuffer = DOWN;break;

case 21:

KeyBuffer = RIGHT;break;

case 25:

KeyBuffer = LEFT;break;

default:break;

}

}

void DrawBoard(void) {

unsigned char n;

for (n = 0; n < 12; n++) {

Lcd\_Rectangle(3 \* n, 0, 3 \* n + 2, 2, 1);

Lcd\_Rectangle(3 \* n, 60, 3 \* n + 2, 62, 1);

}

for (n = 0; n < 20; n++) {

Lcd\_Rectangle(0, 3 \* n, 2, 3 \* n + 2, 1);

Lcd\_Rectangle(33, 3 \* n, 35, 3 \* n + 2, 1);

}

Lcd\_Rectangle(48, 0, 48 + 17, 0 + 17, 1);

Lcd\_WriteStr(3, 2, "Score:");

Lcd\_WriteStr(3, 3, "Level:");

}

void GameOver(void) {

if ((SysFlag & (1 << DEADFLAG)) != 0)

Lcd\_WriteStr(3, 1, "You Fail");

else

Lcd\_WriteStr(3, 1, "You Win");

}

unsigned int

codeMaskTab[16]={0x0001,0x0002,0x0004,0x0008,0x0010,0x0020,0x0040,0x0080,0x0100,0x0200,0x0400,0x0800,0x1000,0x2000,0x4000,0x8000};

void ClearSign(void) {

unsigned char m, n;

for (m = 0; m < 4; m++)

for (n = 0; n < 4; n++) {

if ((Sign[0].dat & MaskTab[4 \* m + n]) != 0)

Lcd\_Rectangle(Sign[0].x + n \* 3, Sign[0].y - 2 - 3 \* m, Sign[0].x + n \* 3 + 2, Sign[0].y - 3 \* m, 0);

}

}

void DrawSign(void) {

unsigned char m, n;

for (m = 0; m < 4; m++)

for (n = 0; n < 4; n++) {

if ((Sign[0].dat & MaskTab[4 \* m + n]) != 0)

Lcd\_Rectangle(Sign[0].x + n \* 3, Sign[0].y - 2 - 3 \* m, Sign[0].x + n \* 3 + 2, Sign[0].y - 3 \* m, 1);

}

}

FixSign(void) {

unsigned char m, n;

for (m = 0; m < 4; m++)

for (n = 0; n < 4; n++) {

if ((Sign[0].dat & MaskTab[4 \* m + n]) != 0) {

num[20 - (Sign[0].y - 2) / 3 + m] |= MaskTab[11 - Sign[0].x / 3 - n];

}

}

}

unsigned char CheckIfDown(void) {

unsigned char m, n;

Sign[1] = Sign[0];

Sign[1].y += 3;

for (m = 0; m < 4; m++)

for (n = 0; n < 4; n++) {

if ((Sign[1].dat & MaskTab[4 \* m + n]) != 0) {

if ((num[20 - (Sign[1].y - 2) / 3 + m] & MaskTab[11 - Sign[1].x / 3 - n]) != 0)

return 0;

}

}

return 1;

}

unsigned char CheckIfLeft(void) {

unsigned char m, n;

Sign[1] = Sign[0];

Sign[1].x -= 3;

for (m = 0; m < 4; m++)

for (n = 0; n < 4; n++) {

if ((Sign[1].dat & MaskTab[4 \* m + n]) != 0) {

if ((num[20 - (Sign[1].y - 2) / 3 + m] & MaskTab[11 - Sign[1].x / 3 - n]) != 0)

return 0;

}

}

return 1;

}

unsigned char CheckIfRight(void) {

unsigned char m, n;

Sign[1] = Sign[0];

Sign[1].x += 3;

for (m = 0; m < 4; m++)

for (n = 0; n < 4; n++) {

if ((Sign[1].dat & MaskTab[4 \* m + n]) != 0) {

if ((num[20 - (Sign[1].y - 2) / 3 + m] & MaskTab[11 - Sign[1].x / 3 - n]) != 0)

return 0;

}

}

return 1;

}

unsigned char CheckIfRoll(void) {

unsigned char m, n;

unsigned int Temp;

Sign[1] = Sign[0];

if (++Sign[1].change > 3)

Sign[1].change = 0;

m = Sign[1].type \* 4 + Sign[1].change;

Temp = (unsigned int) Block[m][0] << 8;

Temp = Temp | Block[m][1];

Sign[1].dat = Temp;

for (m = 0; m < 4; m++)

for (n = 0; n < 4; n++) {

if ((Sign[1].dat & MaskTab[4 \* m + n]) != 0) {

if ((num[20 - (Sign[1].y - 2) / 3 + m] & MaskTab[11 - Sign[1].x / 3 - n]) != 0)

return 0;

}

}

return 1;

}

void DelFull(void) {

unsigned char m, n;

unsigned char Temp;

unsigned char Flag = 0;

Temp = (Sign[0].y - 2) / 3;

if (Temp >= 20)

Temp = 1;

else

Temp = 20 - Temp;

for (n = Temp + 3; n >= Temp; n--) {

if (num[n] == 0xfff) {

Flag = 1;

for (m = n + 1; m <= 19; m++) {

num[m - 1] = num[m];

}

num[m] = 0x801;

Score++;

}

}

if (Flag) {

for (m = Temp; m <= 19; m++)

for (n = 1; n <= 10; n++) {

if ((num[m] & MaskTab[n]) == 0) {

if (Lcd\_ReadPixel(30 - (n - 1) \* 3, 57 - (m - 1) \* 3) != 0) {

Lcd\_Rectangle(30 - (n - 1) \* 3, 57 - (m - 1) \* 3, 30 - (n - 1) \* 3 + 2, 57 - (m - 1) \* 3 + 2,

0);

}

} else {

if (Lcd\_ReadPixel(30 - (n - 1) \* 3, 57 - (m - 1) \* 3) == 0) {

Lcd\_Rectangle(30 - (n - 1) \* 3, 57 - (m - 1) \* 3, 30 - (n - 1) \* 3 + 2, 57 - (m - 1) \* 3 + 2,

1);

}

}

}

}

}

void CreatSign(void) {

unsigned char m, n;

unsigned int Temp;

for (m = 0; m < 4; m++)

for (n = 0; n < 4; n++) {

if ((Sign[2].dat & MaskTab[4 \* m + n]) != 0)

Lcd\_Rectangle(Sign[2].x + n \* 3, Sign[2].y - 2 - 3 \* m, Sign[2].x + n \* 3 + 2, Sign[2].y - 3 \* m, 0);

}

n = Random() \* 28;

Temp = (unsigned int) Block[n][0] << 8;

Temp = Temp | Block[n][1];

Sign[2].dat = Temp;

Sign[2].x = 51;

Sign[2].y = 4 \* 3 + 2;

Sign[2].type = n / 4;

Sign[2].change = n % 4;

for (m = 0; m < 4; m++)

for (n = 0; n < 4; n++) {

if ((Sign[2].dat & MaskTab[4 \* m + n]) != 0)

Lcd\_Rectangle(Sign[2].x + n \* 3, Sign[2].y - 2 - 3 \* m, Sign[2].x + n \* 3 + 2, Sign[2].y - 3 \* m, 1);

}

}

void PrintScore(void) {

unsigned char Str[3];

Str[0] = (Score / 10) | 0x30;

Str[1] = (Score % 10) | 0x30;

Str[2] = 0;

Lcd\_WriteStr(6, 2, Str);

}

void PrintLevel(void) {

unsigned char Str[3];

Str[0] = (Score / 10) | 0x30;

Str[1] = (Score % 10) | 0x30;

Str[2] = 0;

Lcd\_WriteStr(6, 3, Str);

}

void GamePlay(void) {

unsigned char i, m, n;

unsigned int Temp;

SysFlag |= 1 << NEWSIGNFLAG;

InitRandom(TL0);

Lcd\_WriteStr(3, 1, "Playing");

PrintScore();

PrintLevel();

CreatSign();

while (1) {

if ((SysFlag & (1 << NEWSIGNFLAG)) == 1) {

SysFlag &= ~(1 << NEWSIGNFLAG);

Sign[0] = Sign[2];

CreatSign();

Sign[0].x = 12;

Sign[0].y = 14;

for (m = 0; m < 4; m++) {

for (n = 0; n < 4; n++) {

if ((Sign[0].dat & MaskTab[15 - m \* 4 - n]) == 0)break;

}

if (n == 4)

Sign[0].y -= 3;

}

for (m = 0; m < 4; m++)

for (n = 0; n < 4; n++) {

if ((Sign[0].dat & MaskTab[4 \* m + n]) != 0) {

if ((num[20 - (Sign[0].y - 2) / 3 + m] & MaskTab[11 - Sign[0].x / 3 - n]) != 0)

SysFlag |= 1 << DEADFLAG;

}

}

if ((SysFlag & (1 << DEADFLAG)) != 0)break;

DrawSign();

}

switch (KeyBuffer) {

case LEFT:

KeyBuffer = 0;

if ((SysFlag & (1 << PAUSEFLAG)) == 0) {

if (CheckIfLeft()) {

ClearSign();

Sign[0].x -= 3;

DrawSign();

}

}break;

case RIGHT:

KeyBuffer = 0;

if ((SysFlag & (1 << PAUSEFLAG)) == 0) {

if (CheckIfRight()) {

ClearSign();

Sign[0].x += 3;

DrawSign();

}

}break;

case DOWN:

KeyBuffer = 0;

if ((SysFlag & (1 << PAUSEFLAG)) == 0) {

if (CheckIfDown()) {

ClearSign();

Sign[0].y += 3;

DrawSign();

}

}break;

case CHANGE:

KeyBuffer = 0;

if ((SysFlag & (1 << PAUSEFLAG)) == 0) {

if (CheckIfRoll()) {

ClearSign();

if (++Sign[0].change > 3)

Sign[0].change = 0;

i = Sign[0].type \* 4 + Sign[0].change;

Temp = (unsigned int) Block[i][0] << 8;

Temp = Temp | Block[i][1];

Sign[0].dat = Temp;

DrawSign();

}

}break;

case PAUSE:

KeyBuffer = 0;

SysFlag ^= 1 << PAUSEFLAG;

if ((SysFlag & (1 << PAUSEFLAG)) == 0) {

Lcd\_WriteStr(3, 1, " ");

Lcd\_WriteStr(3, 1, "Playing");

} else {

Lcd\_WriteStr(3, 1, " ");

Lcd\_WriteStr(3, 1, "Pause");

}break;

default:break;

}

if ((SysFlag & (1 << PAUSEFLAG)) != 0)continue;

Delay(500);

if (++DelayCnt >= 10) {

DelayCnt = 0;

if (CheckIfDown()) {

ClearSign();

Sign[0].y += 3;

DrawSign();

} else {

FixSign();

DelFull();

PrintScore();

if (Score >= 10) {

SysFlag &= ~(1 << DEADFLAG);break;

}

SysFlag |= 1 << NEWSIGNFLAG;

}

}

}

}

void Main() {

InitCpu();

Lcd\_Reset();

Lcd\_Clear(0);

DrawBoard();

GamePlay();

GameOver();

while (1);

}

//.c主函数

#include "reg52.h"

#include "Lcd12864.h"

#include "Key.h"

#define uchar unsigned char

#define uint unsigned int

static unsigned long Seed = 1;

#define A 48271L

#define M 2147483647L

#define Q (M / A)

#define R (M % A)

double Random(void) {

long TmpSeed;

TmpSeed = A \* (Seed % Q) - R \* (Seed / Q);

if (TmpSeed >= 0)

Seed = TmpSeed;

else

Seed = TmpSeed + M;

return (double) Seed / M;

}

void InitRandom(unsigned long InitVal) {

Seed = InitVal;

}

void Delay(unsigned int t) {

unsigned int i, j;

for (i = 0; i < t; i++)

for (j = 0; j < 10; j++);

}

#define LGOU 0

#define RGOU 1

#define RZ 2

#define LZ 3

#define T 4

#define GUN 5

#define BOX 6

unsigned int idata num[19+2] = {0xfff,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801,0x801, 0xfff };

unsigned char code Block[28][2]={{0x88,0xc0},{0xe8,0x00},{0x62,0x20},{0x02,0xe0},{0x22,0x60},{0x08,0xe0},{0xc8,0x80},{0xe2,0x00},{0x8c,0x40},{0x6c,0x00},{0x8c,0x40},{0x6c,0x00},{0x4c,0x80},{0xc6,0x00},{0x4c,0x80},{0xc6,0x00},{0x04,0xe0},{0x8c,0x80},{0xe4,0x00},{0x26,0x20},{0x44,0x44},{0x0f,0x00},{0x44,0x44},{0x0f,0x00},{0x06,0x60},{0x06,0x60},{0x06,0x60},{0x06,0x60}};

#define PASSSCORE 20

struct Jimu {

unsigned int dat;

char x;

unsigned char y;

unsigned char type;

unsigned char change;

} Sign[3];

unsigned char SysFlag = 0;

#define NEWSIGNFLAG 0

#define DEADFLAG 1

#define PAUSEFLAG 2

unsigned char Score = 0;

unsigned char Level = 1;

unsigned char DelayCnt = 5;

unsigned char KeyBuffer = 0;

#define RESEVER 1

#define CHANGE 2

#define DOWN 3

#define LEFT 4

#define RIGHT 5

#define PAUSE 6

void InitCpu(void) {

TMOD = 0x0;

TH0 = 0;

TL0 = 0;

TR0 = 1;

ET0 = 1;

EA = 1;

}

void Timer0Int(void) interrupt 1 {

switch(OSReadKey())

{

case 9:

KeyBuffer = PAUSE;break;

case 13:

KeyBuffer = CHANGE;break;

case 17:

KeyBuffer = DOWN;break;

case 21:

KeyBuffer = RIGHT;break;

case 25:

KeyBuffer = LEFT;break;

default:break;

}

}

void DrawBoard(void) {

unsigned char n;

for (n = 0; n < 12; n++) {

Lcd\_Rectangle(3 \* n, 0, 3 \* n + 2, 2, 1);

Lcd\_Rectangle(3 \* n, 60, 3 \* n + 2, 62, 1);

}

for (n = 0; n < 20; n++) {

Lcd\_Rectangle(0, 3 \* n, 2, 3 \* n + 2, 1);

Lcd\_Rectangle(33, 3 \* n, 35, 3 \* n + 2, 1);

}

//Lcd\_WriteStr(4,0,"YOLO");

Lcd\_WriteStr(3, 2, "Score:");

Lcd\_WriteStr(3, 3, "Level:");

}

void GameOver(void) {

if ((SysFlag & (1 << DEADFLAG)) != 0)

Lcd\_WriteStr(3, 1, "You Fail");

else

Lcd\_WriteStr(3, 1, "You Pass");

}

unsigned int code

MaskTab[16]={0x0001,0x0002,0x0004,0x0008,0x0010,0x0020,0x0040,0x0080,0x0100,0x0200,0x0400,0x0800,0x1000,0x2000,0x4000,0x8000};

void DrawSign(struct Jimu Temp, unsigned char DrawMode) {

unsigned char m, n;

for (m = 0; m < 4; m++)

for (n = 0; n < 4; n++) {

if ((Temp.dat & MaskTab[4 \* m + n]) != 0)

Lcd\_Rectangle(Temp.x + n \* 3, Temp.y - 2 - 3 \* m, Temp.x + n \* 3 + 2, Temp.y - 3 \* m, DrawMode);

}

}

FixSign(void) {

unsigned char m, n;

for (m = 0; m < 4; m++)

for (n = 0; n < 4; n++) {

if ((Sign[0].dat & MaskTab[4 \* m + n]) != 0) {

num[20 - (Sign[0].y - 2) / 3 + m] |= MaskTab[11 - Sign[0].x / 3 - n];

}

}

}

unsigned char CheckIf(void) {

unsigned char m, n;

for (m = 0; m < 4; m++)

for (n = 0; n < 4; n++) {

if ((Sign[1].dat & MaskTab[4 \* m + n]) != 0) {

if ((num[20 - (Sign[1].y - 2) / 3 + m] & MaskTab[11 - Sign[1].x / 3 - n]) != 0)

return 0;

}

}

return 1;

}

unsigned char CheckIfDown(void) {

Sign[1] = Sign[0];

Sign[1].y += 3;

return CheckIf();

}

unsigned char CheckIfLeft(void) {

Sign[1] = Sign[0];

Sign[1].x -= 3;

return CheckIf();

}

unsigned char CheckIfRight(void) {

Sign[1] = Sign[0];

Sign[1].x += 3;

return CheckIf();

}

unsigned char CheckIfRoll(void) {

unsigned char i;

unsigned int Temp;

Sign[1] = Sign[0];

if (++Sign[1].change > 3)

Sign[1].change = 0;

i = Sign[1].type \* 4 + Sign[1].change;

Temp = (unsigned int) Block[i][0] << 8;

Temp = Temp | Block[i][1];

Sign[1].dat = Temp;

return CheckIf();

}

void DelFull(void) {

unsigned char m, n;

unsigned char Temp;

unsigned char Flag = 0;

Temp = (Sign[0].y - 2) / 3;

if (Temp >= 20)

Temp = 1;

else

Temp = 20 - Temp;

for (n = Temp + 3; n >= Temp; n--) {

if (num[n] == 0xfff) {

Flag = 1;

for (m = n + 1; m <= 19; m++) {

num[m - 1] = num[m];

}

num[m] = 0x801;

Score++;

}

}

if (Flag) {

for (m = Temp; m <= 19; m++)

for (n = 1; n <= 10; n++) {

if ((num[m] & MaskTab[n]) == 0) {

if (Lcd\_ReadPixel(30 - (n - 1) \* 3, 57 - (m - 1) \* 3) != 0) {

Lcd\_Rectangle(30 - (n - 1) \* 3, 57 - (m - 1) \* 3, 30 - (n - 1) \* 3 + 2, 57 - (m - 1) \* 3 + 2,

0);

}

} else {

if (Lcd\_ReadPixel(30 - (n - 1) \* 3, 57 - (m - 1) \* 3) == 0) {

Lcd\_Rectangle(30 - (n - 1) \* 3, 57 - (m - 1) \* 3, 30 - (n - 1) \* 3 + 2, 57 - (m - 1) \* 3 + 2,

1);

}

}

}

}

}

void CreatSign(void) {

unsigned char n;

unsigned int Temp;

DrawSign(Sign[2], 0);

n = Random() \* 28;

Temp = (unsigned int) Block[n][0] << 8;

Temp = Temp | Block[n][1];

Sign[2].dat = Temp;

Sign[2].x = 45;

Sign[2].y = 4 \* 3 + 2;

Sign[2].type = n / 4;

Sign[2].change = n % 4;

DrawSign(Sign[2], 1);

}

void PrintScore(void) {

unsigned char Str[3];

Str[0] = (Score / 10) | 0x30;

Str[1] = (Score % 10) | 0x30;

Str[2] = 0;

Lcd\_WriteStr(6, 2, Str);

}

void PrintLevel(void) {

unsigned char Str[3];

Str[0] = (Level / 10) | 0x30;

Str[1] = (Level % 10) | 0x30;

Str[2] = 0;

Lcd\_WriteStr(6, 3, Str);

}

void GamePlay(void) {

unsigned char m, n;

unsigned int Temp;

SysFlag |= 1 << NEWSIGNFLAG;

InitRandom(TL0);

Lcd\_WriteStr(3, 1, "Playing");

PrintScore();

PrintLevel();

CreatSign();

while (1) {

if ((SysFlag & (1 << NEWSIGNFLAG)) == 1) {

SysFlag &= ~(1 << NEWSIGNFLAG);

Sign[0] = Sign[2];

CreatSign();

Sign[0].x = 12;

Sign[0].y = 14;

for (m = 0; m < 4; m++) {

for (n = 0; n < 4; n++) {

if ((Sign[0].dat & MaskTab[15 - m \* 4 - n]) == 0)break;

}

if (n == 4)

Sign[0].y -= 3;

}

for (m = 0; m < 4; m++)

for (n = 0; n < 4; n++) {

if ((Sign[0].dat & MaskTab[4 \* m + n]) != 0) {

if ((num[20 - (Sign[0].y - 2) / 3 + m] & MaskTab[11 - Sign[0].x / 3 - n]) != 0)

SysFlag |= 1 << DEADFLAG;

}

}

if ((SysFlag & (1 << DEADFLAG)) != 0)break;

DrawSign(Sign[0], 1);

}

switch (KeyBuffer) {

case LEFT:

KeyBuffer = 0;

if ((SysFlag & (1 << PAUSEFLAG)) == 0) {

if (CheckIfLeft()) {

DrawSign(Sign[0], 0);

Sign[0].x -= 3;

DrawSign(Sign[0], 1);

}

} else {

if (++Level >= 10)

Level = 1;

PrintLevel();

}break;

case RIGHT:

KeyBuffer = 0;

if ((SysFlag & (1 << PAUSEFLAG)) == 0) {

if (CheckIfRight()) {

DrawSign(Sign[0], 0);

Sign[0].x += 3;

DrawSign(Sign[0], 1);

}

} else {

if (++Level >= 10)

Level = 1;

PrintLevel();

}break;

case DOWN:

KeyBuffer = 0;

if ((SysFlag & (1 << PAUSEFLAG)) == 0) {

if (CheckIfDown()) {

DrawSign(Sign[0], 0);

Sign[0].y += 3;

DrawSign(Sign[0], 1);

}

}break;

case CHANGE:

KeyBuffer = 0;

if ((SysFlag & (1 << PAUSEFLAG)) == 0) {

if (CheckIfRoll()) {

DrawSign(Sign[0], 0);

if (++Sign[0].change > 3)

Sign[0].change = 0;

m = Sign[0].type \* 4 + Sign[0].change;

Temp = (unsigned int) Block[m][0] << 8;

Temp = Temp | Block[m][1];

Sign[0].dat = Temp;

DrawSign(Sign[0], 1);

}

}break;

case PAUSE:

KeyBuffer = 0;

SysFlag ^= 1 << PAUSEFLAG;

if ((SysFlag & (1 << PAUSEFLAG)) == 0) {

Lcd\_WriteStr(3, 1, " ");

Lcd\_WriteStr(3, 1, "Playing");

} else {

Lcd\_WriteStr(3, 1, " ");

Lcd\_WriteStr(3, 1, "Pause");

}

break;

default:break;

}

if ((SysFlag & (1 << PAUSEFLAG)) != 0)continue;

Delay(500);

if (++DelayCnt >= 2 \* (11 - Level)) {

DelayCnt = 0;

if (CheckIfDown()) {

DrawSign(Sign[0], 0);

Sign[0].y += 3;

DrawSign(Sign[0], 1);

} else {

FixSign();

DelFull();

PrintScore();

if (Score >= PASSSCORE) {

SysFlag &= ~(1 << DEADFLAG);break;

}

SysFlag |= 1 << NEWSIGNFLAG;

}

}

}

}

void Main() {

InitCpu();

Lcd\_Reset();

Lcd\_Clear(0);

DrawBoard();

GamePlay();

GameOver();

while (1);

}

**实验（二）**

//.h文件

#ifndef \_\_KEY\_H\_\_

#define \_\_KEY\_H\_\_

unsigned char OSScanKey(void);

unsigned char OSReadKey(void);

#endif

//.h文件

#ifndef \_\_lcd12864\_H\_\_

#define \_\_lcd12864\_H\_\_

sbit RS = P0 ^7;

sbit RW = P0 ^6;

sbit E = P0 ^5;

sbit PSB = P0 ^4;

sbit RET = P0 ^3;

#define LcdData P2

unsigned char Check\_Busy(void);

void Lcd\_WriteData(unsigned char);

unsigned char Lcd\_ReadData(void);

void Lcd\_WriteCmd(unsigned char);

void Lcd\_PutPixel(unsigned char, unsigned char, unsigned char);

unsigned char Lcd\_ReadPixel(unsigned char, unsigned char);

void Lcd\_HoriLine(unsigned char, unsigned char, unsigned char Length, unsigned char Color);

void Lcd\_VertLine(unsigned char x, unsigned char y, unsigned char Length, unsigned char Color);

void Lcd\_Line(unsigned char x1, unsigned char y1, unsigned char x2, unsigned char y2, unsigned char Color);

void Lcd\_Rectangle(unsigned char x0, unsigned char y0, unsigned char x1, unsigned char y1, unsigned char Color);

void Lcd\_Circle(unsigned char x, unsigned char y, unsigned char r, unsigned char Color);

void Lcd\_Clear(unsigned char);

void Lcd\_WriteStr(unsigned char, unsigned char, unsigned char \*);

void Lcd\_Reset(void);

#endif

//.c文件

#include "Reg52.h"

#include "intrins.h"

#include "Lcd12864.h"

signed char Lcd\_CheckBusy(void) {

unsigned char Busy;

LcdData = 0xff;

RS = 0;

RW = 1;

E = 1;

\_nop\_();

Busy = LcdData & 0x80;

E = 0;

return Busy;

}

void Lcd\_WriteData(unsigned char Data) {

while (Lcd\_CheckBusy());

RS = 1;

RW = 0;

E = 0;

\_nop\_();

\_nop\_();

LcdData = Data;

E = 1;

\_nop\_();

\_nop\_();

E = 0;

}

unsigned char Lcd\_ReadData(void) {

unsigned char Temp;

while (Lcd\_CheckBusy());

LcdData = 0xff;

RS = 1;

RW = 1;

E = 1;

\_nop\_();

Temp = LcdData;

E = 0;

return Temp;

}

void Lcd\_WriteCmd(unsigned char CmdCode) {

while (Lcd\_CheckBusy());

RS = 0;

RW = 0;

E = 0;

\_nop\_();

\_nop\_();

LcdData = CmdCode;

\_nop\_();

\_nop\_();

E = 1;

\_nop\_();

\_nop\_();

E = 0;

}

void Lcd\_WriteStr(unsigned char x, unsigned char y, unsigned char \*Str) {

if ((y > 3) || (x > 7))return;

EA = 0;

switch (y) {

case 0:

Lcd\_WriteCmd(0x80 + x);break;

case 1:

Lcd\_WriteCmd(0x90 + x);break;

case 2:

Lcd\_WriteCmd(0x88 + x);break;

case 3:

Lcd\_WriteCmd(0x98 + x);break;

}

while (\*Str > 0) {

Lcd\_WriteData(\*Str);

Str++;

}

EA = 1;

}

code unsigned int MaskTab[] = {0x0001, 0x0002, 0x0004, 0x0008, 0x0010, 0x0020, 0x0040, 0x0080,0x0100, 0x0200, 0x0400, 0x0800, 0x1000, 0x2000, 0x4000, 0x8000};

void Lcd\_PutPixel(unsigned char x, unsigned char y, unsigned char Color) {

unsigned char z, w;

unsigned int Temp;

if (x >= 128 || y >= 64)

return;

Color = Color % 2;

w = 15 - x % 16;

x = x / 16;

if (y < 32)

z = 0x80;

else

z = 0x88;

y = y % 32;

EA = 0;

Lcd\_WriteCmd(0x36);

Lcd\_WriteCmd(y + 0x80);

Lcd\_WriteCmd(x + z);

Temp = Lcd\_ReadData();

Temp = (unsigned int) Lcd\_ReadData() << 8;

Temp |= (unsigned int) Lcd\_ReadData();

EA = 1;

//如果写入颜色为1

if (Color == 1)

Temp |= MaskTab[w];

else

Temp &= ~MaskTab[w];

EA = 0;

Lcd\_WriteCmd(y + 0x80);

Lcd\_WriteCmd(x + z);

Lcd\_WriteData(Temp >> 8);

Lcd\_WriteData(Temp & 0x00ff);

Lcd\_WriteCmd(0x30);

EA = 1;

}

unsigned char Lcd\_ReadPixel(unsigned char x, unsigned char y) {

unsigned char z, w;

unsigned int Temp;

x = x % 128;

y = y % 64;

w = 15 - x % 16;

x = x / 16;

if (y < 32)

z = 0x80;

else

z = 0x88;

y = y % 32;

Lcd\_WriteCmd(0x36);

Lcd\_WriteCmd(y + 0x80);

Lcd\_WriteCmd(x + z);

Temp = Lcd\_ReadData();

Temp = (unsigned int) Lcd\_ReadData() << 8;

Temp |= (unsigned int) Lcd\_ReadData();

if ((Temp && MaskTab[w]) == 0)

return 0;

else

return 1;

}

void Lcd\_HoriLine(unsigned char x, unsigned char y, unsigned char Length, unsigned char Color) {

unsigned char i;

if (Length == 0)return;

for (i = 0; i < Length; i++) {

Lcd\_PutPixel(x + i, y, Color);

}

}

void Lcd\_VertLine(unsigned char x, unsigned char y, unsigned char Length, unsigned char Color) {

unsigned char i;

if (Length == 0)return;

for (i = 0; i < Length; i++) {

Lcd\_PutPixel(x, y + i, Color);

}

}

void Lcd\_Line(unsigned char x1, unsigned char y1, unsigned char x2, unsigned char y2, unsigned char Color) {

unsigned int x, y;

unsigned int d\_x, d\_y;//d\_x=x2-x1;d\_y=y2-y1;

int err = 0;

unsigned char temp = 0;

if (y2 < y1) {

x = x1;

y = y1;

x1 = x2;

y1 = y2;

x2 = x;

y2 = y;

}

d\_y = y2 - y1;

if (d\_y == 0) {

if (x1 > x2) {

x = x1;

x1 = x2;

x2 = x;

}

for (x = x1; x <= x2; x++)

Lcd\_PutPixel(x, y1, Color);

} else {

if (x2 >= x1) {

temp = 1;

d\_x = x2 - x1;

} else

d\_x = x1 - x2;

x = x1;

y = y1;

Lcd\_PutPixel(x, y, 1);

if (temp && (d\_y <= d\_x))

while (x != x2) {

if (err < 0) {

x = x + 1;

err = err + (y2 - y);

} else {

x = x + 1;

y = y + 1;

err = err + (y2 - y) - (x2 - x);

}

Lcd\_PutPixel(x, y, Color);

}

else if (temp && (d\_y > d\_x))

while (y != y2) {

d\_x = x2 - x;

d\_y = y2 - y;

if (err < 0) {

x = x + 1;

y = y + 1;

err = err + d\_y - d\_x;

} else {

y = y + 1;

err = err - d\_x;

}

Lcd\_PutPixel(x, y, Color);

}

else if (!temp && (d\_y <= d\_x))

while (x != x2) {

d\_x = x - x2;

d\_y = y2 - y;

if (err < 0) {

x = x - 1;

err = err + d\_y;

} else {

x = x - 1;

y = y + 1;

err = err + d\_y - d\_x;

}

Lcd\_PutPixel(x, y, Color);

}

else if (!temp && (d\_y > d\_x))

while (y != y2) {

d\_x = x - x2;

d\_y = y2 - y;

if (err < 0) {

x = x - 1;

y = y + 1;

err = err + d\_y - d\_x;

} else {

y = y + 1;

err = err - d\_x;

}

Lcd\_PutPixel(x, y, Color);

}

}

}

void Lcd\_Rectangle(unsigned char x0, unsigned char y0, unsigned char x1, unsigned char y1, unsigned char Color) {

unsigned char Temp;

if (x0 > x1) {

Temp = x0;

x0 = x1;

x1 = Temp;

}

if (y0 > y1) {

Temp = y0;

y0 = y1;

y1 = Temp;

}

Lcd\_VertLine(x0, y0, y1 - y0 + 1, Color);

Lcd\_VertLine(x1, y0, y1 - y0 + 1, Color);

Lcd\_HoriLine(x0, y0, x1 - x0 + 1, Color);

Lcd\_HoriLine(x0, y1, x1 - x0 + 1, Color);

}

void CircleDot(unsigned char x, unsigned char y, char xx, char yy, unsigned char Color) {

Lcd\_PutPixel((x + yy), (y + xx), Color);

Lcd\_PutPixel((x + xx), (y + yy), Color);

Lcd\_PutPixel((x - xx), (y + yy), Color);

Lcd\_PutPixel((x - yy), (y + xx), Color);

Lcd\_PutPixel((x - yy), (y - xx), Color);

Lcd\_PutPixel((x - xx), (y - yy), Color);

Lcd\_PutPixel((x + xx), (y - yy), Color);

Lcd\_PutPixel((x + yy), (y - xx), Color);

}

void Lcd\_Circle(unsigned char x, unsigned char y, unsigned char r, unsigned char Color) {//中点法画圆

unsigned char xx, yy;

char deltax, deltay, d;

xx = 0;

yy = r;

deltax = 3;

deltay = 2 - r - r;

d = 1 - r;

CircleDot(x, y, xx, yy, Color);

while (xx < yy) {

if (d < 0) {

d += deltax;

deltax += 2;

xx++;

} else {

d += deltax + deltay;

deltax += 2;

deltay += 2;

xx++;

yy--;

}

CircleDot(x, y, xx, yy, Color);

}

}

void Lcd\_Clear(unsigned char Mode) {

unsigned char x, y, ii;

unsigned char Temp;

if (Mode % 2 == 0)

Temp = 0x00;

else

Temp = 0xff;

Lcd\_WriteCmd(0x36);

for (ii = 0; ii < 9; ii += 8)

for (y = 0; y < 0x20; y++)

for (x = 0; x < 8; x++) {

EA = 0;

Lcd\_WriteCmd(y + 0x80);

Lcd\_WriteCmd(x + 0x80 + ii);

Lcd\_WriteData(Temp);

Lcd\_WriteData(Temp);

EA = 1;

}

Lcd\_WriteCmd(0x30);

}

void Lcd\_Reset() {

PSB = 1;

Lcd\_WriteCmd(0x30);

Lcd\_WriteCmd(0x0c);

Lcd\_WriteCmd(0x01);

Lcd\_WriteCmd(0x06);

}

//.c文件

#include "REG52.H"

#include "Key.h"

#define OS\_LONG\_KEY\_EN 1

#define KEY P1

void delays(unsigned int i) {

unsigned int j;

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

for (j = 300; j > 0; j--);

}

unsigned char OSScanKey(void) {

unsigned char Temp;

unsigned char i, key;

KEY = 0xF0;

delays(1);

Temp = KEY;

Temp = Temp & 0xF0;

Temp = ~((Temp >> 4) | 0xF0);

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

if ((Temp & (1 << i)) != 0)break;

}

if (i < 4) {

key = i;

//return i+1;

} else

return 0;

KEY = 0x0F;

delays(1);

Temp = KEY;

Temp = Temp & 0x0F;

Temp = ~(Temp | 0xF0);

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

if ((Temp & (1 << i)) != 0)break;

}

if (i < 4) {

key = key + i \* 4;

return key + 1;

} else

return 0;

}

unsigned char OSReadKey(void) {

static unsigned char KeyEventCnt = 0;

static unsigned char KeySampleCnt = 0;

static unsigned char KeyBuffer = 0;

#define SHORT\_ON\_DITHERING\_COUNTER 3

#define SHORT\_OFF\_DITHERING\_COUNTER 3

#if OS\_LONG\_KEY\_EN > 0

static unsigned int LongKeySampleCnt = 0;

#define LONG\_ON\_DITHERING\_COUNTER 250

#define LONG\_OFF\_DITHERING\_COUNTER 3

#endif

unsigned char KeyTemp;

KeyTemp = OSScanKey();

switch (KeyEventCnt) {

case 0:

if (KeyTemp != 0) {

KeySampleCnt = 0;

KeyBuffer = KeyTemp;

KeyEventCnt = 1;

}

return 0;break;

#if OS\_LONG\_KEY\_EN > 0

case 1:

if (KeyTemp != KeyBuffer) {

KeyEventCnt = 0;

return 0;

} else {

if (++KeySampleCnt > SHORT\_ON\_DITHERING\_COUNTER) {

KeySampleCnt = 0;

KeyEventCnt = 2;

LongKeySampleCnt = 0;

return ((KeyBuffer - 1) << 2) + 1;

} else

return 0;

}break;

case 2:

if (++LongKeySampleCnt > LONG\_ON\_DITHERING\_COUNTER) {

KeySampleCnt = 0;

KeyEventCnt = 3;

return ((KeyBuffer - 1) << 2) + 2;

} else {

if (KeyTemp != KeyBuffer) {

if (++KeySampleCnt > SHORT\_OFF\_DITHERING\_COUNTER) {

KeyEventCnt = 0;

return ((KeyBuffer - 1) << 2) + 3;

} else

return 0;

} else {

KeySampleCnt = 0;

return 0;

}

}break;

case 3:

if (KeyTemp != KeyBuffer) {

if (++KeySampleCnt > LONG\_OFF\_DITHERING\_COUNTER) {

KeyEventCnt = 0;

return ((KeyBuffer - 1) << 2) + 4;

} else

return 0;

} else {

KeySampleCnt = 0;

return 0;

}break;

#else

case 1:

if (KeyTemp != KeyBuffer) {

KeyEventCnt = 0;

//is dithering,return 0

return 0;

} else {

if (++KeySampleCnt >= SHORT\_ON\_DITHERING\_COUNTER) {

KeySampleCnt = 0;

KeyEventCnt = 2;

//sure that key on,return (KeyBuffer-1)<<2+1

return ((KeyBuffer - 1) << 2) + 1;

} else

return 0;

}break;

case 2:

if (KeyTemp != KeyBuffer) {

if (++KeySampleCnt >= SHORT\_OFF\_DITHERING\_COUNTER) {

KeyEventCnt = 0;

return ((KeyBuffer - 1) << 2) + 3;

} else

return 0;

} else {

KeySampleCnt = 0;

return 0;

}break;

#endif

default:break;

}

return 0;

}

//.c文件

#include "reg52.h"

#include "Lcd12864.h"

#include "Key.h"

#define uchar unsigned char

#define uint unsigned int

static unsigned long Seed = 1;

#define A 48271L

#define M 2147483647L

#define Q (M / A)

#define R (M % A)

double Random(void) {

long TmpSeed;

TmpSeed = A \* (Seed % Q) - R \* (Seed / Q);

if (TmpSeed >= 0)

Seed = TmpSeed;

else

Seed = TmpSeed + M;

return (double) Seed / M;

}

void InitRandom(unsigned long InitVal) {

Seed = InitVal;

}

void delay(unsigned int t) {

unsigned int i, j;

for (i = 0; i < t; i++)

for (j = 0; j < 10; j++);

}

void InitCpu(void) {

TMOD = 0x0;

TH0 = 0;

TL0 = 0;

TR0 = 1;

ET0 = 1;

EA = 1;

}

#define N 25

struct Food {

unsigned char x;

unsigned char y;

unsigned char yes;

} food;

struct Snake {

unsigned char x[N];

unsigned char y[N];

unsigned char node;

unsigned char direction;

unsigned char life;

} snake;

unsigned char Flag = 0;

unsigned char Score = 0;

unsigned char Speed = 5;

unsigned char KeyBuffer = 0;

#define FUNC 1

#define UP 2

#define DOWN 3

#define LEFT 4

#define RIGHT 5

#define PASSSCORE 20

void Timer0Int(void) interrupt 1 {

switch (OSReadKey()) {

case 9:

KeyBuffer = FUNC;

if (++Speed >= 10)

Speed = 1;

Flag |= 1 << 1;break;

case 13:

KeyBuffer = DOWN;break;

case 17:

KeyBuffer = UP;break;

case 21:

KeyBuffer = RIGHT;break;

case 25:

KeyBuffer = LEFT;break;

default:break;

}

}

void DrawBoard(void) {

unsigned char n;

for (n = 0; n < 31; n++) {

Lcd\_Rectangle(3 \* n, 0, 3 \* n + 2, 2, 1);

Lcd\_Rectangle(3 \* n, 60, 3 \* n + 2, 62, 1);

}

for (n = 0; n < 21; n++) {

Lcd\_Rectangle(0, 3 \* n, 2, 3 \* n + 2, 1);

Lcd\_Rectangle(90, 3 \* n, 92, 3 \* n + 2, 1);

}

Lcd\_HoriLine(93, 31, 35, 1);

Lcd\_HoriLine(93, 63, 35, 1);

}

void PrintScore(void) {

unsigned char Str[3];

Lcd\_WriteStr(6, 0, "成绩");

Str[0] = (Score / 10) | 0x30;

Str[1] = (Score % 10) | 0x30;

Str[2] = 0;

Lcd\_WriteStr(7, 1, Str);

}

void PrintSpeed(void) {

unsigned char Str[2];

Lcd\_WriteStr(6, 2, "级别");

Str[0] = Speed | 0x30;

Str[1] = 0;

Lcd\_WriteStr(7, 3, Str);

}

void GameOver(void) {

unsigned char n;

Lcd\_Rectangle(food.x, food.y, food.x + 2, food.y + 2, 0);

for (n = 1; n < snake.node; n++) {

Lcd\_Rectangle(snake.x[n], snake.y[n], snake.x[n] + 2, snake.y[n] + 2, 0);

}

if (snake.life == 0)

Lcd\_WriteStr(2, 1, "过关");

else

Lcd\_WriteStr(2, 1, "输了");

Lcd\_WriteStr(1, 2, "游戏结束");

}

void GamePlay(void) {

unsigned char n;

InitRandom(TL0);

food.yes = 1;

snake.life = 0;

snake.direction = DOWN;

snake.x[0] = 6;

snake.y[0] = 6;

snake.x[1] = 3;

snake.y[1] = 6;

snake.node = 2;

PrintScore();

PrintSpeed();

while (1) {

if (food.yes == 1) {

while (1) {

food.x = Random() \* 85 + 3;

food.y = Random() \* 55 + 3;

while (food.x % 3 != 0)

food.x++;

while (food.y % 3 != 0)

food.y++;

for (n = 0; n < snake.node; n++) {

if ((food.x == snake.x[n]) && (food.y == snake.y[n]))break;

}

if (n == snake.node) {

food.yes = 0;break;

}

}

}

if (food.yes == 0) {

Lcd\_Rectangle(food.x, food.y, food.x + 2, food.y + 2, 1);

}

for (n = snake.node - 1; n > 0; n--) {

snake.x[n] = snake.x[n - 1];

snake.y[n] = snake.y[n - 1];

}

switch (snake.direction) {

case DOWN:

snake.x[0] += 3;break;

case UP:

snake.x[0] -= 3;break;

case RIGHT:

snake.y[0] -= 3;break;

case LEFT:

snake.y[0] += 3;break;

default:break;

}

for (n = 3; n < snake.node; n++) {

if (snake.x[n] == snake.x[0] && snake.y[n] == snake.y[0]) {

GameOver();

snake.life = 1;break;

}

}

if (snake.x[0] < 3 || snake.x[0] >= 90 || snake.y[0] < 3 || snake.y[0] >= 60) {

GameOver();

snake.life = 1;

}

if (snake.life == 1)break;

if (snake.x[0] == food.x && snake.y[0] == food.y) {

Lcd\_Rectangle(food.x, food.y, food.x + 2, food.y + 2, 0);

snake.x[snake.node] = 200;

snake.y[snake.node] = 200;

snake.node++;

food.yes = 1;

if (++Score >= PASSSCORE) {

PrintScore();

GameOver();break;

}

PrintScore();

}

for (n = 0; n < snake.node; n++) {

Lcd\_Rectangle(snake.x[n], snake.y[n], snake.x[n] + 2, snake.y[n] + 2, 1);

}

delay(Speed \* 1000);

Lcd\_Rectangle(snake.x[snake.node - 1], snake.y[snake.node - 1], snake.x[snake.node - 1] + 2,

snake.y[snake.node - 1] + 2, 0);

switch (KeyBuffer) {

case FUNC:

KeyBuffer = 0;

if (++Speed >= 10)

Speed = 1;

PrintSpeed();break;

case DOWN:

KeyBuffer = 0;

if (snake.direction != UP)

snake.direction = DOWN;

break;

case UP:

KeyBuffer = 0;

if (snake.direction != DOWN)

snake.direction = UP;

break;

case RIGHT:

KeyBuffer = 0;

if (snake.direction != LEFT)

snake.direction = RIGHT;

break;

case LEFT:

KeyBuffer = 0;

if (snake.direction != RIGHT)

snake.direction = LEFT;

break;

default:break;

}

}

}

void Main() {

InitCpu();

Lcd\_Reset();

Lcd\_Clear(0);

DrawBoard();

GamePlay();

GameOver();

while (1);

}

**六、针对本实验写出实验心得**

做完这次试验我发现贪吃蛇本身是由好几个点连成的一条线段，只要通过延时函数使线段在时间前后往指定的方向进一步就行了，即将所有线段上的点往指定方向移动一段，就会出现贪吃蛇的基本运动。贪吃蛇的方向可以通过设置几个指定的按钮来控制。