综合项目 含多个游戏的小型游戏机设计与实现

姓名 李维唐 学号 2013012291

实验班号 周三下午 机器号 D-045

一、立题背景

随着计算机技术的发展和网络的普及，电子游戏在人们娱乐活动中的比例逐渐增加。电子游戏具有成本较低、简单易上手、沉浸式体验等特点，受到人们喜爱。通过玩电子游戏，可以放松身心，同时增强反应能力和逻辑思考能力等。对于小型游戏机来说，涉及的硬件并不复杂，核心是充分的人机交互，主要难点在于游戏的设计和实现。设计上，游戏应该具有一定的吸引力，其内容应该充实，画面尽可能精致。实现上，利用 C 语言从头实现一些简单的游戏具有一定的难度。

二、功能描述

本产品硬件上主要由 MSP430G2553 单片机、12864B 液晶显示模块、控制器模块和电源模块组成。电源模块给单片机和液晶屏供电，由液晶屏显示游戏画面，玩家操控控制器来进行人机交互，完成游戏过程。

该产品主要有以下功能：

1. 初步的操作系统

在单片机上实现一个初步的人机交互操作系统，单片机启动时，打开游戏选择界面，供玩家选择游戏，在游戏过程中或者游戏结束时，可以退出游戏回到游戏选择界面。

2. 贪吃蛇游戏

贪吃蛇是经典的电子游戏，最早历史可以追溯到 1976 年，玩家需要用方向键控制一条蛇在二维地图上行走，试图吃掉食物以获取更高积分，每吃掉一个食物，蛇的长度会增加 1，如果蛇撞到了墙或者自身，则游戏失败。该游戏规则简单，需要快速的应变，可以锻炼人的反应能力，并且除此之外还对人的逻辑思考能力提出了一定的要求，即：如何制定出合理的蛇行走策略，能够使蛇有较大概率长（zhang）到很长（chang）？本游戏中可以选择贪吃蛇的游戏难度即蛇的行进速度，还以在游戏中随时暂停游戏，并且在游戏结束后能看到自己所获得分数。

3. 推箱子游戏

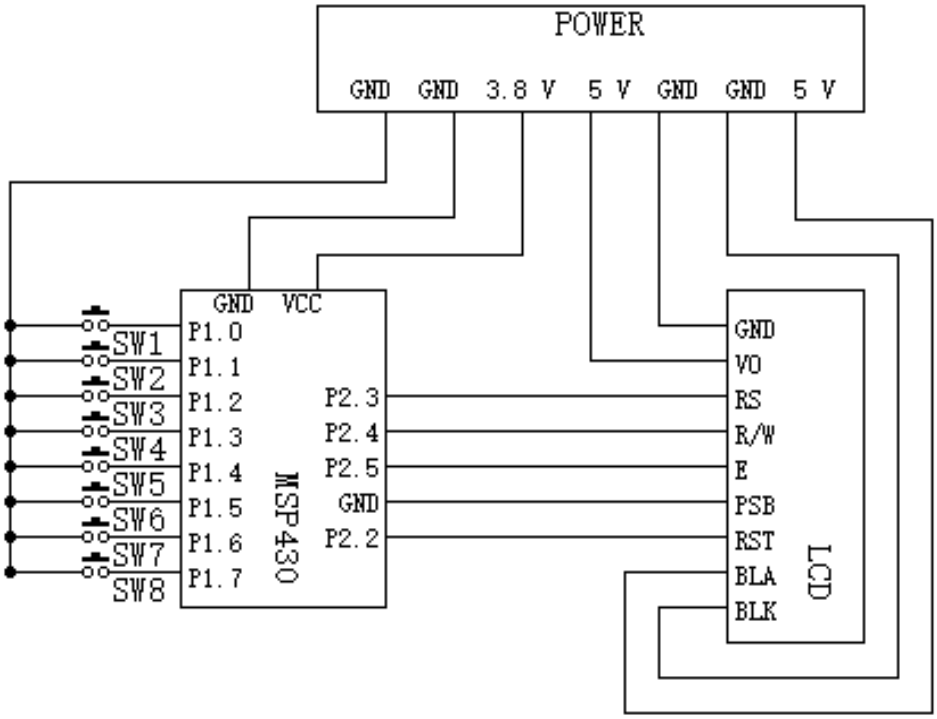
推箱子游戏也是经典的电子游戏，玩家需要用方向键控制一个工人，在狭小的空间内将货箱推到指定的位置。推箱子的策略必须精心制定，否则很容易出现通道堵塞或者箱子无法移动的情况。该游戏规则虽然简单，但是通过不同的地图可以演化出很多变化，趣味无穷。该游戏需要玩家有很强的逻辑推理能力和锲而不舍的精神。本游戏中一共提供 10 个关卡可以自由选择。

4. 其它游戏

其它或经典或新潮且比较容易实现的电子游戏还有很多，如俄罗斯方块、2048、翻转棋等等，在初步的操作系统基础上，后期根据实际项目进度添加。

三、硬件设计

本产品的主要硬件设计如下图所示：

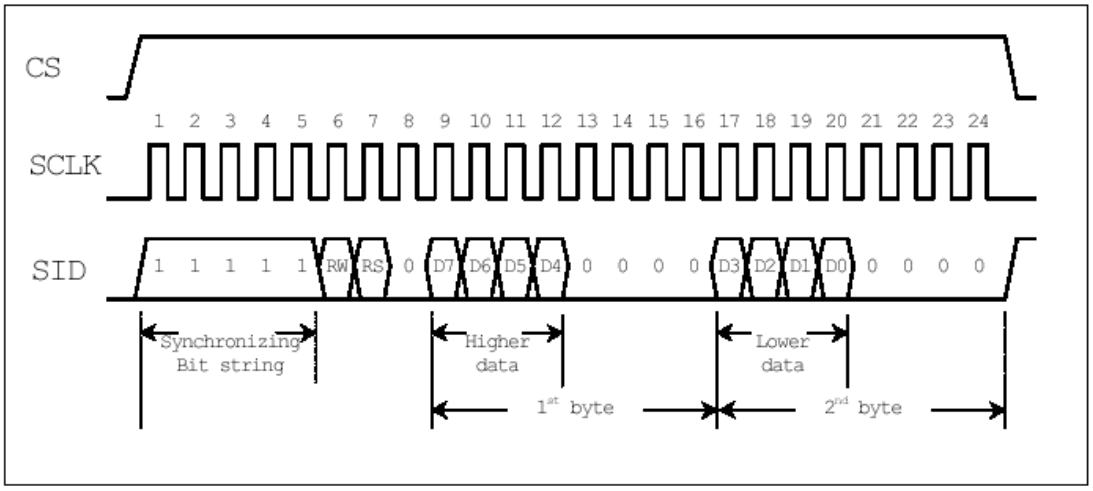


单片机和 LCD 均由电源模块供电，所有模块共地。

游戏机的核心在于充分的人机交互，因此本产品有八个按键占用 P1.0 到 P1.7 八个引脚，以中断方式为玩家提供快速的响应。

本产品中硬件部分较为复杂的是 LCD 的控制。本产品使用串行通信方式控制 LCD，占用单片机 P2.3 到 P2.5 三个引脚，分别连接 LCD 上的 RS、R/W、E 引脚进行串行通信，另 P2.2 连接 RST 控制 LCD 复位。GND 和 VO 为液晶屏显示供电引脚，BLA 和 BLK 为液晶屏背光供电引脚。这两者的供电电压决定了液晶显示的质量，经测试都接 5 V 左右为宜。

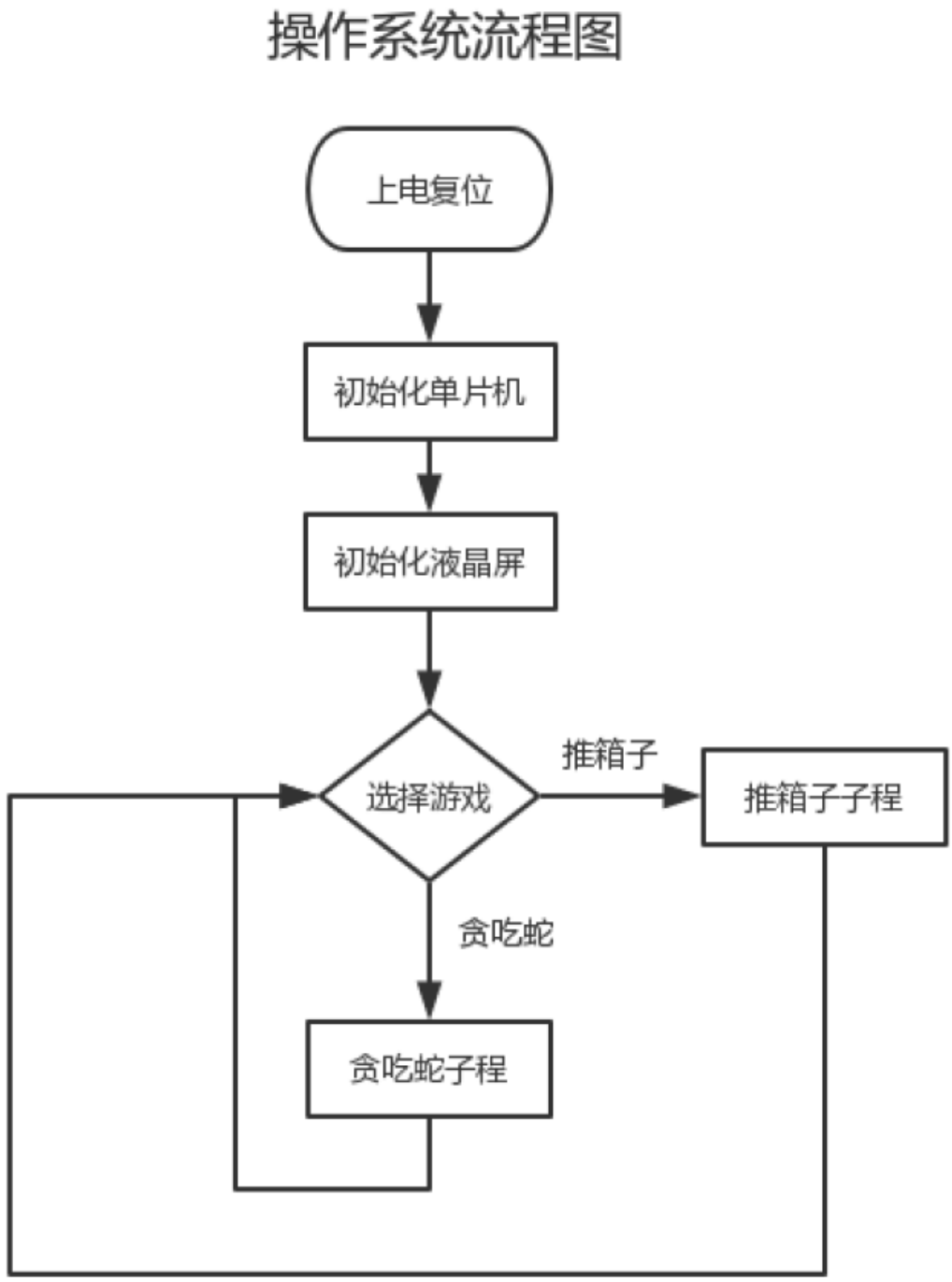
12864B 这款液晶屏含有 128*64 个像素点；自带 GCRAM、DDRAM 和 GDRAM；自带 unicode 字库；支持自定义字库和绘图功能；从基本指令集到扩展指令集共计 18 种指令；支持串行模式和并行模式两种通信方式。其中本产品使用的串行通信模式时序图如下：

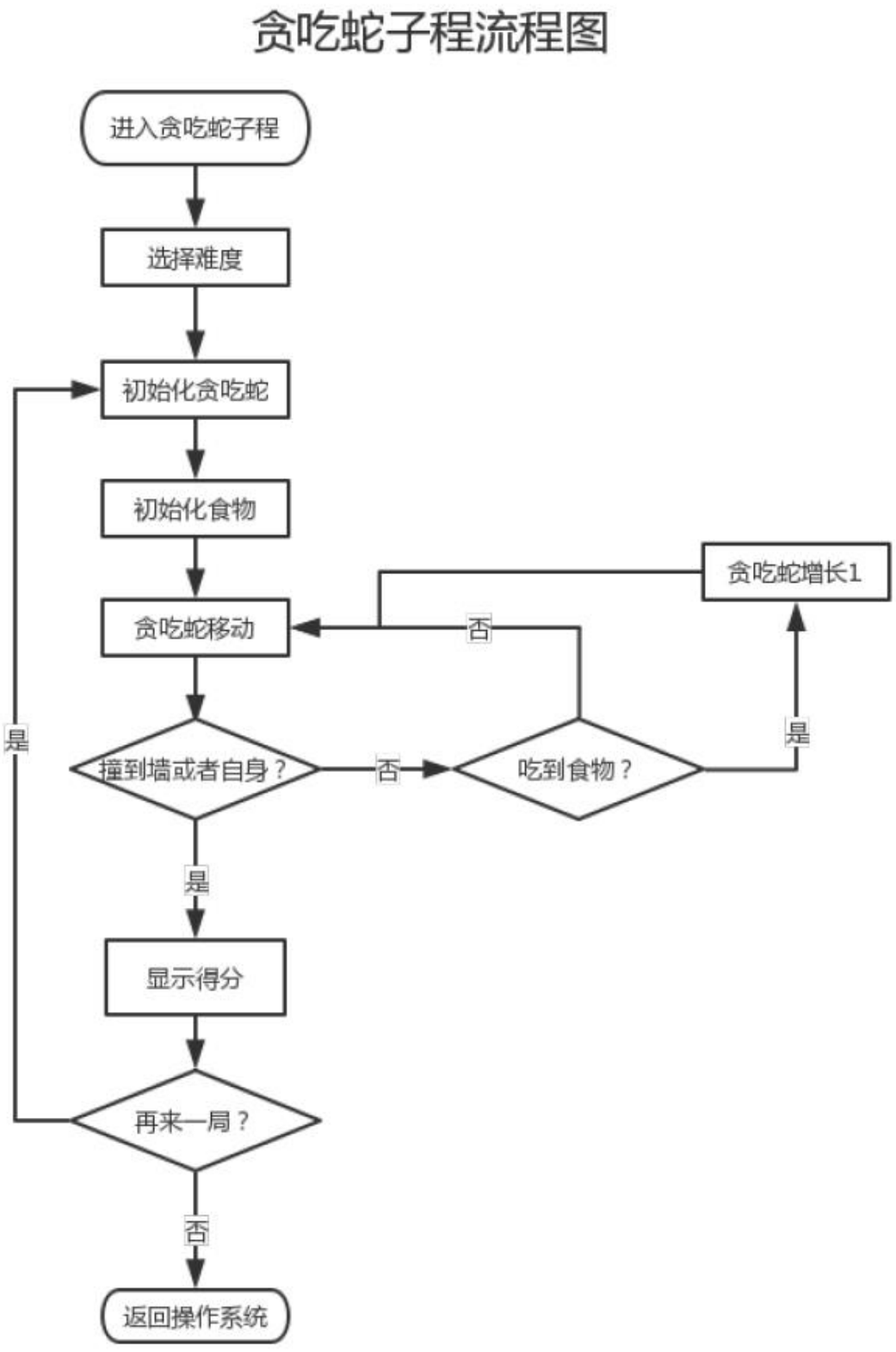


其中 CS 信号接入 RS 引脚，SCLK 信号接入 R/W 引脚，SID 信号接入 E 引脚。在基本指令集下，共可以在固定的位置显示 64 个 ASDII 字符或者 32 个汉字，在扩展指令集下，可以对 128*64 个像素点进行逐一绘制。

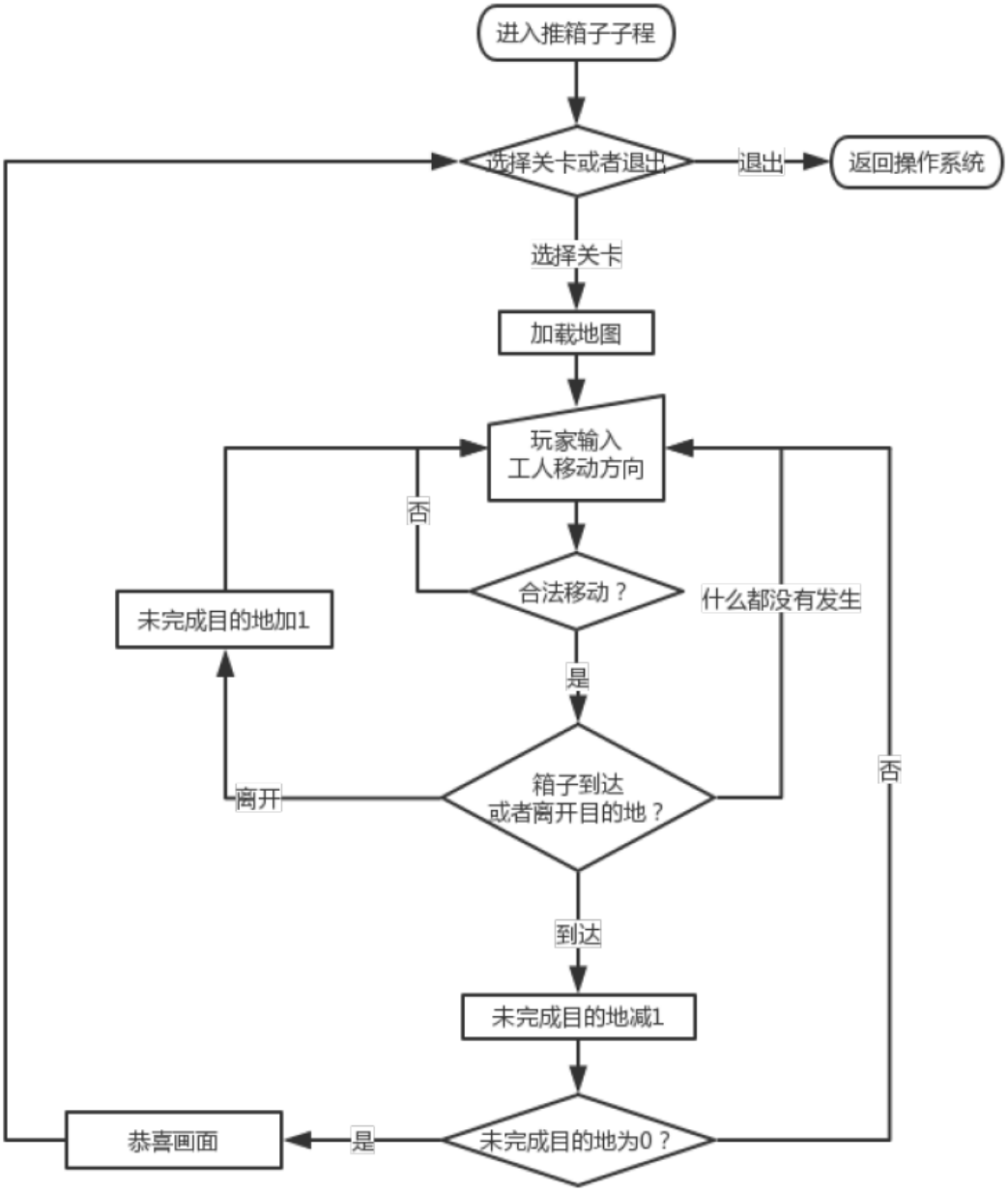
四、软件模块划分

软件目前分为操作系统、贪吃蛇和推箱子三个模块，设计的流程图如下所示（第三个游戏还没有实现）。为了实现快速的人机交互响应，本产品大量使用中断实现产品功能，如暂停、退出等，这些难以体现在流程图中，故这里略去。





其中贪吃蛇方向的改变、暂停等功能以中断方式进行，无法在流程图中显示。



玩家可以以中断方式重置游戏。

五、项目程序清单

1. 程序设计概述

本项目编程难度较大，总体代码数量达到 1500 行左右，分布在 9 个 c 源文件和 h 头文件里。具体的代码收录于附件中，在报告里给出代码结构和主要核心代码，分为以下六个模块：

程序入口：main.c。程序从 main.c 开始执行，进行单片机硬件的初始化和液晶显示屏的初始化，然后进入主菜单，进行游戏的选择。

基本程序部件：basic.h。程序后续实现需要预先定义一个二维点结构和若干全局变量。

硬件抽象层：hardware.h 和 hardware.c。在这些文件中实现了单片机初始化函数、单片

机和 LCD 通信函数、LCD 初始化及重置函数。

贪吃蛇游戏：greedy_snake.h 和 greedy_snake.c。在这些文件中实现了贪吃蛇游戏。

推箱子游戏：push_box.h 和 push_box.c。在这些文件中实现了推箱子游戏。

中断子程：interrupt.c。单片机通过 P1 按键中断进行人机交互，在该文件中定义了 P1 的中断子程。

下面对这六个模块分别进行介绍。

2. 程序入口主要函数，包括：

Main(); //程序入口，初始化单片机和 LCD，并进入主菜单页面

void WriteSymbol(unsigned char line); //写标记符号函数，用于在游戏间进行选择时，调整光标的位置

3. 基本程序部件主要定义，包括：

struct Point; //点结构，带有 x、y 两个变量，用于表示二维平面上的位置

typedef unsigned char Mode;

Mode g_mode; //全局变量，用于记录程序当前位于什么模式，在哪个菜单中

unsigned char g_drctn; //全局变量，用于记录用户输入的方向

4. 硬件抽象层主要定义和函数，包括：

硬件连线的宏定义，将外部输入输出对应于寄存器上的各个位。

void IniMcu(); //初始化单片机的引脚、时钟、定时器

void RstLcd(); //重置 LCD，在 LCD 重置引脚上发送一个脉冲

extern inline void Pulse(); //在 SCLK 上发送一个脉冲，用于串行通信

void SendByte(unsigned char cmd); //向 LCD 发送一个字节

void Send(int cmd); //向 LCD 发送一个指令（两个到三个字节），另对指令进行了宏定义

void IniLcd(); //初始化 LCD，重置之并打开显示

void WriteSentence(char* s,unsigned char row); //在 row 指定行显示字符串 s（16 字节）

void WriteShortSentence(char* s,unsigned char sz, unsigned char row); //写长为 sz 短字符串

void RstGraph(); //重置图形显示，进入图形显示模式并清除所有显示内存

void WritePoint(unsigned char X,unsigned char Y,unsigned char shp); //在(X,Y)位置写 shp 形状，对 shp 进行了宏定义

5. 贪吃蛇游戏主要定义和函数，包括：

struct Snake; //蛇结构，包含贪吃蛇移动、生长必要的参数，矩阵形式实现

unsigned char GsSetShp(struct Snake *S, unsigned char X, unsigned char Y, unsigned char PXshp); //设置(X,Y)位置的形状

void GsIniSnake(struct Snake *S); //初始化蛇

unsigned char GsGrow(struct Snake *S,struct Point *food); //蛇生长，根据 food 位置判断是否吃到了食物

unsigned char GsMove(struct Snake *S,struct Point *food); //蛇移动，调用蛇生长，没有吃到食物，剪去尾巴，否则直接返回

void UpdateFood(struct Snake *S,struct Point *food); //当食物被吃掉后，随机生成新的食物

unsigned char GsSelectDffclty(); //选择难度的页面，让玩家选择游戏难度

void GsMain(); //贪吃蛇游戏主函数，调用难度选择界面，实现游戏和结束界面

void IntGsDs(unsigned char ifg);　　//中断子程中的子函数，用于难度选择界面

void IntGsGm(unsigned char ifg);　　//中断子程中的子函数，用于游戏界面

void IntGsEnd(unsigned char ifg);　　//中断子程中的子函数，用于结束界面

6. 推箱子游戏主要定义和函数，包括：

struct Pb_map;　　//推箱子游戏结构，推箱子是地图型游戏

unsigned char PbSetShp(struct Pb_map *M, unsigned char X, unsigned char Y, unsigned char PXshp);　　//设置(X,Y)位置的形状

unsigned char PbGetShp(const struct Pb_map *M, unsigned char X, unsigned char Y);　　//获得(X,Y)位置的形状

对地图的定义：利用 unsigned char 数组定义了 10 幅推箱子游戏的初始地图，每个大小为 33 Byte，利用关卡数进行索引

void PbIniMap(struct Pb_map *M,unsigned char level);　　//根据关卡初始化地图

void PbWriteLevel(unsigned char lvl);　　//选择关卡界面中，显示当前选择的关卡

unsigned char PbSelectLevel(unsigned char prev_level);　　//选择关卡界面

void PbPush(struct Pb_map* M,const struct Point p1,const struct Point p2);　　//工人不同的移动方式引起不同的地图变化

void PbMain();　　//推箱子游戏主函数，调用关卡选择界面，实现游戏和结束界面

void IntPbLs(unsigned char ifg);　　//中断子程的子函数，用于关卡选择界面

void IntPbGm(unsigned char ifg);　　//中断子程的子函数，用于游戏界面

void IntPbEnd(unsigned char ifg);　　//中断子程的子函数，用于结束界面

7. 中断子程

本项目中只有 P1 中断一种可屏蔽中断源，本文件中定义中断子程函数：

__interrupt void port1_isr();

在不同软件模式下对中断进行不同处理，实现暂停/开始游戏，软件去抖等功能。

六、调试过程中的难点和解决方法

1. 串行并行通信的问题

在于 LCD 通信的过程中，并行通信的速度是要比串行通信快的。经过测试，通过并行通信可以令人满意地对在绘图时对显示屏进行快速的刷新，而通过串行通信则可以感受到明显的闪烁和延时。如此看来，必须要使用并行通信实现单片机和液晶屏的互联。但是并行通信需要占用单片机多达 11 个引脚，这会导致留给控制器系统的引脚最多只有 5 个，难以实现有效的人机交互。这一问题初看无法解决，所以最初我决定使用并行通信的架构，适当简化人机交互。后来我突然想到，提高主系统时钟，在能满足 LCD 时序的前提下，能否实现较快的串行通信呢？经过测试，将主系统时钟从 1 MHZ 提高到 12 MHZ 左右，即可以进行有效地通信，不会有明显的闪烁和延迟。如果再提高频率，则会导致信号失真。通过这样的方式，我只使用 4 个引脚就实现了单片机和 LCD 的通信，为人机交互分配了 8 个引脚，并仍留有余量，可供后续扩展。

2. Malloc 函数的使用问题

这是本次项目实现中最难调试的一个问题了。在项目实现的初期，我首先尝试实现贪吃蛇游戏，在已经声明和定义了贪吃蛇的 Snake 结构之后，创建 Snake 对象以及其相应指针时，我使用了 C 语言中常用于此类情形的 malloc 函数，即

struct Snake * snake_pointer = malloc(sizeof(struct Snake));

随后对其进行初始化，即对 struct Snake 的大部分成员变量（约 200 Byte）赋值为 0。然

而总是无法得到期望的结果。在实际调试中我发现，在 MSP430 中存储器中对应 memory 的部分，即 0x200 到 0x400 的位置，不论是 RAM 还是栈中，经过 struct Snake 的初始化赋值后，都没有明显的变化。我被这个问题困扰了很久，一度认为是需要分配的内存空间过大，导致无法分配的缘故。直到后来，通过查看内存、反汇编等种种手段我发现，当通过 malloc 为 struct Snake*指定了地址后，初始化 Snake 时实际初始化的位置是从 0x000 到约 0x150 的存储器区域，也就是说，snake_pointer 的值并不是预想的 0x200 到 0x400 中的某个值，而是 0。0x000 到 0x200 这一区域是被系统所保留的，对应于内部寄存器，应用程序原则上可以对其进行赋值，但是通过一个 struct Snake 对象来赋值显然是没有意义的。这就产生了一个单片机上的指针越界错误，而且不光是越了数组的界或者栈帧的界，而是把整个内存的界都越了。

如果将这一段代码改为

Struct Snake snake;

Struct Snake * snake_pointer = &snake;

即将 snake 先作为局部变量声明，则这一错误可以避免。

这是我在 windows 和 linux 系统下进行编程的经验盲目移植到单片机上的结果。单片机上没有虚拟存储器系统，malloc 只会从头开始分配内存空间，因此不能使用 malloc 来初始指针。同时我也体会到了 C 语言中指针的强大性和危险性。善用指针，可以完成其它代码很难完成的任务，而如果出现问题，则很难调试且容易引起整个系统的崩溃。

3. 随机数的产生问题

在贪吃蛇游戏中，需要产生随机数来提高游戏性，但是我在最初并没有想到在单片机中产生随机数的方法。C 语言的随机数本质上是伪随机数，一般需要系统时间来提供随机数种子来达到产生随机数的效果。但是在单片机中，没有系统时间的概念。后来，我想到可以模仿系统时间，使用单片机中的定时器系统来完成这一功能。在初始化单片机时，开启单片机的定时器功能，使 TA0R 开始计数，但是不对程序和引脚产生影响。当玩家进入需要产生随机数的贪吃蛇游戏后，TA0R 的值由玩家在主菜单中逗留的时间所决定了，以此作为随机数种子产生随机数，可以令人满意地实现贪吃蛇中所需要的随机数效果。

4. 绘图时写 LCD 的 GDRAM 问题。

在本项目中实现的游戏中，基本上把屏幕视为 8*16 的格点，但是在 LCD 内部的 GDRAM 里，是以 64*8 的形式来进行组织的，水平方向上的地址分辨率（8）小于需求的 16。这意味着，当试图使 LCD 在某个坐标上显示某个形状，也就是写某个地址的 GDRAM 时，每次寻址对应的都是两个格点。通过适当编程，可以在不影响第二个格点的前提下写第一个，但是不可能在不影响第一个格点的前提下写第二个。初步的结局方案有两个，一是顺应 LCD 内部的硬件结构，将屏幕视为 8*8 的格点来编写游戏，但这样实施不仅分辨率、可玩性相对降低，重点是界面非常不美观。另外一个解决方案是读和写交替进行，每次写 GDRAM 时先读取相应内容，然后修改后再写入，但是对硬件软件调试都有一定的要求。最后我采用的方法是，在单片机内存中保存全部的地图形状，每次写都根据保存的地图形状写两个点。这样增大了内存的使用和一定的编程难度，但较好地解决了问题。

七、不足和可改进之处

1. 游戏数量不足

在项目设计中，计划至少加入三个游戏，其中一些游戏可能还带有人机对战功能，要求编写 AI，实现难度较大，因最后时间有限，没有完成。

2. 人机交互还不够充分

游戏机的核心是充分的人机交互，本项目中虽然已经尽可能地最大化人机交互，但是限于时间精力还有很大提升空间。比如，可以引入扬声器或者蜂鸣器，为游戏提供音效支持；可以引入陀螺仪等设备， 为游戏提供重力感应支持等。市面上还有带触摸屏的 LCD，如果采用这样的 LCD 或许可以支持切水果之类的游戏。总之游戏的人机交互有很多方式，而且也是现在大型游戏发展的前沿（例如 VR 技术），在这方面还有很多可以改进。

3. 编程语言的选择

在项目设计与实现的初期，使用 C 语言来进行编程，当项目规模逐渐增大时，越发感觉使用 C++面向对象编程能提高编程的效率、可维护性和可扩展性。然而进行了初步的尝试之后，发现移植难度很大，最后因为时间精力有限，放弃了 C++实现。

4. 游戏平台的移植

现在使用掌上游戏机的人越来越少，其原因是掌上游戏机的功能越来越多地被手机所替代。本项目中虽然可以加入一个蓝牙模块通过手机来进行游戏，但更简单有效的手段是直接在手机上进行游戏机的编程。手机 APP 开发又分为 iOS 开发和 Android 开发两大类，每一类中都有很多学问，这就留待我在现在的游戏机设计与实现的基础上，进一步在今后的学习道路上探索。