中国矿业大学

计算机科学与技术学院

2019 级本科生课程报告

课程名称 硬件课程设计

设计题目 电子密码锁设计（C 语言）

开课学期 2021-2022 年秋季学期

报告时间 2021 年 10 月 23 日

学生姓名 王杰永

学　　号 03190886

班　　级 计算机科学与技术 2019-3 班

专　　业 计算机科学与技术

任课教师 王凯

《硬件课程设计》课程报告评分表

开课学期: 2021-2022 年秋季学期

姓名:王杰永　　学号：03190886　　专业班级：计算机科学与技术 2019-3 班

序号	毕业要求	课程教学目标	考查方式与考查点	占比	得分
1	1.3	**目标 1：**了解微机应用系统解决复杂工程问题的基本方法。掌握微机应用系统硬件电路设计及软件功能需求分析方法和模型。能够针对微机系统应用领域工程需求的系统要求，进行分析与设计。	中期检查与设计文档 掌握解决复杂工程问题的基本方法。微机应用系统软硬件设计相关的理论知识。	10%	
2	4.3	**目标 2：**能够针对硬件电路组成需求描述进行系统硬件设计，能够分析系统功能的软件需求，根据模块设计原则，综合考虑系统的算法模型和软硬件开发，进行合理的方案设计、编程实现、系统测试及对设计方案进行优化。	中期检查与设计文档 考核题目需求分析和功能分析；综合知识应用能力及设计方案的完整性； 考核软件编程及系统调试测试，设计方案进行优化。	30%	
3	9.1	**目标 3：**具备多学科背景知识，并制定项目计划，能够按照标准规范进行设计。能够在多学科背景下具备独立分析问题解决问题的能力。	中期检查与设计文档 考核独立分析问题解决问题的能力	10%	
4	10.3	**目标 4：**掌握设计报告撰写，通过成果演示、陈述发言的清晰表达、回答问题准确性等。	现场验收与答辩 考核编程实现的代码难度和复杂性、设计工作量等；考核设计成果、所涉及的问题答辩。验收设计报告的结构合理性、内容和图表的正确性。验收设计报告排版的规范性。	40%	
5	12.1	**目标 5：**对选题主动通过各种途径寻求解决方法（主动查阅资料、请教老师、同学讨论等）。通过各种资源平台的使用及教师意见的反馈，完成高质量的设计任务，有无创新意识。	现场验收与答辩 考核设计成果完整性；所涉及的设计课题的创新性。	10%	
总成绩				100%	

任课教师：

年　月　　日

目录

1 绪论

本次硬件课程设计的课题是电子密码锁。设计中使用了 8255 芯片，4×4 键盘、74LS273、LCD12864 等器件。其中，使用 8255 控制 4×4 的键盘与 LCD12864 显示屏，使用 74LS274 控制开锁指示继电器与开锁错误的声光报警装置。可以进行密码的设定、修改，同时添加了海关锁特性。最后拓展了密码的加密存储等功能。本次设计选用 C 语言完成。

1.1 问题提出

在日常生活和工作中，传统的防盗锁由于构造简单、安全性较低且寿命短，无法满足大众的需求，特别是配套钥匙的携带极其不方便，给人们带来很大烦恼。随着电子信息技术的发展，我们可以采用电子密码锁来替代传统的防盗锁。

电子密码锁通过内部的数字逻辑电路或单片机芯片工作，采用密码代替钥匙，免去了需要随身携带钥匙的烦恼。

相比于普通锁，电子锁可以随时更换、修改密码，在开锁错误后启动声光报警。因此，电子锁具有功能多样、灵活性高、实用性强等优点，具有广泛的实际应用价值。本课题有良好的研究意义。因而电子密码锁的发展存在很大的空间和市场意义。故本次课题选用现有的资源完成一个电子密码锁。

1.2 设计任务与要求

利用 8255 控制 4×4 键盘与 LCD 显示屏与开锁指示继电器，可以修改密码并具有海关密码，模拟海关锁特性。开关锁状态使用 LCD 屏幕显示。同时具有开锁错误声光报警提示。

2 系统设计需求分析

2.1 系统组成原理及开发平台

本次课程设计使用了 ZK-II_USB 教学实验系统，软件平台选用 VC6.0，选用 Proteus8.6 进行硬件电路图的绘制。使用 C 语言进行软件编写。

2.1.1 开发平台

• 硬件平台选用 ZK-II_USB 教学实验系统，该实验系统上配置了 USB 接口模块，直接与主机（PC）的 USB 接口连接，形成了一整套完整的 USB 接口的微机接口实验系统，该系统由一块 USB 总线接口模块、ZK-II_USB 实验系统及集成环境软件组成，具有高速 USB 下的通信能力，即插即用，接口集成电路丰富，包括：可编程定时器/计数器 8254、可编程并行接口 8255、数/模转换器 DAC0832、模/数转换器 ADC0809 等。外围电路包括：逻辑电平开关、LCD 显示及驱动电路、键盘显示控制电路等。实验程序可以使 8086 汇编和 C 语言编程实验，可以对汇编程序和 C 语言程序进行调试，并且实验台自备电源，具有电源短路保护确保系统安全，使用 USB 接口与 PC 机相连，缺省了打开主机箱安装接口卡的麻烦。本次课程设计我运用到了该实验系统中的 8255 芯片、74LS273 芯片、4×4 键盘和 LCD 显示屏等器件。下图是 ZK-II_USB 实验系统模块结构图：

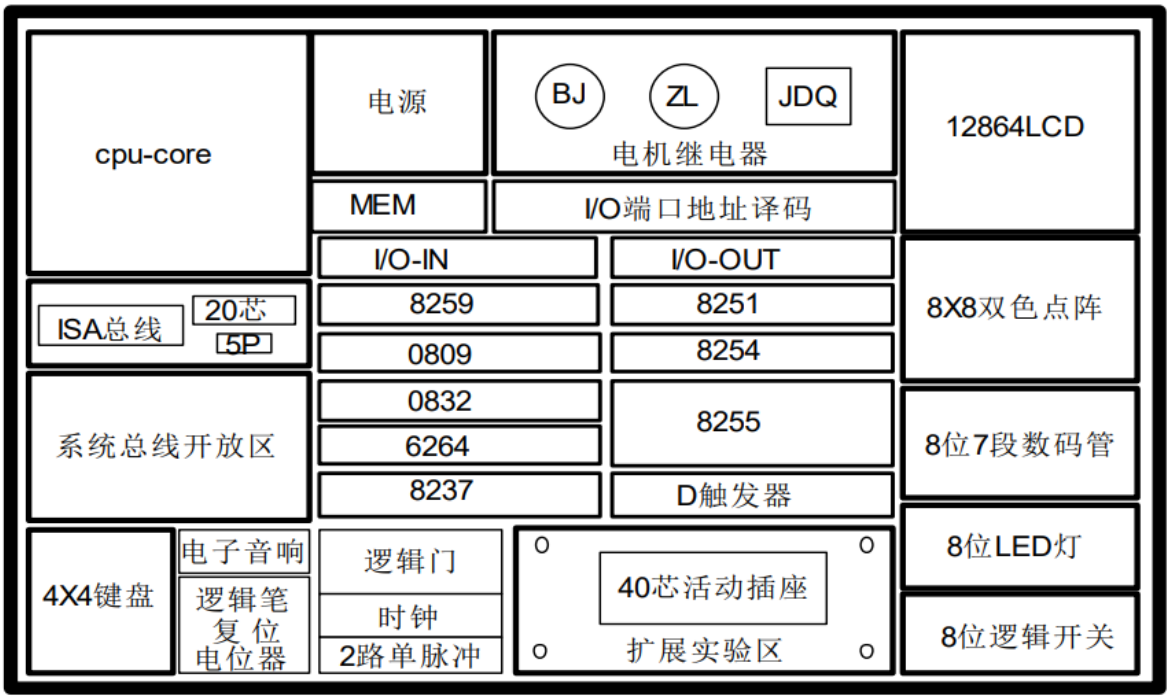

图 2-1 硬件开发平台

• 软件平台选用 VC++6.0。C 语言是一门通用计算机编程语言，与其他计算机编程语言相比，C 语言兼具高级 语言和汇编语言的优点，因此 C 语言不仅可以设计系统语言，而且还能编写计算机硬件 之外的应用程序。由于 C 语言允许直接访问物理地址，可以直接对硬件进行操作，因此 C 语言能够像汇编语言一样对位、字节和地址进行操作。C 语言生成代码质量高，程序执行效率高，一般只比汇编生成的目标代码效率低

10~20%。相对汇编语言来说，C 语言也具有很多优势，如程序可读性好、可维护性强、模块化和可移植性等优点，因此在这次课程设计中我运用了 C 语言作为编程语言。

• 本次课题选用 Proteus 绘制电路原理图。Proteus 是英国著名的 EDA 工具(仿真软件)，从原理图布图、代码调试到单片机与外围电路协同仿真，一键切换到 PCB 设计，真正实现了从概念到产品的完整设计。是世界上唯一将电路仿真软件、PCB 设计软件和虚拟模型仿真软件三合一的设计平台，其处理器模型支持 8051、HC11、PIC10/12/16/18/24/30/DSPIC33、AVR、ARM、8086 和 MSP430 等，2010 年又增加了 Cortex 和 DSP 系列处理器，并持续增加其他系列处理器模型。在编译方面，它也支持 IAR、Keil 和 MATLAB 等多种编译器。

2.1.2 系统设计电路原理图

本次课题中，8086 使用最小工作模式模式，通过 8086 控制键盘、LCD 等芯片进行工作。系统整体的电路原理图如图 2-2 所示。

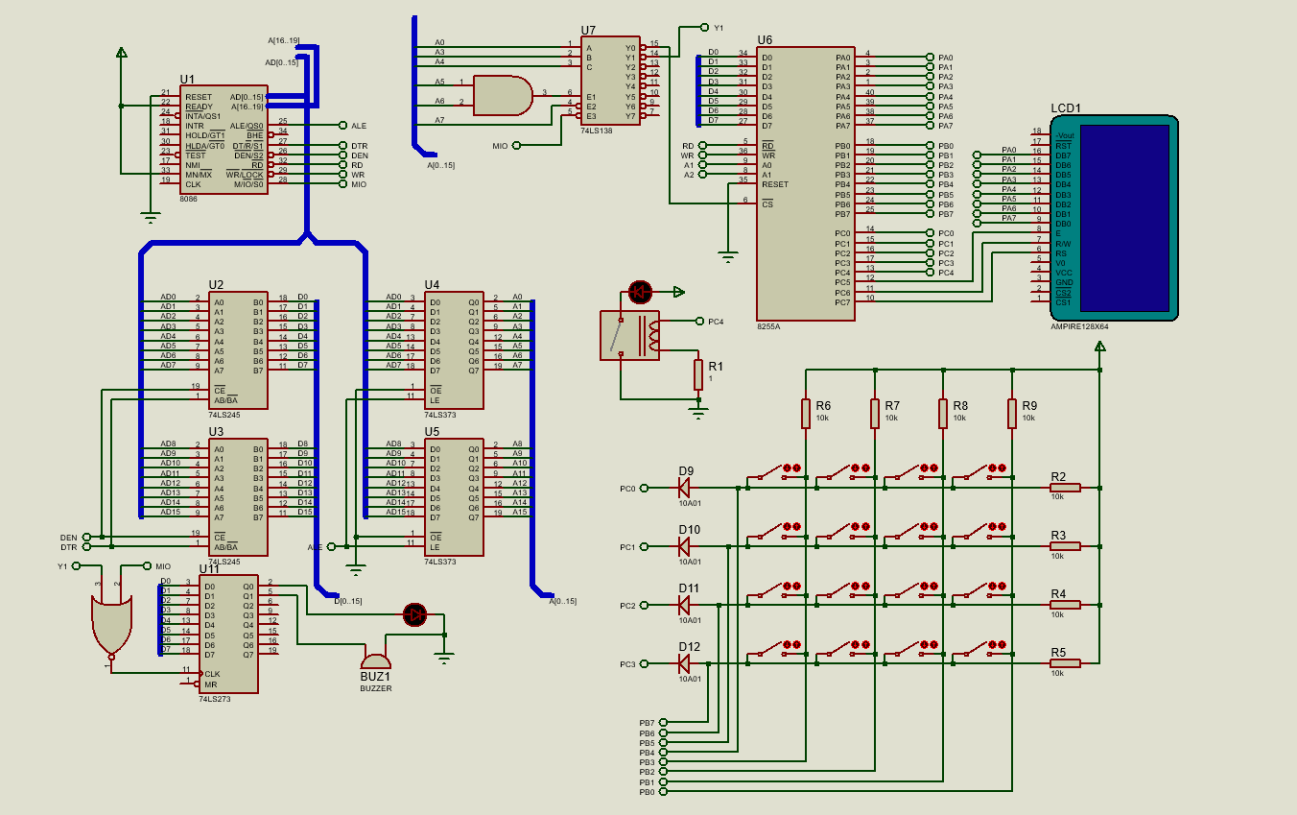

图 2-2 完整的电路原理图

2.2 系统工作业务流程图

针对设计任务与需求，综合考虑键盘、LCD 显示屏、继电器及声光报警以及密码相关功能，绘制出了如图 2-3 所示的业务流程图。

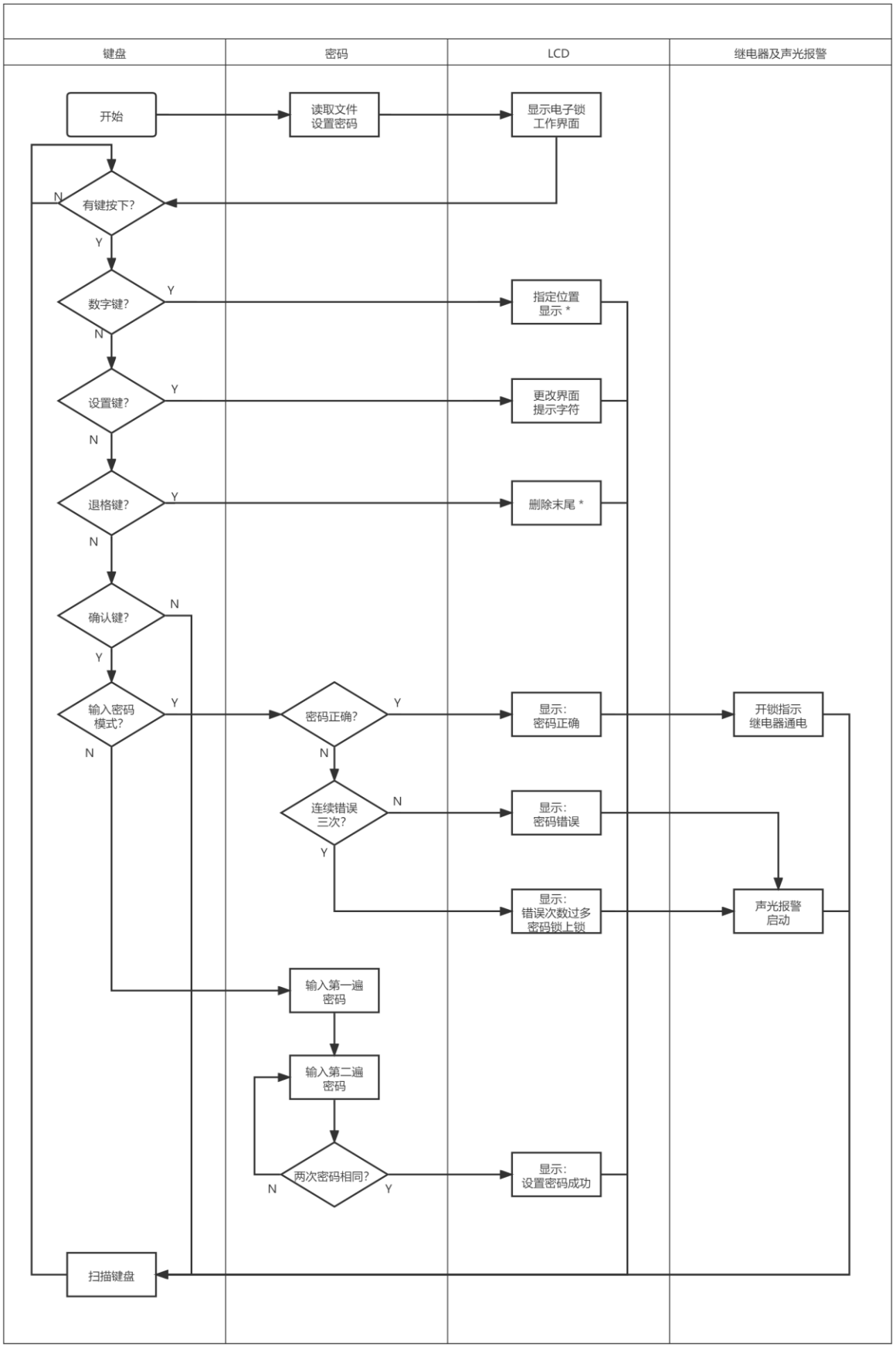

图 2-3 系统的工作业务流程图

2.3 系统的用例图

针对设计需求与设计目标，还可以绘制出系统设计的用例图。在图 2-4 展示的系统用例图中，参与者有两种——普通用户与海关人员。对于普通用户而言，有“密码输入”与“密码修改”两个用例；对于海关人员而言，有“海关密码输入”用例。同时，两种参与者均可以进行“开锁”。

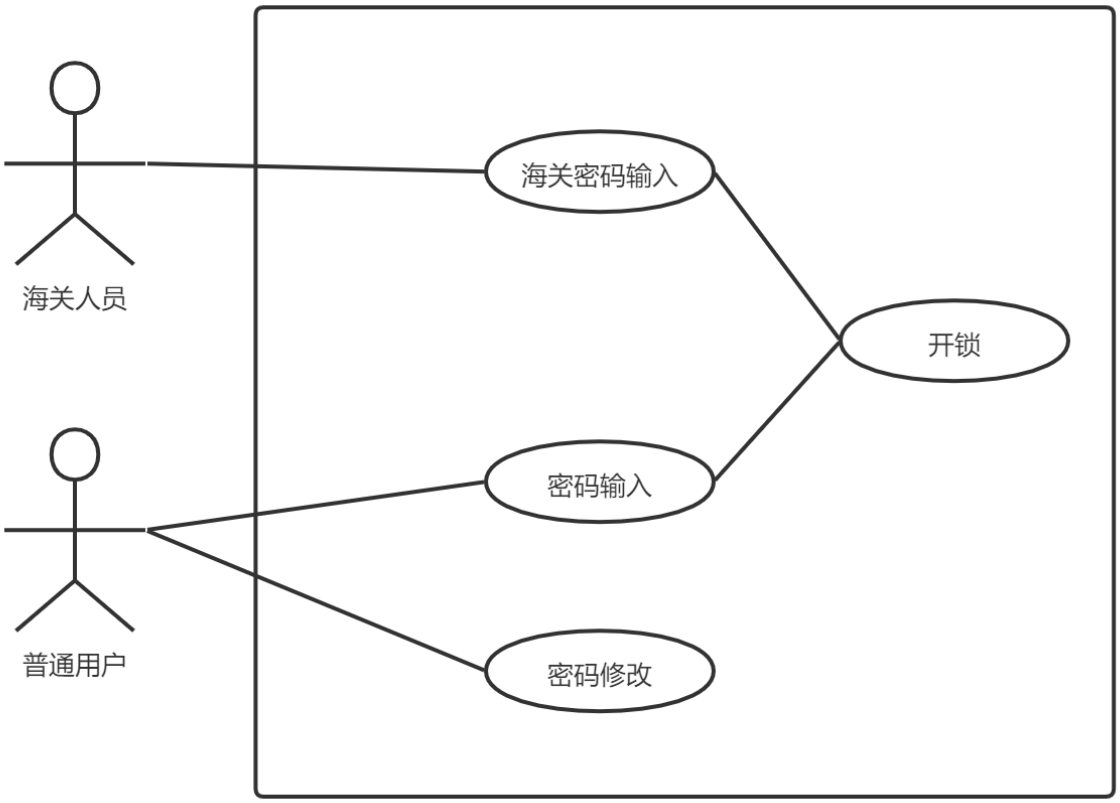

图 2-4 系统设计用例图

3 系统的总体设计

3.1 系统基本功能

本课题“电子密码锁设计”，要求完成的基本功能有：使用键盘键入密码、修改密码，同时与 LCD12864 联动显示，加入海关密码特性，并在开锁成功时为开锁指示继电器通电，开锁失败时有声光报警效果。

3.2 系统拓展功能

除了上述基本功能之外，还加入了若干拓展功能。

①对于密码的存储问题，由于直接将密码存储在“.txt”文件中及其不安全，对于第三方使用者很容易从中找到用户设定的密码，故采取特定算法进行加密；

②在输入密码时，若密码连续输入错误，则可以很大程度上认为有人在暴力破解密码。因此当连续输入密码错误三次时，密码锁进入“休眠”状态五分钟，防止可能到来的持续性暴力破解；

③在设置密码时，必须在输入密码后，再输入一遍相同的密码即确认密码，否则设置不成功。

③最后，对主程序进行若干次测试，模拟各种非正常输入并提供有效的应对策略，完善程序的健壮性。

3.3 系统功能层次图

在分析本课题的需求以及目标后，将系统所要实现的功能划分成了键盘控制模块、密码模块、LCD 显示模块和开锁提示模块共计四大模块。其中，密码模块又包括密码的设定、密码的修改、密码的加密以及海关密码。开锁提示模块又包括开锁错误的声光报警模块以及开锁指示继电器模块。系统的功能模块图如下。

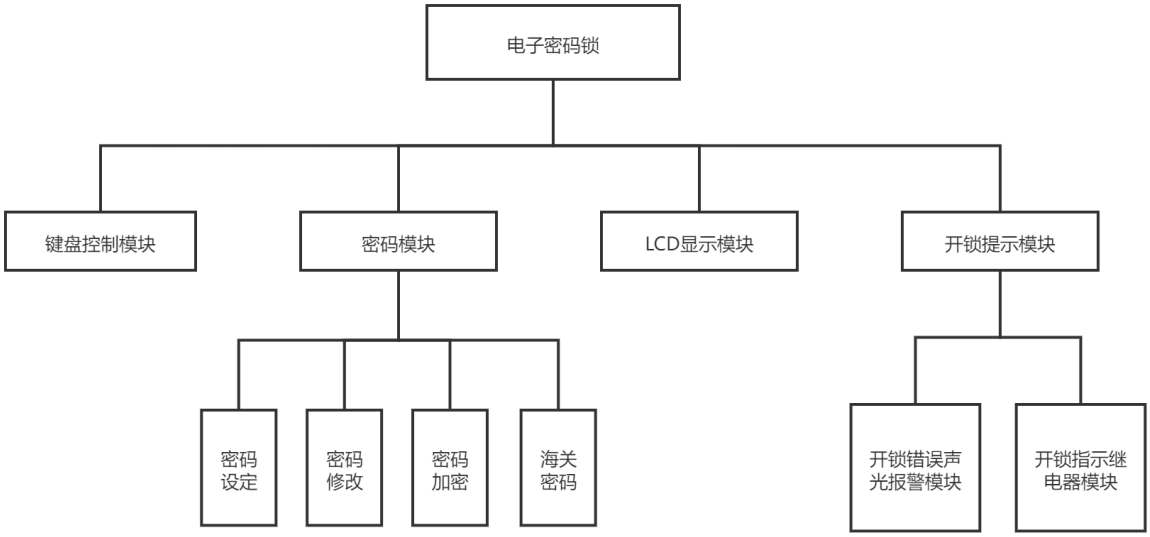

图 3-1 系统的功能模块图

4 系统的详细设计

4.1 键盘控制模块设计

该课题我选用了 4×4 的机械键盘，共有 4 条行线与 4 条列线。通过按钮是否按下可以使得对应行线与对应列线是否连通。通过 8255 控制键盘与 CPU 的数据交换。键盘的 8 条行列线与 8255 的 B 端口相连，供 CPU 读入键盘的行列线编码。四条行线同时与 8255C 端口低四位相连，供 CPU 控制键盘的工作状态。其硬件原理图如下。

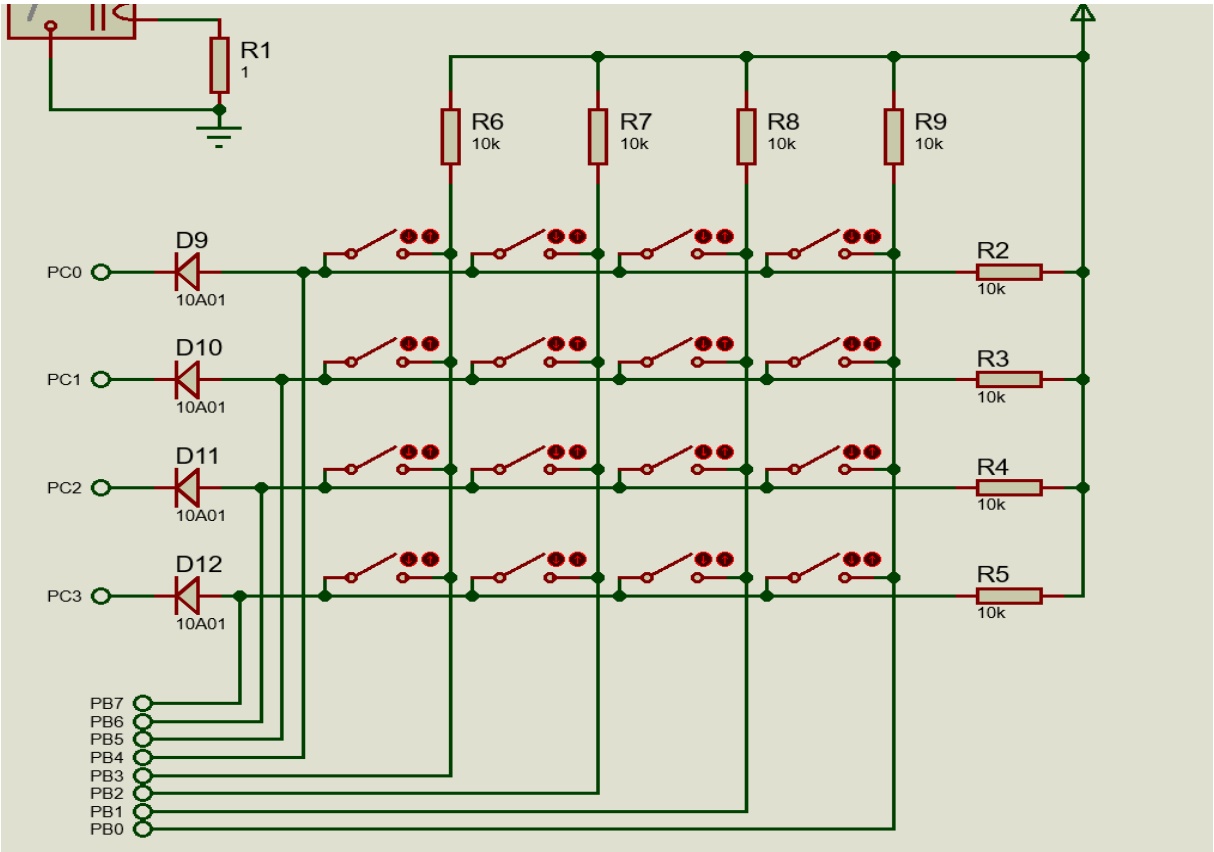

图 4-1　4×4 键盘与 8255 的硬件原理图

4.1.1 程序流程图

键盘的相关程序的工作原理描述如下：

①.在正常工作时，通过 8255 为键盘行线送入低电平，由于按键未按下，所有列线呈现高电平。此时我们可以判断出键盘所有按键均松开。

②.之后我们继续循环读入键盘的行列线编码，在行线为 0 的条件下，当列线不全为 1，我们便可以判断出某一按钮被按下。

③.经过一小段软延时后，再次读取列线状态，若仍然不全为 1，我们可以真正确定有一按键被按下。否则为轻微的抖动导致的行列线短暂接触。

④.接下来逐行拉低行线，读取列线，若得到非全 1 的结果，则可以确定是当前行的某一按键被按下。通过查询预先定义的键盘行列线编码表，可以唯一确定是哪一个按键被按下。

⑤.返回①。

下图是键盘控制的算法流程图。

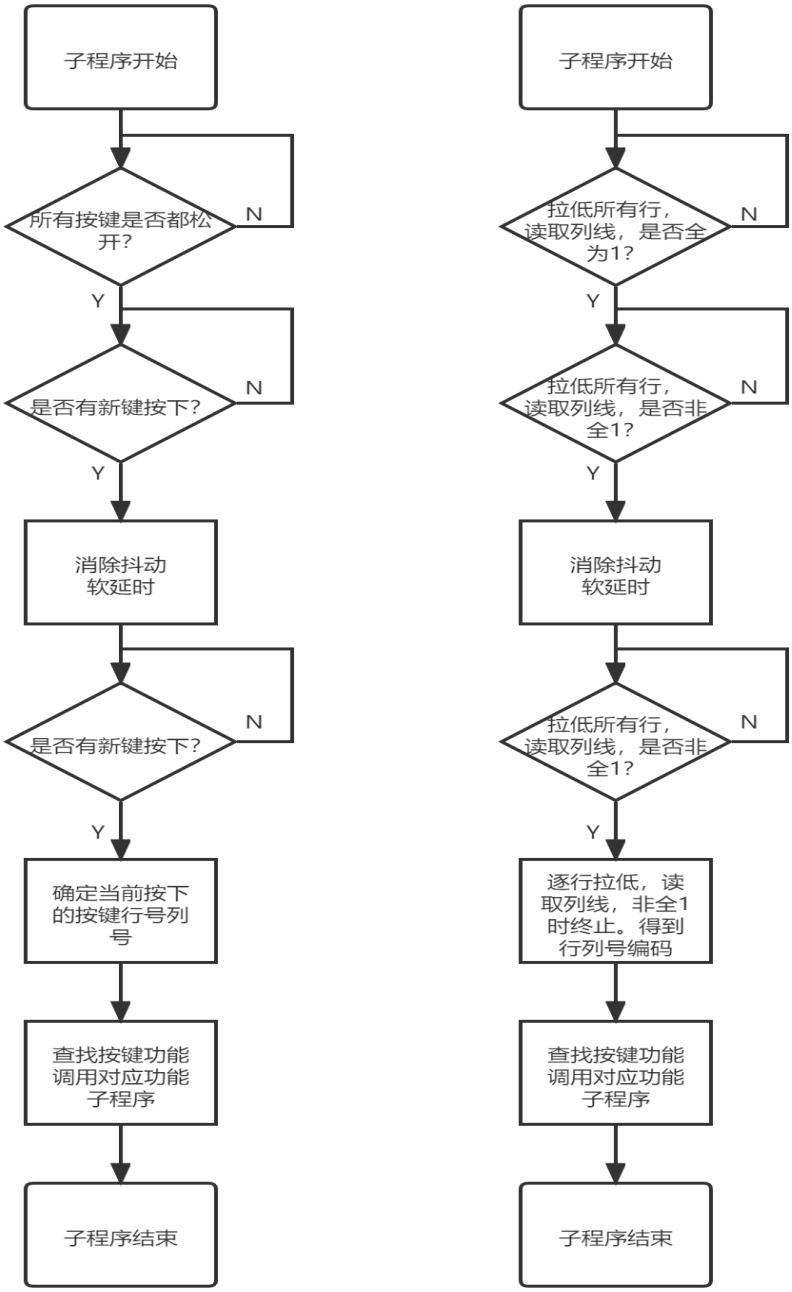

图 4-2 键盘控制的算法流程图

4.1.2 系统功能描述

该模块实现了循环检测键盘有无按下，并在某一按键按下时正确返回对应按键的唯一确定编码。为 LCD 显示模块的显示功能打下基础。

4.1.3 运行界面截图

在程序运行中，每当键盘按键被按下时，通过 C 语言的 printf 语句，打印对应按键

编码的十进制值，同时取出高四位与低四位打印。控制台输出如下图。

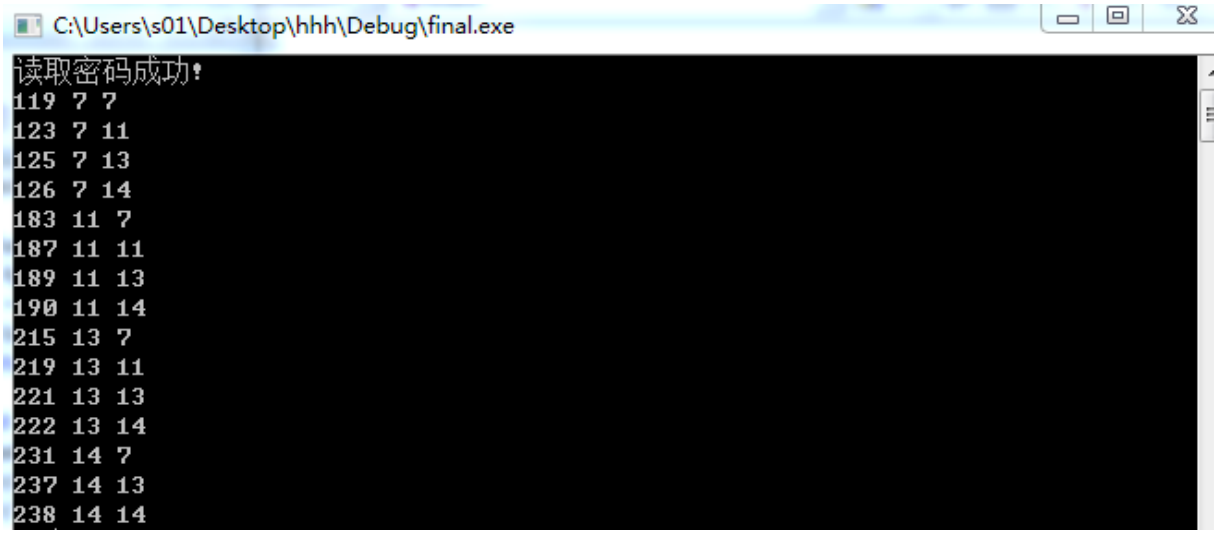

图 4-3 键盘控制模块的运行界面

总共输出 16 行，从上至下依次为 0~F 按键的编码。每行输出三个数字，第一个数字为按键的八位编码，后两位数字分别是编码的高四位与低四位值。例如，按键 0 位于 4×4 键盘的第一行第一列，其编码为 0111 0111，转换十进制为 119，高四位为 7，低四位为 7。

4.2 LCD 显示模块设计

该课题选用的是 LCD12864-ST9720 芯片。该课题需要使用 12864 进行汉字显示。

LCD12864 用于显示汉字时，其屏幕可以划分为四行八列的显示区域，共计显示 32 个汉字。芯片内部已经为每一个显示区域的地址进行了编码。表 4.1 为 32 个汉字显示区域的地址码。

表 4.1 LCD 显示区域的地址码

行编号	地址							
1	80H	81H	82H	83H	84H	85H	86H	87H
2	90H	91H	92H	93H	94H	95H	96H	97H
3	88H	89H	8AH	8BH	8CH	8DH	8EH	8FH
4	98H	99H	9AH	9BH	9CH	9DH	9EH	9FH

同样，采用 8255 作为 CPU 与 LCD12864 的接口芯片。

8255 的 A 端口作为 LCD 与 CPU 数据交换的八位端口。LCD 的三条控制线——使能端 E、读/写控制 R/W、指令/数据控制 RS 分别与 8255 的 PC5、PC6、PC7 相连。最终 LCD 显示模块的硬件原理图如下。

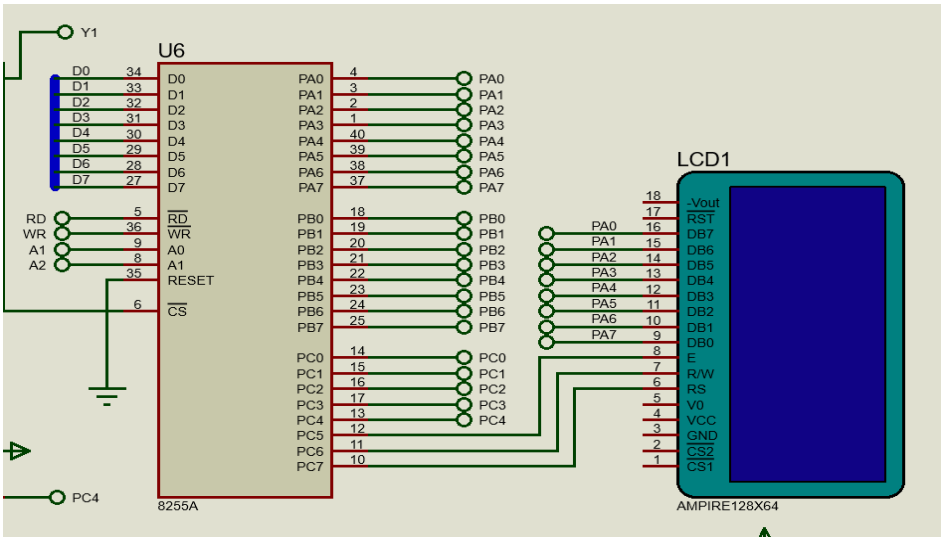

图 4-4　LCD 与 8255 的硬件原理图

4.2.1 程序流程图

LCD 的数据端口连接了 8255 的 A 端口，由控制端口 R/W、RS 控制当前操作是读写指令还是读写数据。

若要向 LCD 写入某一单字节指令，只需把指令写入 8255 的 A 端口，并将 R/W、RS 置 0，最后通过控制使能端 E 产生一个下降沿，即可使 LCD 执行相应指令。写入数据同理。以上写指令与写数据的共同步骤通过构造两个函数 cmdSetup()、dataSetup()来完成。

需要注意的是，显示一个汉字需要写入两字节数据，故芯片规定了要先写高字节，再写低字节。同时，从表 1 中可以看出，显示区域的位置编码也并非连续，需要注意。模块的算法流程图如下：

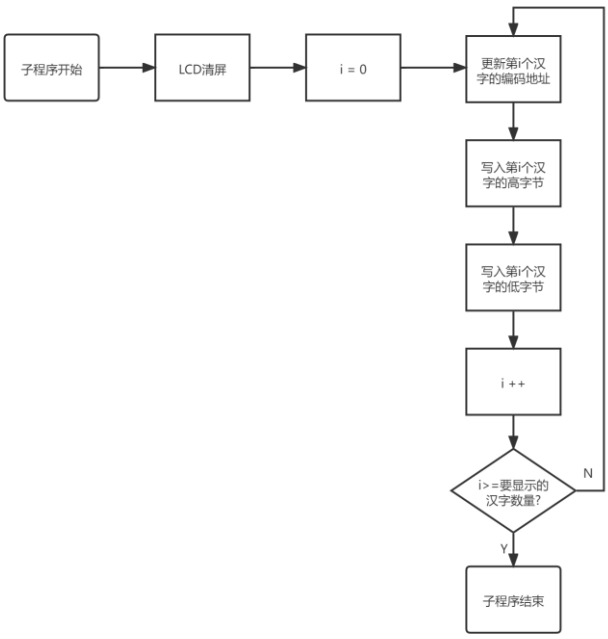

图 4-5 LCD 显示模块算法流程图

4.2.2 系统功能描述

根据 LCD 只能在特定位置显示总计 32 个汉字的特定，我绘制了想要显示出的四幅固定图案。如下图所示：

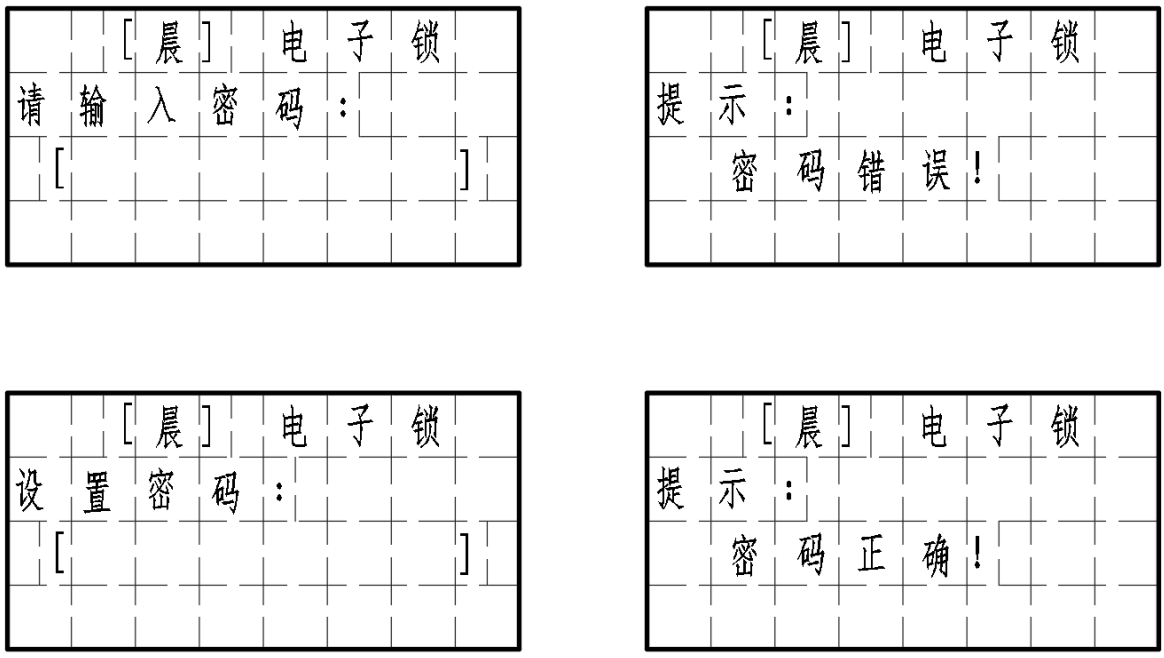

图 4-6 预定义的显示图

该模块除了可以在特定时间显示上面四图中的一个外，还实现了对输入密码的动态显示，与其余各个模块均有联系。

4.2.3 运行界面截图

系统开机，LCD 显示欢迎界面与“请输入密码：”。若按下“设置”按键，LCD 显示“设置密码”；

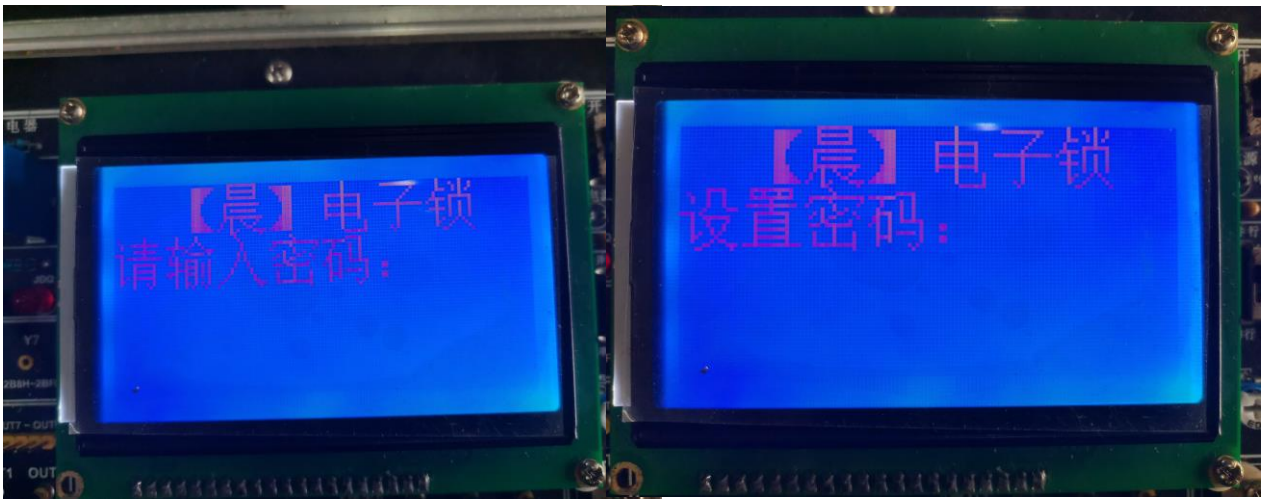

图 4-7 LCD 显示图 1

当按下确定键，根据输入的 6 位密码的正确与否，LCD 分别显示“密码正确”与“密码错误”。

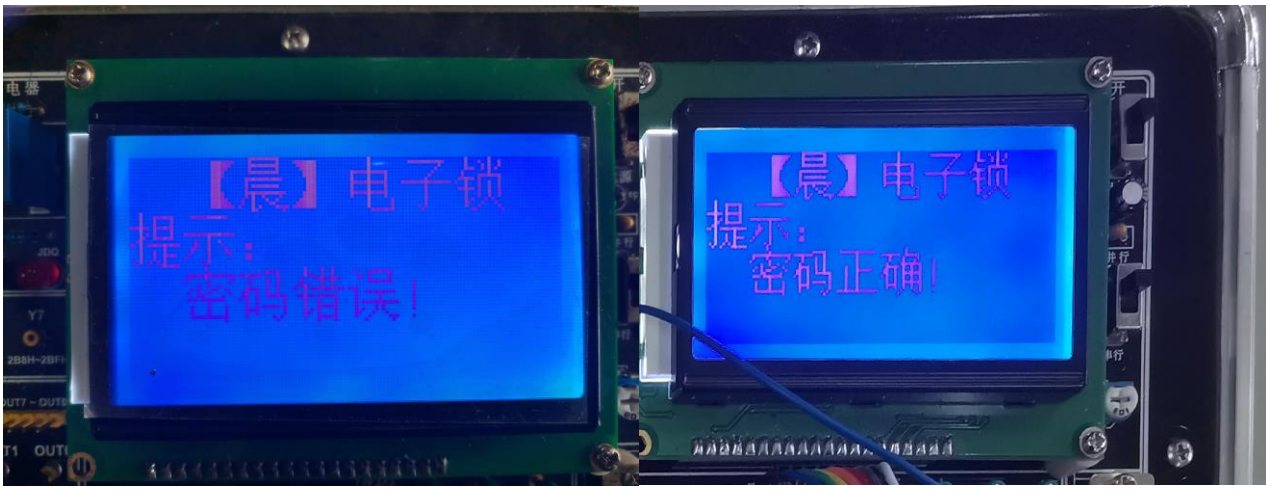

图 4-8 LCD 显示图 2

由于设置密码需要输入两次相同的密码，当第二次输入的密码与第一次不同时，LCD 显示“ ”。

当设置密码成功时，LCD 显示“设置成功”。

图 4-9　LCD 显示图 3

除此之外，为了保护用户的隐私与安全 LCD 动态显示键入的密码时，密码统一采用“*”显示。

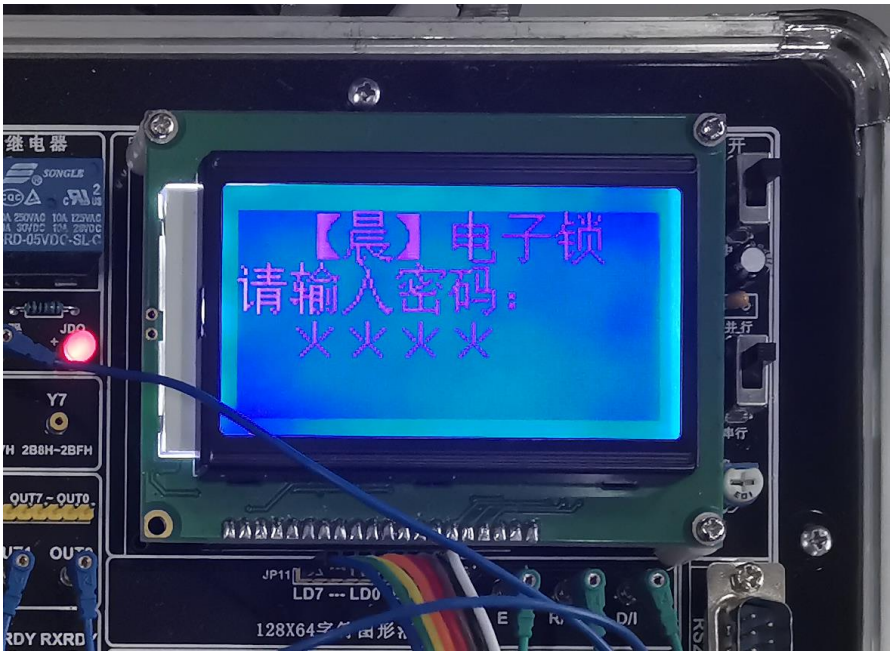

图 4-10　LCD 显示图 4

当输入密码连续错误三次时，LCD 显示“警告：错误次数过多！密码锁已被锁定！”。

图 4-11　LCD 显示图 5

4.3 开锁提示模块设计

该模块下需要实现两个功能——开锁指示继电器与开锁错误时的声光报警。

由于 8255C 端口仅剩 PC4 一个，不足以控制继电器、LED 与蜂鸣器，因此采用 74LS273 作为 CPU 的一个输出接口芯片进行拓展。

将 LED 灯连接 273 的 OUT0，蜂鸣器连接 273 的 OUT1，开锁指示继电器连接 273

的 OUT2 端口。硬件原理如下图：

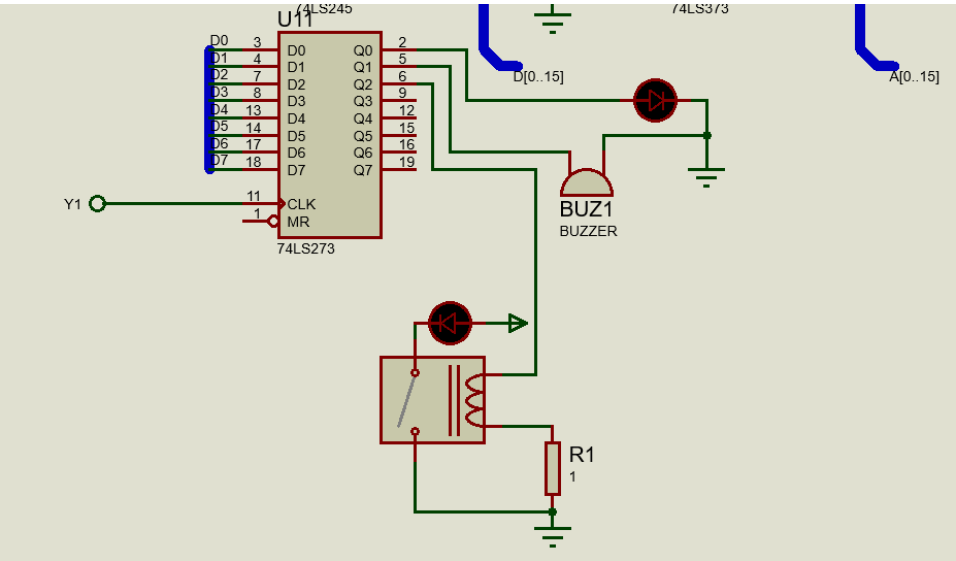

图 4-12 开锁提示模块的硬件原理图

4.3.1 程序流程图

该模块的程序较为简单，当输入密码正确时，使开锁指示继电器通电；当输入密码错误时，LED 亮起若干秒，蜂鸣器鸣笛若干秒后停止，达到开锁错误的声光报警功能。程序流程图如下：

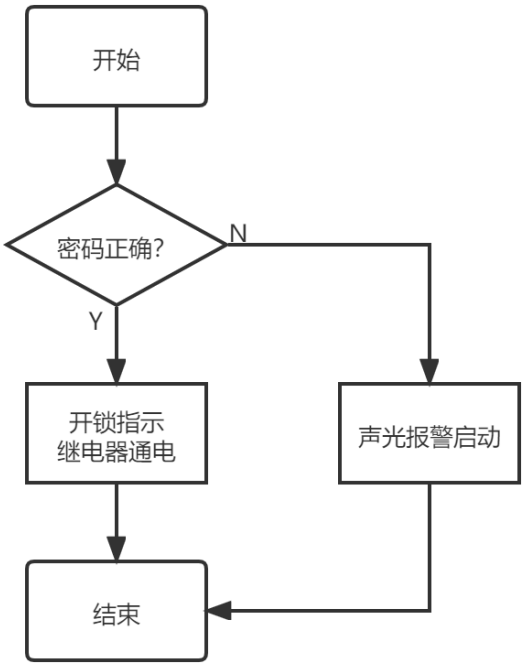

图 4-13 开锁提示模块程序流程图

4.3.2 系统功能描述

该模块实现了开锁成功与否的相关提示功能。当确认键被按下且密码正确时，通过

273 为继电器送入高电平，实现开锁功能；当确认键被按下且密码错误时，通过 273 为 LED 与蜂鸣器送入高电平，实现声光报警功能。

4.3.3 运行界面截图

当输入密码正确时，从图 4-14 可以看出，位于试验箱上方的开锁指示继电器通电。

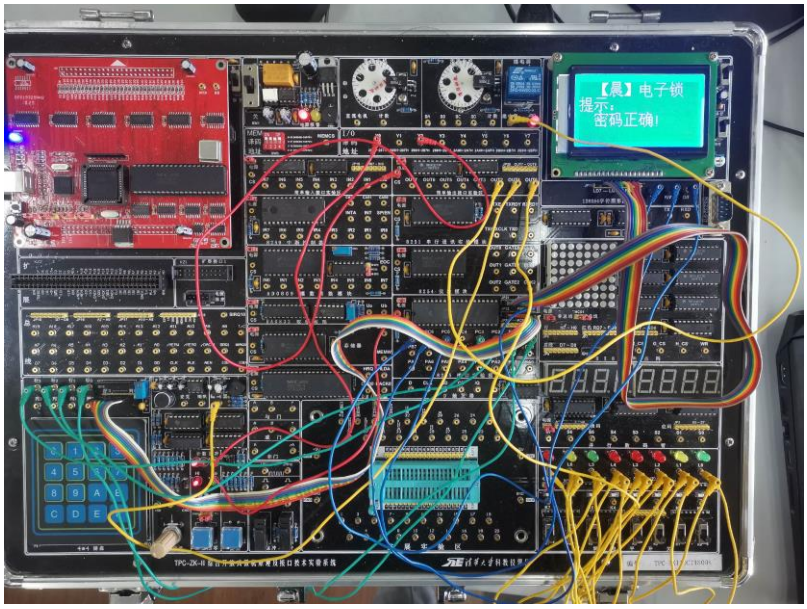

图 4-14 开锁指示继电器

当密码输入错误时，从图 4-15 可以看出，位于试验箱右下方的 8 个 LED 灯全亮，同时左侧蜂鸣器响，达到声光报警的功能。

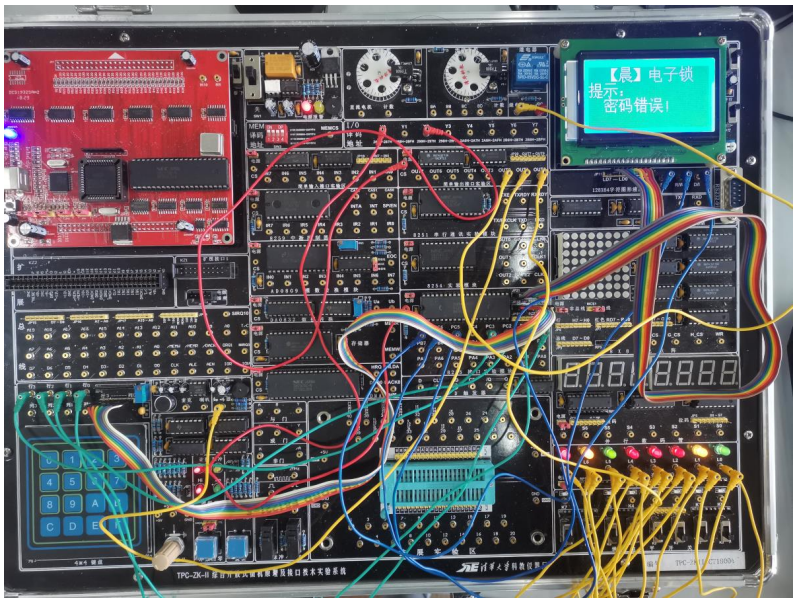

图 4-15 声光报警

4.4 密码模块设计

密码的设定、修改、加密以及海关密码主要是通过软件实现。

4.4.1 程序流程图或算法流程图

对于密码的输入正误判断，当按下确认键且当前为“输入”模式时，将预设置密码与当前输入密码比对，一致则给继电器通电。不一致则启动声光报警，并且启动计数功能，若连续三次输入错误密码，则电子密码锁会自锁五分钟。同时，加入了海关特性。在程序初始化时预设置一个海关密码，当输入的密码与海关密码相同时，无论用户设置的密码是什么，锁均会打开。

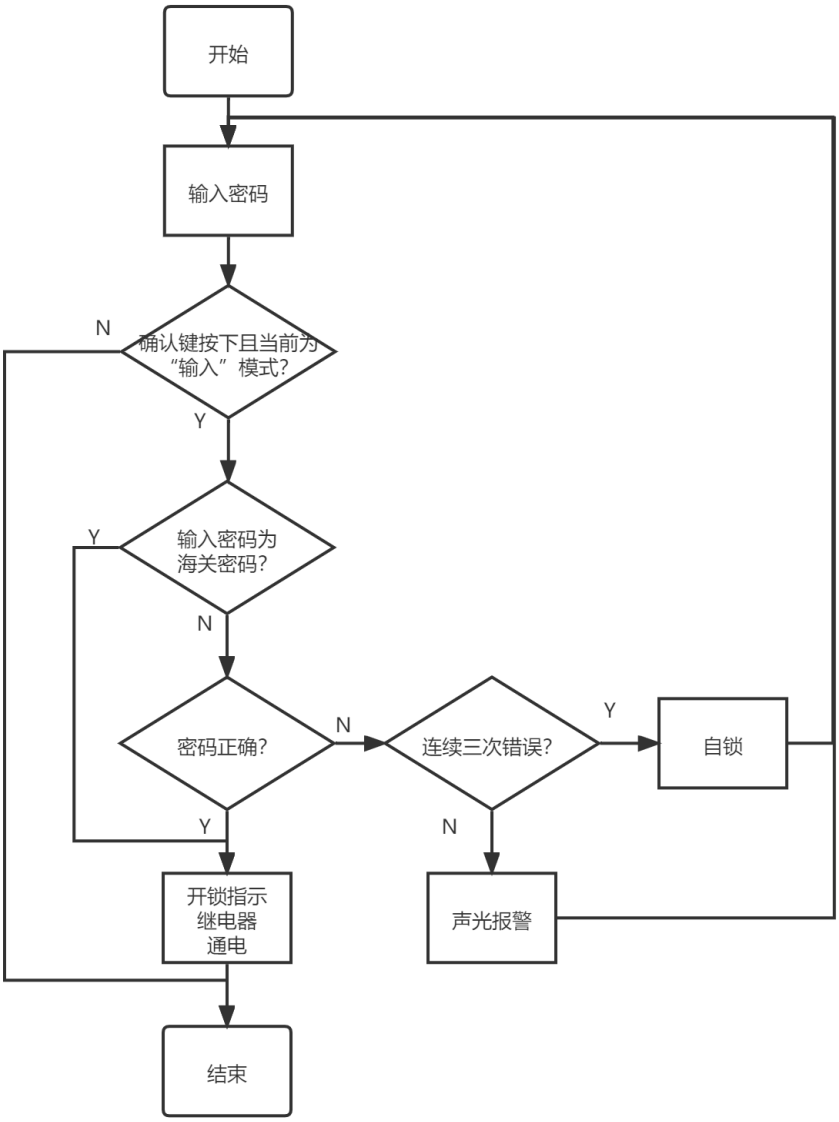

图 4-16 密码的输入与海关密码的程序流程图

对于密码的修改，当按下确认件且当前为“修改”模式时，记录当前输入的密码，并让用户再次输入，若两次一致则设置成功，修改密码文件；否则重新输入第二遍密码。

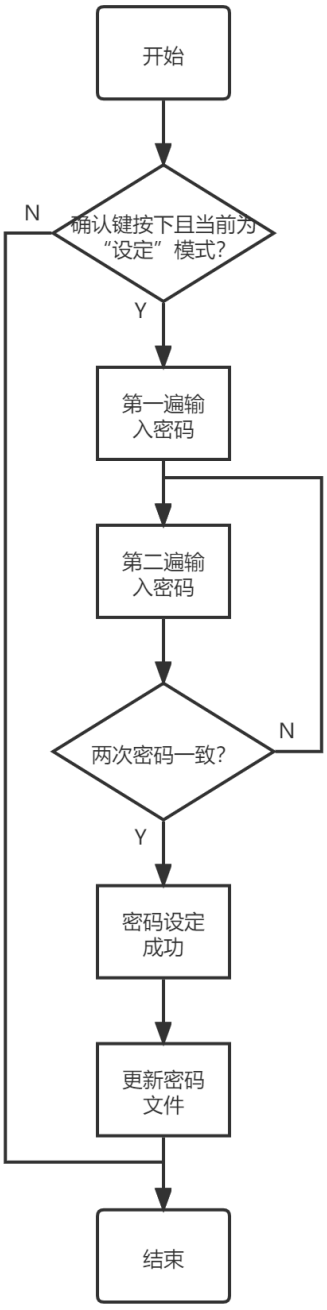

图 4-17 修改密码的程序流程图

最后，对于密码的加密，由于输入的数字所在的范围是 0~9，为了方便保存密码文件以及后期读取文件，我自定义了加密函数，该函数对于输入的一位数字可以输出一个唯一确定的一位数字与之对应。

若输入的是奇数，则输出原数字减一；若输入的是偶数，则输出原数字加一。这样完成了密码加密。对于上述加密规则，解密就变得很容易。输入的加密后的数字是偶数，则原数字为加密数字加一；输入的加密后的数字是奇数，则原数字为加密数字减一。可以看出，加密与解密的规则完全一样。故加密与解密算法的程序流程图完全相同，如下：

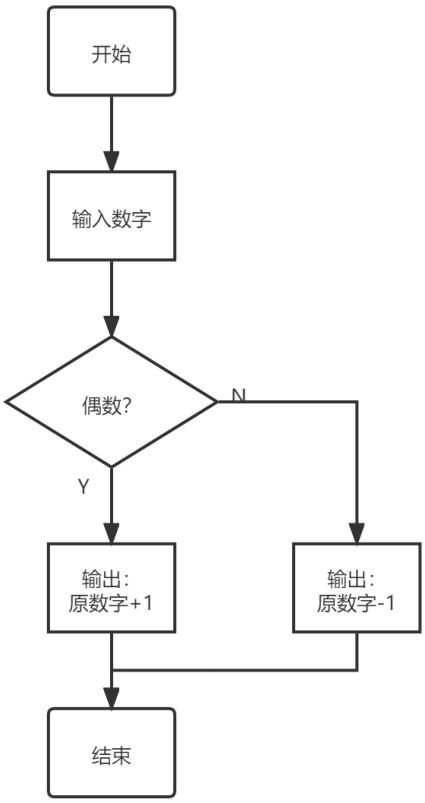

图 4-18 密码的加密程序流程图

4.4.2 系统功能描述

该模块实现了密码输入的正误判断，密码的自定义修改，海关锁特性以及密码的加密功能。同时兼具了连续输入错误密码的自锁功能。

4.4.3 运行界面截图

对于密码输入的正误判断，LCD 显示“开锁成功”或者“密码错误”，运行界面截图如图 4-13、图 4-14 所示。

当密码连续输入错误三次时，LCD 显示“错误次数过”，运行界面如图 4-10 所示。

对于密码加密模块，我首先编写了加密与解密程序并使用子程序进行了 0~9 共计 10 个数字的测试，可以看出每一个数字都有唯一确定的加密数字与其对应。

```
原数字：1    加密：0    解密：1
原数字：2    加密：3    解密：2
原数字：3    加密：2    解密：3
原数字：4    加密：5    解密：4
原数字：5    加密：4    解密：5
原数字：6    加密：7    解密：6
原数字：7    加密：6    解密：7
原数字：8    加密：9    解密：8
原数字：9    加密：8    解密：9
```

图 4-19 加密解密程序的测试

启动程序，设置密码为”0 0 1 2 0 2”后，打开保存密码的文件”code.txt”，其中保存的结果为 1 1 0 3 1 3，与预期结果一致。

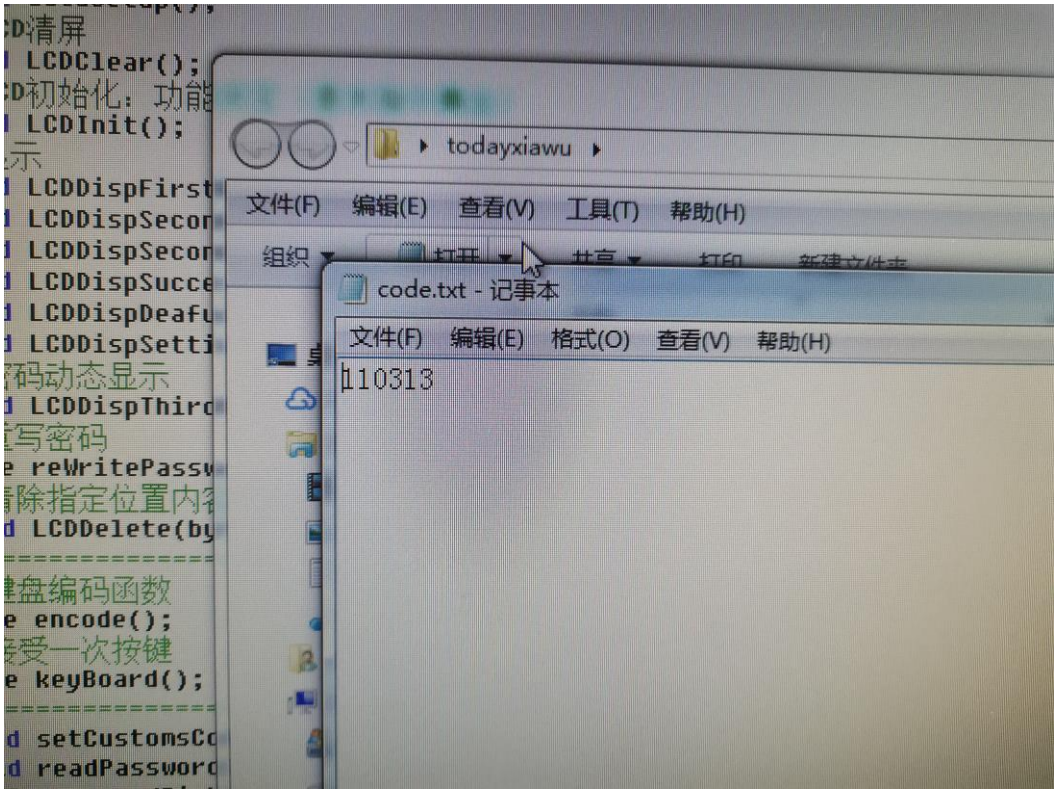

图 4-20 密码的加密存储

5 系统测试

在编码完成后，设计了若干测试样例，对系统的测试结果如表 5.1 所示。

表 5.1 系统测试结果

测试的功能	测试步骤	预期结果	实测结果
键盘控制	①按下数字键：0 1 2 3 4 5 6 7 8 9 ②按下功能键：A B C D E F	①控制台正确输出各按键的编码 ②LCD 屏幕上显示最多六位"*"标识	通过，与预期结果一致
LCD 显示	①启动程序 ②依次按下数字键：5 4 8 ③按下”设置“键	①启动程序后，LCD 第一行显示"[晨]电子锁"，第二行显示”请输入密码：“。 ②LCD 第三行显示三个"*"标识。 ③LCD 第二行显示”设置密码："	通过，与预期结果一致
密码正确的判断	①输入正确的密码：0 0 1 2 0 2 ②按下“确认”键	LCD 显示“密码正确”，开锁指示继电器通电。	通过，与预期结果一致
密码错误的判断	①输入错误的密码：0 5 4 8 3 5 ②按下“确认”键	LCD 显示“密码错误“，声光报警启动	通过，与预期结果一致
连续三次密码错误，密码箱自锁	①输入错误的密码：0 5 4 8 3 5 ②按下“确认”键 ③再重复上述操作两次	LCD 显示”警告，错误次数过多！密码锁已锁定！"，同时键盘失效五分钟	通过，与预期结果一致
修改密码，且两次输入密码必须一致	①按下设置键 ②输入设置的新密码：0 5 4 8 3 5 ③再次输入密码确认：0 0 0 0 0 0 ④再次输入密码确认：0 5 4 8 3 5	①在输入确认密码时，由于两次密码不一致，需要重新输入 ②重新输入与第一遍密码相同的确认密码 ③LCD 显示”设置成功“	通过，与预期结果一致
海关特性	①输入海关密码:0 1 1 0 1 6 ②按下”确认“键	LCD 显示”密码正确“，开锁指示继电器通电	通过，与预期结果一致
密码加密	①按下设置键 ②输入设置的新密码：0 5 4 8 3 5 ③再次输入密码确认：0 5 4 8 3 5 ④查看保存密码的文件'code.txt'	"code.txt"中保存的密码应该为：1 4 5 9 2 4	通过，与预期结果一致

6 系统设计结果分析及结论

在本课题中，我使用了 8255、74LS273、4×4 键盘与 LCD12864 等芯片。通过 8255 作为 CPU 与键盘、LCD 等外设的接口芯片。但是由于 8255 的管脚不足，又加入了 74LS273 作为 CPU 与继电器、声光报警装置的接口芯片。系统的硬件连线实物图如下：

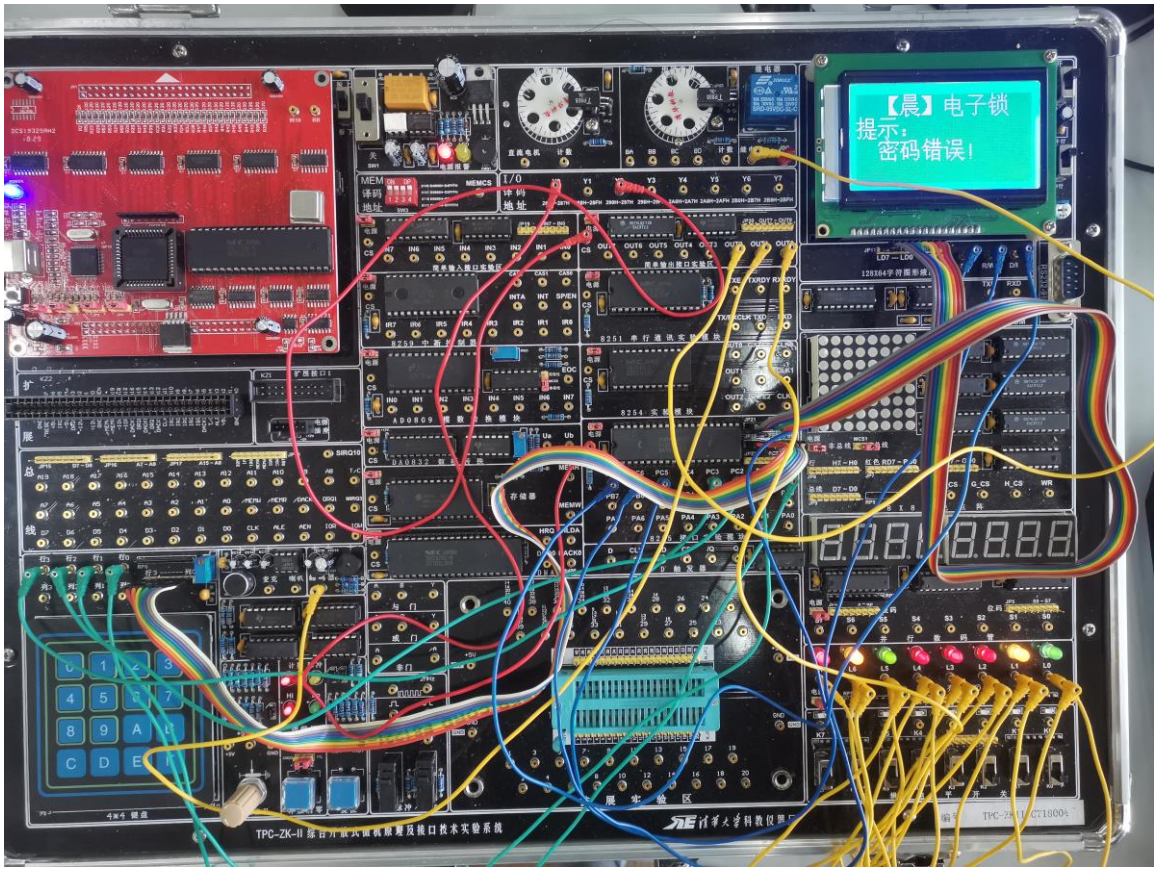

图 6-1 系统的完整电路图

在硬件连线的基础上，通过编写代码，我实现了如下功能：

①由键盘作为系统的输入，按下任何按键时，由控制台打印出该按键的编码，如图 4-3 所示，可以看出键盘控制模块的功能完全实现。通过不断扫描键盘有无按键被按下，并当按键被按下时转入编码程序，实现了上述功能。

②当输入的六位密码正确时，此时按下确定键，开锁指示继电器通电，LCD 屏幕显示“密码正确”，如图 4-14 所示；而当密码输入错误时，声光报警启动，LCD 屏幕显示“密码错误”，如图 4-15 所示。通过简单的条件程序结构，实现了上述功能。

③当输入连续三次输入密码错误时，此时密码锁启动自锁功能，声光报警启动，LCD 屏幕显示“错误次数过多！密码锁已被锁定！”，此时键盘失效五分钟，如图 4-11 所示。定义全局整型变量 warning 用于连续错误计数，每当输入错误时计数值自减 1，当密码正确时恢复初值 3。当自减为 0 时，程序跳转，启动自锁功能。

④当按下“设置”键时，此时密码锁进入“设置密码”模式，输入六位密码并按下“确定”键，密码锁会记忆住本次预设置的密码，接下来进行第二次确认密码的输入。若输入错误则需重新输入，反之新密码设置成功，如图 4-9 所示。通过定义布尔型标志变量 flag 确定密码锁的工作模式，并使用顺序程序结构来完成两次密码的输入。

⑤当输入的密码是预定义的海关密码时，继电器通电，LCD 显示“密码正确”。如图 4-8 所示。通过预定义海关密码，并在判断当前密码是否正确时先进入海关密码的判

断，若与海关密码匹配则无需判断用户自定义密码，直接开锁。

⑥通过设置文件”code.txt”来保存用户自设置的密码。每次程序运行会先读取该文件中的密码，之后再判断用户输入的密码是否与该密码相同。

⑦在设置模式下，且成功设置了密码时，程序会对输入的新密码进行保存。保存前要对新密码进行加密存储，如图 4-19 所示；同样的，在读取密码时，先调用解密程序后，再转入主程序的运行。

⑧完善了程序的健壮性。在编码完成后，运行若干次程序。在运行中很容易发现许多小”bug”。例如，输入密码时，若密码已经满六位，继续输入会把输入的字符显示在非密码输入的位置；同理，在输入栏没有密码时，多次按下删除按键，会导致下一次的输入字符显示在非密码输入的位置。诸如此类小”bug”有很多，但都是很容易解决的。解决后程序的健壮性明显得到改善。可以任由用户随意操作了。

7 设计体会

通过参与本次课程设计，我深深的感受到了课程设计的艰辛。从选择课题到需求分析，从理论上的设计到实验室里的亲手实践，每一项都感受到了曾经所学课程、曾经所做的实验从没有过的压力。

本次课程设计，我成功在 Proteus8.6 的系统上搭建了 8086 的最小模式工作图，对微型计算机的工作原理有了更好的理解。在最小模式基础上，使用 Proteus 自带的汇编语言编译器系统，实现了 8086、8255、小灯泡等的协同工作。不禁感慨到，要是上学期就知道了这个软件，那微机实验对自己一定会很简单。

而后，在 8086 最小模式连线图的基础上，添加了 8255、LCD12864、138 译码器等各类芯片，对这些接口芯片与外围设备也更加熟练的掌握。最终完成了 850 行的 C 语言程序也让我有了一些小小的满足感。深感软硬结合的有趣之处。

衷心感谢学校给予我们这样一个机会去应用所学的理论知识成为实践。

8 参考文献

[1]朱玉璟,李大,孙文杰,王晶.锁具之电子锁技术综述[J].科技致富向导,2014(21):285-286.
[2]孙万麟.基于 Proteus 的多功能电子密码锁控制系统设计及实现[J].机械制造与自动化,2021,50(04):216-218.
[3]黄田,杨婷婷,姜少维,李佳康,董莉霞.基于单片机的电子密码锁设计[J].软件,2020,41(10):102-104.
[4]周荷琴,吴秀清 主编.微机原理与接口技术.合肥:中国科学技术出版社 2007
[5]王俊博.关于 FPGA 的电子密码锁系统的设计[J].科学技术创新,2019(26):89-90.

9 代码

```
#include <stdio.h>
#include <conio.h>
#include <vector>
#include <Windows.h>
#include <iostream>
#include "ApiExUsb.h"
#pragma comment(lib,"ApiExUsb.lib")
#pragma warning(disable:4996)

#define PORT_A 0x280
#define PORT_B 0x281
#define PORT_C 0x282
#define PORT_CTL 0x283
#define PORT_273 0x290

using namespace std;

vector<int> PASSWORD;
vector<int> password;
vector<int> customsCode;

//开始时读一次数据
void test(){
	byte data;
	if(!PortReadByte(PORT_B, &data)){
		while (!kbhit()) printf("读数据错误\n");
	}
}
//测试
void dispCurrentPassword();

//初始化
void init();
void formatDisp(byte data);

/*=================LCD 部分=======================*/
//向 LCD 写指令
void cmdSetup();
//向 LCD 写数据
void dataSetup();
//LCD 清屏
void LCDClear();
```

```
//LCD 初始化：功能设定（基本指令集合）
void LCDInit();
//显示
void LCDDispFirstRow();
void LCDDispSecondRow();
void LCDDispSecondRow_1();
void LCDDispSuccess();
void LCDDispDeafuat();
void LCDDispSetting();
void LCDWarning();
//密码动态显示
void LCDDispThirdRow(byte data, byte currentLoc, bool flag=true);
//重写密码
byte reWritePassword();
//清除指定位置内容
void LCDDelete(byte currentLoc);
/*=================键盘部分=======================*/
//键盘编码函数
byte encode();
//接受一次按键
byte keyBoard();
/*=================密码部分=======================*/
void setCustomsCode();
void readPassword();
bool passwordRight();
void savePassword();
int Go(int num);
int Ret(int num);
/*=================继电器部分=======================*/
void deviceStart();
/*=================声光报警部分=======================*/
void light_sound_warning();

int warning;


int main() {
    warning = 3;
    byte table[16] = { 0x77, 0x7B, 0x7D, 0x7E, 0xB7, 0xBB, 0xBD, 0xBE,
                       0xD7, 0xDB, 0xDD, 0xDE, 0xE7, 0xEB, 0xED, 0xEE };


    byte data;
    if (!Startup())
```

```
{
    printf("ERROR: Open Device Error!\n");
    return 0;
}
init();
test();


LCDDispFirstRow();
LCDDispSecondRow();

byte currentLoc = 0x88;         //0x89H ~ 0x8EH       6 个数  +5

readPassword();
setCustomsCode();

byte lastNum = 0xDD;
bool r = false;
vector<int> temp;
/*
当前模式
1——解锁
2——设定密码
 */
int state = 1;
while(!kbhit()){
    byte num = keyBoard();
    //A  退格
    if(num == 0xDD){
        LCDDelete(currentLoc);
        if(currentLoc != 0x88){
            currentLoc--;
        }
        if(password.size() != 0)
            lastNum = table[password[password.size() - 1]];
    }
    //E  确认密码
    else if(num ==    0xEE){
        //state==1      解锁
        if(state == 1){
            bool flag = passwordRight();
            if(flag){
                printf("密码正确，开锁成功！\n");
                warning = 3;
```

```
            deviceStart();
            LCDDispSuccess();
            Sleep(2000);
            LCDDispSecondRow();
            currentLoc = reWritePassword();
        }
        else{
            printf("密码错误！\n");
            dispCurrentPassword();
            light_sound_warning();
            //LCDDispDeafuat();
            //Sleep(2000);
            LCDDispSecondRow();
             reWritePassword();
        }
        //if(currentLoc >= 0x89 && currentLoc <= 0x8E)
        //    LCDDispThirdRow(lastNum, currentLoc, false);
        currentLoc = 0x88;
    }
    //state==2    设定
    else if(state == 2){
        printf("设定\n");

        if(!r && password.size() == 6){
            for(int i=0; i<password.size(); i++)
                temp.push_back(password[i]);
            password.clear();
            reWritePassword();
            r = true;
            currentLoc = 0x88;
        }
        else if(r && password.size() == 6)
        {
            bool rr = true;
            printf("%d %d\n", password.size(), temp.size());
            for(int i = 0; i < 6; i++){
                if(temp[i] != password[i])
                    rr = false;
            }
            if(!rr)
            {
                printf("两次密码不一致！\n");
                reWritePassword();
                currentLoc = 0x88;
```

```
                    continue;
                }
                dispCurrentPassword();
                savePassword();
                readPassword();
                printf("设置成功！\n");
                LCDDispSetting();
                Sleep(2000);
                LCDDispSecondRow();
                 reWritePassword();
                currentLoc = 0x88;
                temp.clear();
                r = false;
                state = 1;
            }
        }
    }
    //C 切换模式——设定/输入密码
    else if(num == 0xE7){
        if(state == 1){
            LCDDispSecondRow_1();
            state = 2;
        }
        else if(state == 2){
            LCDDispSecondRow();
            state = 1;
        }

        if(currentLoc >= 0x89 && currentLoc <= 0x8E)
                LCDDispThirdRow(lastNum, currentLoc, false);
    }
    //显示数字
    else
    {
        if(currentLoc < 0x8E)
            currentLoc++;
        printf("currentLoc:%d\n", currentLoc);
        LCDDispThirdRow(num, currentLoc);
        lastNum = num;
    }
}

printf("按任意键退出\n");
while(!kbhit());
```

```
    return 0;
}

void init(){
    //8255
    PortWriteByte(PORT_CTL, 0x82);          //10000010
    //LCD
    LCDInit();
    //273
    PortWriteByte(PORT_273, 0x00);
}
void formatDisp(byte data){
    byte low = data & 0x0F;
    byte high = data & 0xF0;
    printf("data:%d    high:%d    low:%d\n", data, high>>4, low);
}



/*=================LCD 部分=========================*/
void cmdSetup() {
    /*
    8255C 端口：
    PC5---E
    PC6---R/W          0
    PC7---RS(D/I)      0
    */
    PortWriteByte(PORT_C, 0x00);
    Sleep(5);
    PortWriteByte(PORT_C, 0x20);          //0010 0000
    Sleep(5);
    PortWriteByte(PORT_C, 0x00);          //0000 0000
    Sleep(5);
}

void dataSetup() {
    /*
    8255C 端口：
    PC5---E
    PC6---R/W          0
    PC7---RS(D/I)      1
    */
    PortWriteByte(PORT_C, 0x80);          //1000 0000
    Sleep(5);
```

```
        PortWriteByte(PORT_C, 0xA0);            //1010 0000
        Sleep(5);
        PortWriteByte(PORT_C, 0x80);            //1000 0000
       Sleep(5);
}

void LCDClear() {
        //基本指令集——1.清除显示
        //将指令写入 LCD 数据口
        PortWriteByte(PORT_A, 0x01);            //0000 0001
        cmdSetup();
}

void LCDInit() {
       LCDClear();
        //基本指令集——6.功能设定
        PortWriteByte(PORT_A, 0x30);            //0011 0000
        cmdSetup();
       PortWriteByte(PORT_A, 0x0C);        //显示游标啥的
       cmdSetup();
}

void LCDDispFirstRow() {

        //第一行的汉字编码
        byte table[8][2] = {
                0xA1, 0xA0,
                0xA1, 0xBE,
                0xB3, 0xBF,
                0xA1, 0xBF,
                0xB5, 0xE7,
                0xD7, 0xD3,
                0xCB, 0xF8,
                0xA1, 0xA0
        };

        //第一行基地址
        byte baseAddr = 0x80;

        //显示第一行 8 个汉字
        for (int i = 0; i < 8; i++) {

                //调用 CMDSetup，设定 DDRAM 的地址
                PortWriteByte(PORT_A, baseAddr);            //0x90 是第一行首地址
```

```
            cmdSetup();
            baseAddr = baseAddr + 1;
            //送高字节
            PortWriteByte(PORT_A, table[i][0]);
            dataSetup();
            Sleep(20);
            //送低字节
            PortWriteByte(PORT_A, table[i][1]);
            dataSetup();
            Sleep(20);
        }
}

void LCDDispSecondRow(){

        //第二行的汉字编码
        byte table[8][2] = {
            0xC7, 0xEB,
            0xCA, 0xE4,
            0xC8, 0xEB,
            0xC3, 0xDC,
            0xC2, 0xEB,
            0xA3, 0xBA,
            0xA1, 0xA0,
            0xA1, 0xA0
        };

        //第二行基地址
        byte baseAddr = 0x90;

        //显示第二行 8 个汉字
        for (int i = 0; i < 8; i++) {

            //调用 CMDSetup，设定 DDRAM 的地址
            PortWriteByte(PORT_A, baseAddr);            //0x90 是第一行首地址
            cmdSetup();
            baseAddr = baseAddr + 1;
            //送高字节
            PortWriteByte(PORT_A, table[i][0]);
            dataSetup();
            Sleep(20);
            //送低字节
            PortWriteByte(PORT_A, table[i][1]);
            dataSetup();
```

```
            Sleep(20);
        }
}

void LCDDispSecondRow_1(){
    //第二行的汉字编码
     byte table[8][2] = {
            0xC9, 0xE8,
            0xD6, 0xC3,
            0xC3, 0xDC,
            0xC2, 0xEB,
            0xA3, 0xBA,
            0xA1, 0xA0,
            0xA1, 0xA0,
            0xA1, 0xA0
     };

     //第二行基地址
     byte baseAddr = 0x90;

     //显示第二行 8 个汉字
     for (int i = 0; i < 8; i++) {

            //调用 CMDSetup，设定 DDRAM 的地址
            PortWriteByte(PORT_A, baseAddr);            //0x90 是第一行首地址
            cmdSetup();
            baseAddr = baseAddr + 1;
            //送高字节
            PortWriteByte(PORT_A, table[i][0]);
            dataSetup();
            Sleep(20);
            //送低字节
            PortWriteByte(PORT_A, table[i][1]);
            dataSetup();
            Sleep(20);
     }
}

void LCDDelete(byte currentLoc){
    PortWriteByte(PORT_A, currentLoc);
    cmdSetup();
    PortWriteByte(PORT_A, 0xA1);
    dataSetup();
    PortWriteByte(PORT_A, 0xA0);
```

```
    dataSetup();
    if(password.size() != 0)
        password.pop_back();
}

void LCDDispThirdRow(byte data, byte currentLoc, bool flag){
    //数字编码
     byte table[10][2] = {
          0xA3, 0xB0,
          0xA3, 0xB1,
          0xA3, 0xB2,
        0xA3, 0xB3,
        0xA3, 0xB4,
        0xA3, 0xB5,
        0xA3, 0xB6,
        0xA3, 0xB7,
        0xA3, 0xB8,
        0xA3, 0xB9,
     };
    //键盘与数字的映射
    byte key_val_map[10] = {0x77, 0x7B, 0x7D, 0x7E, 0xB7, 0xBB, 0xBD, 0xBE, 0xD7,
0xDB};
    //要显示的数字
    int num = -1;
    for(int i = 0; i < 10; i++){
        if(key_val_map[i] == data)
            num = i;
    }

    //添加记录
    if(flag){
        if(password.size() == 6)
            password.pop_back();
        password.push_back(num);
    }

    //指定位置显示指定数字
    printf("In fun:%d\n", currentLoc);
    PortWriteByte(PORT_A, currentLoc);
    cmdSetup();
    //PortWriteByte(PORT_A, table[num][0]);
    PortWriteByte(PORT_A, 0xA3);
    dataSetup();
    //PortWriteByte(PORT_A, table[num][1]);
```

```
    PortWriteByte(PORT_A, 0xAA);
    dataSetup();
}

byte reWritePassword(){
    byte currentLoc = 0x89;
    while(currentLoc <= 0x8E){
        PortWriteByte(PORT_A, currentLoc);
        cmdSetup();
        PortWriteByte(PORT_A, 0xA1);
        dataSetup();
        PortWriteByte(PORT_A, 0xA0);
        dataSetup();
        currentLoc++;
    }
    password.clear();
    return currentLoc - 1;
}

void LCDDispSuccess(){
    //汉字编码
     byte table[16][2] = {
          0xCC, 0xE1,
          0xCA, 0xBE,
          0xA3, 0xBA,
          0xA1, 0xA0,
          0xA1, 0xA0,
        0xA1, 0xA0,
        0xA1, 0xA0,
          0xA1, 0xA0,

        0xA1, 0xA0,
        0xC3, 0xDC,
          0xC2, 0xEB,
        0xD5, 0xFD,
        0xC8, 0xB7,
        0xA3, 0xA1,
        0xA1, 0xA0,
        0xA1, 0xA0
     };

     //第二行基地址
     byte baseAddr = 0x90;
```

```
    //显示
    for (int i = 0; i < 16; i++) {

        //调用 CMDSetup，设定 DDRAM 的地址
        PortWriteByte(PORT_A, baseAddr);          //0x90 是第一行首地址
        cmdSetup();
        baseAddr = baseAddr + 1;
        //送高字节
        PortWriteByte(PORT_A, table[i][0]);
        dataSetup();
        Sleep(20);
        //送低字节
        PortWriteByte(PORT_A, table[i][1]);
        dataSetup();
        Sleep(20);

       //换行
       if(baseAddr == 0x98)
           baseAddr = 0x88;
    }
}

void LCDDispDeafuat(){
    //汉字编码
    byte table[16][2] = {
          0xCC, 0xE1,
          0xCA, 0xBE,
          0xA3, 0xBA,
          0xA1, 0xA0,
          0xA1, 0xA0,
       0xA1, 0xA0,
       0xA1, 0xA0,
        0xA1, 0xA0,

       0xA1, 0xA0,
       0xC3, 0xDC,
        0xC2, 0xEB,
       0xB4, 0xED,
       0xCE, 0xF3,
       0xA3, 0xA1,
       0xA1, 0xA0,
       0xA1, 0xA0
    };
```

```
        //第二行基地址
        byte baseAddr = 0x90;

        //显示
        for (int i = 0; i < 16; i++) {

            //调用 CMDSetup，设定 DDRAM 的地址
            PortWriteByte(PORT_A, baseAddr);            //0x90 是第一行首地址
            cmdSetup();
            baseAddr = baseAddr + 1;
            //送高字节
            PortWriteByte(PORT_A, table[i][0]);
            dataSetup();
            Sleep(20);
            //送低字节
            PortWriteByte(PORT_A, table[i][1]);
            dataSetup();
            Sleep(20);

           //换行
           if(baseAddr == 0x98)
                baseAddr = 0x88;
        }
}

void LCDDispSetting(){
      //汉字编码
       byte table[16][2] = {
             0xCC, 0xE1,
             0xCA, 0xBE,
             0xA3, 0xBA,
             0xA1, 0xA0,
             0xA1, 0xA0,
           0xA1, 0xA0,
           0xA1, 0xA0,
            0xA1, 0xA0,

           0xA1, 0xA0,
           0xC9, 0xE8,
            0xD6, 0xC3,
           0xB3, 0xC9,
           0xB9, 0xA6,
           0xA3, 0xA1,
           0xA1, 0xA0,
```

```
        0xA1, 0xA0
    };

    //第二行基地址
    byte baseAddr = 0x90;

    //显示
    for (int i = 0; i < 16; i++) {

        //调用 CMDSetup，设定 DDRAM 的地址
        PortWriteByte(PORT_A, baseAddr);            //0x90 是第一行首地址
        cmdSetup();
        baseAddr = baseAddr + 1;
        //送高字节
        PortWriteByte(PORT_A, table[i][0]);
        dataSetup();
        Sleep(20);
        //送低字节
        PortWriteByte(PORT_A, table[i][1]);
        dataSetup();
        Sleep(20);

        //换行
        if(baseAddr == 0x98)
            baseAddr = 0x88;
    }
}

/*=================键盘部分=======================*/
byte keyBoard()
{
    byte data;    //寄存器

    //向所有行送 0
    PortWriteByte(PORT_C, 0x00);
    //读列，查看所有键是否松开
    while (1) {
        if (!PortReadByte(PORT_B, &data)) {
            printf("QQQQ");
        }
        //读取四根列线，全为高电位则所有键松开。
        data = data & 0x0f;
        if (data != 0x0f)       //有键按下则进入循环
            continue;
```

```
    //判断是否有键按下
    while (1) {

        if (!PortReadByte(PORT_B, &data)) {
            printf("读数据错误\n");
        }
        data = data & 0x0f;
        if (data != 0x0f)
            break;
    }
    //有键按下，延迟 20ms，消除抖动
    Sleep(20);
    //再查列，是否仍有键按下
    if (!PortReadByte(PORT_B, &data)) {
        printf("第二次错误\n");
    }
    data = data & 0x0f;
    if (data == 0x0f)       //无键按下则重新读列
        continue;
    else {
        break;                  //有键按下则退出大循环
    }
}
//仍有键按下，确定键位
byte location = 0xFE;
while (1) {

    //使一行变为低电平
    PortWriteByte(PORT_C, location);
    //读入键盘状态
    PortReadByte(PORT_B, &data);
    //只截取列值
    data = data & 0x0F;
    //是否均为 1？
    if (data == 0x0F) {
        //均为 1，不是该行按键按下
        location = location << 1;           //左移，判断下一行
        location = location + 1;
        continue;
    }
    else {
        //不均为 1，该行按键被按下
        //while(1) printf("编码！！！\n");
        return encode();
```

```
            }
        }
        return 0x00;
}

//键盘编码函数 01110111
byte encode() {
        byte data;
        byte table[16] = { 0x77, 0x7B, 0x7D, 0x7E, 0xB7, 0xBB, 0xBD, 0xBE,
                                0xD7, 0xDB, 0xDD, 0xDE, 0xE7, 0xEB, 0xED, 0xEE };
        PortReadByte(PORT_B, &data);
        byte high = (data & 0xF0) >> 4;
        byte low    = data & 0x0F;
        printf("%d %d %d\n", data, high, low);
        for (int i = 0; i < 16; i++) {
            if (table[i] == data) {
                PortWriteByte(PORT_A, data);
                return data;
            }
        }
        return 0x00;
}

/*=================密码部分=======================*/
void readPassword(){
        PASSWORD.clear();
        FILE* fp = fopen("code.txt", "r");
        if(fp == NULL){
            printf("open error\n");
            return;
        }
        char ch;
        while((ch = fgetc(fp)) != EOF){
            PASSWORD.push_back(Ret(ch - '0'));
        }
        fclose(fp);
        printf("读取密码成功!\n");
}

bool passwordRight(){
        printf("当前密码：");
        for(int j=0; j<password.size();j++)
            cout<<password[j]<<" ";
        printf("\n");
```

```
    if(password.size() != 6)
        return false;
    //判断海关密码
    bool flag = true;
    for(int k = 0; k < customsCode.size(); k++){
        if(password[k] != customsCode[k])
            flag = false;
    }
    if(flag)
        return true;

    //判断普通密码
    for(int i = 0; i < password.size(); i++){
        if(password[i] != PASSWORD[i])
            return false;
    }
    return true;
}

void setCustomsCode(){
    customsCode.push_back(0);
    customsCode.push_back(1);
    customsCode.push_back(1);
    customsCode.push_back(0);
    customsCode.push_back(1);
    customsCode.push_back(6);
}
void savePassword(){
    FILE* fp = fopen("code.txt", "w");
    if(fp == NULL){
        printf("open error\n");
        return;
    }
    for(int i = 0; i < 6; i++)
        fputc(Go(password[i]) + '0', fp);
    fclose(fp);
    printf("密码保存成功！\n");
}

int Go(int num){
    return num % 2 == 0 ? num + 1 : num - 1;
}

int Ret(int num){
```

```
    return num % 2 == 0 ? num + 1 : num - 1;
}
/*=================继电器部分=======================*/
void deviceStart(){
    if(!PortWriteByte(PORT_273, 0x04)){       //00000100
        printf("继电器启动失败\n");
    }
    else
        printf("继电器启动成功\n");
}
/*=================声光报警部分=======================*/
void light_sound_warning(){
    if(!PortWriteByte(PORT_273, 0x03)){       //00010000
        printf("声光报警启动失败\n");
    }
    else
        printf("声光报警启动！\n");
    warning--;
    if(warning == 0){
        LCDWarning();
        warning = 1;
    }
    else
        LCDDispDeafuat();
    Sleep(2000);
    PortWriteByte(PORT_273, 0x00);
    for(int i = 0; i < 16; i++){
        LCDDelete(0x90 + i);
    }
    LCDDelete(0x88);
}

//测试
void dispCurrentPassword(){
    printf("当前密码：");
    for(int i = 0; i < password.size(); i++)
        printf("%d ", password[i]);
    printf("\n");
}


/*=================娴嬭瘜闆ㄥ垛=======================*/
void LCDWarning(){
     LCDDispFirstRow();
```

```
LCDDelete(0x92);
LCDDelete(0x93);
LCDDelete(0x94);
LCDDelete(0x95);

//调用 CMDSetup，设定 DDRAM 的地址
 PortWriteByte(PORT_A, 0x90);          //0x90 是第一行首地址
 cmdSetup();
 //送高字节
 PortWriteByte(PORT_A, 0xBE);
 dataSetup();
 Sleep(20);
 //送低字节
 PortWriteByte(PORT_A, 0xAF);
 dataSetup();
 Sleep(20);
//调用 CMDSetup，设定 DDRAM 的地址
 PortWriteByte(PORT_A, 0x91);          //0x90 是第一行首地址
 cmdSetup();
 //送高字节
 PortWriteByte(PORT_A, 0xB8);
 dataSetup();
 Sleep(20);
 //送低字节
 PortWriteByte(PORT_A, 0xE6);
 dataSetup();
 Sleep(20);
 //警告
 //密码错误过多!
 //密码锁已被锁定!
 byte table[16][2] = {
        0xB4, 0xED,
        0xCE, 0xF3,
        0xB4, 0xCE,
        0xCA, 0xFD,
        0xB9, 0xFD,
        0xB6, 0xE0,
        0xA3, 0xA1,
        0xA1, 0xA0,

        0xC3, 0xDC,
        0xC2, 0xEB,
        0xCB, 0xF8,
        0xD2, 0xD1,
```

```
            0xB1, 0xBB,
            0xCB, 0xF8,
            0xB6, 0xA8,
            0xA3, 0xA1,
        };
        byte baseAddr = 0x88;
        for (int i = 0; i < 16; i++) {
            PortWriteByte(PORT_A, baseAddr);
            cmdSetup();
            baseAddr = baseAddr + 1;
            if(baseAddr == 0x90) baseAddr = 0x98;
            PortWriteByte(PORT_A, table[i][0]);
            dataSetup();
            Sleep(20);
            PortWriteByte(PORT_A, table[i][1]);
            dataSetup();
            Sleep(20);
        }
}
```