大连理工大学

硕士学位论文

基于51单片机的贪吃蛇游戏

姓名：赵子翔

申请学位级别：硕士

专业：软件工程

指导教师：陈志奎

20090606

摘　要

随着科技的发展，现代生活节奏越来越快，人们的工作生活压力也随之加大。设计一款操作简单，生动新颖，娱乐性强，便于携带的小游戏，在繁忙的工作生活之余玩玩这款小游戏，不仅可以调节人们的情绪，使人心情舒畅，还能健脑益智，为更好地投入工作学习做好准备。

本文基于单片机设计的贪吃蛇游戏，除了具有传统意义上的贪吃蛇游戏的特点：吃豆子蛇身增长，得分；分数达到一定等级进行升级以后，蛇运动速度加快等以外，本次设计加入了游戏暂停，中途退出，地图选择，背景提示音效等功能。为节省存储空间，游戏算法上进行了新的设计，定义一个一维数组，利用位操作存储读取蛇头的运动状态信息。

具体实现上，硬件系统平台采用 51 系列单片机，搭载 LCD，键盘和扬声器，构成了一个轻巧便携的游戏机系统。51 系列单片机技术成熟，功能强大，应用广泛。使用单片机作为控制核心，可以简化硬件电路，采用软件编程控制单片机实现硬件电路的功能，降低能耗，降低成本。软件采用 C 语言编程，方便灵活，大大加快了软件开发速度，缩短了开发周期，并且便于移植。为提高开发效率和硬件稳定性，采用了功能强大的硬件仿真软件 Proteus，依托该软件提供的仿真环境搭配 Keil 作为软件开发调试环境进行仿真调试成功。

文中具体介绍了使用到的各种硬件的特性，游戏的各种功能与详细设计，软件的具体设计思路，各模块的详细介绍，部分模块的程序流程图，状态迁移图，关键部分代码的详细讲解等。

关键词：贪吃蛇；51 单片机；游戏

Greedy Snake Game Based on 51MCU

Abstract

With the development of science and technology, more and more fast-paced modern life, the pressure of people also increase. Designing a simple game with vivid, novel, entertaining and easy to carry on, it can satisfy people's needs. On a busy life, to play this simple game,not only may adjust people's mood, but also make people smart. Relaxing yourself like this, it can improve working and learning efficiency.

In this paper, the game of Greedy Snake is such a game based on MCU. In addition to the traditional sense of the Greedy Snake game features: Snake body grows and scores with eating a bean; Scores reached a certain number to upgrade the level and speed, many new features are added, for instance, game pause or exit midway, map choice, rich audio and other functions. In order to save the storage space, the algorithms of game makes a new design.By defining a one-dimensional array, snake head's motion states are stored and read in this array with bit operation.

The hardware system uses 51 MCU platform with two LCDs, keyboards and a speaker, ultimately forming a compact portable game system. 51 MCU's technology is mature, powerful and widely used. The use of MCU as control core, you can simplify the hardware circuit, achieve hardware functions of the circuit with software programming to control the MCU,and reduce energy consumption and costs. Software use C language programming, it is convenient and flexibly easy to transplant. In order to enhance the efficiency and hardware stability of development, using a powerful hardware emulation software Proteus, based on the software simulation environment provided with the Keil software as a debugging environment to debug successfully.

The article introduces the use of a variety of hardware features, the game features and the detailed design, software design of the specific ideas, details of various modules, some modules of the program flow chart, state transition graph, a key part of the code details and so on.

Key Words: Greedy Snake; 51MCU; Game

大连理工大学学位论文独创性声明

作者郑重声明：所呈交的学位论文，是本人在导师的指导下进行研究工作所取得的成果。尽我所知，除文中已经注明引用内容和致谢的地方外，本论文不包含其他个人或集体已经发表的研究成果，也不包含其他已申请学位或其他用途使用过的成果。与我一同工作的同志对本研究所做的贡献均已在论文中做了明确的说明并表示了谢意。

若有不实之处，本人愿意承担相关法律责任。

学位论文题目：基于引擎片机的贪吃蛇游戏

作 者 签 名：赵子翔　　日期：2009 年 6 月 10 日

大连理工大学学位论文版权使用授权书

本人完全了解学校有关学位论文知识产权的规定，在校攻读学位期间论文工作的知识产权属于大连理工大学，允许论文被查阅和借阅。学校有权保留论文并向国家有关部门或机构送交论文的复印件和电子版，可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印、或扫描等复制手段保存和汇编本学位论文。

学位论文题目：基于51单片机的贪吃蛇游戏

作者签名：赵子翔　　日期：2009年6月10日

导师签名：[illegible]　　日期：2009年6月10日

1 绪论

1.1 研究背景与意义

随着社会的发展，人们生活的步调日益加快，越来越多的人加入了全球化的世界。人们不再拘泥于一小块天地，加班，出差成了现代人不可避免的公务。而此时一款可以随时随地娱乐的游戏成了必需品。贪吃蛇这一游戏简单易行，操作方便，娱乐性较强，吸引了不少人。这一款游戏紧紧地抓住了人们的心理，虽然简单，却其乐无穷，在人们不断追求更多的欲望下，该游戏给人们带来了追逐的快感，以及成功后的满足感，对于一直处于高压下的现代人是很好的放松工具。

当前科学技术飞速发展，特别是微电子技术，计算机软件与应用技术的发展，使得人们的日常生活丰富多彩。单片微型计算机（简称单片机）作为微型计算机家族的一员，以其独特的结构，良好的稳定性，便宜的价格在嵌入式领域广泛应用。与传统的 PC 上设计的贪吃蛇游戏不同，本次作者利用 Proteus 硬件仿真软件，采用单片机、液晶显示屏、扬声器、按键等搭建硬件平台，C 语言编程，实现便携地贪吃蛇游戏。

传统的贪吃蛇游戏只有单纯的吃豆子，得分，升级以后蛇运动的速度加快等功能。本次作者对贪吃蛇游戏进行了升级，出上述基本功能外，针对现有硬件条件，加入地图选择，游戏中途暂停与退出，各种背景音的播放（包括吃豆子背景音、错误提示背景音、升级背景音、游戏结束背景音、游戏通关背景音等），背景音静音与否的选择等，使玩家的游戏体验更上一层楼。

1.2 单片机发展状况

单片微型计算机（Single-Chip Microcomputer）简称单片机（MCU）。它是在一块芯片上集成了中央处理单元（CPU）、振荡器电路、只读存储器（ROM）、随机存取存储器（RAM）、并行/串行 I/O 接口、可编程定时器/计数器等，有的甚至包含了 A/D 转换器。总之，这么一块小小的单片机芯片，就相当于一台微型计算机，它具有体积小、重量轻、单一电源、低功耗、功能强、价格低廉、运算速度快、抗干扰能力强、可靠性高等特点。1974 年，美国仙童（Fairchild）公司生产出世界上第一块单片机，短短几十年的时间，单片机如雨后春笋一般，大量涌现出来。目前，已经出现了 4 位、8 位和 16 位单片机，甚至 32 位超大规模集成电路单片机也已问世，性能也在不断地提高。

国内从 80 年代起开始了单片机的热潮，二十多年过去了，单片机从研究所走出来，成为与日常生活中的一个不可缺少的部件。早些时候单片机种类稀少，开发工具奇缺。8035、8048、Z80 等在现在主流市场上基本已没有踪影，用汇编语言开发产品的艰苦工

作也逐步被C语言取代。硬件方面日趋多样化，4位、8位、16位、32位等型号共同并存，在不同的领域存在，如家电、玩具、工业设备、仪器、通讯。价格也从几元到几百元不等。每一种单片都有它所擅长的领域，如PIC系列较多用于电话机、玩具，51系列较多用于设备控制和仪器，DSP较多用于DVD、通讯等。软件方面发展主要为汇编语言、C语言、嵌入式操作系统。速度、稳定性特别要求的场合较多采用汇编语言和C语言，如电机控制，UPS控制、信号处理等。功能复杂、内容较多的系统多采用嵌入式操作系统，如PDA、电子词典、游戏机等。以后的发展中，各类型号的单片机种类会进一步增加，而开发工具和过程会逐步趋向于统一，软件和硬件差别会更加难以区分[1]。

1.3 LCD发展状况

液晶显示器TFT LCD，全称为薄膜晶体管液晶显示器（Thin Film Transistor Liquid Crystal Displayer），一般简称LCD（Liquid Crystal Display）。超薄体形、低功耗、低辐射、无闪烁、完全物理平面、低反光、清晰的字符显示等等，都是大家非常熟悉的液晶显示器LCD优点。最简单的液体晶体管就是我们常见的小型计算器以及电子手表上面的液晶字符屏幕。他是把有机液晶原料夹在两片透明的玻璃或者有机玻璃中。没有电流通过的时候，长棒状的原料晶体分子是无规则排列的，光线无法随意透过玻璃，外表看上去就是黑色。通电的时候，液晶原料排列顺序随电流极向改变，光线在规则排列的晶体分子中可以透过，液晶管由原来的非透明状态变成透明状态。通过把液晶材料进行不同的排列，组成不同的字符形状，就能通过电流控制其开关显示，以显示出我们说需要的字符。液晶技术发展的早期，由于液晶管的稳定性以及生产技术，还不能大量大规模的生产，直到了英国的科学家发明了用“联苯（Biphenyl）”作为液晶管的原料，这个问题才得以解决。1970年，弗格森制造了第一台能够工作的LCD，而在此之前的所谓LCD都是耗电量大而且对比度极低的昂贵设备。到了1971年，这种新的液晶显示器开始普遍地为人们接受。当然，那时候的LCD还是单色产品，但是已经不是简单的字符型液晶屏幕了。LCD技术是把液晶灌入两片偏振玻璃之间。所谓偏振玻璃，就是光线通过这样的玻璃之后，就会从球面波或者高斯球面波，变成只在一个平面上振动的波，称为偏振光。偏振光只能通过相应方向的偏振玻璃，如果偏振玻璃的偏振方向和偏振光线的有一定的夹角，就会减弱偏振光强度，甚至偏振光无法通过。如果大家对这方面有兴趣，可以参阅有关的大学物理书籍。夹住液晶的两片偏振玻璃，假设为a、b，他们的偏振方向会设置为90度夹角。光线通过第一片偏振玻璃a后，假设这X方向偏振，通过液晶后，液晶通电流之后，在电场极化作用下，呈规则排列，X偏振光不会有任何改

变，投射到 b 玻璃上。而 b 玻璃的偏振方向为 Y，就是 X+90 度，X 偏振的光线无法通过，在 b 玻璃外面看上去就是黑色了。而如果液晶没有电场作用，就是没有通电流，通过无规则排列的液晶，X 偏振光的偏振方向会发生改变，旋转 90 度，旋转后 X 偏振光的偏振方向刚好和 b 偏振玻璃的偏振方向一样，就是 X+90=Y，光线就能通过 b 玻璃了[2]。

1.4 作者的主要工作

贪吃蛇游戏是一款经典的小游戏，前人根据不同的需求，使用不同的编程语言和算法实现过该游戏。本次作者基于 51 单片机这一常用的硬件平台，充分发挥其性能，在嵌入式平台利用 C 语言编程实现这款经典游戏。

由于嵌入式平台对于硬件资源有着相对于 PC 机开发的应用软件更为苛刻的要求，特别是在 RAM/ROM 的存储空间大小上，所以本次作者将重点放在如何处理蛇的运动轨迹方面，采用了一个无符号 char 型一维数组来存放蛇头的运动轨迹，并考虑到蛇身的最大长度，将该数组定义为 54，充分节省了有限的存储空间，通过处理按键，完成了对蛇运动的控制以及游戏控制。显示游戏信息和游戏运行画面的 LCD1602 和 LCD12864 使用广泛，技术相对成熟，故 LCD 底层驱动采用前人成果，并在此基础上设计了游戏的运行界面。单片机背景音的产生于播放也参考了许多他人的思想。

1.5 本文结构

本文按照整个系统的开发流程，首先进行游戏的需求分析，然后介绍系统的架构设计，包括论文使用的硬件结构设计，软件的结构设计与模块划分，最后是各模块的软件详细设计与实现。

2 贪吃蛇游戏需求分析

贪吃蛇是一款经典小游戏，游戏的规则是：玩家通过方向键（上，下，左，右）来控制蛇移动，在地图上吃豆子。吃掉豆子后蛇身加长，并且会增加相应分数，达到一定分数以后升级，升级后，蛇身运动速度加快。蛇运动时撞到墙壁（屏幕框），障碍物或者蛇身结束游戏。

本次贪吃蛇游戏主要功能如下：选择游戏地图（4幅地图）；蛇控制（移动、吃豆子等）；游戏信息（游戏时间、级别、分数、背景音开关等）；游戏界面（游戏运行、暂停、通关等界面）；游戏背景音（吃豆子、升级、通关、失败等背景音）。

2.1 游戏信息显示界面与分数等级计算规则

游戏时间显示游戏已经运行时间，以分钟：秒形式显示（00:00）。

游戏分数显示是根据等级、蛇吃豆子的多少增加分数、显示分数。蛇吃一个豆子，分数加10。分数从0开始，通关后分数为2100。

游戏等级显示是根据蛇吃豆子增加分数到某个特定值增加等级数。本次设计默认等级从1开始，吃10个豆子升到第二级，在此基础吃20个豆子升级到第三级，以此类推。升级到第六级吃60个豆子即可通关（共吃210个豆子）。

用户选择背景音，则屏幕右上方有喇叭图标处于打开状态显示；用户选择静音模式，则屏幕右上方喇叭图标处于静音状态显示。

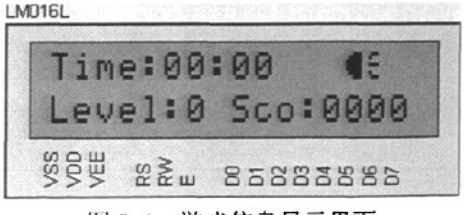

图2.1 游戏信息显示界面

Fig. 2.1 Game Information Display Interface

2.2 游戏界面状态显示

上电开机后，LCD12864界面全屏显示游戏的欢迎信息"Welcome to Snake World!"，蛇的图像，以及点击"start"键的提示信息。通过单击开始键可以进入游戏地图选择界面。开机界面如图所示：

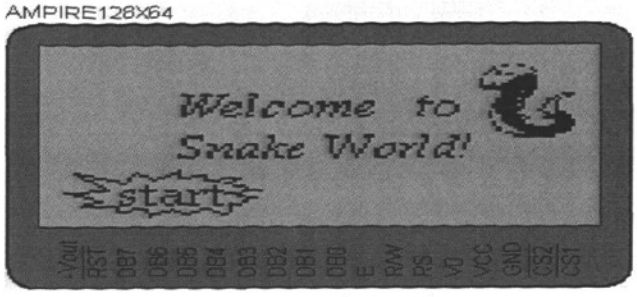

图 2.2　开机界面

Fig. 2.2　Boot Interface

游戏地图选择界面以图形的方式显示，一共 4 个地图。用键盘按键选择地图，单击开始键进入游戏界面。游戏地图选择界面如图所示：

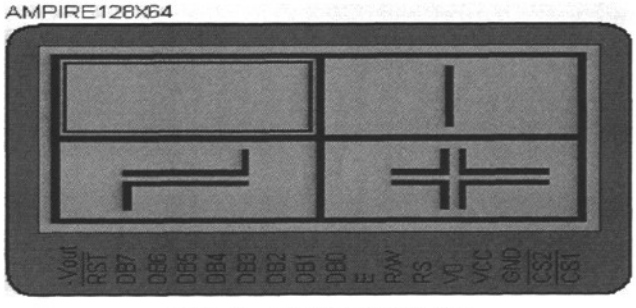

图 2.3　游戏地图选择界面

Fig. 2.3　Game Map Selection Interface

游戏界面划分成游戏区与信息区。游戏开始时，游戏区中央显示有一个长为3个单位的蛇，豆子以及选择的地图。信息区显示一个蛇图片。游戏准备运行界面如图所示：

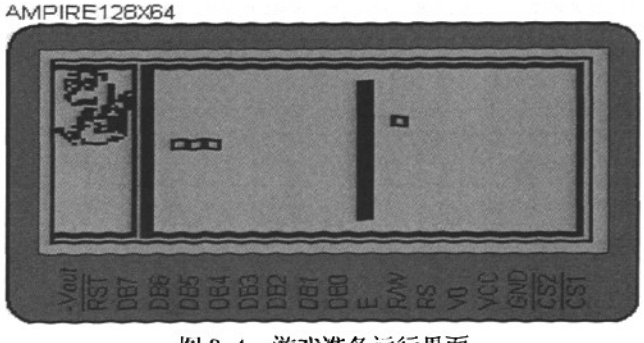

图 2.4　游戏准备运行界面

Fig. 2.4　Game Ready Interface

游戏时游戏区由键盘控制蛇的运动方向吃豆子。信息区显示"COME ON!"及蛇图片。游戏运行界面如图所示：

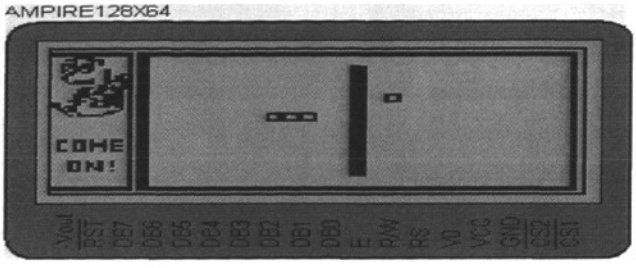

图 2.5　游戏运行界面

Fig. 2.5　Game Running Interface

游戏暂停时，游戏区蛇的运动停止。信息区显示"EXIT YES NO"。可以选择退出。游戏暂停界面如图所示：

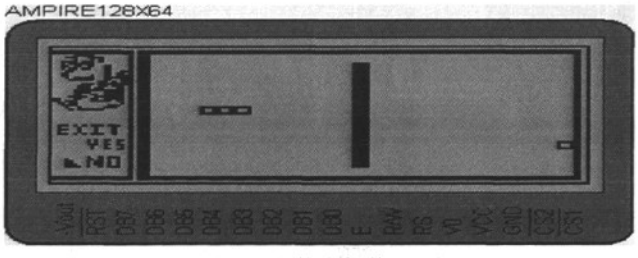

图 2.6　游戏暂停界面

Fig. 2.6　Game Pause Interface

游戏通关时，蛇的运动停止，全屏显示"Congratulation!"及礼花背景。游戏通关界面如图所示：

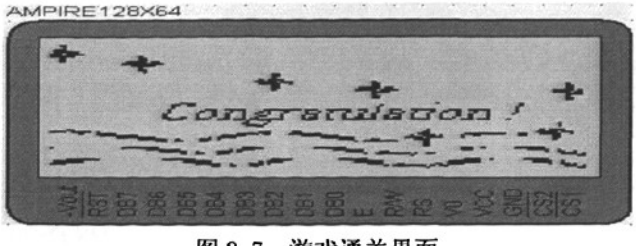

图 2.7　游戏通关界面

Fig. 2.7　Game Clear Interface

游戏结束时，全屏显示"GAME OVER!"。游戏结束界面如图所示：

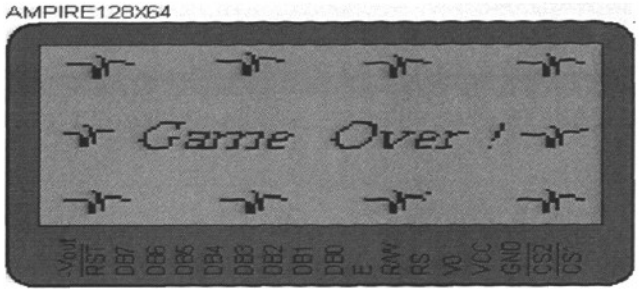

图 2.8　游戏结束界面

Fig. 2.8　Game Over Interface

2.3　游戏处理

上电以后，初始化游戏。根据玩家选择的游戏地图，初始化地图，并且在游戏区域固定位置出现蛇，蛇的长度为三个单位。在随机位置出现第一个豆子。

蛇的移动：使蛇按照方向键的方向移动。

开始游戏时，蛇头和蛇尾已经固定，按开始键游戏开始，蛇只能向左或者向右转，不能向后。

刷新豆子：豆子被吃掉后在随机位置出现豆子。豆子不能出现在和蛇身或者障碍物重合，否则重新刷新豆子。

吃豆子：蛇头吃到豆子，蛇身变长一格（在蛇头加）。

分数、等级和蛇身运动速度：游戏等级最高是六级，默认等级从 1 开始，吃 1 个豆子增加 10 分。第一级吃 10 个豆子升到第二级，第二级吃 20 个豆子升级到第三级，以此类推，第六级吃 60 个豆子，即可通关。随着等级的增加，蛇运动速度也逐渐加快

判定死亡：当蛇头碰到屏幕边缘，碰到障碍物，或者碰到蛇自己的身体时，蛇死亡，游戏结束。

2.4　背景音处理块

游戏通关，播放游戏通关背景音。

游戏结束，播放游戏结束背景音。

游戏升级，播放游戏升级背景音。

吃到豆子，播放吃豆子背景音。

错误提示，播放无效按键背景音。

2.5 键盘控制块

启动/暂停控制：点击开始键进入地图选择界面，再点开始键进入游戏界面；游戏中，使用该键，实现暂停/开始控制。

方向控制：用上、下、左、右键选择地图。用上、下、左、右键控制蛇头运动方向。游戏中暂停时，上、下键选择"EXIT YES NO"，左右键无效。

声音控制：用静音键控制背景音是否打开。

3 系统架构设计

3.1 软件开发环境

在本次开发中，采用了专门用于 MCS-51 系列单片机软件开发的 C51 语言，这种语言与普通 C 语言相同，并提供了针对单片机的常量定义、库函数等等。C 是一种源于编写 UNIX 操作系统的语言，它是一种结构化语言，可产生紧凑代码。由于汇编语言程序的可读性和可移植性都较差，用汇编语言编写单片机应用系统程序的周期长，且调试和排错也比较困难。而一般效率高的高级语言难以实现汇编语言对于计算机硬件直接进行操作（如对内存地址的操作移位操作等的功能）而 C 语言既具有一般高级语言的特点，又能直接对计算机的硬件进行操作，并且采用 C 语言编写的程序能够很容易地在不同类型的计算机之间进行移植[3]，因此许多以前只能采用汇编语言来解决的问题现在可以改用 C 语言来解决。

开发环境选择了 Keil uVision2 编译调试，硬件仿真软件 Proteus 7.1 仿真运行。这样做的目的是节省成本和缩短开发调试时间。Keil 软件是目前最流行开发 MCS-51 系列单片机的软件，提供了包括 C 编译器、宏汇编、连接器、库管理和一个功能强大的仿真调试器等在内的完整开发方案，通过一个集成开发环境（uVision）将这些部份组合在一起，全 Windows 界面。另外重要的一点，只要看一下编译后生成的汇编代码，就能体会到 Keil C51 生成的目标代码效率非常之高，多数语句生成的汇编代码很紧凑，容易理解[4]。在开发大型软件时更能体现高级语言的优势。Proteus 软件是英国 Labcenter 公司开发的电路分析与实物仿真软件。它运行于 Windows 操作系统上，可以绘制电路原理图，仿真、分析各种模拟器件和集成电路，支持主流单片机系统和多种外围芯片的仿真，提供软件调试功能，支持第三方的软件编译和调试环境，如 Keil C51 uVision2[5]。

3.2 硬件结构设计

针对第二章的需求分析，系统采用的硬件设备主要包括 51 系列单片机，LCD1602，LCD12864，按键，扬声器。

51 系列单片机采用了飞利浦（PHILIPS）公司生产的型号为 P87C51RD2 的低功耗高性能 CMOS 型 8 位单片机。选择该型号单片机的原因是，它内置了 64K bytes 的 OTP 只读程序存储器(ROM)和 1K bytes 的随机存取数据存储器(RAM)，能满足游戏对存储空间的要求。另外配备了 32 个可编程的 I/O 端口，3 个 16 位定时器/计数器，一个 7 中断源 4 优先级嵌套中断结构，一个全双工串行通信口，片内震荡器及时钟电路[6]。系统利用 51 单片机作为总的控制驱动单元，两个 LCD 分别显示游戏信息和游戏界面。LCD1602

显示游戏相关信息，包括游戏运行时间，游戏等级，游戏得分，静音图标；LCD12864显示游戏运行界面。按键分为六个，包括控制上下左右蛇运动方向的四键，游戏开始\暂停键，游戏静音键。扬声器用于发出背景音。依据上述设计的总体硬件电路图如下图所示。

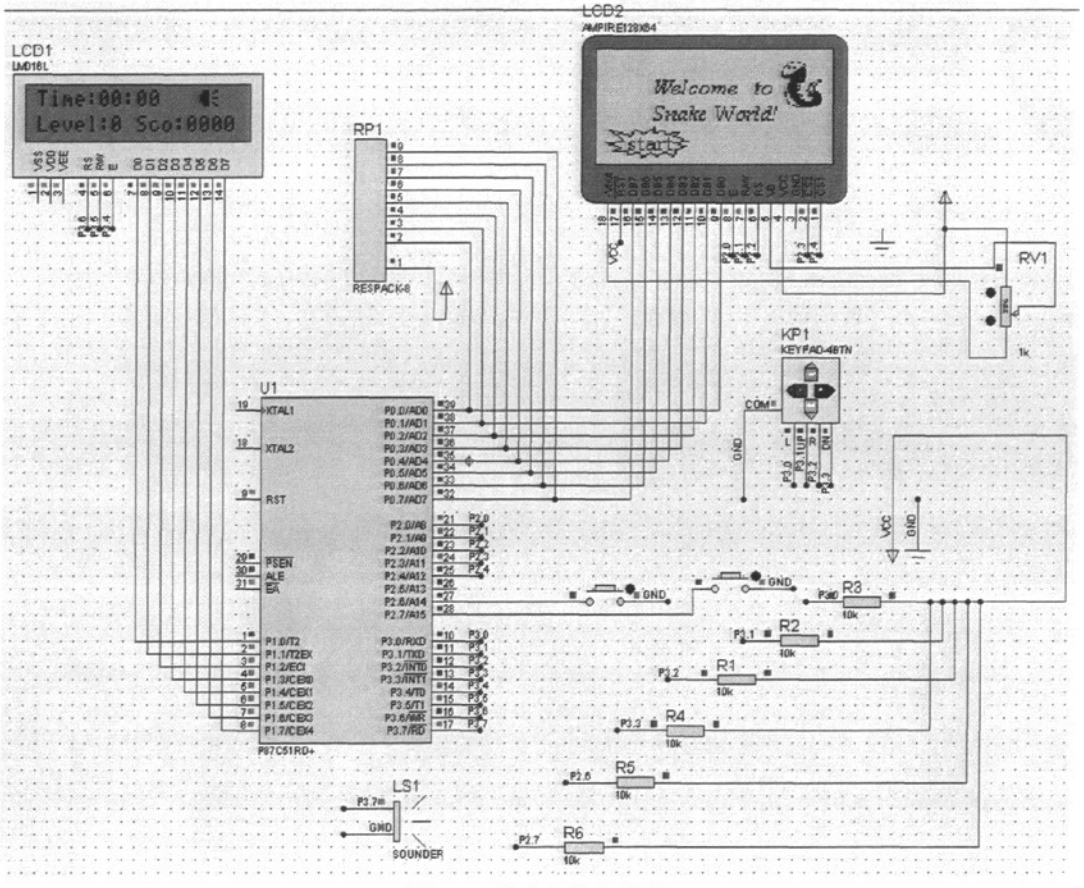

图 3.1　总体硬件电路图

Fig. 3.1　Total Hardware Circuit Diagram

其主要运行原理是使用单片机的I/O口驱动LCD，向LCD的数据口写数据或指令，使LCD显示相应的游戏信息和游戏界面。单片机采用查询方式扫描键盘，当有键按下的时候，单片机读取键值，按照按键的功能进入不同的游戏状态。背景音利用定时器0和定时器1的中断，查询方式产生。蛇的运动和游戏时钟则是采用定时器2中断产生的，当满足一定条件时通过单片机I/O口送到LCD上进行显示[7]。详细实现将在软件详细设

计中介绍。

3.3 软件结构设计

针对第二章的需求分析，将贪吃蛇游戏按照功能划分成如下几个模块：

（1）主函数模块

主函数是整个程序运行的一个缩影，是一个无限循环的程序。完成的操作包括初始化一系列硬件软件，利用定时中断完成对游戏时钟的控制，调用其他模块完成按键处理，蛇运动与游戏的处理。

（2） 按键模块

按键模块分为按键检测模块和按键处理模块两部分。

按键检测模块初始化按键对应的管脚，利用查询方式检测具体是哪个按键被按下，将该按键对应的变量值改变。

按键处理模块针对不同状态下，对按键做不同的处理，具体划分成七个状态，对应七个函数，分别是游戏欢迎状态，游戏选择地图状态，游戏等待开始状态，游戏中状态，游戏中暂停状态，游戏通关状态，游戏结束状态。

（3） LCD1602 显示模块

该模块通过获取相关变量的当前值，负责将当前的游戏运行时间，游戏得分，游戏等级，静音图标显示在 LCD1602 显示屏上。

（4） LCD12864 显示模块

该模块负责显示游戏界面。针对七种状态（游戏欢迎状态，游戏选择地图状态，游戏等待开始状态，游戏中状态，游戏中暂停状态，游戏通关状态，游戏结束状态），在 LCD12864 上时时刷新显示不同的画面。

（5） 蛇运动控制模块

该模块负责对蛇的动作与状态的控制，包括游戏开始时对蛇的初始化，蛇移动的处理，对存储地图的数组中的一位做置 1，置 0 或查询操作，蛇吃食物后的处理，根据当前蛇头坐标及蛇头运动方向，更新下一步蛇头的坐标。

（6） 游戏控制模块

该模块负责游戏的控制，具体包括在屏幕上随机生成新豆子；吃到豆子以后分数增加，等级增加；对游戏时钟的控制。

（7） 背景音模块

该模块定义不同的背景音，在不同游戏状态发出对应背景音。背景音包括无效按键背景音，吃豆子背景音，升级背景音，游戏通关背景音，游戏结束背景音。

4 软件详细设计与实现

4.1 游戏设计思想

按照需求分析，游戏设计重点要解决游戏界面的显示和蛇身运动处理的问题。

游戏采用两个LCD屏幕显示游戏界面。LCD1602分两行显示游戏信息。第一行显示游戏运行时间，静音图标；第二行显示游戏等级，游戏得分；LCD12864显示游戏运行界面。设定LCD12864以4x4大小为一个单位点阵，定义贪吃蛇的每一节蛇身大小为一个单位点阵，豆子的大小，游戏地图中障碍物的大小和一节蛇身大小相同，也是一个单位点阵。LCD12864要显示各种游戏状态对应的界面，所以定义数组来存储各种界面对应的位图数据。定义函数实现在LCD12864的某一坐标位置增加一个单位点阵或删除一个单位点阵的操作，这样通过调用该函数实现增加蛇头，消除蛇尾，实现在游戏界面上蛇运动。

关于蛇身运动处理，游戏采用定时器2的中断方式定时，当1 s的定时时间到的时候，蛇向前运动一个单位点阵。蛇在运动的过程中，主要需要处理如下几个问题：

（1） 运动处理。根据用户按键的键值蛇身进行柔体传动。所谓柔体传动，指贪吃蛇运动的时候并不是整条蛇向一个方向运动的，而是在每个时钟到来时，由蛇头带动每个点阵的方向都向下一个点阵传播，然后自己向新的方向运动一步。运动后，下一个点阵由于得到了上一个点阵的方向，就按照此方向同样地运动一步。所以，它会马上填补上一个点阵的位置，如此类推。实际上在设计贪吃蛇的时候，只需要把蛇尾的那个点阵去掉，然后在蛇头的新方向上放一个点阵就可以了。因此定义一个函数用于更新点的坐标，只要知道蛇头或蛇尾的坐标和运动方向，就可以调用该函数更新蛇头或蛇尾的坐标，同时调用相关显示函数，实现蛇运动处理。

蛇头的方向可以通过按键捕获实时获得。而蛇尾的方向获得要困难得多。为节省空间，定义一个无符号char型一维数组aucSnake[54]按位存储蛇头方向信息。蛇身运动期间需要在该数组中记录下每次蛇身移动一步，蛇头的运动方向。定义两个变量分别代表蛇头和蛇尾，每次存储蛇头运动方向后，代表蛇头的变量加一，以便在下一位置继续存储蛇头运动方向；代表蛇尾的变量则指向数组的开始位置，通过该变量读取数组中蛇尾的运动方向（因为存在该数组中蛇头的运动方向，随着蛇头坐标的移动，存在数组开始位置的运动方向即是蛇尾的运动方向），通过读取的蛇尾运动方向便可以更新蛇尾坐标。

（2） 吃到豆子的处理。蛇头坐标与豆子坐标相等，则吃到豆子。如果吃到豆子，在蛇头位置增加一个单位点阵，并更新豆子的坐标。

（3） 放置新的豆子。通过更新豆子的坐标实现放置新的豆子。放置豆子的过程中，还需要判断新豆子的坐标是否与蛇身或障碍物坐标重叠，如果重叠，则需要重新放置和判断，直到新的豆子不与蛇身或障碍物坐标重叠为止。

（4） 死亡处理。蛇运动的过程中，若蛇头碰到墙，障碍物或自己的身体，则游戏结束。在游戏中，任何时候按下开始/暂停键，则游戏暂停，可以进一步选择是否退出游戏。其中游戏暂停的处理是通过关闭定时器 2 实现的[8-10]。

4.2 贪吃蛇游戏中的各种状态

贪吃蛇操作过程中有多种状态，采用宏定义，使各种状态一目了然，思路清晰。同时在编写代码时，采用匈牙利命名规则命名宏，变量，函数名，方便阅读修改[11]。

贪吃蛇游戏中的各种状态如下：

按键状态：检测具体按到哪个按键，将每个键对应定义一个常量，具体定义如下：

```
#define KEY_NULL   (INT8U)0          //无按键状态
#define KEY_UP     (INT8U)1          //按上键
#define KEY_DOWN   (INT8U)4          //按下键
#define KEY_LEFT   (INT8U)2          //按左键
#define KEY_RIGHT (INT8U)3           //按右键
#define KEY_START (INT8U)5           //按开始键
#define KEY_SOUND (INT8U)6           //按静音键
```

蛇头的运动状态除了上下左右运动外，还有一个反方向运动的处理。比如当前蛇头运动状态向右，这时点击左键，蛇头不会向相反方向调转。此时定义的状态就是蛇头反方向运动。

```
#define UP         (INT8U)0          //蛇头向上
#define DOWN       (INT8U)3          //蛇头向下
#define LEFT       (INT8U)1          //蛇头向左
#define RIGHT      (INT8U)2          //蛇头向右
#define OPPOSITE     (INT8U)4        //蛇头反方向运动
```

游戏的七种状态：游戏欢迎状态，游戏选择地图状态，游戏等待开始状态，游戏中状态，游戏中暂停状态，游戏通关状态，游戏结束状态。

```
#define GAME_STAT_WELCOME    (INT8U)0     //七种游戏状态
#define GAME_STAT_CHANGEMAP (INT8U)1
```

```
#define GAME_STAT_WAIT      (INT8U)2
#define GAME_STAT_DONE      (INT8U)3
#define GAME_STAT_DEAD      (INT8U)4
#define GAME_STAT_GAMING    (INT8U)5
#define GAME_STAT_PAUSE     (INT8U)6
```

选择四种地图，声音静音与否的状态，五种游戏背景音：

```
#define MAP_0           (INT8U)0            //四个地图
#define MAP_1           (INT8U)1
#define MAP_2           (INT8U)2
#define MAP_3           (INT8U)3
#define MAP_UNCHOOSE    (INT8U)4            //未选择地图

#define SOUND_ON        (INT8U)1            //声音开关
#define SOUND_OFF       (INT8U)0

#define SOUND_UP_LEVEL (INT8U)1             //游戏升级背景音
#define SOUND_EAT       (INT8U)2            //蛇吃豆子背景音
#define SOUND_ERR       (INT8U)3            //游戏错误背景音
#define SOUND_DONE      (INT8U)4            //游戏通关背景音
#define SOUND_DEAD      (INT8U)5            //游戏结束背景音
```

4.3 主函数模块详细设计与实现

主函数模块主要包括两个函数：void main(void)和 void T2_ISR(void)

main( )函数是整个程序的入口地址，程序从此函数开始执行，将各个模块串联起来，实现游戏的各种功能。首先完成软件硬件的初始化任务，然后进入一个无限循环，反复检测有无按键、处理按键、处理蛇运动、处理时钟节拍这四件事情[7]。

代码如下：

```
void main(void)
{
    GameInit();                             //硬件和全局变量初始化
    T2Init();                               //定时器 2 初始化
    while(1)
```

```
    {
        if(GS_ucKeyValue == KEY_NULL)
        {
            ISO_CLOSE();                    //关闭中断
            Key_Get();                      //获取按键
            ISO_OPEN();                     //打开中断
        }
        if(GS_ucKeyValue != KEY_NULL)
        {
            KeyManage_Task();
        }
        if(ucSnakeFlag == GO && GS_ucGameState == GAME_STAT_GAMING)
        {
            SnakeControl_SnakeTask();
            ucSnakeFlag = STOP;
        }
        if(ucTimeFlag == GO && GS_ucGameState == GAME_STAT_GAMING)
        {
            GameControl_TimeTask();
            ucTimeFlag = STOP;
        }
    }
}
```

T2_ISR( )函数是一个中断服务子程序，利用 T2 定时器工作在方式 1 状态，设置 TL2,TH2 初值为十进制数 15536，单片机的晶振频率是 12MHz，因此 T2 定时器每记满 50000 个数进入中断子程序需要 50ms，定时 1s 只要进入 20 次中断子程序即可，游戏运行时间的时钟节拍就是利用 T2 定时器计算出来的。其中有两个标志位初始状态都是 STOP，ucSnakeFlag 是蛇移动速度的标志位，ucTimeFlag 是时钟节拍标志位。定义了一个变量 uiT2Cou 计数，uiT2Cou 与 20 模除为 0，修改时钟节拍标志位 ucTimeFlag 为 GO，屏幕时间显示加 1；同理，蛇根据等级不同，每秒移动速度不同，修改蛇运动标志位 ucSnakeFlag 为 GO。只有在游戏运行中状态且这两个标志位修改为 GO 时，才能进行蛇运动的控制和游戏时钟的控制。蛇运动的快慢则取决于数组 auiSpeed 里面的值。

代码如下：

```
void   T2_ISR(void) interrupt 5 using 2          //T/C2 中断服务程序入口
```

```
{
    ISO_CLOSE();                                    //关闭中断
    TF2 = 0;                                        //T2 定时器的中断标志位置 0
    ++uiT2Cou;                                      //计数器+1
    if(uiT2Cou%(auiSpeed[ucGameLevel-1]) == 0)      //根据等级调整蛇的速度
    {
        ucSnakeFlag = GO;
    }
    if(uiT2Cou%20 == 0)                             //时间每 1 秒走一次
    {
        ucTimeFlag = GO;
    }
    if(uiT2Cou >= 60480)                            //计数器置 0
    {
        uiT2Cou = 0;
    }
    ISO_OPEN();                                     //打开中断
}
```

4.4 按键模块详细设计与实现

该模块包括两部分：按键检测和按键处理。

以下是按键相关的全局变量：

```
INT8U   data GS_ucKeyValue;             //存放当前按键
INT8U   data GS_ucGameState;            //游戏的状态标志
INT8U   data GS_ucMapNumber;            //游戏用到的地图号
INT8U   data GS_ucSoundSwitch;          //声音开关标志
INT8U   data GS_ucDirection;            //蛇头当前的方向
INT8U   data ucCheck = CONT;        //记录玩家选择继续(CONT)或者退出(EXIT)
```

4.4.1 按键检测模块

按键检测模块提供接口函数 Key_Init( )和 Key_Get( )。六个按键对应单片机的六个管脚，Key_Init( )用来初始化各管脚，将各管脚置 1，即各管脚为高电平。Key_Get( )将检

测这些管脚的电平值，当某一管脚置 0，即该管脚为低电平时，将管脚对应的宏定义的值赋给全局变量 GS_ucKeyValue。模块流程图如下图所示。

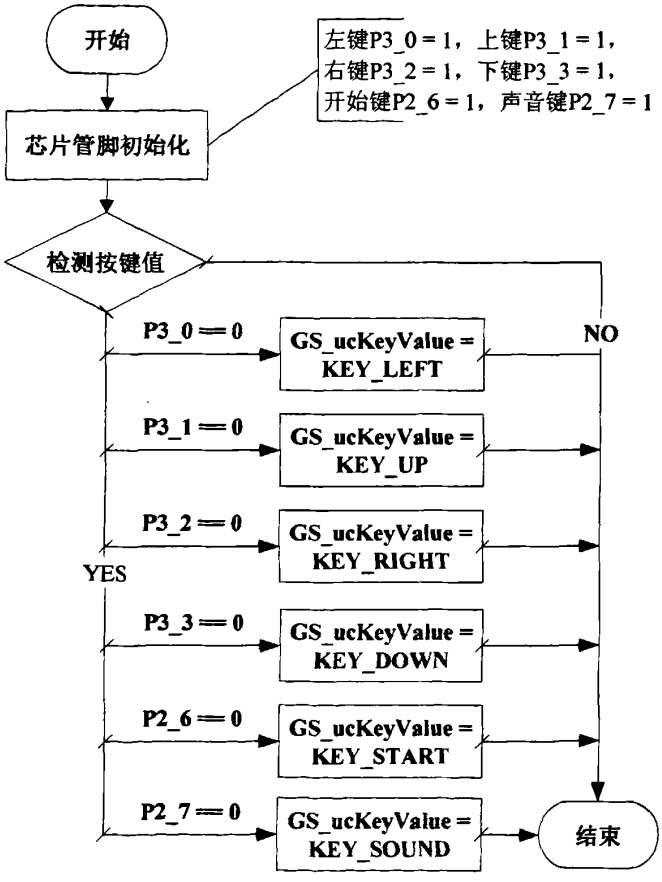

图 4.1　按键检测模块流程图
Fig. 4.1　Key Detection Module Flow Chart

4.4.2　按键处理模块

按键处理模块，主要实现对按键的处理，游戏有六个按键，分别是“方向上键”，“方向下键”，“方向左键”，“方向右键”，“声音开关键”，“确定键（开始/暂停键）”，在游戏的不同阶段，按键有不同的功能，比如方向键，在游戏过程中，起到控制蛇运动方向的作用，而在游戏开始前，选择地图的时候，它又用来选择地图，所以要根据游戏状态的不同，对按键做不同的处理。游戏分为七种状态：

（1）　欢迎游戏状态（GAME_STAT_WELCOME）

点击确定键进入选择地图画面，点击声音开关键选择是否静音，其他按键无效。

（2） 选择地图状态（GAME_STAT_CHANGEMAP）

点击方向键选择游戏地图，点击确定键确定，点击声音开关键选择是否静音。

（3） 游戏等待状态（GAME_STAT_WAIT）

点击确定键进入游戏状态，点击声音开关键选择是否静音，其他按键无效。

（4） 游戏进行状态（GAME_STAT_GAMING）

点击方向键控制蛇运动，点击确定键开始/暂停游戏，点击声音开关键选择是否静音。

（5） 游戏暂停状态（GAME_STAT_PAUSE）

点击方向上下键选择，点击确定键决定退出还是返回游戏，点击声音开关键选择是否静音，其他按键无效。

（6） 游戏通关状态（GAME_STAT_DONE）

点击确定键返回欢迎画面，点击声音开关键选择是否静音，其他按键无效。

（7） 游戏结束状态（GAME_STAT_DEAD）

第六种和第七种状态的按键处理是一样的，另外，在各种状态下都可以进行的对声音开关键的处理，因此将这个模块的功能分为七个函数，分别对他们进行处理。函数原型声明如下：

```
void KeyManage_Welcome(void)          //欢迎界面时的按键处理
void KeyManage_ChangeMap(void)        //选择地图状态时的按键处理
void KeyManage_Wait(void)             //游戏等待状态时的按键处理
void KeyManage_Gaming(void)           //游戏进行时的按键处理
void KeyManage_Pause(void)            //游戏暂停状态时的按键处理
void KeyManage_Done(void)             //游戏通关时的按键处理
void KeyManage_Dead(void)             //游戏结束时的按键处理
```

另外定义了函数 void KeyManage_Task(void)来调用上述定义的七个函数进行按键的处理。

下面四个函数是在上面七种游戏状态中都要调用的。

```
void KeyManage_SoundManage(void)          //游戏声音开关键的处理
void KeyManage_ChangeGameStat(INT8U stat) //修改游戏的状态
void KeyManage_PutKeyValue(void)          //将按键标志置 0
void GameEnd(INT8U stat)                  //游戏通关或结束处理
```

GameEnd( )函数根据传入的参数判断当前的游戏状态，分两种状态：游戏结束时在

大屏上显示游戏结束界面，播放游戏结束的背景音，并修改游戏状态的全局变量；游戏通关时在大屏上显示游戏通关界面，播放游戏通关的背景音，并修改游戏状态的全局变量。为了更好的说明该模块功能，下面列出了按键处理模块的流程图和状态迁移图。

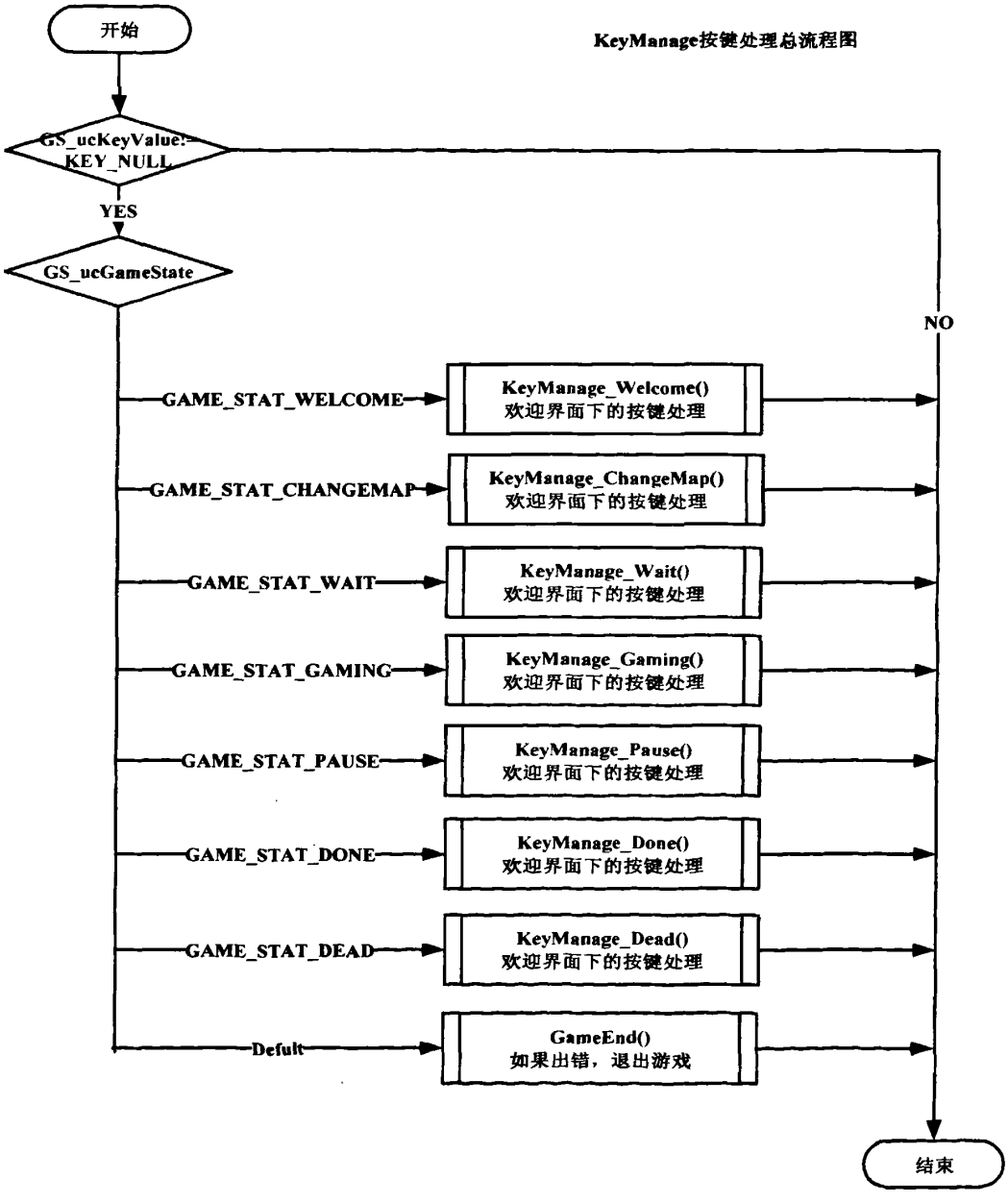

图 4.2　按键处理模块流程图
Fig. 4.2　Key Processing Module Flow Chart

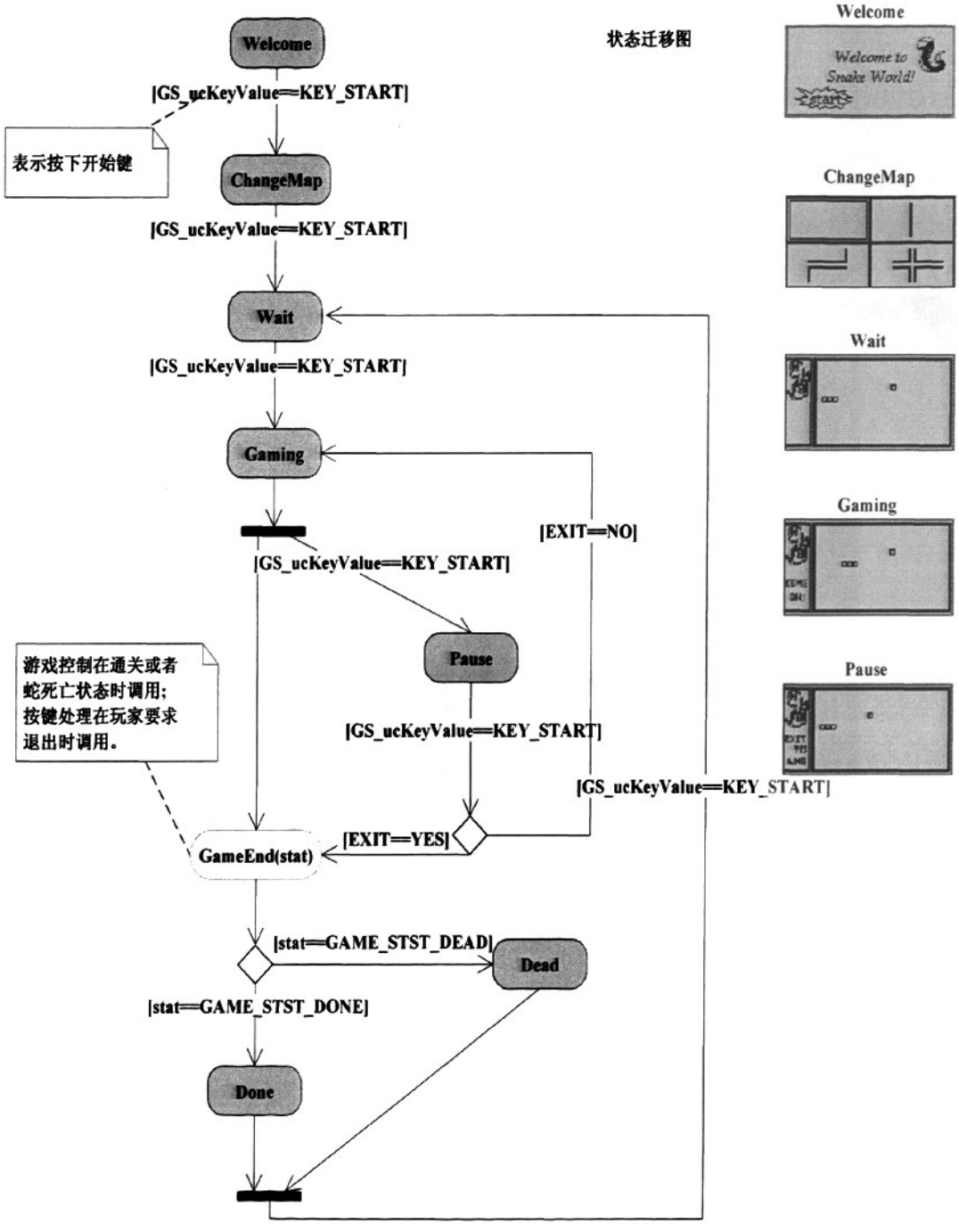

图 4.3　按键处理模块状态迁移图

Fig. 4.3　Key Processing Module State Transition Diagram

4.5 LCD1602 显示模块详细设计与实现

LCD1602（以后简称小屏）为两行显示，每行 16 个字符，5 伏供电，通过 8 条数据线和 3 条控制线与微控制器相连，通过送入的数据和指令进行工作。LCD 控制器内包含有显示数据 RAM(DDRAM)，字符发生器 ROM(CGROM)，字符发生器 RAM(CGRAM)。DDRAM 是用来寄存待显示的代码；CGROM 用来提供用户所需字符库；CGRAM 可以存储 8 个用户自定义的字符图形 RAM。字符显示位置，第一行从 0x00 到 0x0F；第二行从 0x40 到 0x4F。不能显示汉字，内置含 128 个字符的 ASCII 字符集字库，只有并行接口，无串行接口[12]。LCD1602 有 14 个引脚，其定义如下：

1 脚 VSS：电源+5V

2 脚 VDD：地

3 脚 VEE：液晶显示对比度调节端

4 脚 RS：寄存器选择。RS=0 时，选择指令寄存器；RS=1 时，选择数据寄存器

5 脚 R/W：读写信号线。R/W=1 时，读操作；R/W=0 时，写操作

6 脚 E：显示板控制使能端

7～14 脚 D0-D7：双向三态 I/O 线。其中 D7 用于读/写忙检测[13]

根据上述管脚功能介绍，硬件管脚相关定义如下：

```
sbit   P3_6 = P3^6;
sbit   P3_5 = P3^5;
sbit   P3_4 = P3^4;
sbit   P1_7 = P1^7;

#define    LCD1602_RS          P3_6
#define    LCD1602_RW          P3_5
#define    LCD1602_EN          P3_4
#define    LCD1602_BUSY        P1_7

#define FIRST_LINE    0x80          //第一行第一个点阵位置的十六进制表示
#define SECOND_LINE 0xC0            //第二行第一个点阵位置的十六进制表示
```

LCD1602 显示模块在进行游戏初始化时初始化 LCD1602，并显示静态游戏信息。具体功能如下：

刷新游戏运行时间

刷新游戏当前等级

刷新游戏所得分数

刷新游戏音效开关小喇叭图形

游戏初始化以后，游戏时间，游戏等级，游戏分数均为 0，，背景音图标显示为打开，具体格式如图所示。

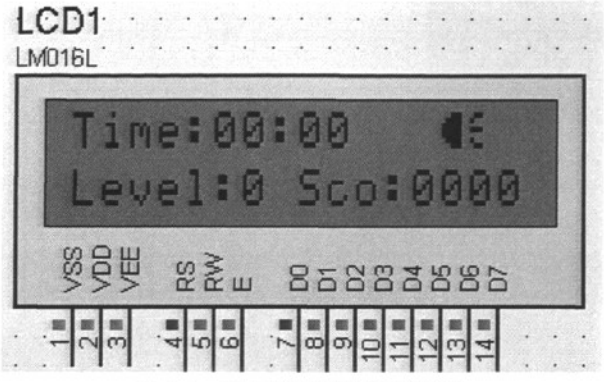

图 4.4　初始化显示的游戏信息

Fig. 4.4　Display Initialization Game Information

在游戏运行过程中，根据需要刷新游戏运行时间、游戏当前等级、游戏分数和背景音静音信息。游戏初始化时初始化小屏并显示静态游戏信息。

这些游戏信息的字符都可以通过调用 CGROM 中提供给用户的字符库显示出来，所以可以直接定义字符数组。只有背景音提示图标需要用户自定义。LCD1602 是按照 5x7 点阵，横向取模，字节正序的方式显示一个字符。定义两个字模数组 Close[8]和 Open[8]，将数据写入 CGRAM 中的相应位置，之后便可以像调用 CGROM 中本身提供的用户的字符库一样调用这两个用户自定义的字符了。以下列出了对应的字符数组定义。

```
static INT8U code Time[10]={ 'T', 'i', 'm', 'e', ':', '0', '0', ':', '0', '0' };
static INT8U code Level[7]={ 'L', 'e', 'v', 'e', 'l', ':', '0' };
static INT8U code Sco[8]={ 'S', 'c', 'o', ':', '0', '0', '0', '0' };
static INT8U code Close[8]={ 0x07,0x0F,0x1F,0x1F,0x1F,0x0F,0x07,0x00 };
static INT8U code Open[8]={ 0x0C,0x10,0x00,0x1C,0x00,0x10,0x0C,0x00 };
```

具体功能函数如下：

LCD1602 显示模块的内部接口函数如下所示，为底层的驱动函数，对 LCD1602 进行基本操作，外部函数通过调用下面的内部函数实现相应功能[5]。

```
void InitLCD(void)                          //初始化 LCD 屏幕
void WriteCmd(INT8U command, INT8U flag)//通过内部指令向 LCD 写入指令
void WriteData(INT8U data, INT8U flag)      //通过内部指令向 LCD 写入数据
void SetXY(INT8U x, INT8U y)                //定位到屏上由坐标 X、Y 指定的位置
void CheckBusy(void)                        //检查 LCD 屏是否状态为忙，忙则等待
void ShowDefaultInfo(void)                  //显示静态游戏信息，即游戏运行时不变的信息
void ShowSound(void)                        //显示表示背景音的喇叭图形
```

这里重点说明 ShowSound( )函数的实现。代码如下：

```
void ShowSound(void)
{
    INT8U data i=0;
    WriteCmd(0x40,1);                       //发出指令向 CGRAM 写入数据
  for(;i<8;i++)
  {
      WriteData(Close[i],0);
  }
  for(i=0;i<8;i++)
  {
      WriteData(Open[i],0);
  }                                         //以上向CGRAM写入16字节数据
  WriteCmd(0x80,1);                         //向 DDRAM 写入数据
  SetXY(13,0);
  WriteData(0,0);                           //将写入 CGRAM 的数据显示在 LCD 上
  SetXY(14,0);
  WriteData(1,0);
}
```

该函数首先调用 WriteCmd( )设置向 CGRAM 写数据，从地址 0 的位置开始。然后利用两个 for 循环将静音标志的字符写入到 CGRAM 中地址 0 和 1 的位置。继续调用

WriteCmd( )函数设置为相DDRAM写数据，定位要显示的位置写入数据，完成了用户自定义字符的显示过程。

外部接口函数如下：

//初始化游戏信息，提供给GameInit模块调用

INT8U LCDInfo_InitGameInfo(void)

函数首先调用函数InitLCD( )初始化与LCD1602硬件相关的各管脚信息，然后通过WriteCmd( )函数向屏幕写入指令，设置屏幕属性；设置16x2显示，5x7点阵，8位数据接口；不显示光标，光标不闪烁；清屏；当读写一个字符后地址指针加1，光标加1。然后调用ShowDefaultInfo( )显示静态游戏信息，包括时间，分数，游戏等级。

/刷新LCD1602的相应位置，显示要显示的时间

INT8U LCDInfo_TimeDisplay(const INT8U minute, const INT8U second)

该函数用于显示要显示的参数。函数传入的参数是分钟和秒。参数合法就显示当前时间。首先在小屏上调用函数SetXY( )确定位置，然后通过除以10和模除10计算出分钟数和秒数的个位十位，进行写数据，显示到小屏上。

//刷新LCD1602的相应位置，显示要显示的等级

INT8U LCDInfo_LevelDisplay(const INT8U level)

该函数用于显示要显示的等级。函数传入的参数是等级。参数合法就显示当前等级首先在小屏上调用函数SetXY( )确定位置，然后进行写数据，显示到小屏上。

//刷新LCD1602的相应位置，显示要显示的分数

INT8U LCDInfo_ScoresDisplay(const INT16U scores)

该函数用于显示要显示的分数。函数传入的参数是游戏分数。参数合法就显示当前游戏分数（游戏分数大于等于0，小于等于2100即为合法）。首先在小屏上调用函数SetXY( )确定位置，然后通过除和模除计算出游戏分数的千位、百位、十位、个位。进行写数据，显示到小屏上。

//刷新LCD1602的相应位置，显示表示背景音开或关的喇叭图形

INT8U LCDInfo_SoundSwitch(const INT8U SoundSwitch)

该函数用于显示要显示的背景音状态图标。函数传入参数SoundSwitch，判断该变

量状态，然后在小屏上调用函数 SetXY( )确定位置，进行写数据，显示到小屏上背景音开或关的喇叭图形。函数流程图如下图所示。

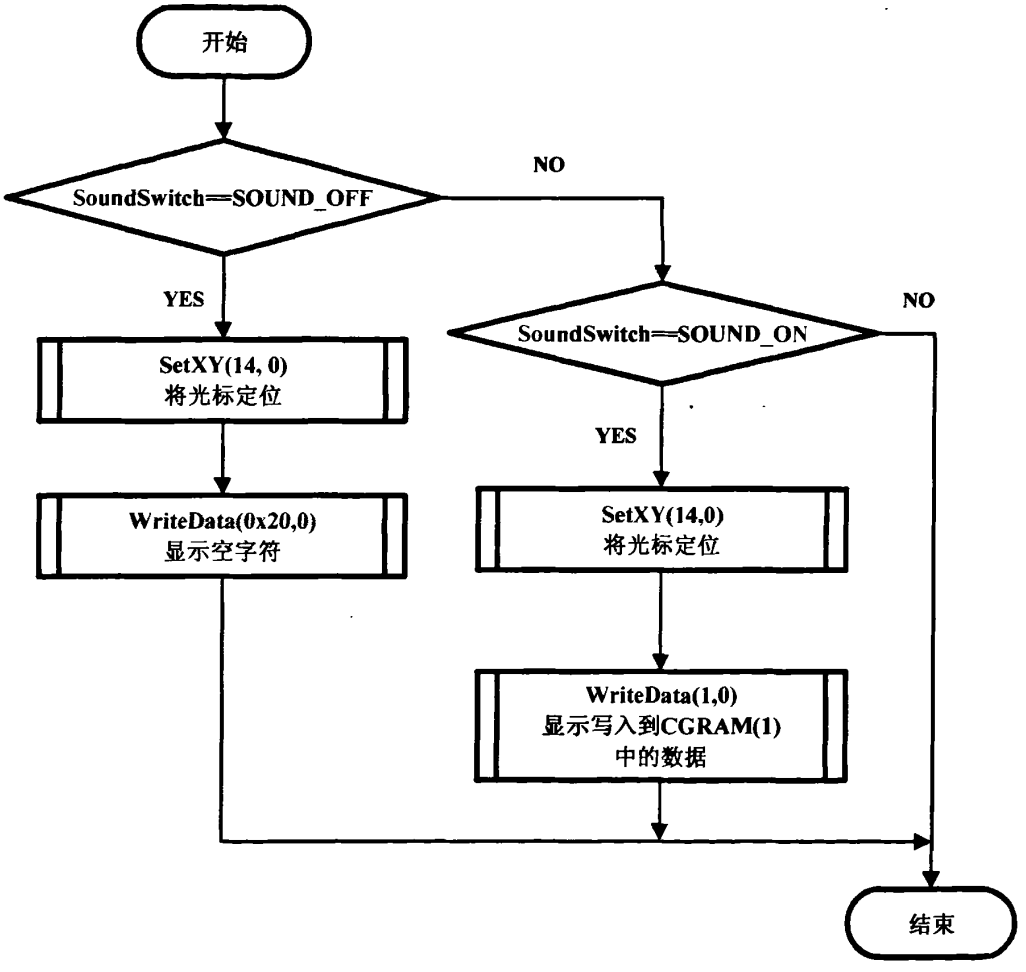

图 4.5 LCDInfo_SoundSwitch 函数流程图
Fig. 4.5 LCDInfo_SoundSwitch Function Flow Chart

4.6 LCD12864 显示模块详细设计与实现

LCD12864（以后简称大屏）功能强大，5V 电压驱动，带背光，内置 8192 个 16x16 点阵、显示容量为 128x64，内部由 A，B 两屏左右分布组成。LCD12864 有 18 个引脚，其定义如下：

1 脚$\overline{CS1}$：片选 A 屏，低电平有效

2脚$\overline{CS2}$：片选B屏，低电平有效

3脚GND：地

4脚VCC：电源

5脚VO：液晶显示对比度调节端

6脚RS：寄存器选择。RS=0时，选择指令寄存器；RS=1时，选择数据寄存器

7脚R/W：读写信号线。R/W=1时，读操作；R/W=0时，写操作

8脚E：显示板控制使能端

9～16脚DB0-DB7：双向三态I/O线

17脚$\overline{RST}$：复位端，低电平有效

18脚Vout：对比度调节供电

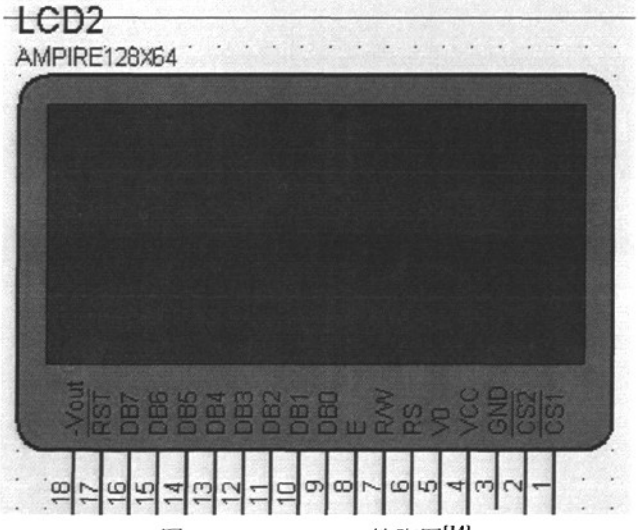

图4.6　LCD12864管脚图[14]

Fig. 4.6　LCD12864 Pin Diagram[14]

LCD12864并行方式时，数据线DB0～DB7用来传送数据和命令。CSl和CS2为液晶显示器的左右半屏的选择端口。液晶的亮度可以使用改变VO的输入电压来改变[14]。

LCD12864显示的点阵方式是128x64。横向0—127共128个点阵，分为左右两屏，各占0-63共64个点阵；纵向0—63共64个点阵，每8个纵向点阵（8个点阵可以看成8位，8位为1字节）构成一页，将屏幕分成8页，并按照纵向取模，字节倒序的方式显示一字节内容[14]。由于将4x4点阵定义为单位点阵的大小，并定义单位点阵为豆子的

大小，因此将屏幕按 4x4 比例从 128x64 缩小为 32x16（即将 4x4 点阵看成 LCD12864 上的一个最小点阵）。将游戏界面划分成游戏区与信息区，其中游戏区为 25x16（其中去掉上下边框，游戏区的实际范围是 25x14），信息区为 7x16。本模块提供了 LCD12864 的初始化，显示游戏界面的函数，显示选框的函数、显示游戏状态的函数（游戏运行或暂停）和显示图形的函数。具体功能如下：

（1） 对 LCD12864 的初始化

（2） 在 LCD12864 上显示图形（三角或正方形）

（3） 在 LCD12864 上显示选框

（4） 在 LCD12864 上显示各种界面

根据上述管脚功能介绍，硬件管脚相关定义如下：

```
sbit   P2_2 = P2^2;
sbit   P2_1 = P2^1;
sbit   P2_0 = P2^0;
sbit   P2_4 = P2^4;
sbit   P2_3 = P2^3;

#define    MAINLCD_RS          P2_2;          //硬件对应的管脚
#define    MAINLCD_RW          P2_1;
#define    MAINLCD_EN          P2_0;
#define    MAINLCD_CSA         P2_4;
#define    MAINLCD_CSB         P2_3;
```

利用 P0 口对应的八个管脚与 LCD12864 的八个数据管脚相连接。利用此八个管脚输入不同的高低电平，进行对 LCD12864 的各种控制，包括打开关闭屏幕，设置页，设置行等。具体定义如下：

```
#define    MAINLCD_DATA        P0
#define    SET_ON              0x3F          //LCD12864 打开
#define    SET_OFF             0x3E          //LCD12864 关闭
#define    SET_PAGE            0xb8          //设置页数
#define    SET_COL             0x40          //设置行数
#define    SINGLE_SCREEN_COL   64            //单屏幕最大列数
```

MAP_NUMBER为游戏中全屏显示的界面个数，一共五个，具体是：游戏欢迎界面，选择游戏地图，游戏运行界面，游戏结束界面，游戏通关界面。GAMESEG_START_COL和GAMESEG_START_ROW设定了游戏区的起始列与起始行。FIGURE_NUMBER为存储图形（三角，空心正方形和实心正方形）位图数组的行下标，OPTION _NUMBER为存储选择Yes和No箭头在大屏上位置的数组的下标。SCREEN_SIZE为存储全屏显示游戏五幅位图数组的列下标。具体定义如下：

```
#define   MAP_NUMBER          5          //游戏相关信息
#define   GAMESEG_START_COL   6
#define   GAMESEG_START_ROW   1
#define   FIGURE_NUMBER       3
#define   OPTION_NUMBER       2
#define   SCREEN_SIZE         1024       //屏幕的大小（字节）
```

OPT_MAP_NUMBER为存储游戏信息区三角形箭头显示位置的数组的下标，同时又代表游戏运行时的游戏信息区对应的两种状态：游戏运行与游戏暂停的信息区的显示。其他宏定义包括对信息区的起始页，起始列，信息区的列大小，页大小的定义。其中信息区起始页被定义为4，因为在游戏运行或暂停时信息区范围内的0—3页显示静态的蛇图像，只有信息区范围内的4—7页会因为游戏的运行或暂停而显示不同的图像，所以只要修改信息区范围内4—7页的显示内容即可。具体定义如下：

```
//可修改的信息区的信息
#define   OPT_MAP_NUMBER      2
#define   MODIFY_START_X      1          //信息区的起始列
#define   MODIFY_START_Y      4          //信息区的起始页
#define   MODIFY_SIZE_X       19         //信息区的列大小
#define   MODIFY_SIZE_Y       4          //信息区的页大小
```

选择游戏地图时，具体选择哪个地图的选框的长度与宽度。具体定义如下：

```
#define   CHOOSE_LENTH        4          //选框的长度
#define   CHOOSE_WIDE         59         //选框的宽度
```

按照4x4的点阵大小定义的基本图形。三角形用作“选择是否退出游戏”的箭头标记；空心正方形用来表示蛇身和豆子；实心正方形表示障碍物（组成地图的基本点）。

具体定义如下：

```
#define FIGURE_STRIANGLE (INT8U)0          //三角形
#define FIGURE_SQUARE    (INT8U)1          //空心正方形
#define FIGURE_BALK      (INT8U)2          //实心正方形
```

另外定义了一些数组，用于存储显示各种图形的位图。具体说明如下：

const INT16U code GS_auiMap[4][MAX_X]

以 4x4 大小为单位点阵，按此比例缩小（即将 4x4 点阵看成 LCD12864 上的一个最小点阵），游戏区变为 25x16（此处不忽略边框，因为边框被看做地图的一部分）。该数组按照 25x16 的比例存储四幅游戏地图的位图。其中 MAX_X 的值是 25。

static const INT8U code aucFigure[FIGURE_NUMBER][4]

存储用于显示三角形，空心正方形和实心正方形的位图。利用三角形可以表示“选择是否退出游戏”的箭头标记；空心正方形用来表示蛇身和豆子；实心正方形表示障碍物（组成地图的基本点）。

static const Point code astOptPos[OPTION_NUMBER]

存储在信息区指向 YES 和 NO 的坐标地址。

static const Point code astMapOpt[4]

存储在选择游戏地图界面，四幅地图在大屏上的起始地址。

INT8U code aucFullScreen[MAP_NUMBER][SCREEN_SIZE]

存储全屏显示五种游戏状态的位图。

INT8U aucInfoMes[OPT_MAP_NUMBER][MODIFY_SIZE_X*MODIFY_SIZE_Y]

存储游戏运行和游戏暂停时，信息区显示信息的位图。游戏运行时显示”COME ON!”；游戏暂停时显示”EXIT YES NO”。

static INT8U * pAllMap[MAP_NUMBER+OPT_MAP_NUMBER]

是一个指针数组，存储五个游戏状态对应的全屏图片位图的指针和游戏运行，暂停时信息区位图的指针。

LCD12864 显示模块的内部接口函数如下所示，为底层驱动函数，对 LCD12864 进行基本操作，外部函数通过调用下面的内部函数实现相应功能。

void MWriteCmd(INT8U ucCmd); //向大屏进行写命令

void MWriteData(INT8U ucData); //向大屏进行写数据

void ReadByte(INT8U ucPage,INT8U ucCol,INT8U * ucData);//根据传入的前两个参数 ucPage 和 ucCol，读出大屏对应页，对应列上一个字节的信息，并存储到指针 ucData 所指向的变量里。

INT8U DisplayByte(INT8U ucPage,INT8U ucCol,INT8U ucData);//显示一个字节的数据，即纵向八个点阵。

INT8U DisplayFigure(Point ptPos,INT8U ucType,INT8U ucMode);//在指定坐标位置，显示或清除一个4x4的图形(显示或清除的图形包括三角形,空心正方形和实心正方形)。

INT8U DisplayGsMap(void); //在游戏区显示选择的对应地图

外部接口函数如下：

//初始化 LCD

void MainLCD_Init(void)

该函数初始化大屏，设置全局变量 GS_ucMapNumber 为未选择地图，并在大屏上显示游戏欢迎界面，初始化指针数组 pAllMap[7]的值为五个游戏状态对应的全屏图片位图的指针和游戏运行，暂停时信息区位图的指针。。

//在屏幕上全屏显示游戏的界面

INT8U MainLCD_DisplayFullScreen(INT8U ucPic)

通过游戏界面的编号 ucPic，全屏显示游戏某种状态对应的界面。通过调用 Display Byte( )函数读取 aucFullScreen[5][1024]数组里存储的位图信息完成界面显示。如果玩家选择了游戏地图，同时显示当前的游戏地图。

//在信息区显示游戏的选择界面和加油界面

INT8U MainLCD_ModifyInfoSection (INT8U ucPic)

通过游戏界面的编号 ucPic，在信息区显示游戏是否退出的选择界面或加油界面。通过调用 DisplayByte( )函数读取指针数组 pAllMap[7]完成显示。

//在信息区退出的界面，询问是否退出

INT8U MainLCD_ChooseOpt(INT8U ucDirect)

根据按键的方向 ucDirect 调用 DisplayFigure( )函数，在信息区相应的新坐标位置显

示三角形箭头选项，旧坐标位置清除三角形箭头选项。流程图如下所示。

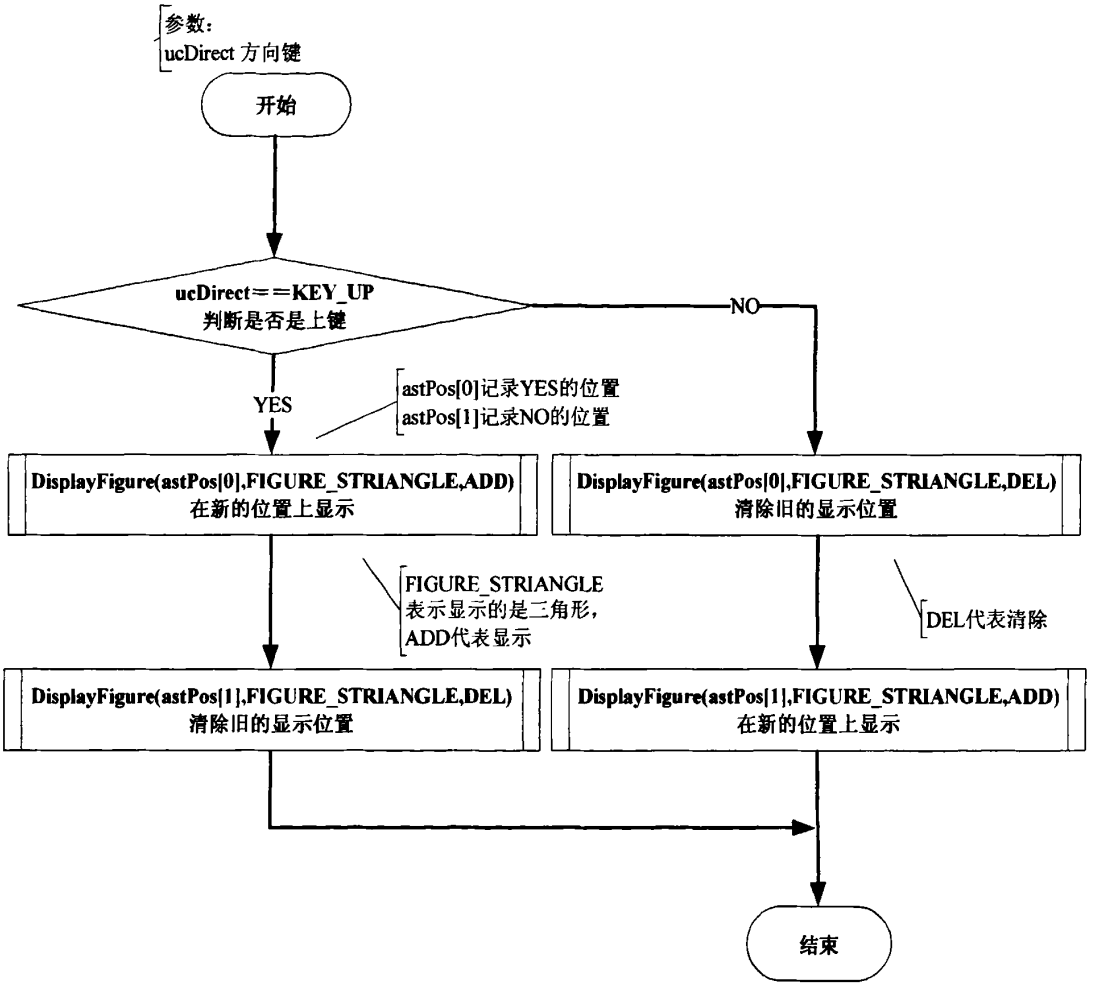

图 4.7　MainLCD_ChooseOpt 函数流程图
Fig. 4.7　MainLCD_ChooseOpt Function Flow Chart

//在选择的游戏地图上画边框，代表选中此地图

INT8U MainLCD_ChooseMap(INT8U ucOld,INT8U ucNew)

ucOld 是要删除的地图编号，ucNew 是要选择的地图编号。通过掩码数组进行逻辑运算，删除和显示上下两行和左右两行的边框。流程图如下所示。

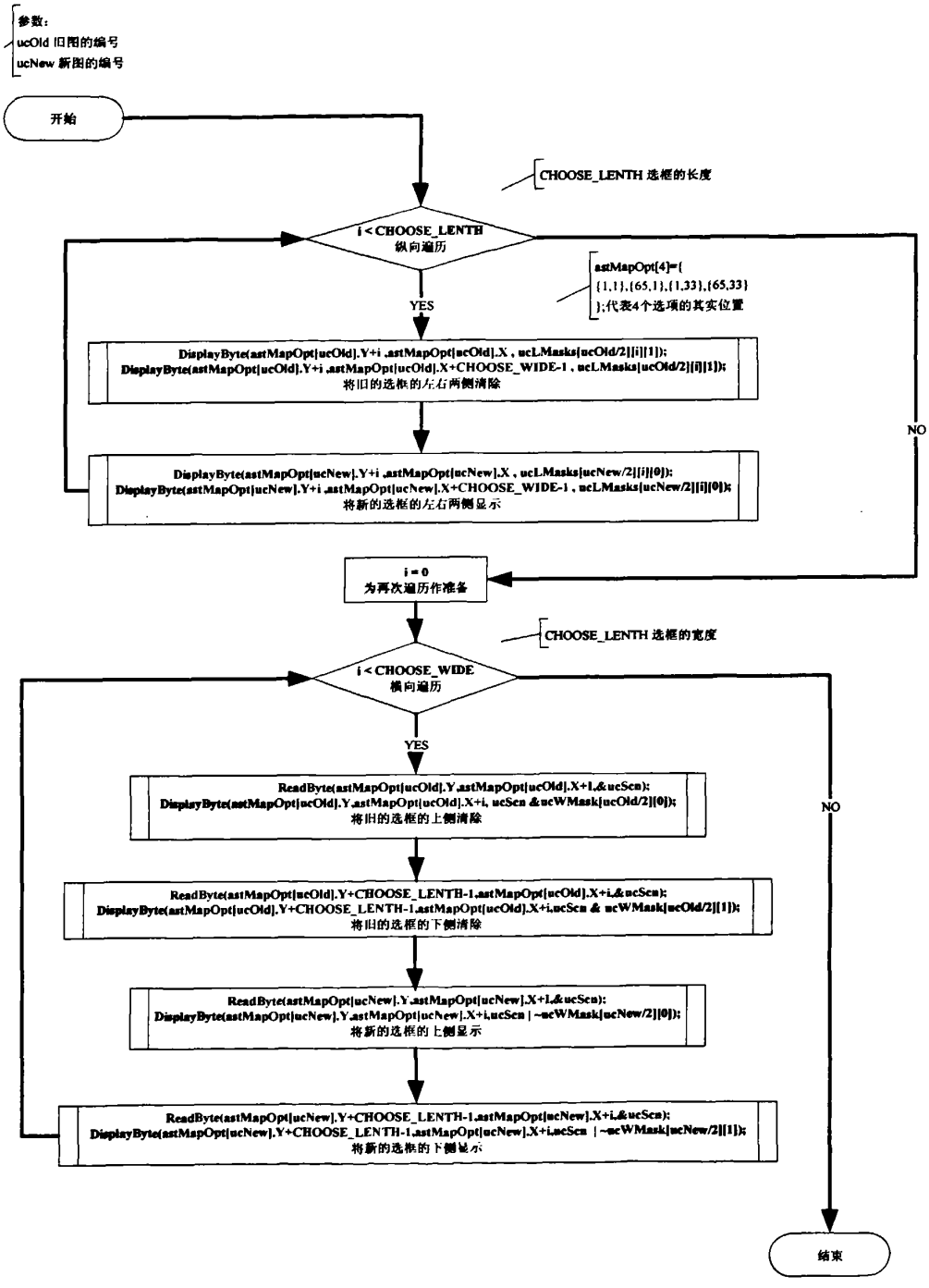

图 4.8　MainLCD_ChooseMap 函数流程图

Fig. 4.8　MainLCD_ChooseMap Function Flow Chart

4.7 蛇运动控制模块详细设计与实现

本模块主要进行蛇运动的控制，触发各种事件的响应。

主要开发内容包括蛇的方向、蛇的速度、蛇的位置的控制，吃食物、撞边界、撞自己判断并调用其它模块进行相应操作。根据不同的等级，使贪吃蛇以不同的速度前进。具体功能如下：

（1） 贪吃蛇按一定方向移动

（2） 贪吃蛇根据按键改变方向

（3） 贪吃蛇判断吃食物

（4） 贪吃蛇判断撞边界

（5） 贪吃蛇判断撞自己

（6） 修改游戏地图标记表

相关变量的定义：

```
const INT8U    code ucEns = 0x03;              //掩码
const INT16U code uiEns = 0x8000;
INT16U data uiGameScores;                      //游戏分数
INT8U    data ucGameLevel;                     //游戏等级
#define MAX_X   25                             //地图的极限坐标
#define MAX_Y   14
#define SNAKE_LENGTH   216                     //蛇的极限长度为213(为被4整除取216)
#define ARR_LENGTH     (SNAKE_LENGTH/4)        //蛇数组的大小
typedef struct {
    INT8U x;
    INT8U y;
} Point;                                       //定义点坐标的结构体
Point   data stHead;                           //蛇头坐标
Point   data stEnd;                            //蛇尾坐标
Point   data stFood;                           //豆子坐标
INT16U data uiT;                               //蛇（蛇头）运动轨迹数组的下标
INT16U data uiE;                               //蛇（蛇尾）运动轨迹数组的下标
INT16U data uiTime;                            //游戏时间
INT16U auiMap[MAX_X];                          //地图数组
```

INT8U aucSnake[ARR_LENGTH]; //蛇运动轨迹数组

//初始化与蛇有关的信息

void SnakeControl_Init(void)

负责蛇的初始化。具体包括初始化豆子的位置，蛇头蛇尾坐标，游戏分数，游戏时间初始化为0，游戏等级初始化为1。根据四副地图中具体选择了哪副地图，初始化地图数组，同时初始化蛇运动轨迹数组，并在大屏上初始化蛇[15][16]。

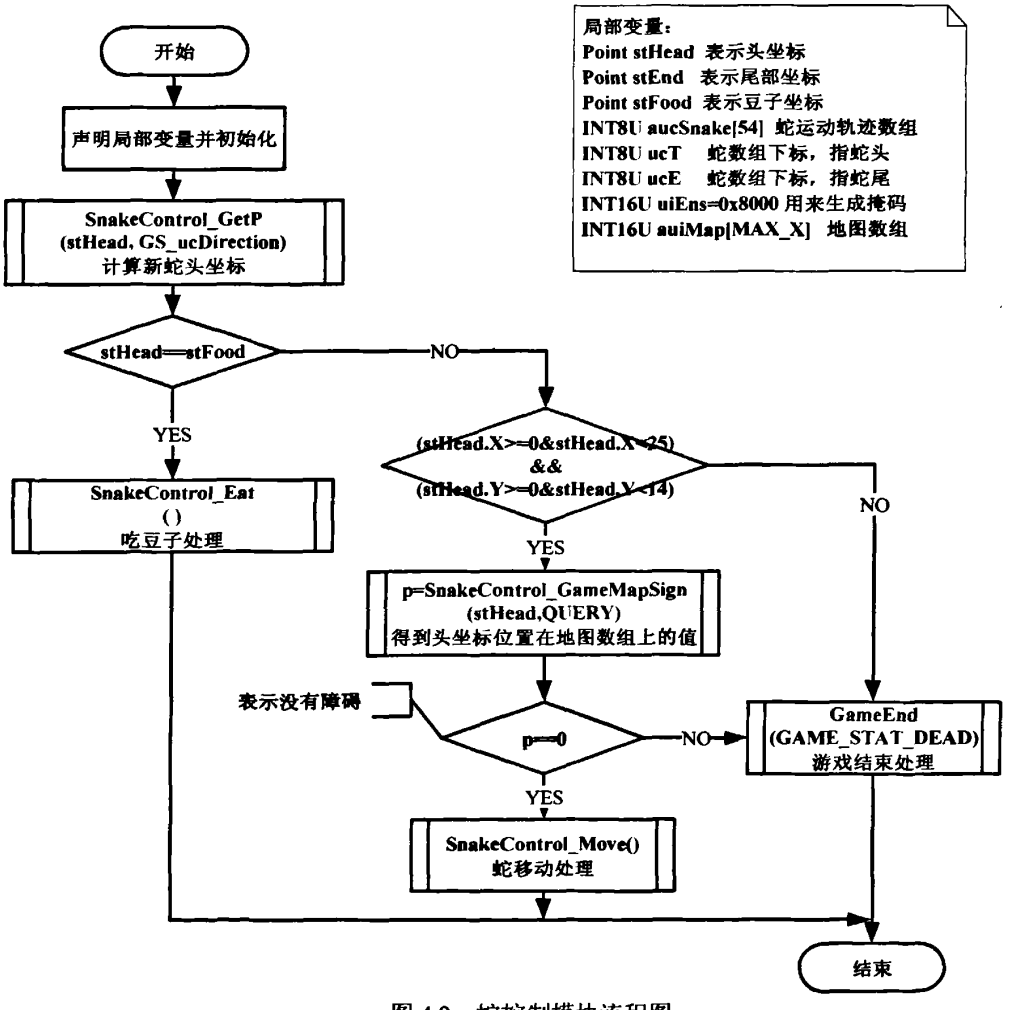

图 4.9 蛇控制模块流程图

Fig. 4.9 Snake Control Module Flow Chart

//记录游戏地图中蛇、障碍的位置，蛇移动时查询、修改具体点的位值

INT16U SnakeControl_GameMapSign(Point stP,INT8U ucMode)

给地图数组的一位做置 1，置 0，查询的操作。该函数根据参数不同，可以实现对地图数组不同操作。如果对地图数组的一位做增加操作，则利用掩码查询地图数组相应位未被使用，并将其置 1；如果对地图数组的一位做删除操作，则利用掩码查询地图数组相应位已经被使用，并将其置 0；如果对地图数组的一位做查询操作，则利用掩码查询地图数组相应位，将其坐标值返回。流程图如下图所示。

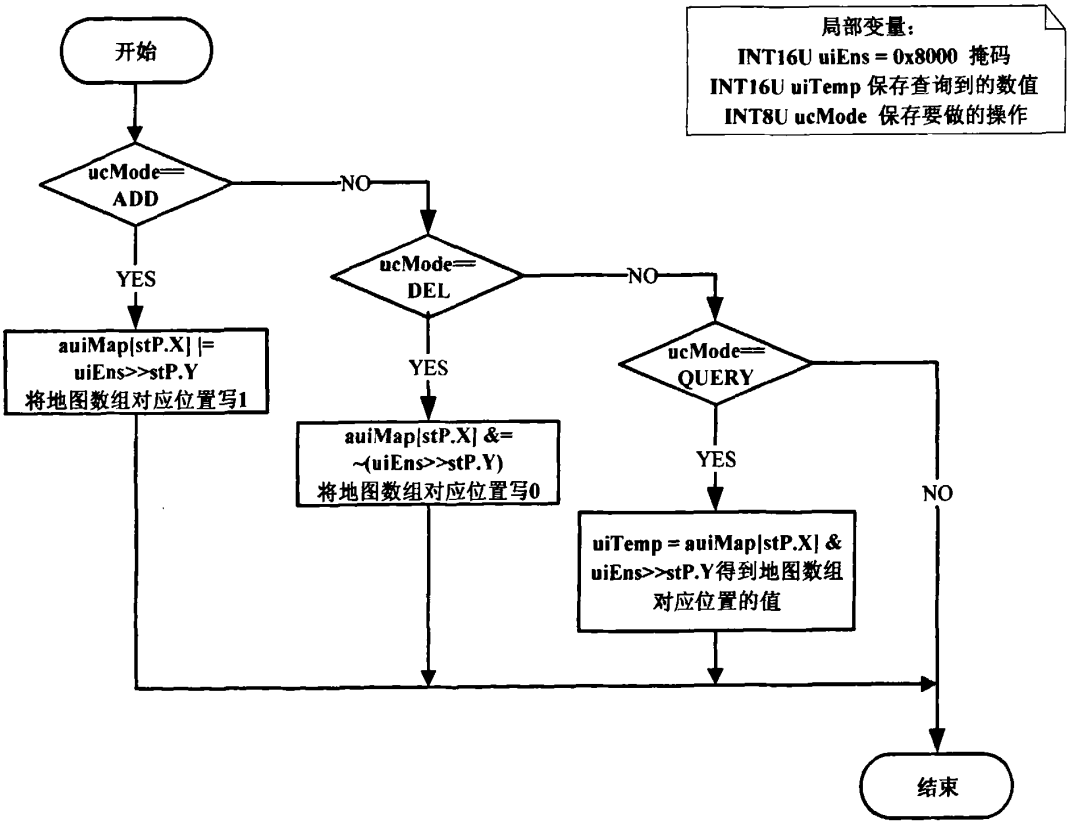

图 4.10　SnakeControl_GameMapSign 函数流程图

Fig. 4.10　SnakeControl_GameMapSign Function Flow Chart

//根据运动方向，更新坐标

void SnakeControl_GetP(Point* stP, INT8U ucDirection)

通过给定点坐标和一个方向，更新坐标。传递给函数的两个参数，一个是蛇头（或蛇尾）当前坐标，一个是蛇头（或蛇尾）当前运动方向。通过判断当前蛇头（或蛇尾）

运动方向（上，下，左，右），增加x坐标或y坐标，如果当前蛇头（或蛇尾）运动方向不是上述四种，则游戏结束。

图 4.11　SnakeControl_Move 函数流程图
Fig. 4.11　SnakeControl_Move Function Flow Chart

//蛇移动的处理
INT8U SnakeControl_Move(void)
蛇移动的处理。蛇移动一步，就调用 SnakeControl_GameMapSign( )函数将蛇尾坐标

在地图数组上删除；调用 MainLCD_DisplaySquare( )函数删除蛇尾在大屏上显示的点阵。通过蛇（蛇尾）运动轨迹数组的下标 uiE 和掩码 ucEns，从蛇运动轨迹数组 aucSnake[54]计算出蛇尾方向。首先左移蛇尾下标 uiE 若干位，通过掩码筛出蛇尾方向是哪两位，通过与操作从 aucSnake[54]数组里提出蛇尾的方向，再右移与刚才左移位数相同的次数，就完成了蛇尾方向的计算。代码如下：

```
ucTempDirection = (  aucSnake[uiE>>2] & (ucEns<<(2*(uiE%4))) )>>(2*(uiE%4));
```

根据蛇尾方向调用函数 SnakeControl_GetP( )更新蛇尾坐标，将新蛇头位置在地图数组上置 1，并在大屏上显示新蛇头，流程图如上图所示。

```
//蛇吃食物后的处理
INT8U SnakeControl_Eat(void)
```

蛇吃食物后的处理。蛇吃食物以后，调用 GameControl_Score( )函数增加游戏分数，当游戏分数增加到 2100（即吃了 210 个豆子以后），游戏通关，退出该函数。否则，调用 SnakeControl_GameMapSign( )函数，给地图数组上写 1，表示蛇身增加一位，调用函数 Sound_Play( )播放吃豆子背景音，调用 GameControl_SetFood( )随机生成新豆子。

```
//处理蛇的各种状态
void SnakeControl_SnakeTask(void)
```

该函数针对在游戏运行时蛇的各种状态，调用各种函数处理这些状态。

首先详细讲解在游戏运行时如何将蛇头的方向存在一个一维数组 aucSnake[54]中。

代码如下所示：

```
//通过方向写蛇运动数组
aucSnake[(uiT%SNAKE_LENGTH)>>2] &= ~(ucEns<<(2*(uiT%4)));
aucSnake[(uiT%SNAKE_LENGTH)>>2] |= (GS_ucDirection<<(2*(uiT%4)));
uiT = (uiT+1)%SNAKE_LENGTH;
```

[17]

定义了一个数组 INT8U aucSnake[ARR_LENGTH]，用来存储蛇的运动轨迹。蛇头的运动轨迹和蛇身蛇尾的运动轨迹一致，所以该数组只存储蛇头的当前运动轨迹即可。蛇的运动轨迹包括上、下、左、右四种状态，程序中定义了四个宏：

```
#define U (INT8U)0                    //存在蛇数组中表示蛇头和蛇尾的方向
#define D (INT8U)3
#define L (INT8U)1
#define R (INT8U)2
```

0、3、1、2 这四个数对应四个方向，即上、下、左、右四种状态，转换成 16 进制

数位为0x00、0x11、0x01、0x10。为节省空间，程序设计成按位存储，即一个数组元素为一个字节8位，2位存储一个蛇的运动轨迹，一个数组元素可以存储四个蛇的运动轨迹。这样算来，定义该数组大小为54，就可以同时存储216蛇头运动轨迹。

代码第一句的作用是将数组下标元素的对应两位清零，然后记录当前蛇头的方向到这两位，将uiT加1。这样就完成了将蛇头的方向存在一维数组aucSnake[54]中。

接下来调用函数KeyManage_PutKeyValue( )将按键标志清零，调用SnakeControl _GetP( )函数更新蛇头坐标。如果蛇头坐标与豆子坐标相等，则调用SnakeControl_Eat( )函数处理吃到豆子以后的操作。如果蛇头超出大屏边界，游戏结束；否则查看是否撞到障碍或蛇身，撞到同样游戏结束，否则调用SnakeControl_Move( )函数处理蛇移动[18-20]。

4.8 游戏控制模块详细设计与实现

游戏控制模块的主要功能是控制游戏运行时间、游戏分数、游戏级别等与小屏相关的信息，以及控制豆子在大屏的显示。具体功能如下：

（1） 在大屏空位置上随即放置豆子

（2） 更新游戏分数，并在小屏显示当前游戏分数

（3） 更新游戏等级，并在小屏显示当前游戏等级

（4） 更新游戏时间，并在小屏显示游戏的运行时间

（5） 当音效开启的时候，吃豆子和升级要产生相应的声音

具体函数讲解如下：

//随机生成豆子并显示

INT8U GameControl_SetFood (void)

在大屏空位置上，随机放置豆子，调用MainLCD_DisplaySquare( )显示新豆子，函数返回新豆子的坐标。这里详细讲解随机放置豆子的原理。LCD12864的点阵是128x64，当前我们定义4x4点阵为单位大小（即一个豆子的大小），这样屏幕的坐标被等比缩小为25x16，如前所述，去掉上下边框，实际游戏区的大小是25x14，都在就是这个范围内随机生成。我们利用srand(TL2)和rand( )函数可以生成从TL2到32767之间的随机数，而TL2是定时器2的低八位，一直变化，这样更体现了随机的可信度。

之后利用i = rand( )%25随机出x的坐标，把当前的地图数组i位置的元素同0xFFFF比较，判断豆子的x坐标值为i时，y是否有空位。如果有空位，则利用0x00FF != (auiMap[i]>>8)语句判断豆子的x坐标值为i时，高8为是否有空位。有空位则在0到7（无空位在8到13之间生成随机数）之间随机生成y坐标值j，判断i，j相交位置是否

有空位，有空位则在此位置生成豆子，反之重新随机生成 y 坐标值 j。

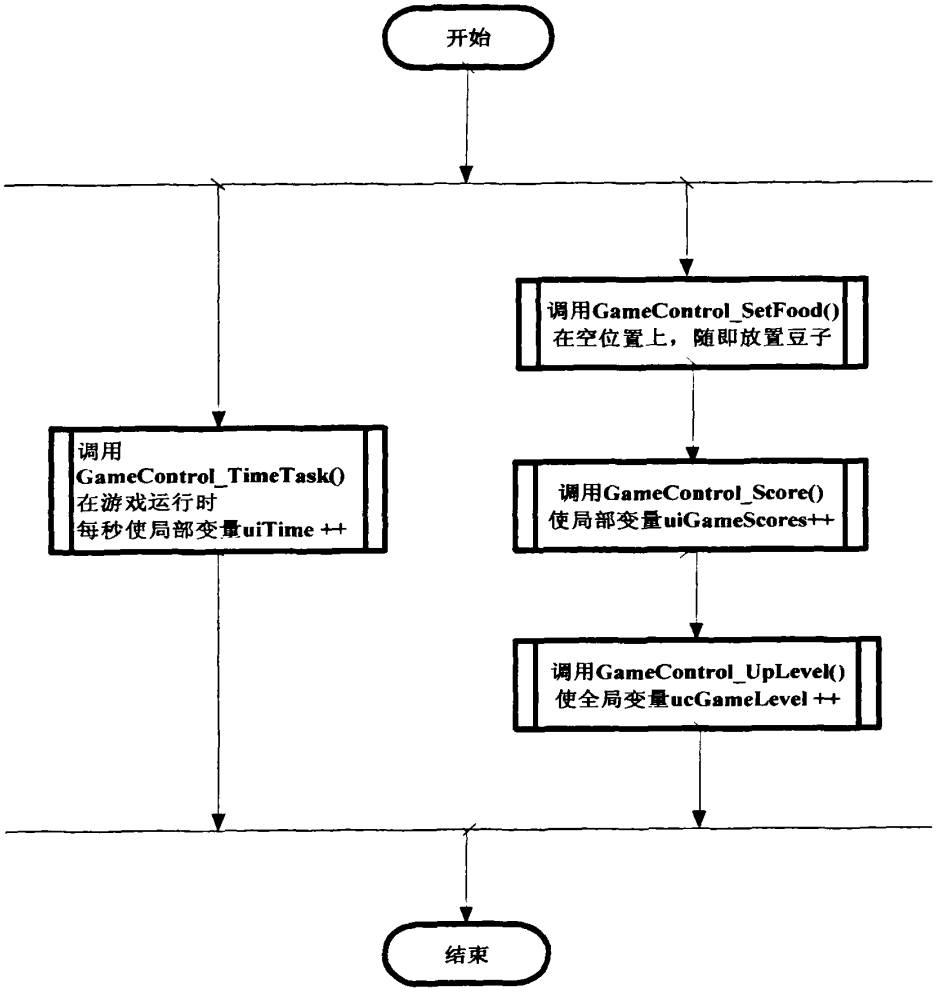

图 4.12　游戏控制模块流程图
Fig. 4.12　Game Control Module Flow Chart

//更新游戏分数

INT8U GameControl_Score (void)

更新小屏的游戏分数。每次蛇头坐标与豆子坐标重合，就更新一次调用一次该函数。函数调用一次，游戏分数 uiGameScores 加 10，并调用 LCDInfo_ScoresDisplay( )函数在小屏上刷新分数。当游戏分数 uiGameScores 为 100，300，600，1000，1500 时，GameControl

UpLevel( )函数被调用，使 ucGameLevel 加 1。

//更新游戏等级
INT8U GameControl_UpLevel (void)
更新小屏的游戏等级，并产生升级的声音。每次调用该函数，将修改游戏等级 ucGameLevel 加 1，调用 LCDInfo_ScoresDisplay( )函数在小屏上刷新分数，并调用 Sound_Play( )函数发出游戏升级背景音。

//更新游戏时间
void GameControl_TimeTask(void)
小屏时间每秒加 1。只要游戏处于运行，uiTime 每秒加 1（通过 T2 定时器实现准确定时），并将 uiTime 以 uiTime/60 : uiTime%60 的格式，更新小屏的游戏时间。

4.9 背景音模块详细设计与实现

4.9.1 单片机实现播放音乐的原理

单片机演奏音乐都是单音频率，因此单片机奏乐只需弄清楚两个概念即可，也就是音调和节拍。音调表示一个音符唱多高的频率，节拍表示一个音符唱多长的时间。

在音乐中所谓“音调”，其实就是我们常说的“音高”。在音乐中常把中央 C 上方的 A 音定为标准音高，其频率 f=440Hz。当两个声音信号的频率相差一倍时，也即 f2=2f1 时，则称 f2 比 f1 高一个倍频程，在音乐中 1（do）与$\dot{1}$，2（来）与$\dot{2}$……正好相差一个倍频程，在音乐学中称它相差一个八度音。在一个八度音内，有 12 个半音。以 1—i 八音区为例， 12 个半音是：1—#1、#1—2、2—#2、#2—3、3—4、4—#4，#4—5、5 一#5、#5—6、6—#6、#6—7、7—i。这 12 个音阶的分度基本上是以对数关系来划分的。如果我们只要知道了这十二个音符的音高，也就是其基本音调的频率，我们就可根据倍频程的关系得到其他音符基本音调的频率。

知道了一个音符的频率后，要产生音频脉冲，只要算出某一音频的脉冲（1/频率），然后将此周期除以 2，即为半周期的时间，利用定时器计时这个半周期的时间，每当计时到后就将输出脉冲的 I/O 反相，然后重复计时此半周期的时间再对 I/O 反相，就可以在 I/O 脚上得到此频率的脉冲。

对于音乐的节拍，在单片机上控制一个音符唱多长可采用循环延时的方法来实现。首先，就需要确定一个基本时长的延时程序，比如说以十六分音符的时长为基本延时时

间，那么，对于一个音符，如果它为十六分音符，则只需调用一次延时程序，如果它为八分音符，则只需调用二次延时程序，如果它为四分音符，则只需调用四次延时程序，依次类推。通过上面关于一个音符音调和节拍的确定方法，我们就可以在单片机上实现演奏音乐了[21]。

4.9.2 功能实现

该模块实现播放背景音的功能。在蛇吃到豆子、游戏升级、游戏通关、游戏失败、用户无效按键时分别提示不同的音乐。基于单片机播放音乐的理论基础，利用定时器 0 作为音符定时器，工作于方式 1，中断操作，16 位定时器；利用定时器 1 作为音长定时器，工作于方式 1，查询操作，16 位定时器。具体的实现方法为：将乐谱中的每个音符的音调及节拍变换成相应的音调参数和节拍参数，按位存储，存放在数组中，通过程序取出一个音符的相关参数，播放该音符，该音符唱完后，接着取出一个节拍的相关参数，以此类推。乐曲结束用节拍参数为 00H 来表示[22]。该模块对应流程图如下：

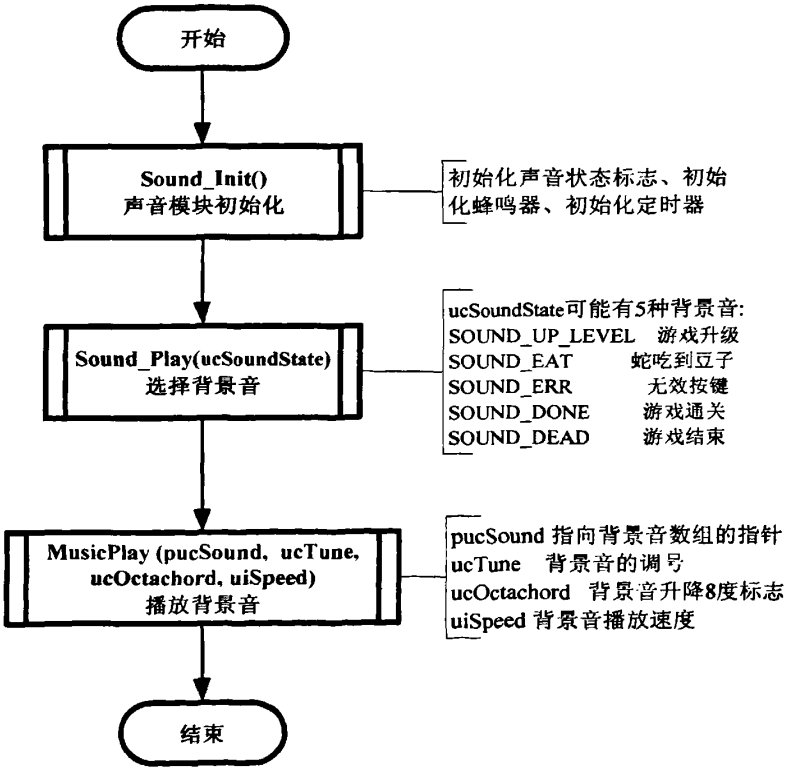

图 4.13 背景音模块流程图

Fig. 4.13 Backgroud Sound Module Flow Chart

下面列出了背景音模块用到的相关定义：

```
#define SOUND_ON      (INT8U)1                  //背景音开关
#define SOUND_OFF     (INT8U)0

#define SOUND_UP_LEVEL (INT8U)1          //5 种背景音
#define SOUND_EAT       (INT8U)2
#define SOUND_ERR       (INT8U)3
#define SOUND_DONE      (INT8U)4
#define SOUND_DEAD      (INT8U)5
```

对应五种背景音，分别定义五个一维数组存放背景音的音调与音长。

```
static INT8U code aucMusicErr[];                //无效按键背景音
static INT8U code aucMusicEat[];                //吃豆子背景音
static INT8U code aucMusicLevel[];              //游戏升级背景音
static INT8U code aucMusicDone[];               //游戏通关背景音
static INT8U code aucMusicDead[];               //游戏结束背景音

static INT16U code auiFreTab[12];               //原始频率表
static INT8U code aucSignTab[7] = { 0,2,4,5,7,9,11 };  //1~7 在频率表中的位置
static INT8U code aucLengthTab[7]= { 1,2,4,8,16,32,64 };      //音符节拍

static INT8U data ucSound_TH0,ucSound_TL0;   //音符定时器初值暂存
static INT8U data ucSound_TH1,ucSound_TL1;   //音长定时器初值暂存

sbit   sBeep = P3^7;                          //定义输出管脚，输出周期不同的方波
```

具体定义如下四个函数：

```
//背景音模块初始化
INT8U Sound_Init(void)
```

初始化时，将音效设置为打开状态，设置定时器状态与初值等。

```
//中断函数，将对应管脚周期性置反
void BeepTimer0(void) interrupt 1 using 2
```

采用定时器 0 工作于方式 1 状态，将对应管脚周期性置反，并重新对定时器赋初值。

//根据游戏状态的不同，调用内部函数播放不同的乐曲
INT8U Sound_Play(INT8U const ucSoundState)
函数根据传入的参数不同，调用 MusicPlay( )函数播放不同的背景音。流程图如下：

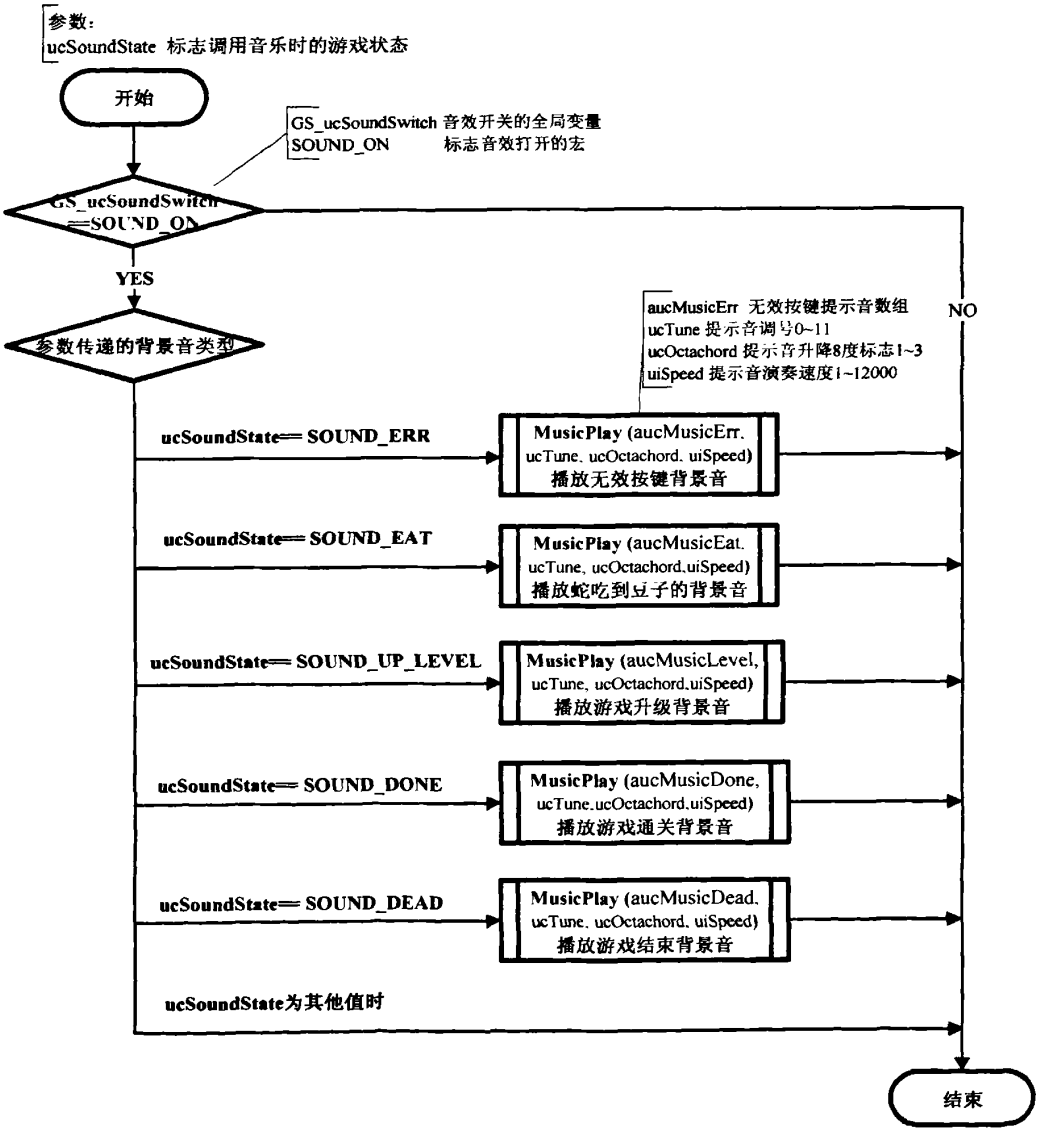

图 4.14 Sound_Play 函数流程图
Fig. 4.14 Sound_Play Function Flow Chart

//按照指定音调、音长播放背景音

INT8U MusicPlay (INT8U const * pucSound, INT8U const ucTune, INT8U const ucOctachord, INT16U const uiSpeed)

函数根据传入的参数，将音调节拍等分解进行处理，转化成音乐播放。流程图如下：

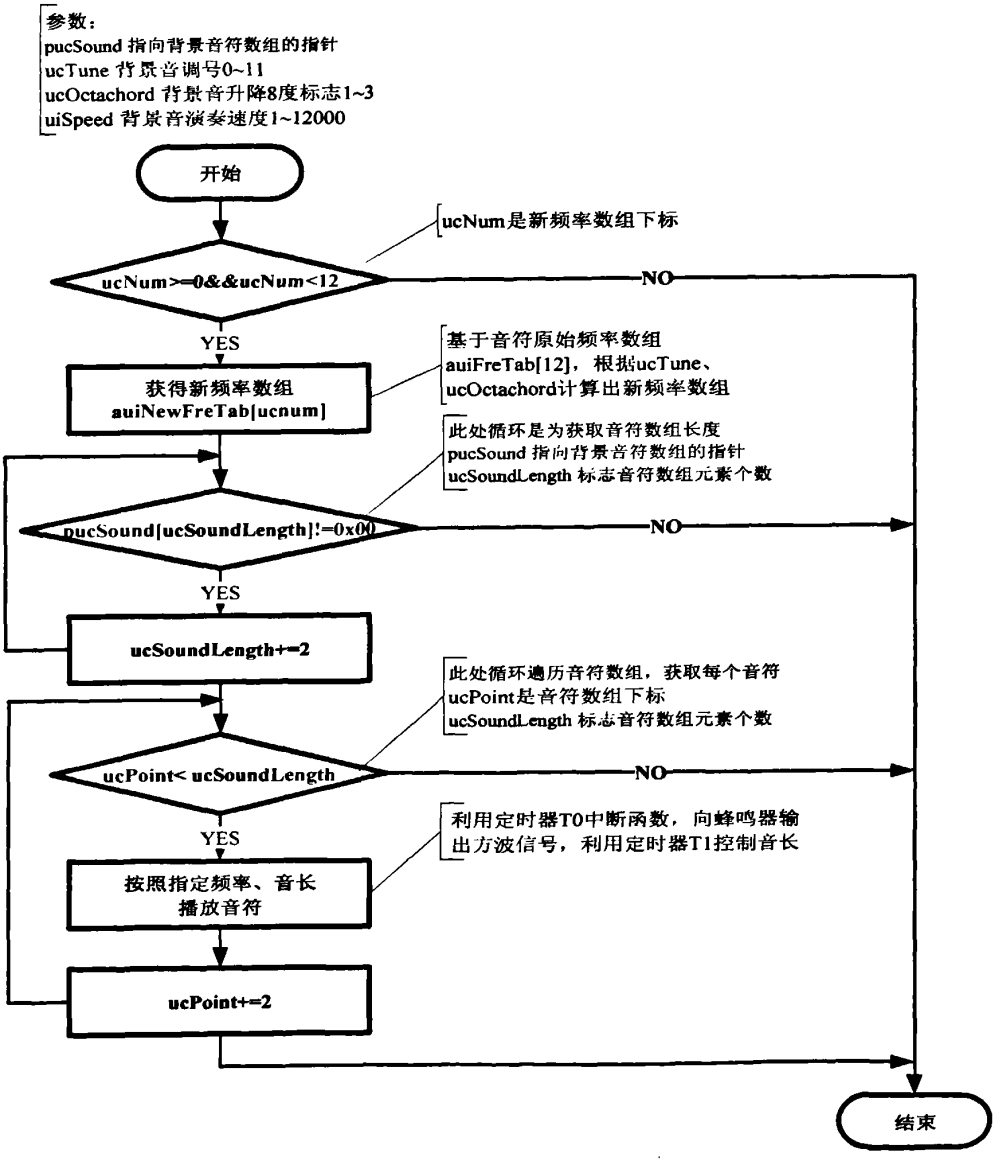

图 4.15 MusicPlay 函数流程图

Fig. 4.15 MusicPlay Function Flow Chart

结　论

基于51单片机设计的贪吃蛇游戏，可以充分发挥单片机的性能，体现嵌入式系统功耗低，节能，便携性好的特点，给人们的日常生活带来轻松快乐。本次论文按照设计要求完成了以下工作：

（1） 根据实际要求进行了系统硬件电路的设计。以P87C51单片机为核心，扩展了外围电路，加入LCD，扬声器，按键等构成了贪吃蛇游戏的硬件系统。

（2） 软件方面整个游戏采用C语言编写，大大加快了软件开发速度，缩短了开发周期，而且C语言编程可以方便地在各种型号的单片机上移植。

（3） 游戏设计上考虑到嵌入式系统存储容量有限的特点，利用一个一维数组存储蛇头的运动轨迹，大大节省了存储器空间的使用。代码编写上参考了许多良好的编程规范，代码可读性增强。

（4） 游戏除了拥有传统贪吃蛇游戏的功能外，又加入了地图选择的功能，给用户带来更多的挑战。丰富的背景音在游戏上给予用户更多的提示，并根据个人需要可以随时选择打开或者关闭背景音。

当然，由于条件所限，这里还有很多不足。例如不能进行全程持续背景音操作；游戏的界面应该更加绚丽多彩等，随着学习的深入以后可以在此基础上继续拓展，丰富功能。

参 考 文 献

[1] 郭天祥.新概念51单片机C语言教程:入门、提高、开发、拓展全攻略[M].北京:电子工业出版社,2009.

[2] 龙脉工作室.51单片机C语言应用开发技术大全[M].北京:人民邮电出版社,2008.

[3] 普拉塔.C Primer Plus:第5版[M].北京:人民邮电出版社,2005.

[4] Keil Software-Cx51编译器用户手册[M/OL].2001.

[5] 周润景,张丽娜,刘印群.PROTEUS入门使用教程[M].北京:机械工业出版社,2007.

[6] P87C51RA2/RB2/RC2/RD2 DATA SHEET[M/OL].2003.

[7] 樊永显,许勇,张向文等.基于STC89C54RC/RD+单片机的游戏机系统设计[J].湖南工业大学学报,2007,21(5):66-69.

[8] 金春霞,白秋产.基于J2ME技术手机游戏开发与实现[J].计算机与数字工程,2008,36(4):177-179.

[9] 王宏宇.VF游戏设计—贪吃蛇[J].中国科技信息,2007(7):91-92.

[10] 李德建,姚远程,周东杰.基于SOPC架构的贪吃蛇游戏研究与设计实现[J].科技创新导报,2008(31):26-27.

[11] 林锐,韩永泉.高质量程序设计指南:C++/C语言[M].北京:电子工业出版社,2007.

[12] Specification for LCD Module TS1620-1[M/OL].SHENZHEN TECHSTAR ELECTRONICS CO.,LTD.

[13] SMC1602A LCM使用说明书[M/OL].长沙太阳人电子有限公司,2001.

[14] Specification for LCD Module AG12864C[M/OL].晶采光电科技股份有限公司,2003.

[15] 李振军,成良玉.基于MIDP的Java手机游戏开发方法的分析与实现[J].计算机应用,2004,24(3):237-241.

[16] 唐天兵,严毅,陈纬文.基于J2ME的手机游戏开发与实现[J].广西大学学报,2005,30(7):51-53.

[17] 拉伯罗斯.嵌入式实时操作系统uC/OS-II[M].北京:北京航空航天大学出社,2003.

[18] WolfW.Hardware - Software Co - Design of Embedded Systems Proceedings[J].the IEEE,2004,82(7):123-126.

[19] KuoKai Shyu.A Newly Robust Controller Design for the Position Control of Permanent-Magnet Synchronous Motor[J].IEEE Trans.Ind.Electron,2002,49(3):558-564.

[20] Karl J Astrom.The Application of Keil and Proteus in MCU Game Desisn[J].the IEEE,2001,80(5):23-27.

[21] 谢少伟.基于MCS-51单片机功能完善的音乐播放程序设计[J].电子技术,2007,36(11):36-39.

[22] 王健,林原珩.带音效的手机游戏的设计与实现[J].长春师范学院学报,2008,27(5):53-55.

致　谢

经过几个月的努力，学位论文终于画上了一个圆满的句号。本次项目涉及到软件与硬件方面的许多知识，从开始准备，构思到最后完成，经历了许多东西，学到了许多知识，在此也要感谢许多人。

整个论文从准备期开始，查阅了大量资料，学习了许多编程技巧和思想。在硬件电路与软件编写上，都遇到了很多问题，通过和导师、同学、朋友的交流，解决了问题，其中经历的许多困难也磨练了自己的意志，对以后的工作学习会有很大的帮助。

最后要衷心感谢导师和曾经帮助过我的同学朋友们，没有你们耐心地指导帮助是不会有我今天的成果的，谢谢！