项目开发

电 子 密 码 锁

谷林海 等

No. 1

电子密码锁需求分析

题目： 电子密码锁的设计

小组成员：

谷林海

顾友峰

兰顺福

葛振涛

目录

1、引言

1.1 编写目的

信息电子技术高度发展的今天，利用电子产品取代传统的产品已经成为一种必然的趋势，大量电子产品的应用使得用户可以更加便捷安全的处理相关事项。电子密码锁由于其保密性高，使用灵活性好，安全系数高，从而受到了广大用户的信赖。

编写该需求分析的目的是为设计一套电子密码锁，通过撰写该需求分析使开发者对开发电子密码锁当中所要解决的问题进行详细的分析弄清楚问题的要求，包括需要实现怎么输入，最后所做的电子密码锁要实现哪些的功能，以及相关的性能指标等等。本文档可供用户、系统分析人员、程序设计人员及系统测试人员阅读和参考。

1.2 背景

这次待开发的系统名称是：电子密码锁
开发者：谷林海、顾友峰、兰顺福、葛振涛
该系统采用 Keil C 和 Protues 软件进行开发和仿真，并用 protel 进行相关 PCB 板的设计，通过仿真结果来模拟实物运行的过程。

1.3 参考资料

《新概念 51 单片机 C 语言教程：入门、提高、开发、拓展全攻略》电子工业出版社 郭天祥
《51 单片机典型应用开发范例大全》 中国铁道出版社 郑峰
《单片机原理及其接口技术（第 3 版）》 清华大学出版社 胡汉才

2、任务概述

2.1 开发目的

当今社会计算机的迅猛发展，数字电子技术已经进入人们生活的各个领域，电子产品也渐渐地应用于安全防范领域。有需求就必有发展，不同种类的锁也应运而生，有密码锁、磁性锁、电子锁、激光锁、声控锁等等，它们的实现在传统

钥匙的基础上加了一组或多组密码，不同声音，不同磁场，不同声波，不同光束光波，不同图像（如指纹、眼底视网膜等）来控制锁的开启。

电子密码锁由于其保密性高，使用灵活性好，安全系数高，受到了广大用户的信赖。电子密码锁是一种通过密码输入来控制电路或是芯片工作，从而控制机械开关的闭合，完成开锁、闭锁任务的电子产品。它的种类很多，有简易的电路产品，也有基于芯片的性价比较高的产品。现在应用较广的电子密码锁是以芯片为核心，通过编程来实现的。其性能和安全性已大大超过了机械锁。其特点如下：

1）保密性好，编码量多。随机开锁成功率几乎为零。

2）密码可变，用户可以随时更改密码，防止密码被盗，同时也可以避免因人员的更替而使锁的密级下降。

3）误码输入保护，当输入密码多次错误时，报警系统自动启动。

4）无活动零件，不会磨损，寿命长。

5）使用灵活性好，不像机械锁必须佩带钥匙才能开锁。

6）电子密码锁操作简单易行，一学即会。

现在电子密码锁出现了很多的种类，功能日益强大，使用更加方便，安全保密性更强，由以前的单密码输入发展到现在的，密码加感应元件，实现了真正的电子加密，用户只有密码或电子钥匙中的一样，是打不开锁的，随着电子元件的发展及人们对保密性需求的提高出现了越来越多的电子密码锁。

2.2 总体目标

1、设计一套电子密码锁，通过设计训练，了解产品设计的整个过程。

2、掌握电子产品设计中的基本技术。

3、掌握电子产品设计中相关软件工具的使用。

4、掌握产品设计过程中的文档写作方法和规范要求。

5、掌握电子产品的开发、调试方法。

2.3 基本功能

设计一套基本的密码锁，基本功能有：

1、设置固定 6 位的密码，密码通过键盘输入，若密码正确，则将锁打开。若密码不正确，则重新输入，若三次没有输入正确的密码，则键盘自动锁定。

2、具有开锁、超时报警、超次锁定、管理员解密、设置密码、修改用户密码基本的密码锁的功能，可以对报警时间、超时时间、输入次数、解锁时间进行设置。

3、密码可以由用户自己修改设定，锁打开后才能修改密码。修改密码之前必须再次输入密码，在输入新密码时候需要二次确认，以防止误操作。

4、密码的输入、清除、更改功能：（a）密码输入功能：按下一个数字键，一个“*”就显示在最右边的数码管上，同时将先前输入的所有“*”向左移动一位。（b）当按下左键时，清楚前面的一位数值（c）密码更改功能：将输入的值作为新的密码。

5、开锁功能：当按下开锁键，系统将输入与密码进行检查核对，如果正确锁打开，否则不打开，并发出报警提示。如果键入完密码后不按确认键系统会当做放弃开锁处理。

6、可以设置超时时间与自锁时间；

7、具有掉电存储、光提示等功能。

8、具有显示时钟的功能，并且可以修改时间。

9、各个操作，不管成功、失败都有指示作为交互。

2.4 运行环境

2.3.1 设备

用于系统仿真用的计算机一台，以及 Keil C 开发工具、Proteus 仿真软件和 Protel 仿真软件。

2.3.2 支持软件

Proteus 仿真软件、Keil C 编程开发软件以及 Protel 仿真软件。

Keil C51 是美国 Keil Software 公司出品的 51 系列兼容单片机 C 语言软件开发系统，与汇编相比，C 语言在功能上、结构性、可读性、可维护性上有明显的优势，因而易学易用。用过汇编语言后再使用 C 来开发，体会更加深刻。

Keil C51 软件提供丰富的库函数和功能强大的集成开发调试工具，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。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 electronics 公司出版的 EDA 工具软件（该软件中国总代理为广州风标电子技术有限公司）。它不仅具有其它 EDA 工具软件的仿真功能，还能仿真单片机及外围器件。它是目前最好的仿真单片机及外围器

件的工具。虽然目前国内推广刚起步，但已受到单片机爱好者、从事单片机教学的教师、致力于单片机开发应用的科技工作者的青睐。Proteus 是世界上著名的 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。是目前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 年即将增加 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 MPLAB 等多种编译器。

Ptotel 软件是 Altium 公司在 80 年代末推出的 EDA 软件，在电子行业的 CAD 软件中，它当之无愧地排在众多 EDA 软件的前面，是电子设计者的首选软件，它较早就在国内开始使用，在国内的普及率也最高，有些高校的电子专业还专门开设了课程来学习它，几乎所有的电子公司都要用到它，许多大公司在招聘电子设计人才时在其条件栏上常会写着要求会使用 Ptotel。

2.5 功能模块划分

该系统由几个子模块共同构成：

按键模块：用户可以通过键盘输入数字到微处理器模块中，当系统判断为正确密码后，系统自动解锁。使用确定按键结束密码输入，使用返回键可以返回前面某处重新输入密码。使用上下键选择需要设置的项目。

报警模块：首先，在进入输入密码界面时，如果在 30 秒内没有完成正确的密码输入界面会自动跳转回开始欢迎界面；其次，在密码输入错误时系统会提示“密码错误!!”并且提示剩余输入次数，如果连续 3 次输入错误 系统会提示“非法入侵，键盘已经锁定”，此时数字键盘被锁定，无法完成数字键操作，用户只能通过点击“返回”键，返回到菜单主页。

液晶显示模块：系统可以采用液晶显示界面及相关的信息。主要显示内容有：1、欢迎使用信息 2、当前时间（精确到秒） 3、开锁提示信息 4、报警提示信息 5、密码修改信息 6、相关管理信息。

电子开锁模块：当用户在指定次数内输入正确密码后，微处理器模块通过控制电子开锁模块开锁；否则电子开锁模块一直保持系统闭锁状态

时钟模块：该模块和微处理器模块相连，主要用来为系统提供时间的查询和修改。

No. 2

电子密码锁总体设计

题目： 电子密码锁的设计

小组成员：

谷林海

顾友峰

兰顺福

葛振涛

目录

1.概述

随着电子技术和计算机技术的飞速发展，单片机性能不断完善，性能价格比显著提高，技术日趋完善。由于单片机具有体积小、重量轻、价格便宜、功耗低、控制功能强及运算速度快等特点，因而在国民经济建设、军事及家用电器等各个领域均得到了广泛的应用。本设计利用单片机及附加电子元器件实现数据采集和控制算法，来完成某一实际功能，检验并提高同学对整体电路设计和把握能力，了解单片机系统设计流程，以及电路板的实际制作和调试能力。同时也加强对数字电路、单片机和微机原理等课程知识的实际应用能力，也为同类产品的进一步发展奠定理论和实践基础。

随着人们生活水平的提高和安全意识的加强，对安全的要求也就越来越高。锁自古以来就是把守护门的铁将军，人们对它要求甚高，既要安全可靠的防盗，又要使用方便，这也是制锁者长期以来研制的主题。随着电子技术的发展，各类电子产品应运而生，电子密码锁就是其中之一。据有关资料介绍，电子密码锁的研究从 20 世纪 30 年代就开始了，在一些特殊场所早就有所应用。这种锁是通过键盘输入一组密码完成开锁过程。研究这种锁的初衷，就是为提高锁的安全性。由于电子锁的密钥量（密码量）极大，可以与机械锁配合使用，并且可以避免因钥匙被仿制而留下安全隐患。电子锁只需记住一组密码，无需携带金属钥匙，免除了人们携带金属钥匙的烦恼，而被越来越多的人所欣赏。电子锁的种类繁多，例如数码锁，指纹锁，磁卡锁，IC 卡锁，生物锁等。但较实用的还是按键式电子密码锁。

20 世纪 80 年代后，随着电子锁专用集成电路的出现，电子锁的体积缩小，可靠性提高，成本较高，是适合使用在安全性要求较高的场合，且需要有电源提供能量，使用还局限在一定范围，难以普及，所以对它的研究一直没有明显进展。

在我国电子锁整体水平尚处于国际七十年代左右，电子密码锁的成本还很高，市场上仍以按键电子锁为主，按键式和卡片钥匙式电子锁已引进国际先进水平，现国内有几个厂生产供应市场。但国内自行研制开发的电子锁，其市场结构尚未形成，应用还不广泛。国内的不少企业也引进了世界上先进的技术，发展前景非常可观。希望通过不断的努力，使电子密码锁在我国也能得到广泛应用。

此次设计采用以 AT89C51 单片机为核心的单片机控制方案。选用单片机 AT89C51 作为本设计的核心元件，利用单片机灵活的编程设计和丰富的 I/O 端口，及其控制的准确性，实现基本的密码锁功能。在单片机的外围电路外接输入键盘用于密码的输入和一些功能的控制，外接字符型液晶显示器用于显示。

2.系统设计

2.1 系统总设计结构

系统总体设计结构图，由液晶显示，单片机控制电路，键盘输入电路，报警电路，电子开锁电路组成。总体设计结构如图 2-1：

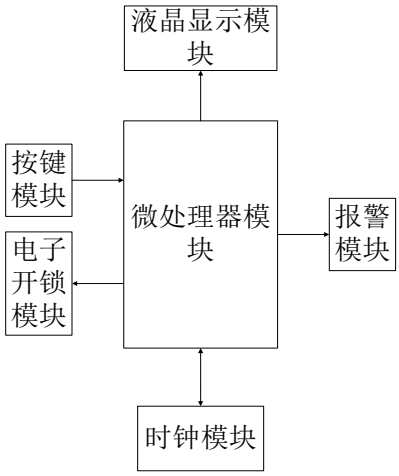


图 2-1 系统总体设计图

（1） AT89C51 单片机控制模块
（2） 报警模块
（3） 电子开锁模块
（4） 按键模块
（5） 时钟模块
（6） 液晶显示模块

2.2 硬件电路功能

键盘电路：用户可以通过键盘输入数字到单片机中，当系统判断为正确密码后，系统自动解锁。使用向左退格键可以清除当前密码，进而重新输入密码。当用户每输错一次密码，显示屏提示剩余输入次数，当三次输入错误，键盘自锁。用户输入正确密码后，可以使用上下键选择需要设置的项目。

报警电路：首先，按菜单键进入输入密码界面时，如果在 30 秒内没有完成正确的密码输入界面会自动跳转回开始欢迎界面；其次，在密码输入错误时系统会提示“密码输入错误”并且提示剩余输入次数，此时 LED 亮红灯，如果连续 3 次输入错误 系统会提示“非法入侵，键盘已锁定”，此时数字键盘被锁定，无法完成数字键操作，用户只能等待系统自行返回菜单主页。

液晶显示电路：系统采用液晶显示界面。液晶屏显示内容有：1、欢迎使用信息 2、当前时间（精确到秒） 3、开锁提示信息 4、报警提示信息 5、密码修改信息

电子开锁电路：当用户在指定次数内输入正确密码后，单片机通过控制电子开锁电路开锁；否则一直保持系统闭锁状态

时钟电路：该电路和单片机相连，主要用来为系统提供时间和修改。如 30 秒输入倒计时、实时获取时间

3. 设计方案简介

本设计采用以单片机为核心的控制方案。

由于单片机种类繁多，各种型号都有其一定的应用环境，因此在选用时要多加比较，合理选择，以期获得最佳的性价比。一般来说在选取单片机时从下面几个方面考虑：性能、存储器、运行速度、I/O 口、定时/计数器、串行接口、模拟电路功能、工作电压、功耗、封装形式、抗干扰性、保密性，除了以上的一些的还有一些最基本的比如：中断源的数量和优先级、工作温度范围、有没有低电压检测功能、单片机内有无时钟振荡器、有无上电复位功能等。在开发过程中单片机还受到：开发工具、编程器、开发成本、开发人员的适应性、技术支持和服务等等因素。

基于以上因素本设计选用单片机 AT89C51 作为本设计的核心元件，利用单片机灵活的编程设计和丰富的 I/O 端口，及其控制的准确性，实现基本的密码锁功能。在单片机的外围电路外接输入键盘用于密码的输入和一些功能的控制，外接 AT24C02 芯片用于密码的存储，外接液晶显示器用于显示作用。当用户需要开锁时，先按键盘开锁键之后按键盘的数字键 0－9 输入密码。密码输完后按下确认键，如果密码输入正确则开锁，不正确，显示密码错误，重新输入密码，当输三次密码都错误则发出报警；当用户需要修改密码时，先按下键盘设置键后输入原来的密码，只有当输入的原密码正确后才能设置新密码。新密码输入无误后按确认键使新密码将得到存储，密码修改成功。当用户在开锁状态下时，可以通过密码重置键进入密码重置模式，连续两次输入密码正确之后，重置密码成功，原密码失效，新密码生效。当用户在开锁状态下时，可以通过进入管理员权限修改模式，可以修改以上权限的时间限定长度。

4.各电路设计

4.1 电子开锁电路设计

在本次设计中，基于节省材料的原则，暂时用发光二极管代替电磁锁，发光管亮，表示开锁；灭，表示没有开锁。当 P3.7 口输出低电平时，二极管发绿光，表示开锁。

4.2 按键电路设计

方案一：使用独立式按键来控制 使用独立式按键来控制数码管的显示，这样需要很多的按键，每个按键实现一个能，易于控制，程序编写简单，但是每个按键都要接上拉电阻，占用了单片机大量的 I/O 接口资源，要对单片机外扩 I/O 口，并且在电路焊接方面又不方便，还要浪费大量的资源，提高了成本。

方案二：采用矩阵式键来控制 把按键按行列组成矩阵，在行列交点上都对应有一个键，这样使用的按键要少，为判定有无键被按下以及被按键的位置，这种称为键扫描法。这样虽然提高了编程难度，但是节约了单片机大量的 I/O 口，免去了上拉电阻为焊接带来了方便，提高了整块电路板的美观度。

由于本设计所用到的按键数量较多而不适合用独立按键式键盘，所以考虑采用的是矩阵式按键键盘。其原理图如下：

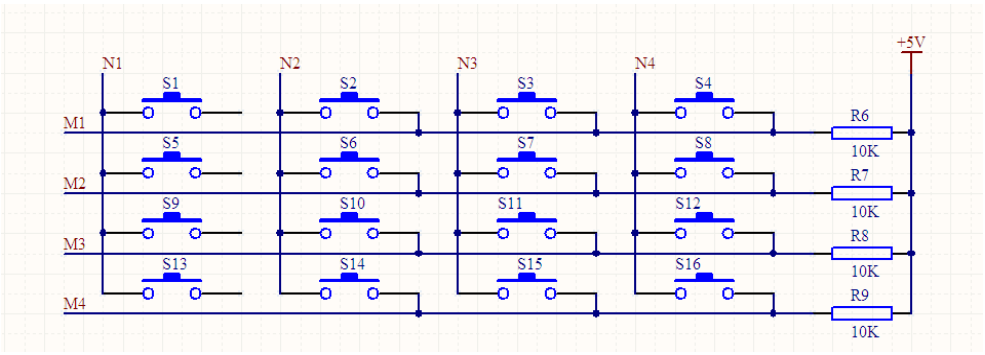


图 4.1 矩阵键盘

每一条水平（行线）与垂直线（列线）的交叉处不相通，而是通过一个按键来连通，利用这种行列式矩阵结构只需要 N 条行线和 M 条列线，即可组成具有 N×M 个按键的键盘。

在这种行列式矩阵键盘非键盘编码的单片机系统中，键盘处理程序首先执行等待按键并确认有无按键按下的程序段。

当确认有按键按下后，下一步就要识别哪一个按键按下。对键的识别通常有两种方法：一种是常用的逐行扫描查询法；另一种是速度较快的线反转法。

对照图 4.1 所示的 4×4 键盘，说明线反转个工作原理。

首先辨别键盘中有无键按下，有单片机 I/O 口向键盘送全扫描字，然后读入行线状态来判断。方法是：向行线输出全扫描字 00H，把全部列线置为低电平，然后将列线的电平状态读入累加器 A 中。如果有按键按下，总会有一根行线电平被拉至低电平从而使行线不全为 1。

判断键盘中哪一个键被按下使通过将列线逐列置低电平后，检查行输入状态来实现的。方法是：依次给列线送低电平，然后查所有行线状态，如果全为 1，则所按下的键不在此列；如果不全为 1，则所按下的键必在此列，而且是在与零电平行线相交的交点上的那个键。

具体的功能设计如表 4.1：

表 4.1 按键功能

按 键	键 名	功 能 说 明
1—9 键	数字键	输入密码
* 键	菜单键	进入输入密码界面
上下左右键	选择键	选择不同设置界面
# 键	返回键	使显示器返回上一菜单

4.3 报警电路

报警模块有喇叭，三极管电阻、以及三只 LED 灯等组成。当输入三次错误后，报警电路会会发出报警声提示密码错误。报警部分由陶瓷压电发声装置及外围电路组成，加电后不发声，当有键按下时，“叮”声，每按一下，发声一次，密码正确时，P1.5 输出低电平，三极管 Q3 发射极和集电极不导通，蜂鸣器不发声直接开锁，当密码输入错误时，单片机的 P1.5 引脚为高电平，三极管 Q3 发射极和集电极导通使喇叭发出噪鸣声报警。如图 4-2 所示：

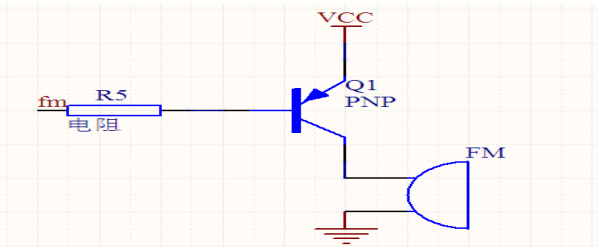


图 4-2 报警电路原理图

4.4 液晶显示电路设计

显示系统的方案比较

方案 1：用数码管或 LED 显示。

方案 2：用液晶 1602 显示。

方案 3：用液晶 12864 显示。

时钟和温度的显示可以用数码管或 LED，而且价格便宜。但是数码管的只能显示简单的设计的系统，与我们设计要求也不相符。有汉字需要显示，所以液晶显示器比较好，它能显示更多的数据，用 1602 液晶显示数据有限，显示数据的可读性不好，而用可以显示汉字的 12864 液晶显示器则可以增加显示信息的可读性。因此我们选择方案三密码锁采用的显示方式是用液晶显示，该显示器应用广泛，功能强大，性价比高。可通过单片机把需要的的内容输入到 12864 中显示。

液晶屏上如何显示一些汉字或图画，这也许是所有 LCD12864 初学者都最先思考的一个问题。再数字电路中，所有数据都是由 0 和 1 保存的，同样 LCD 也利用了这一方法。再点阵 LCD 上显示的只有两种颜色，因此可利用 0 和 1 来表示这两种颜色。假设空格是由 16*16 个 0 组成的，再显示 16*16 的字体时，将其中某些点置为 1 便可再视觉上形成一个汉字，这些二进制数称为代位码。而这些由 0 和 1 转换而成的 16 进制数据便是字模。不同的汉字有不同的字模，相同的汉字不同的字体也有不同的字模。而将字模设为 16*16 像素是因为这样基本可以将汉字显示清楚准确，更高像素则更为清楚准确，但是却更多地占用了 LCD 的面积。与汉字不同的是，一个字符只需要 16*8 的像素便即可。如何将这 16*16 或者 16*8 个 0、1 保存下来是初学者所需要了解的，假设要再 LCD12864 屏幕上准确正确的显示出汉字，则需要将 16*16 的汉字分为两行，每行由 16 列组成，这 16 列每列存再 8 个点，用 2 位 16 进制数（8 位二进制数）表示这 8 个点，16 个 16 进制数可表示 1 行，32 个 16 进制数则能表示整个汉字。通过 LCD12864，则可将这些字模信息还原成汉字或图像。

主要显示内容有：1、欢迎使用信息 2、当前时间（精确到秒） 3、开锁提示信息 4、报警提示信息 5、密码修改信息 6、相关管理信息。

其与单片机的接口电路如下

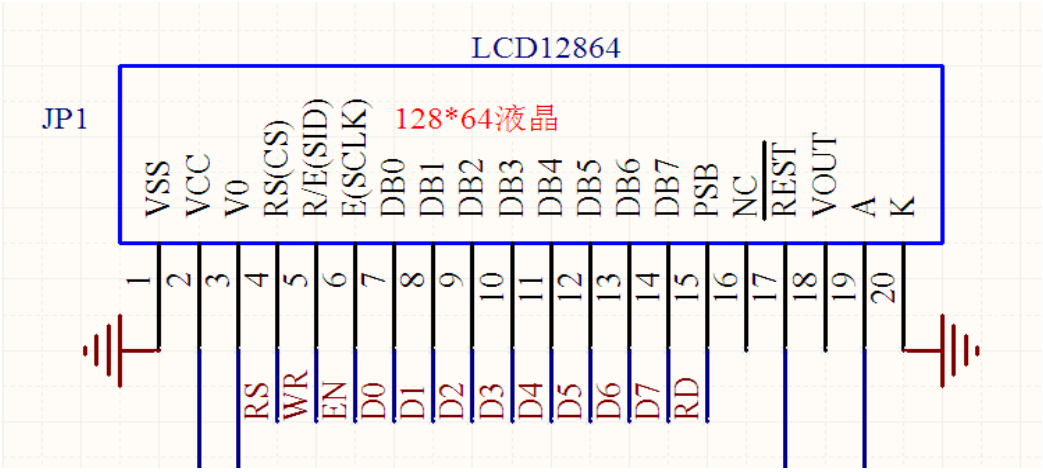


图 4-3 12864 液晶显示接口电路图

4.5 时钟电路设计

方案 1：采用 DS1302 为计时时钟芯片

该芯片是串行电路，与单片机接口简单，且能实时获取点电脑时间，但需另备电池和 32.768kHz 晶振，因焊接工艺和晶振质量等原因会导致精度降低。

方案 2：采用采用单片机的内部定时器来实现时间的显示

采用单片机的内部定时器来实现时间的显示，程序设计比较复杂，硬件电路简单，但硬件电路的成功率低，精度差。

综合考虑，本设计采用 DS1302 作为计时时钟。

我们采用具有涓细电流充电能力的低功耗实时时钟电路 DS1302。它可以对年、月、日、周日、时、分、秒进行计时，且具有闰年补偿等多种功能。它采用主电源和备用电源双电源供电。它的工作电压范围 2.0~5.5V，在 2.2V 时，小于 300nA。它内部含有 31 个字节的静态 RAM，可提供用户访问。

DS1302 可以对年、月、日、周日、时、分、秒进行计时,可以达到我们设计的基本的要求。内部的寄存器为我们调时，闹钟定时提供了寄存空间。备用电源也实现了当系统断电我们采用具有涓细电流充电能力的低功耗实时时钟电路 DS1302。它可以对年、月、日、周日、时、分、秒进行计时，且具有闰年补偿等多种功能。它采用主电源和备用电源双电源供电。它的工作电压范围 2.0~5.5V，在 2.2V 时，小于 300nA。它内部含有 31 个字节的静态 RAM，可提供用户访问。DS1302 可以对年、月、日、周日、时、分、秒进行计时,可以达到我们设计的基本的要求。内部的寄存器为我们调时，闹钟定时提供了寄存空间。备用电源也实

现了当系统断电后，时钟仍然可以保持。而且它是串行接口，与单片机通信所需要的接口少。不像 DS12887 等芯片并行通信需要很多 IO 口。

4.6 单片机主控制电路的设计

4.6.1 单片机选择：

方案 1　采用 51 系列单片机作为系统控制器

单片机算术运算功能强　软件编程灵活、自由度大　可用软件编程实现各种算法和逻辑控制。由于其功耗低、体积较小、技术成熟和成本低等优点　在各个领域应用广泛。而且抗干扰性能好。

方案 2　采用凌阳系列单片机作为系统的控制器

凌阳系列单片机可以实现各种复杂的逻辑功能，模块大，密度高。它将所有器件集成在一块芯片上减少了体积，提高了稳定性。凌阳系列单片机提高了系统的处理速度，适合作为大规模实时系统的控制核心。 因 51 单片机价格比凌阳系列低得多且本设计不需要很高的处理速度从经济和方便使用角度考虑本设计选择了方案 1。

4.6.2 51 系列单片机概述：

AT89C51 是一种带 4K 字节闪烁可编程可擦除只读存储器（FPEROM—Falsh Programmable and Erasable Read Only Memory）的低电压，高性能 CMOS8 位微处理器，俗称单片机。AT89C51 是一种带 4K 字节闪烁可编程可擦除只读存储器的单片机。单片机的可擦除只读存储器可以反复擦除 1000 次。该器件采用 ATMEL 高密度非易失存储器制造技术制造，与工业标准的 MCS-51 指令集和输出管脚相兼容。由于将多功能 8 位 CPU 和闪烁存储器组合在单个芯片中，ATMEL 的 AT89C51 是一种高效微控制器，为很多嵌入式控制系统提供了一种灵活性高且价廉的方案。

AT89C51，它是美国 ATMEL 公司生产的低电压，高性能 CMOS8 位单片机，片内含 4kbytes 的可反复擦写的 Flash 只读程序存储器（ROM）和 128bytes 的随机存取数据存储器（RAM），器件采用 ATMEL 公司的高密度、非易失性存储技术生产，与标准 MCS-51 指令系统及 8052 产品引脚兼容，片内置通用 8 位中央处理器（CPU）和 Flash 存储单元，功能强大 AT89C51 单片机适合于许多较为复杂控制应用场合。 本设计的程序代码比较多，我们选择 AT89C51 控制芯片。

主要性能参数：

- 与 MCS-51 产品指令和引脚完全兼容
- 4K 字节可重擦写 Flash 闪速存储器
- 1000 次擦写周期
- 全静态擦写周期：0Hz—24MHz
- 三级加密程序存储器
- 128*8 字节内部 RAM
- 32 个可编程 I/O 口线
- 两个 16 位定时/计数器
- 6 个中断源
- 低功耗空闲和掉电模式

4.7 最终方案

经过反复论证　最终确定了如下方案：

（1）　采用 AT89C51 单片机作为主控制器

（2）　用 DS1302 为计时时钟芯片

（3）　用液晶 12864 作为显示模块

（4）　按键模块采用 4*4 矩阵键盘

5．开发平台介绍

这次设计涉及软件开发和仿真，不涉及到具体电路板的焊接和制作。下面对所用到的平台进行简介。

5.1 系统概述

Keil C51 是美国 Keil Software 公司出品的 51 系列兼容单片机 C 语言软件开发系统，与汇编相比，C 语言在功能上、结构性、可读性、可维护性上有明显的优势，因而易学易用。用过汇编语言后再使用 C 来开发，体会更加深刻。

Keil C51 软件提供丰富的库函数和功能强大的集成开发调试工具，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。在开发大型软件时更能体现高级语言的优势。

5.2 Keil C51 单片机软件开发系统的整体结构

C51 可在两种开发环境下使用，其中 uVision 与 Ishell 分别是 C51 for Windows 和 for Dos 的集成开发环境(IDE)，可以完成编辑、编译、连接、调试、仿真等整个开发流程。开发人员可用 IDE 本身或其它编辑器编辑 C 或汇编源文件。然后分别由 C51 及 A51 编译器编译生成目标文件(.OBJ)。目标文件可由 LIB51 创建生成库文件，也可以与库文件一起经 L51 连接定位生成绝对目标文件(.ABS)。ABS 文件由 OH51 转换成标准的 Hex 文件，以供调试器 dScope51 或 tScope51 使用进行源代码级调试，也可由仿真器使用直接对目标板进行调试，也可以直接写入程序存贮器如 EPROM 中。

5.3 proteus 仿真平台简介

Proteus 软件是英国 Labcenter electronics 公司出版的 EDA 工具软件（该软件中国总代理为广州风标电子技术有限公司）。它不仅具有其它 EDA 工具软件的仿真功能，还能仿真单片机及外围器件。它是目前最好的仿真单片机及外围器件的工具。虽然目前国内推广刚起步，但已受到单片机爱好者、从事单片机教学的教师、致力于单片机开发应用的科技工作者的青睐。Proteus 是世界上著名的 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。是目前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 年即将增加 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 MPLAB 等多种编译器。

Proteus 与其它单片机仿真软件不同的是，它不仅能仿真单片机 CPU 的工作情况，也能仿真单片机外围电路或没有单片机参与的其它电路的工作情况。因此在仿真和程序调试时，关心的不再是某些语句执行时单片机寄存器和存储器内容的改变，而是从工程的角度直接看程序运行和电路工作的过程和结果。对于这样的仿真实验，从某种意义上讲，是弥补了实验和工程应用间脱节的矛盾和现象。

(1)proteus 的工作过程

运行 proteus 7.0 professional 的 ISIS 7 professional 程序后，进入该仿真软件的主界面。选择 file 菜单下的新建 new design，在选择 DEFAULT,点击 OK，即

新建完成。接下来通过工具栏中的 p(从库中选择元件命令)命令，在 pick devices 窗口中输入 keywords 选择电路所需的元件，放置元件并调整其相对位置，元件参数设置，元器件间连线；完成仿真电路连接后，双击 AT89C51，出现 Edit component 窗口，在 Program File 栏双击文件夹图标，选择前面生成的 hex 文件；通过 debug 菜单的相应命令仿真程序和电路的运行情况。

(2)Proteus 软件所提供的元件资源

Proteus 软件所提供了 30 多个元件库，数千种元件。元件涉及到数字和模拟、交流和直流等。

(3)Proteus 软件所提供的仪表资源

对于一个仿真软件或实验室，测试的仪器仪表的数量、类型和质量，是衡量实验室是否合格的一个关键因素。在 Proteus 软件包中，不存在同类仪表使用数量的问题。Proteus 还提供了一个图形显示功能，可以将线路上变化的信号，以图形的方式实时地显示出来，其作用与示波器相似但功能更多。

(4)Proteus 软件所提供的调试手段

Proteus 提供了比较丰富的测试信号用于电路的测试。这些测试信号包括模拟信号和数字信号。对于单片机硬件电路和软件的调试，Proteus 提供了两种方法：一种是系统总体执行效果，一种是对软件的分步调试以看具体的执行情况。对于总体执行效果的调试方法，只需要执行 debug 菜单下的 execute 菜单项或 F12 快捷键启动执行，用 debug 菜单下的 pause animation 菜单项或 pause 键暂停系统的运行；或用 debug 菜单下的 stop animation 菜单项或 shift-break 组合键停止系统的运行。其运行方式也可以选择工具栏中的相应工具进行。 对于软件的分步调试，应先执行 debug 菜单下的 start/restart debugging 菜单项命令，此时可以选择 step over 、step into 和 step out 命令执行程序(可以用快捷键 F10、F11 和 ctrl+F11)，执行的效果是单句执行、进入子程序执行和跳出子程序执行。在执行了 start / restart debuging 命令后，在 debug 菜单的下面要出现仿真中所涉及到的软件列表和单片机的系统资源等，可供调试时分析和查看。

No.3

电子密码锁电路设计

题目： 电子密码锁的设计

小组成员：

谷林海

顾友峰

兰顺福

葛振涛

目录

第一章 绪论

1.1 电子密码锁简介

随着人们生活水平的提高，如何实现家庭防盗这一问题也变的尤其的突出，传统的机械锁由于其构造的简单，被撬的事件屡见不鲜，电子锁由于其保密性高，使用灵活性好，安全系数高，受到了广大用户的喜爱。

电子密码锁是一种通过密码输入来控制电路或是芯片工作，从而控制机械开关的闭合，完成开锁、闭锁任务的电子产品。它的种类很多，有简易的电路产品，也有基于芯片的性价比较高的产品。现在应用较广的电子密码锁是以芯片为核心，通过编程来实现的。其性能和安全性已大大超过了机械锁。其特点如下：

1）保密性好，编码量多，远远大于弹子锁。随机开锁成功率几乎为零。

2）密码可变，用户可以随时更改密码，防止密码被盗，同时也可以避免因
人员的更替而使锁的密级下降。

3）误码输入保护，当输入密码多次错误时，报警系统自动启动。

4）无活动零件，不会磨损，寿命长。

5）使用灵活性好，不像机械锁必须佩带钥匙才能开锁。

6）电子密码锁操作简单易行，一学即会。

1.2 电子密码锁的发展

随着社会物质财富的日益增长和人们生活水平的提高，安全防盗已成为现代居民最关心的社会问题之一。而锁自古以来就是把守门户的铁将军，人们对它要求甚高，既要安全可靠地防盗，又要使用方便，这也是制锁着长期以来研制的主题。

目前，最常用的锁是 20 世纪 50 年代意大利人设计的机械锁，其结构简单，使用方便，价格便宜。但在使用中暴露了很多缺点：一是机械锁是靠金属制成的钥匙上的不同齿形与锁芯的配合来工作的。据统计，每 4000 把锁中就有两把锁的钥匙齿牙相同或类似，故安全性低。根据国外的统计资料，装有电子防盗装置的商业区或居民区盗窃犯罪率平均下降 30%左右。二是钥匙一旦丢失，无论谁捡到都可以将锁打开。三是机械锁的材料大多为黄铜，质地较软，容易损坏。四是机械锁钥匙易于复制，不适于诸如宾馆等公共场合使用。

出于安全，方便等方面的需要，许多智能锁（如指纹辨别，IC 卡识别）已相继问世，但这类产品的特点是针对特定指纹或有效卡，只能使用于保密要求高且仅供个人使用的箱，柜，房间。另外，卡片式的 IC 卡易丢失，加上其成本一般较高，在一定程度上限制了这类产品的普及和推广。

随着人们生活水平的提高，电子密码防盗作为防盗卫士的作用越来越重要。电子密码锁用密码代替钥匙，不但省去了佩戴钥匙的烦恼，也从根本上解决了普通门锁保密性差的缺点。如果采用 4 位密码，则密码组合可达到 10000，每增加 1 位，密码组合就增加 10 倍，同时可设多组密码，其中一组是管理密码，可以增加用户密码又清除所有用户的密码。笔者设计的电子密码防盗锁利用串行 EEPROM 存储器，将设计的电子密码存入 EEPROM 中，从而克服了旧式电子密码锁电路断电后所设置密码丢失的缺点。另外，该锁还具有报警等辅助功能，是典型的机电一体化产品。

在安全技术防范领域，具有防盗报警功能的电子密码锁逐渐代替传统的机械锁，克服了机械式安全性能差的缺点，使密码锁无论在技术上还是在性能上都大大提高一步。随着大规模集成电路技术的发展，特别是单片机的问世，出现勒带位处理器的智能密码锁，它除具有电子密码锁的功能外，还引入勒智能化管理，专家分析等功能，从而使密码锁具有很高的安全性，可靠性，应用日益广泛。鉴于目前的技术水平与市场的接收程度，电子密码锁是电子防盗产品的主流。

出于安全，方便等方面的需要，许多电子密码锁已相继问世。但这类产品的特点是针对特定的有效卡，指纹，或声音有效，且不能实现远程控制，只能适用于保密要求高且供个人使用的箱子，柜子，房间等。由于数字，字符，图像，人体生物特征和时间等要素均可成为钥匙的电子信息，组合使用这些信息能够使电子密码锁获得更高的保密性，如防范很高的金库，需要使用复合信息密码的电子密码锁，这样对盗贼而言是道高一尺，魔高一丈。组合使用信息也能够使电子密码锁获得无穷扩展的可能，使产品多样化，让用户有更多的选择。可以看出把电子信息组合使用作为密码是电子密码锁以后发展的方向。

1.3 本设计所实现的目标

1、设计一套电子密码锁，通过设计训练，了解产品设计的整个过程。

2、掌握电子产品设计中的基本技术。

3、掌握电子产品设计中相关软件工具的使用。

4、掌握产品设计过程中的文档写作方法和规范要求。

5、掌握电子产品的开发、调试方法。

第二章 各部件概述

2.1 设计思路

根据设计要求，首先从整体上规划好整个系统的功能和性能，然后再对系统进行划分，将比较复杂的系统分解为多个相对独立的子系统，特别注意对各个子系统与系统、子系统与子系统之间的接口关系进行精心设计以及技术指标的合理分解。然后再由子系统到部件、部件到具体元器件的选择和调试。各部件或子系统各自完成后再进行系统联调，直到完成总体目标。

2.2 工作原理

本设计主要由单片机、矩阵键盘、液晶显示器、串行通信、开锁、报警和密码存储等部分组成。其中矩阵键盘用于输入数字密码和进行各种功能的实现。由用户通过连接单片机的矩阵键盘输入密码，后经过单片机对用户输入的密码与自己保存的密码进行对比，从而判断密码是否正确，然后控制引脚的高低电平传到开锁电路或者报警电路控制开锁还是报警。

本系统共有两部分构成，即硬件部分与软件部分。其中硬件部分由电源输入部分、键盘输入部分、密码存储部分、复位部分、晶振部分、显示部分、报警部分、开锁部分组成，软件部分对应的由主程序、初始化程序、LCD 显示程序、键盘扫描程序、启动程序、关闭程序、建功能程序、密码设置程序、EEPROM 读写程序和延时程序等组成。根据上述功能介绍，密码锁系统可以分为用户密码输入、显示和控制报警 3 大功能，因此可以键盘模块、显示模块、报警模块、电子锁控制模块和单片机模块。

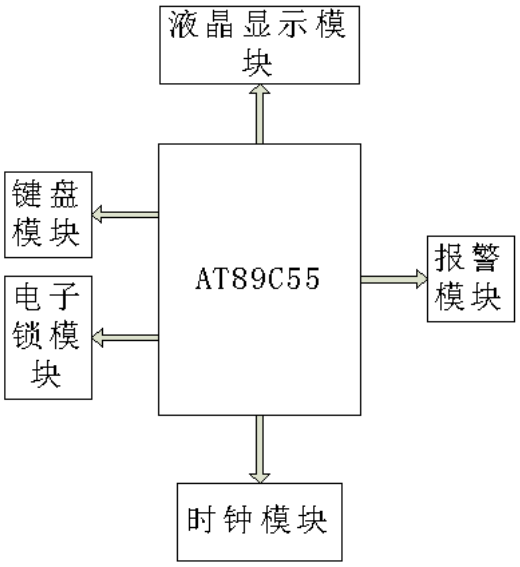


图 1 电子密码锁原理框图

2.3 电路设计

2.3.1 电源电路

实现用三端稳压器 7805 实现>5V 转换为 5V，在 PCB 板上，输入信号滤波出来之后总是要经过一段走线才进入 7805，这段走线中可能会有杂波，所以，在进入 7805 之前输入端要用电容进行一次滤波。而输出端的两个电容分别滤除高频杂波和低频杂波。这样使得滤波效果比较好，并且可以得到比较平滑的电压信号。其电路图如图 2 所示：

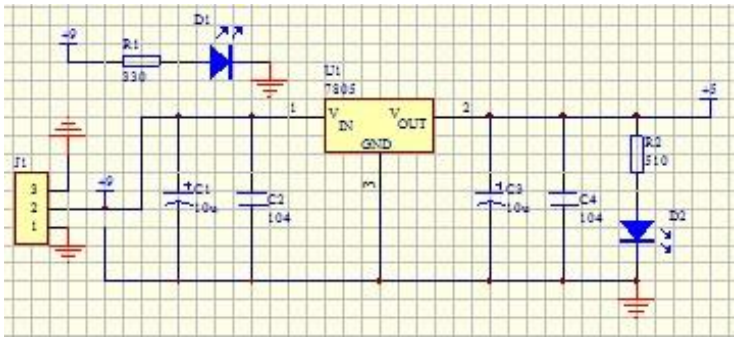


图 2

2.3.2 单片机最小系统电路

在 MCS-51 单片机系列中，种类多，价格便宜，我们对 51 系列单片机比较了

解，适用范围广，实物图连接的电路比较简单，用 80C51 芯片可以构成最小系统，用这种芯片构成的单片机最小系统简单、可靠。因此决定使用此方案。

单片机最小系统电路主要由单片机、下载口、振荡电路和复位电路构成。AT89C51 单片机为40 引脚双列直插芯片，有四个I/O口，分别是P0、P1、P2、P3，每一条I/O 线都能独立地作为输出或输入使用。单片机的最小系统，如下图2 所示

2.3.2.1　AT89C51单片机介绍

AT89C51是美国ATMEL公司的低电压，高性能CMOS8位单片机，片内含4k bytes的可反复擦写的只读程序存储器（PEROM）和128 bytes的随机存取数据存储器（RAM），器件采用ATMEL公司的高密度、非易失性存储技术盛产，兼容标准MCS-51指令系统，片内置通用8位中央处理器（CPU）和Flash存储单元，功能强大的AT89C51单片机可以为您提供许多高性价比的应用场合，可灵活应用于各种控制领域。

主要性能参数：

1、与MCS-51产品指令系统完全兼容

2、4k字节可重擦写Flash闪速存储器

3、1000次擦写周期

4、全静态操作：0Hz—24MHz

5、三级加密程序存储器

6、128×8字节内部RAM

7、32个可编程I/O口线

8、2个16位定时器/计数器

9、6个中断源

10、可编程串行UART通道

11、低功耗空闲和掉电模式

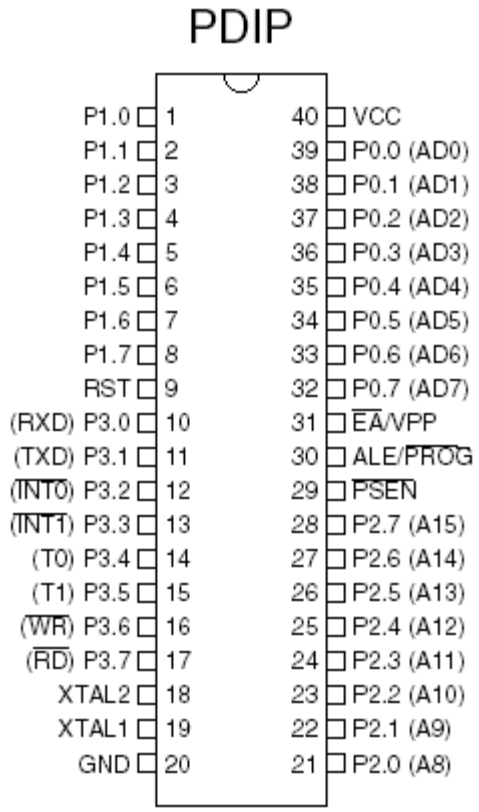


图 3

功能特性概述

AT89C51提供以下标准功能：4k字节Flash闪速存储器，128字节内部RAM，32个I/O口线，两个16位定时/计数器，一个5向量两级中断结构，一个全双工串行通信口，片内振荡器及时钟电路。同时，AT89C51可降至0Hz的静态逻辑操作，并

支持两种软件可选的节电工作模式。空闲方式停止CPU的工作，但允许RAM，定时/计数器，串行通信口及终端系统继续工作。掉电方式保存RAM中的内容，但振荡器停止工作并进制其它所有部件工作指导下一个硬件复位。

引脚功能说明

Vcc：电源电压

GND：地

P0口：P0口是一组8位漏极开路型双向I/O口，也即地址/数据总线复用口。作为输出口用时，每位能吸收电流的方式驱动8个TTL逻辑门开路，对端口写“1”可作为高阻抗输入端用。在访问外部数据存储器或程序存储器时，这组口线分时转换地址（低8位）和数据总线复用，在访问期间激活内部上拉电阻。在Flash编程时，P0口接收指令字节，而在程序校验时，输出指令字节，校验时，要求外接上拉电阻。

P1口:P1是一个带内部上拉电阻的8位双向I/O口,P1的输出缓冲级可驱动(吸收或输出电流）4个TTL逻辑门电路。对端口写“1”，通过内部的上拉电阻把端口拉到高电平，此时可做输入口。做输入口使用时，因为内部存在上拉电阻，某个引脚被外部信号拉低时会输出一个电流。Flash编程和程序校验期间，P1接收低8位地址。

P2口：P2是一个带有内部上拉电阻的8位双向I／O口，P2的输出缓冲级可驱动（吸收或输出电流）4个TTL逻辑门电路。对端口写“1”，通过内部的上拉电阻把端口拉到高电平，此时可作输入口，作输入口使用时，因为内部存在上拉电阻，某个引脚被外部信号拉低时会输出一个电流。在访问外部程序存储器或16位地址的外部数据存储器（例如执行MOVX@DPTR指令）时，P2口送出高8位地址数据。在访问8 位地址的外部数据存储器（如执行MOVX@RI 指令）时，P2 口线上的内容（也即特殊功能寄存器（SFR）区中R2寄存器的内容），在整个访问期间不改变。Flash编程或校验时，P2亦接收高位地址和其它控制信号。

P3口：P3口是一组带有内部上拉电阻的8 位双向I／O 口。P3 口输出缓冲级可驱动（吸收或输出电流）4 个TTL逻辑门电路。对P3 口写入“1”时，它们被内部上拉电阻拉高并可作为输入端口。作输入端时，被外部拉低的P3 口将用上拉电阻输出电流（IIL）。P3口除了作为一般的I／O口线外，更重要的用途是

它的第二功能，如下表所示：

端口引脚	第二功能
P3.0	RXD（串行输入口）
P3.1	TXD（串行输入口）
P3.2	$\overline{INT0}$
P3.3	$\overline{INT1}$
P3.4	T0（定时/计数器0外部输入）
P3.5	T1（定时/计数器1外部输入）
P3.6	$\overline{WR}$（外部数据存储器写选项）
P3.7	$\overline{RD}$（外部数据存储器读选项）

P3口还接收一些用于Flash闪速存储器编程和程序校验的控制信号。

RST：复位输入。当振荡器工作时，RST引脚出现两个机器周期以上高电平将使单片机复位。

$ALE/\overline{PROG}$：当访问外部程序存储器或数据存储器时，ALE（地址锁存允许）输出脉冲用于锁存地址的低8位字节。即使不访问外部存储器，ALE 仍以时钟振荡频率的1／6 输出固定的正脉冲信号，因此它可对外输出时钟或用于定时目的。要注意的是：每当访问外部数据存储器时将跳过一个ALE脉冲。对Flash存储器编程期间，该引脚还用于输入编程脉冲（$\overline{PROG}$）。如有必要，可通过对特殊功能寄存器（SFR）区中的8EH单元的D0 位置位，可禁止ALE 操作。该位置位后，只有一条MOVX和MOVC指令ALE才会被激活。此外，该引脚会被微弱拉高，单片机执行外部程序时，应设置ALE无效。

$\overline{PSEN}$：程序储存允许（$\overline{PSEN}$）输出是外部程序存储器的读选通信号，当AT89C51 由外部程序存储器取指令（或数据）时，每个机器周期两次PSEN有效，即输出两个脉冲。在此期间，当访问外部数据存储器，这两次有效的$\overline{PSEN}$信号不出现。。

EA／VPP：外部访问允许。欲使CPU仅访问外部程序存储器（地址为0000H—FFFFH），EA端必须保持低电平（接地）。需注意的是：如果加密位LB1被编程，

复位时内部会锁存EA端状态。如EA端为高电平（接VCC端），CPU则执行内部程序存储器中的指令。Flash存储器编程时，该引脚加上+12V的编程允许电源Vpp，当然这必须是该器件是使用12V编程电压Vpp。

2.3.2.2 内部时钟振荡电路

内部振荡电路是采用单片机内部振荡器来产生工作所需的时钟。51系列单片机内包含一个高增益的单级反相放大器，引脚XTAL1和XTAL2分别为片内反相放大器的输入端口和输出端口。当单片机工作于内部振荡模式的时候，只需在XTAL1引脚和XTAL2引脚连接一个晶体振荡器，并通过两个电容接地后即可。使用时，对电容的选择有一定的要求。电容一般选择C6=C7=30pF。

在实际的硬件电路板设计时，应该保证外接的振荡器和电容尽可能的靠近单片机的XTAL1和XTAL2引脚。这样可以减少寄生电容的影响使振荡器能够可靠稳定的为单片机CPU提供时钟信号。如果振荡器连接不当，会导致电路不起振，没有时钟信号产生。

2.3.2.3 单片机的复位电路

我们采用手动加上电复位电路，其基本的原理是利用RC电路的充放电效应。当按下复位开关的时候，VCC通过一个电阻之间连接到RST引脚，给RST一个高电平，按键松开的时候，RST引脚恢复为低电平，从而完成复位。

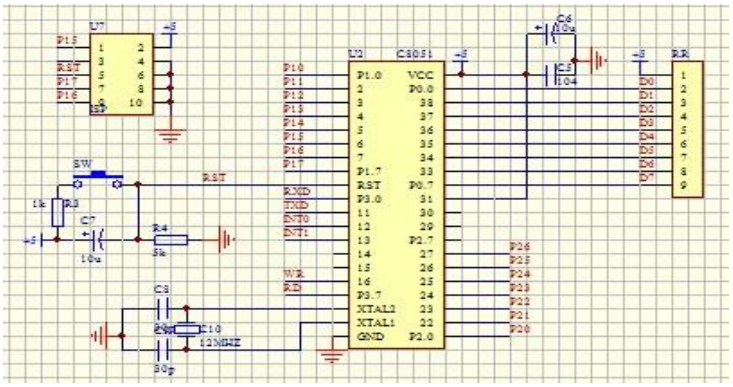


图 4

2.3.3 键盘输入电路

对于比较简单的系统，如果需要的按键比较的少，单片机引脚比较宽裕，则可以使用独立式按键结构。对于比较复杂的系统或者是按键比较多的场合，可以

采用矩阵式键盘。因此我们的按键模块采用 4×4 行列矩阵结构，在矩阵键盘结构中，采用行线和列线交叉的形式，每个交叉点即为一个按键。交叉点的行列线是不连接的，当按键按下的时候，此交叉点处的行线和列线导通。4×4 矩阵式键盘可以组成 16 个按键。在矩阵键盘的设计中，我们采用扫描法，扫描法是在程序中逐行或者逐列扫描查询键盘接口，根据端口的输入情况，判断是哪一个按键被按下，然后分别调用不同的按键处理子程序。以列扫描为例：在使用列扫描时，应将矩阵式键盘的行线通过上拉电阻接电源。此时，如果没有按键按下，则对应的行线为高电平；如果有按键按下，对应交叉点的行线和列线短路，行线的输出依赖于与此行连接的列的电平状态。由此，逐列扫描键盘，便可以实现矩阵式键盘的检测。图 5 示：

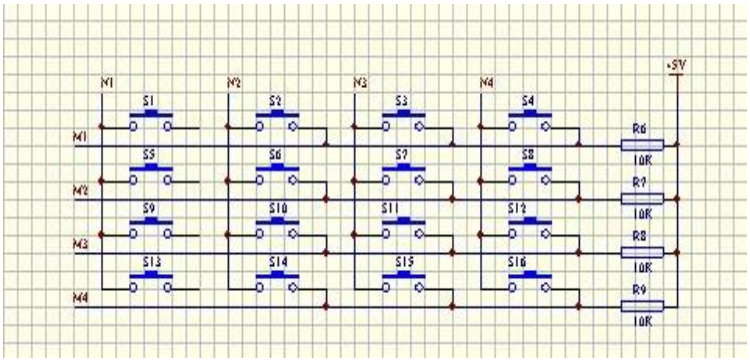


图 5

2.3.4 时钟电路模块

我们使用实时时钟芯片DS1302，DS1302是美国DALLAS Semiconductor 公司推出的一款实时时钟芯片，以计时准、接口简单、使用方便、工作电压范围宽和低功耗等优点，得到了广泛的应用。DS1302的主要特点有：

（1）、DS1302采用3线串行接口，占用引脚少。

（2）、DS1302内部集成了可编程日历时钟，用户可以根据需要设置。

（3）、DS1302内部集成了31个字节的RAM。

（4）、DS1302的日历时钟可自动进行瑞年补偿。

（5）、DS1302支持双电源供电，可以使用外部电源和备份电源。

（6）、DS1302具有对备份电池进行涓流充电的功能，可以有效延长电池的使用寿命。

Vcc1为电源输入引脚1，单电源供电时接Vcc1脚，双电源供电时用于接备份电源。Vcc2为电源输入引脚2，单电源供电时不接，双电源供电时用于接主电源，工作电压范围为2.5到5.5V。

外部主控制器，即8051单片机，可以通过RST、SCLK和I/O引脚来实现数据的传输。其中，SCLK为串行数据的同步时钟信号，由外部主控制器产生，I/O为双向串行数据传送信号，低电平有效，即RST=0表示允许通信，RST=1表示禁止通信。

实时时钟芯片DS1302芯片内部计时电路振荡源有外部晶振提供，一般在X1和X2 引脚外接32.768kHz的石英晶振。Vcc1除了外接供电电源之外，还可以为备份电池充电，DS1302内部的涓流充电器在主电源工作正常时间向备份电池充电，这样可以延长电池的使用时间。

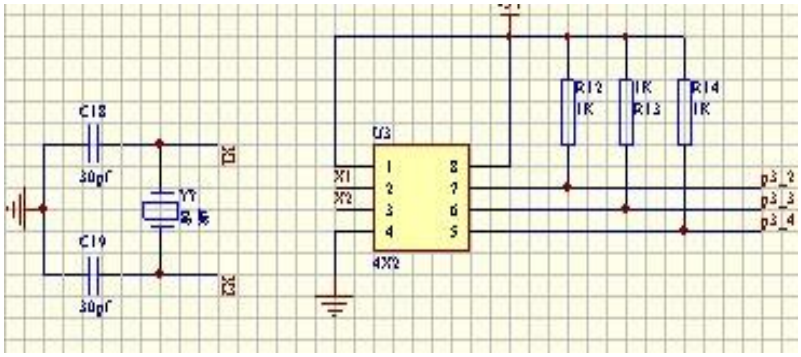


图 6

2.3.5 显示电路模块

带中文字库的128X64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。由该模块构成的液晶显示方案与同类型的图形点阵液晶显示模块相比，不论硬件电路结构或显示程序都要简洁得多，且该模块的价格也略低于相同点阵的图形液晶模块。12864的基本特性有:（1）、低电源电压（VDD:+3.0--+5.5V）（2）、显示分辨率:128×64点 （3）、内置汉字字库，提供8192个16×16点阵汉字(简繁体可选) （4）、内置 128个16×8点阵字符 （5）、2MHZ时钟频率 （6）、

显示方式：STN、半透、正显 （7）、驱动方式：1/32DUTY，1/5BIAS （8）、视角方向：6点 （9）、背光方式：侧部高亮白色LED，功耗仅为普通LED的1/5—1/10 （10）、通讯方式：串行、并口可选 （11）、内置DC-DC转换电路，无需外加负压 （12）、无需片选信号，简化软件设计（13）、工作温度：0℃ - +55℃ ，存储温度：-20℃ - +60℃

控制器接口信号说明：

1、RS，R/W的配合选择决定控制界面的4种模式：

RS	R/W	功能说明
L	L	MPU写指令到指令暂存器（IR）
L	H	读出忙标志（BF）及地址记数器（AC）的状态
H	L	MPU写入数据到数据暂存器（DR）
H	H	MPU从数据暂存器（DR）中读出数据

2、E信号

E状态	执行动作	结果
高——>低	I/O 缓 冲——>DR	配合/W进行写数据或指令
高	DR——>I/O缓冲	配合R进行读数据或指令
低/低——>高	无动作	

● 忙标志:BF　　BF标志提供内部工作情况.BF=1表示模块在进行内部操作,此时模块不接受外部指令和数据.BF=0时,模块为准备状态,随时可接受外部指令和数据.利用STATUS RD 指令,可以将BF读到DB7总线,从而检验模块之工作状态.

● 字型产生ROM（CGROM）　　字型产生ROM（CGROM）提供8192个此触发器是用于模块屏幕显示开和关的控制。DFF=1为开显示（DISPLAY ON),DDRAM 的内容就显示在屏幕上,DFF=0为关显示(DISPLAY OFF)。DFF 的状态是指令DISPLAY ON/OFF和RST信号控制的。

● 显示数据RAM（DDRAM）模块内部显示数据RAM提供64×2个位元组的空间，最多可控制4行16字（64个字）的中文字型显示，当写入显示数据RAM时，可分别显示CGROM与CGRAM的字型;此模块可显示三种字型,分别是半角英数字型(16*8)、CGRAM字型及CGROM的中文字型，三种字型的选择，由在DDRAM中写入的编码选择，在0000H—0006H的编码中（其代码分别是0000、0002、0004、0006共4个）将选择CGRAM的自定义字型，02H—7FH的编码中将选择半角英数字的字型，至于A1以上的编码将自动的结合下一个位元组,组成两个位元组的编码形成中文字型的编码BIG5（A140—D75F），GB（A1A0-F7FFH）。

● 字型产生RAM(CGRAM)　　字型产生RAM提供图象定义(造字)功能, 可以提供四组16×16点的自定义图象空间,使用者可以将内部字型没有提供的图象字型自行定义到CGRAM中,便可和CGROM中的定义一样地通过DDRAM显示在屏幕中。

● 地址计数器AC地址计数器是用来贮存DDRAM/CGRAM之一的地址,它可由设定指令暂存器来改变，之后只要读取或是写入DDRAM/CGRAM的值时，地址计数器的值就会自动加一，当RS为“0”时而R/W为“1”时，地址计数器的值会被读取到DB6——DB0中。

光标/闪烁控制电路

此模块提供硬体光标及闪烁控制电路，由地址计数器的值来指定DDRAM中的光标或闪烁位置。

12864 液晶具有如下的特性：

1 提供8 位，4 位并行接口及串行接口可选

2 并行接口适配M6800 时序

3 自动电源启动复位功能

4 内部自建振荡源

64×16 位字符显示RAM（DDRAM 最多16 字符×4 行，LCD 显示范围16×2 行）（改为半角输入）

2M 位中文字型ROM（CGROM），总共提供8192 个中文字型（16×16 点阵）

16K 位半宽字型ROM(HCGROM)，总共提供126 个西文字型（16×8 点阵）

64×16 位字符产生RAM（CGRAM）

15×16 位总共240 点的ICON RAM（ICONRAM）

为了提高密码锁的密码显示效果能力。本设计的显示部分由液晶显示器LCD12864 取代普通的数码管来完成。只有按下键盘上的开启按键后，显示器才处于开启状态。同理只有按下关闭按键后显示器才处于关闭状态。否则显示器将一直处于初始状态，当需要对密码锁进行开锁时，按下键盘上的开锁按键后利用键盘上的数字键 0－9 输入密码，每按下一个数字键后在显示器上显示一个*，输入多少位就显示多少个*。DB0-DB7 口作为并行的数据，连接单片机 P0 口；发左半屏数据选通 CS1，右半屏数据选通 CS2；E 作为串/并工作方式选择端口,R/W 作为读写控制口。其与单片机的连接电路如下图 7 所示：

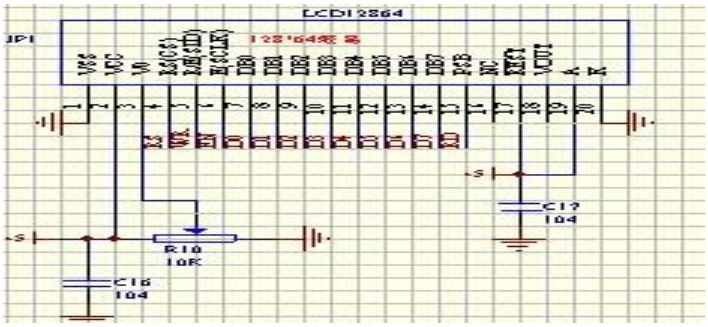


图 7

2.3.6 MAX232 与单片机通信电路

89C51 的 RX、TX 接 MAX232，MAX232 的 13 和 14 引脚接 DB9 的 3、2 引脚组成串口通信接线。串口通信原理，单片机和 PC 机通信要经过 TTL 电平转换。这里串行通信使用 RS-232 标准。这里只用 3 个引脚构成串口通信，2 脚发送数据 TXD，3 脚接收数据 RXD 和 5 脚接地。单片机的串口通信，在单片机芯片中，UART 已集成在其中，做为组成部分，构成一个串行口。单片机串口通信的波特率设置

是有单片机的定时记数器 0 提供时钟。串口通信用到串口发送缓冲寄存器 SBUF 和及串行通信控制寄存器 SCON。下图是 MAX232 的基本接线图如图 8。

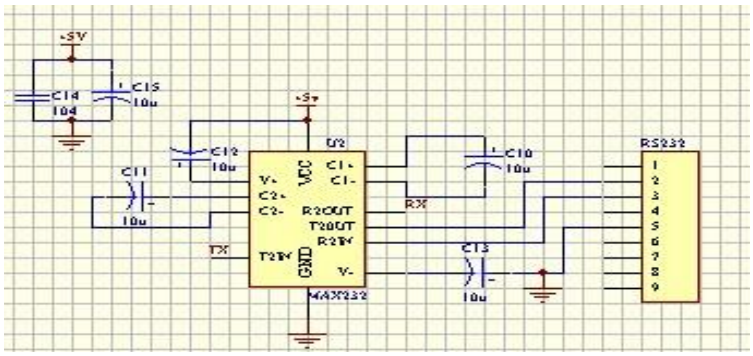


图 8

2.3.7 掉电存储电路

存储系统采用AT24C02，具有很好的可靠性参数如下：

工作温度工业级-55°C～ +125°C

商业级0℃～+75°C

贮存温度-65℃～+150°C

各管脚承受电压-2.0 <VCC<+2.0V

VCC 管脚承受电压-2.0 ～+7.0V

封装功率损耗（Ta=25°C） 1.0W

输出短路电流100mA

下图 1、2、3 脚是三条地址线，用于确定芯片的硬件地址，在 AT89S51 上它们都能接地。 AT24C02 中带有片内地址寄存器，每写入或读出一个数据字节后，该地址寄存器自动加 1，以实现对下一个储存单元的读写，所有字节均以单一操作方式读取。

AT24C02 与单片机的连接电路如下图 9 所示：

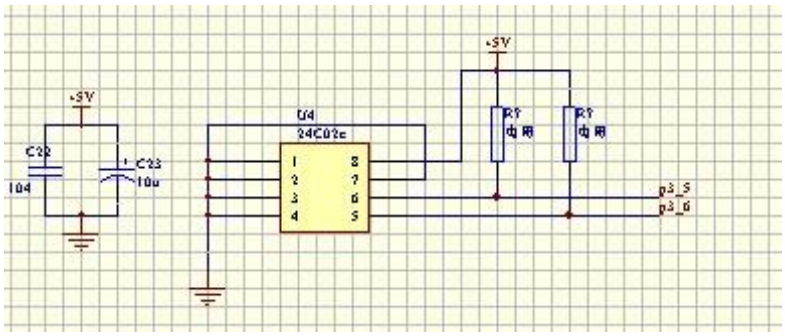


图 9

2.3.8 开锁控制电路

89C51 的 RX、TX 接 MAX232，MAX232 的 13 和 14 引脚接 DB9 的 3、2 引脚组成串口通信接线。串口通信原理，单片机和 PC 机通

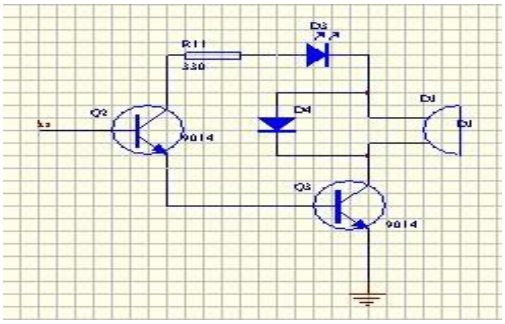


图 10

2.3.9 报警控制电路

报警电路主要由 PNP 三极管和蜂鸣器构成。当对其 1 号引脚施加 5V 电压时，便会鸣叫。由下图可知，当 fm 输出低电平时，三极管饱和导通，蜂鸣器鸣叫；当 fm 输出高电平时，三极管饱截止，蜂鸣器停止鸣叫。通过控制 fm 输出低电平的时间长短来控制蜂鸣器长叫或短叫。电路图如图 11 所示：

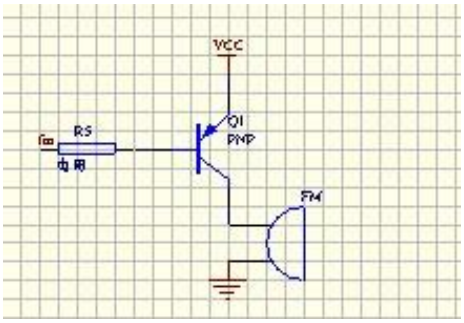


图 11

2.3.10 指示灯电路

89C51 的 RX、TX 接 MAX232，MAX232 的 13 和 14 引脚接 DB9 的 3、2 引脚组成串口通信接线。串口通信原理，单片机和 PC 机通

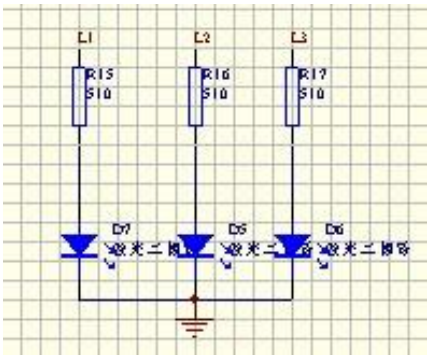


图 12

第三章 总原理图及 PCB 板的设计

3.1 系统原理电路图

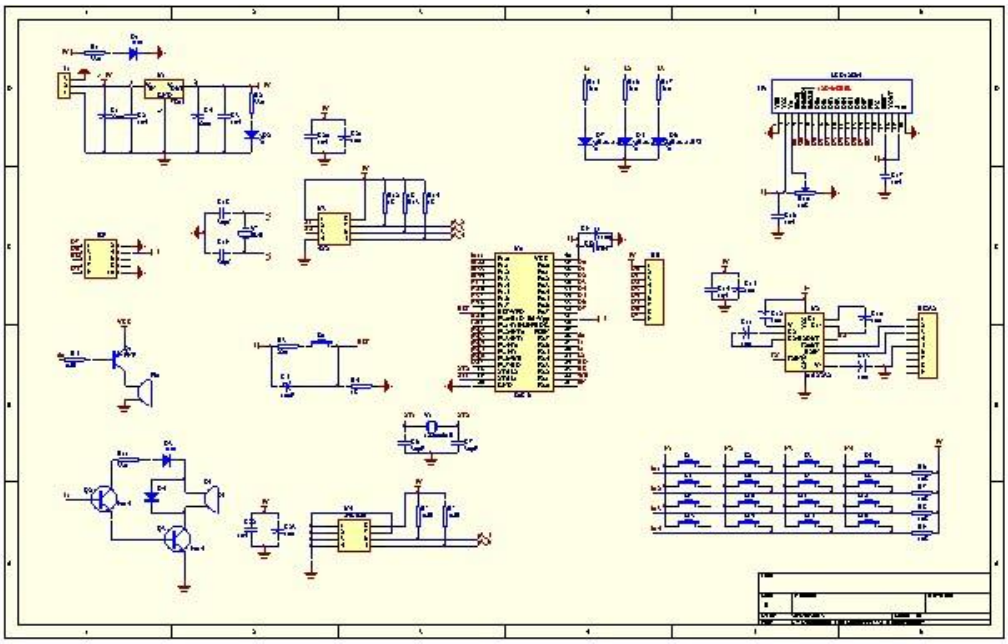


图 13

3.2 系统 PCB 电路图

元件布局基本规则 ：

1. 按电路模块进行布局，实现同一功能的相关电路称为一个模块，电路模块中的元件应采用就近集中原则，同时数字电路和模拟电路分开。

2.定位孔、标准孔等非安装孔周围 1.27mm 内不得贴装元、器件，螺钉等安装孔周围 3.5mm（对于 M2.5）、4mm（对于 M3）内不得贴装元器件。

3. 卧装电阻、电感（插件）、电解电容等元件的下方避免布过孔，以免波峰焊后过孔与元件壳体短路。

4. 元器件的外侧距板边的距离为 5mm。

5. 贴装元件焊盘的外侧与相邻插装元件的外侧距离大于 2mm。

6. 金属壳体元器件和金属件（屏蔽盒等）不能与其它元器件相碰，不能紧贴印制线、焊盘，其间距应大于 2mm。定位孔、紧固件安装孔、椭圆孔及板中其它方

孔外侧距板边的尺寸大于 3mm。

7. 发热元件不能紧邻导线和热敏元件；高热器件要均衡分布。

8. 电源插座要尽量布置在印制板的四周，电源插座与其相连的汇流条接线端应布置在同侧。特别应注意不要把电源插座及其它焊接连接器布置在连接器之间，以利于这些插座、连接器的焊接及电源线缆设计和扎线。电源插座及焊接连接器的布置间距应考虑方便电源插头的插拔。

9. 其它元器件的布置： 所有 IC 元件单边对齐，有极性元件极性标示明确，同一印制板上极性标示不得多于两个方向，出现两个方向时，两个方向互相垂直。

10、板面布线应疏密得当，当疏密差别太大时应以网状铜箔填充，网格大于 8mil(或 0.2mm)。

11、贴片焊盘上不能有通孔，以免焊膏流失造成元件虚焊。重要信号线不准从插座脚间穿过。

12、贴片单边对齐，字符方向一致，封装方向一致。

13、有极性的器件在以同一板上的极性标示方向尽量保持一致。

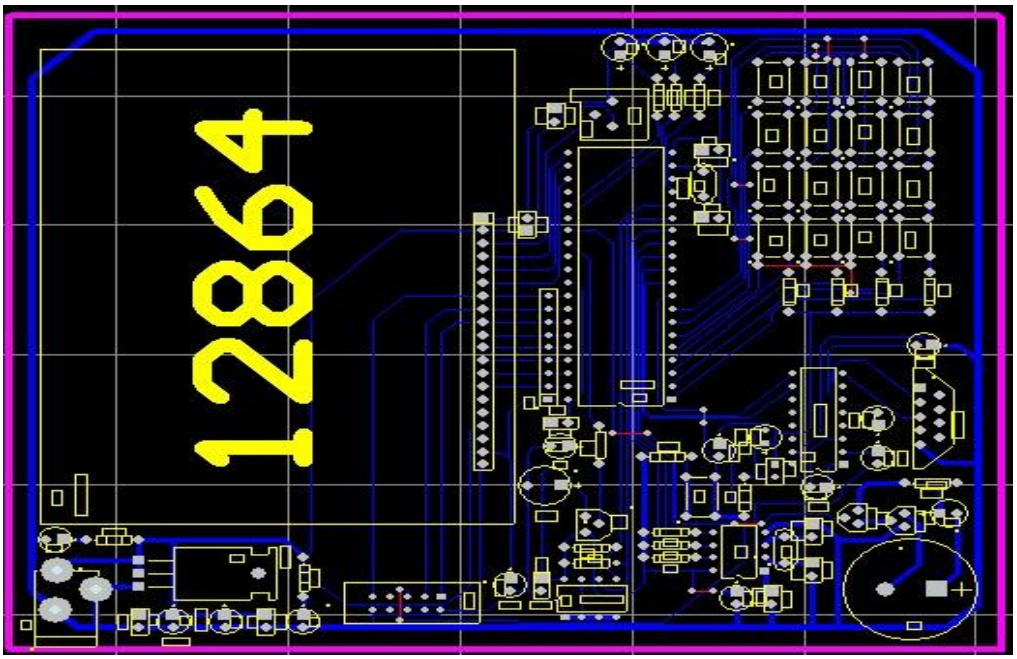


图 14

3.3 电子密码锁的操作界面

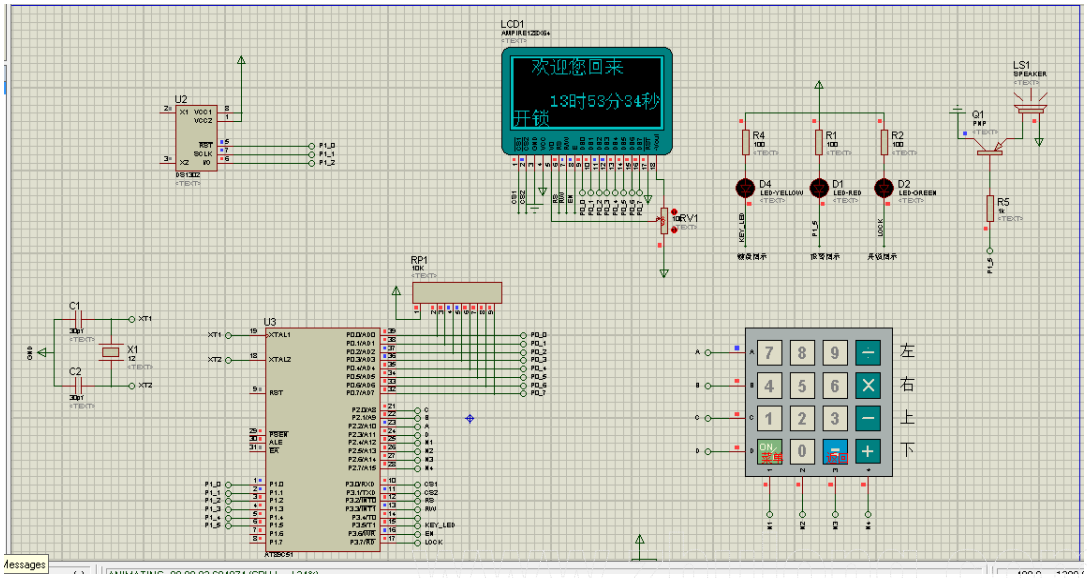


图 15

No. 4

电子密码锁模块流程图

题目： 电子密码锁的设计

小组成员：

谷林海

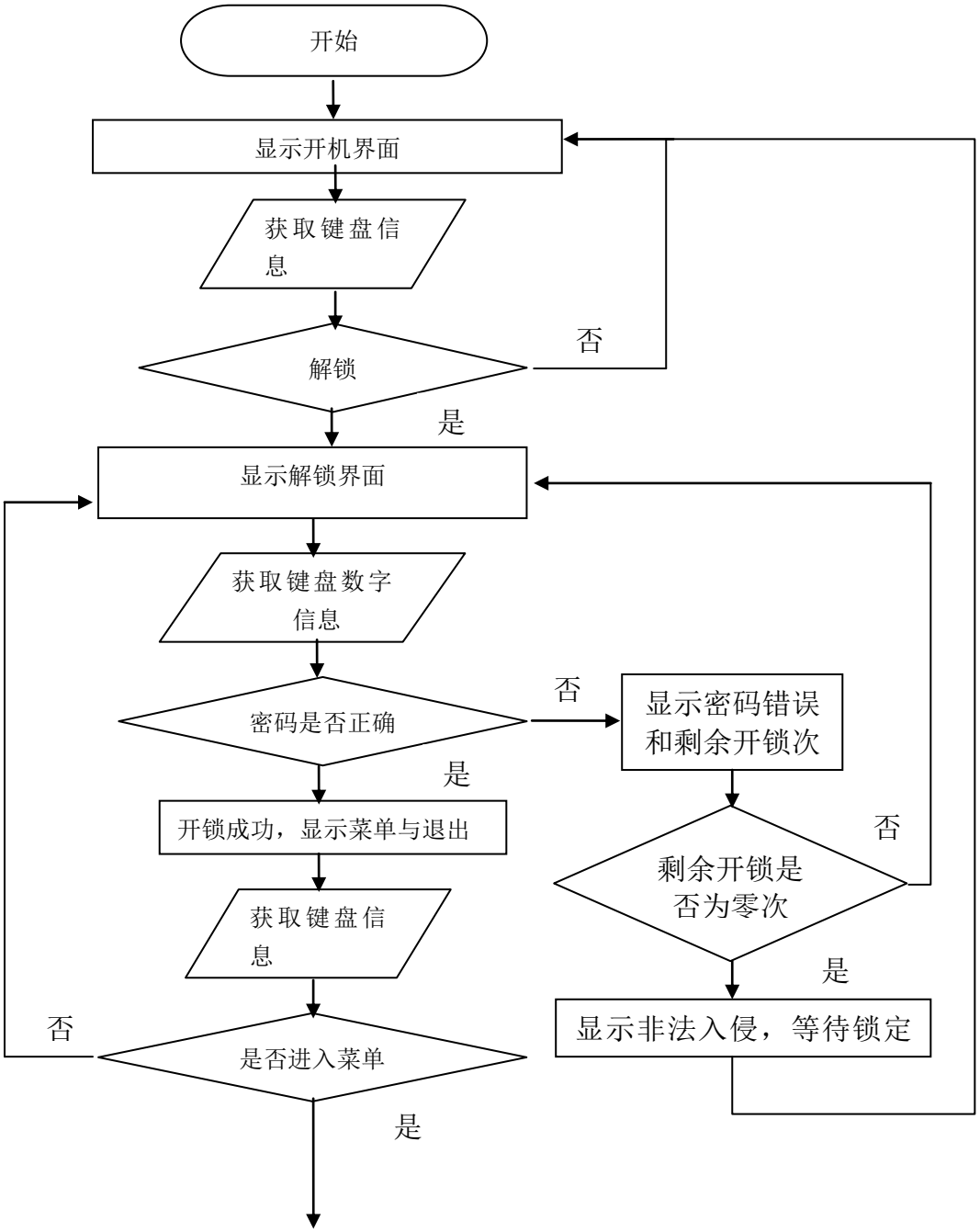
顾友峰

兰顺福

葛振涛

目录

1、显示模块


开始
显示开机界面
获取键盘信息
解锁
否
是
显示解锁界面
获取键盘数字信息
密码是否正确
否
显示密码错误和剩余开锁次
是
开锁成功，显示菜单与退出
剩余开锁是否为零次
否
获取键盘信息
是
显示非法入侵，等待锁定
否
是否进入菜单
是

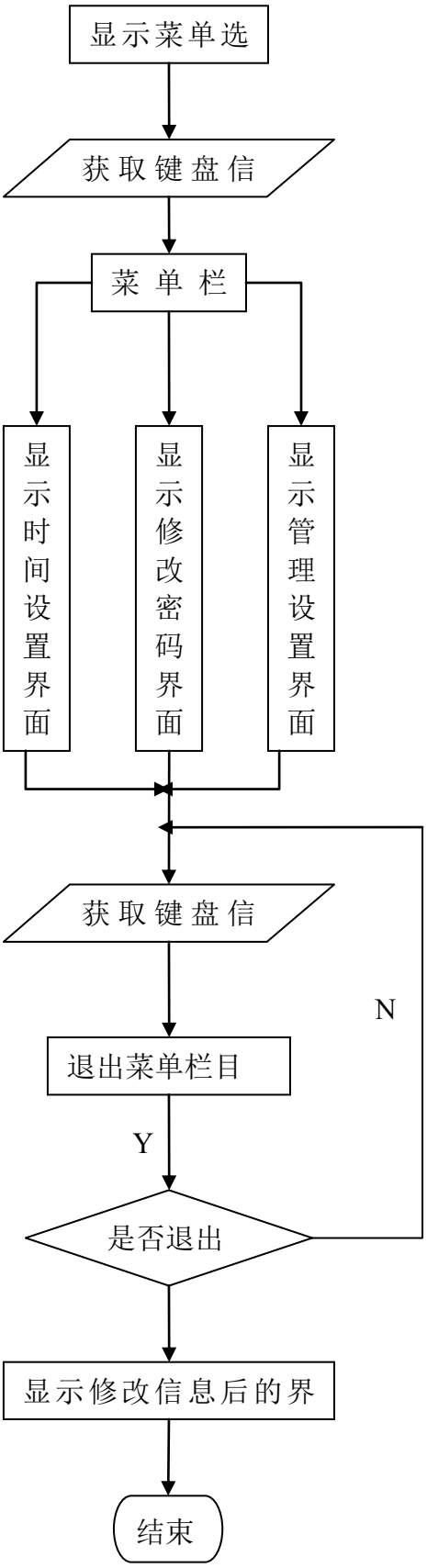
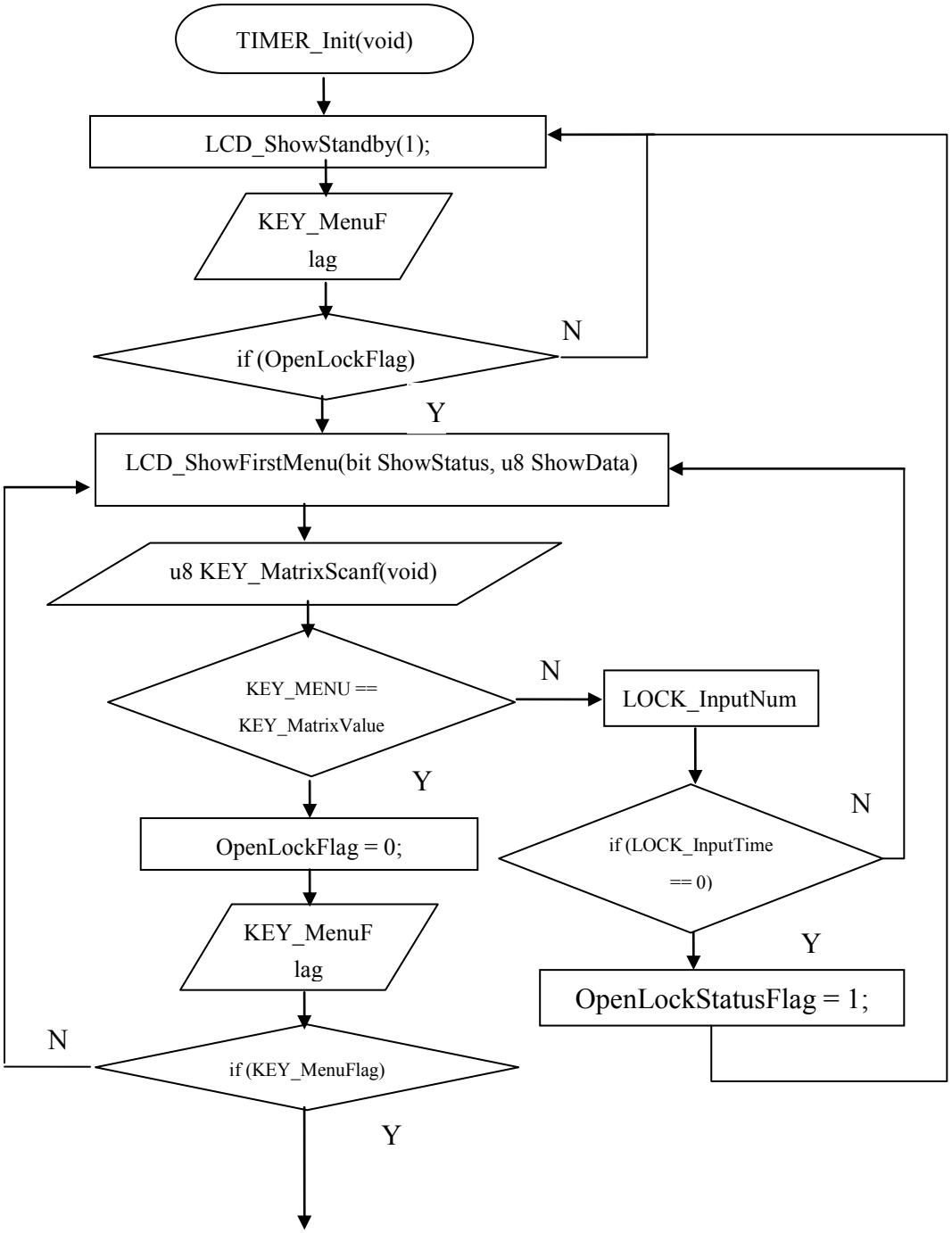


图 1.1 显示模块的逻辑流程图


TIMER_Init(void)
LCD_ShowStandby(1);
KEY_MenuF lag
if (OpenLockFlag)
N
Y
LCD_ShowFirstMenu(bit ShowStatus, u8 ShowData)
u8 KEY_MatrixScanf(void)
KEY_MENU == KEY_MatrixValue
N
LOCK_InputNum
Y
OpenLockFlag = 0;
if (LOCK_InputTime == 0)
N
Y
KEY_MenuF lag
OpenLockStatusFlag = 1;
N
if (KEY_MenuFlag)
Y

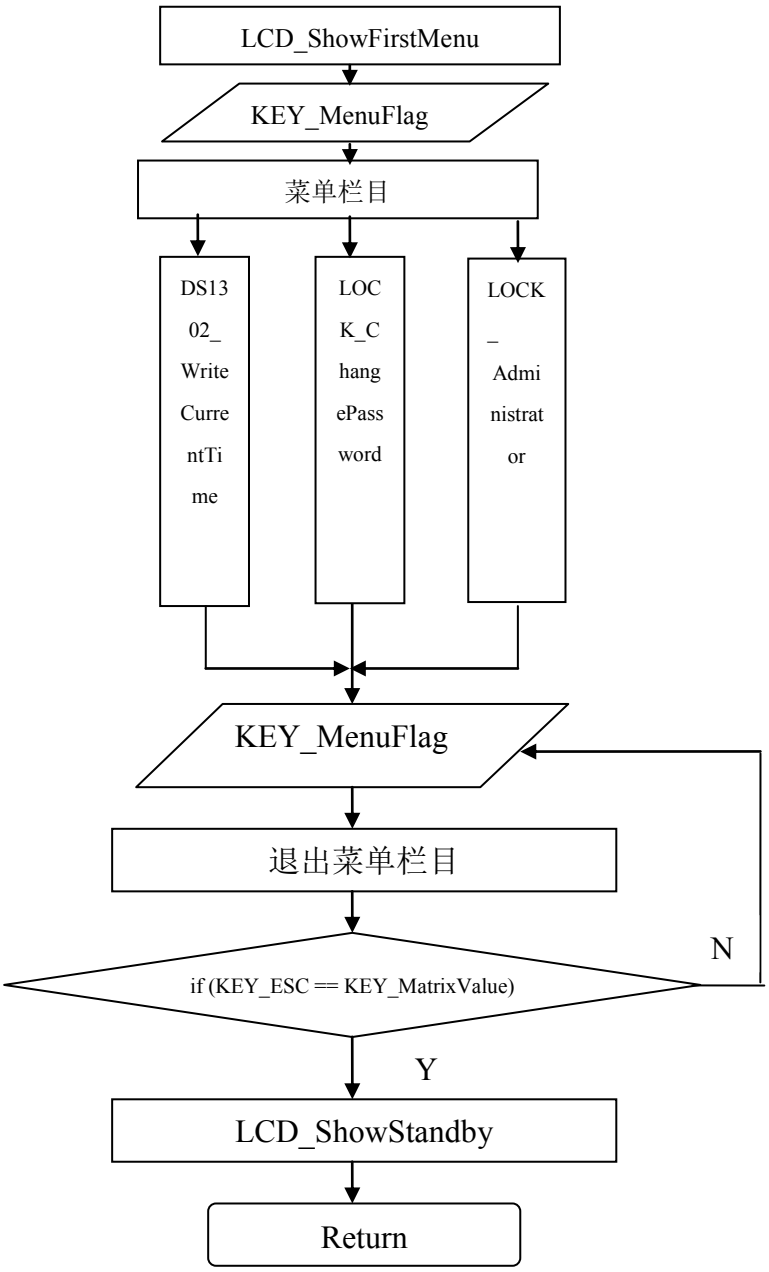


图 1.2 显示模块的程序流程图

以上是显示部分的流程图。首先在系统开始运行后，LCD12864 显示开机欢迎界面，界面包括实时时间显示，欢迎您回来的语句以及开锁推出选项。在键盘没有按下开锁确认的情况下，屏幕保持上述界面，等待开锁确认后，界面跳转为解锁，显示密码输入提示信息和倒计时 30 秒提示信息，并开始倒计时，在倒计时结束时，如果没有确定正确的密码输入，界面显示输入失败，提示剩余合法输入次数，显示 30 秒后，跳转界面至开机画面。如果在 30 秒内输入了正确的密码并按下确认键，界面将显示第一级菜单，菜单内容为三个部分：时间调整、密码

修改和管理员权限。通过键盘控制画面上光标的上下位置，通过确认键可以选择需要的菜单项目。如果进入时间调整菜单，界面跳转，显示年、月、日、时、分、秒。通过键盘的上下左右键，选择修改项目，光标所到处，该处时间闪烁显示，用户通过键盘数字键可输入数字，界面将输入的数字在闪烁处显示出来。用户调整完时间后，选择该菜单的确认项，完成一次对时间的修改。界面跳转回一级菜单。如果用户选择密码修改项，LCD12864 将跳转至密码修改的二级菜单，菜单由原始密码、新密码、重复输入新密码组成，用户通过键盘上下依次选择输入项目栏，通过数字键输入相关信息，如果用户原始密码输入错误，画面跳转显示密码错误后再返回至该二级目录。当所有信息输入正确后，用户点击确认键，画面跳转显示设置成功，通过倒计时后，自动跳转至一级菜单。同理，当用户在一级菜单选择管理员权限栏时，显示跳转至该栏的二级菜单，显示内容有报警时间设置、超时时间设置、超时次数设置、自锁时间设置四项内容。用户通过键盘正确的操作，可以设置相关项目内容。同意，点击确认键后设置成功，画面跳转显示设置成功后 10 秒，再自动跳转回一级菜单。除此之外，每一级菜单的右下角都有显示返回选项，选择退出键屏幕将显示上一级显示内容。

2、键盘模块

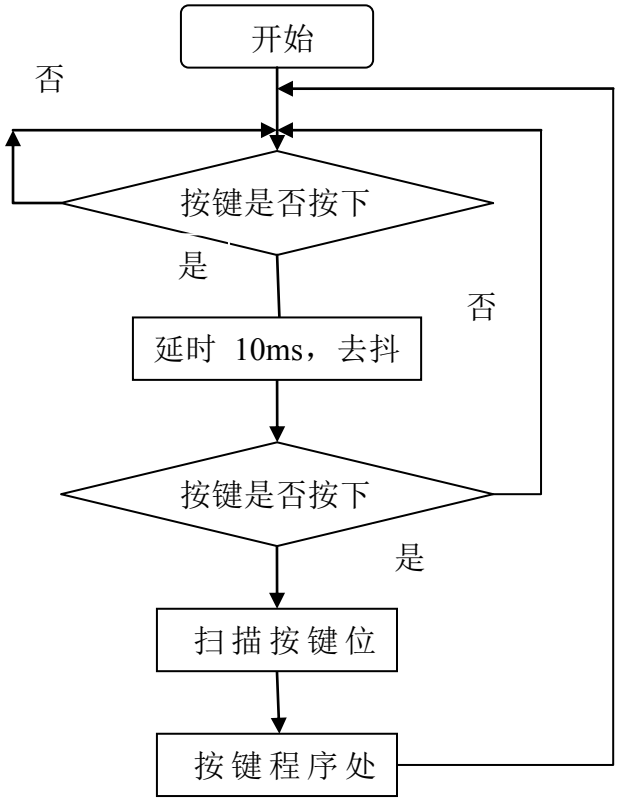


图 2.1 盘扫描的逻辑流程图

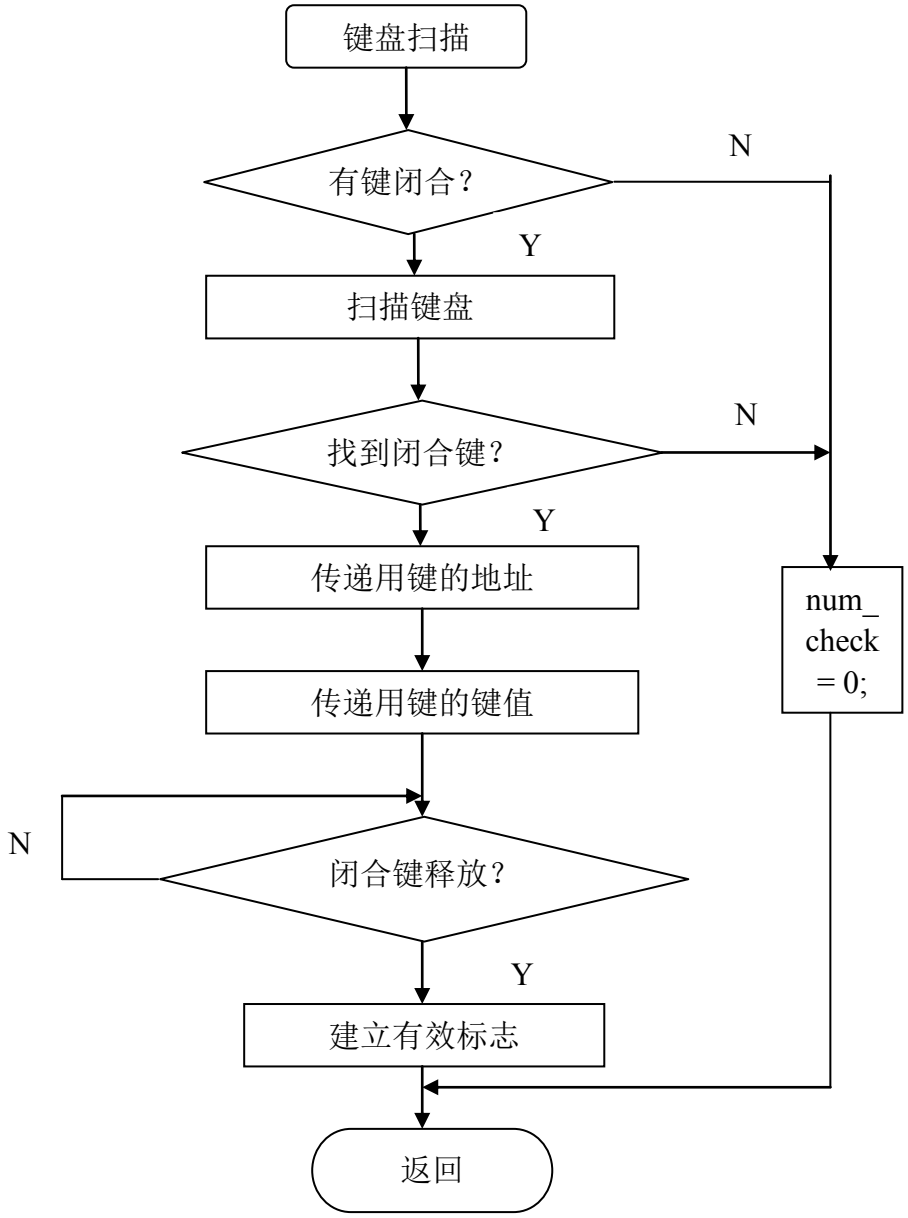


图 2.2 键盘模块逻辑流程图

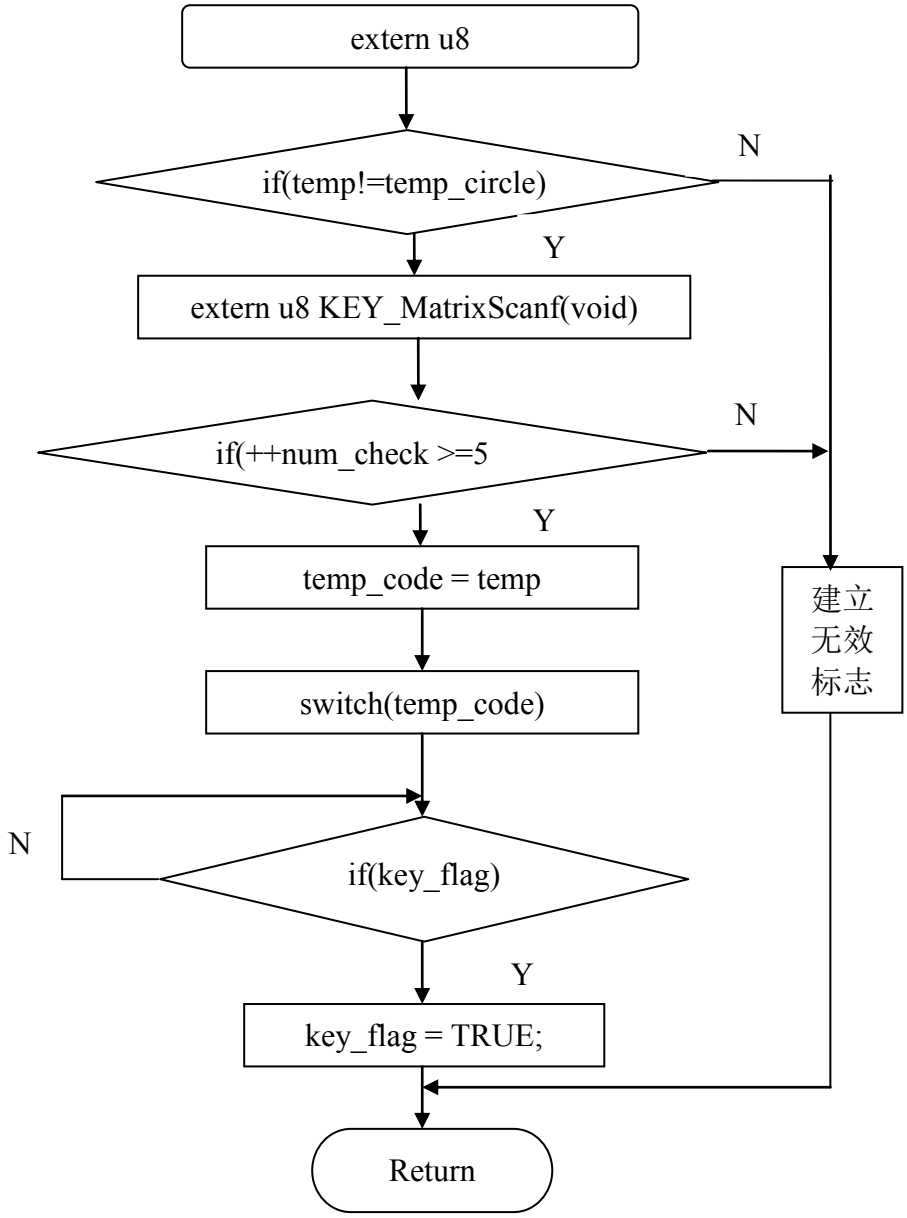


图 2.3 键盘模块程序流程图

以上部分是键盘模块的流程图，在系统开始运行后键盘 KEY 开始键盘扫描，至系统运行结束时扫描截止。通过键盘反馈电平变化，判断是否有键盘按下事件发生，如果有键盘按下，开始扫描键盘；如果没有键盘按下，则建立无效的标志信息，键盘继续扫描。扫描键盘时，如没有找到闭合键，则也建立无效的标志信息，键盘继续扫描；如果找到了闭合键，则首先确定该键地址，然后确定该键的键值。同时，不断扫描该键是否释放闭合，当扫描到该键闭合释放，则建立有效的标志信息。

3、定时模块

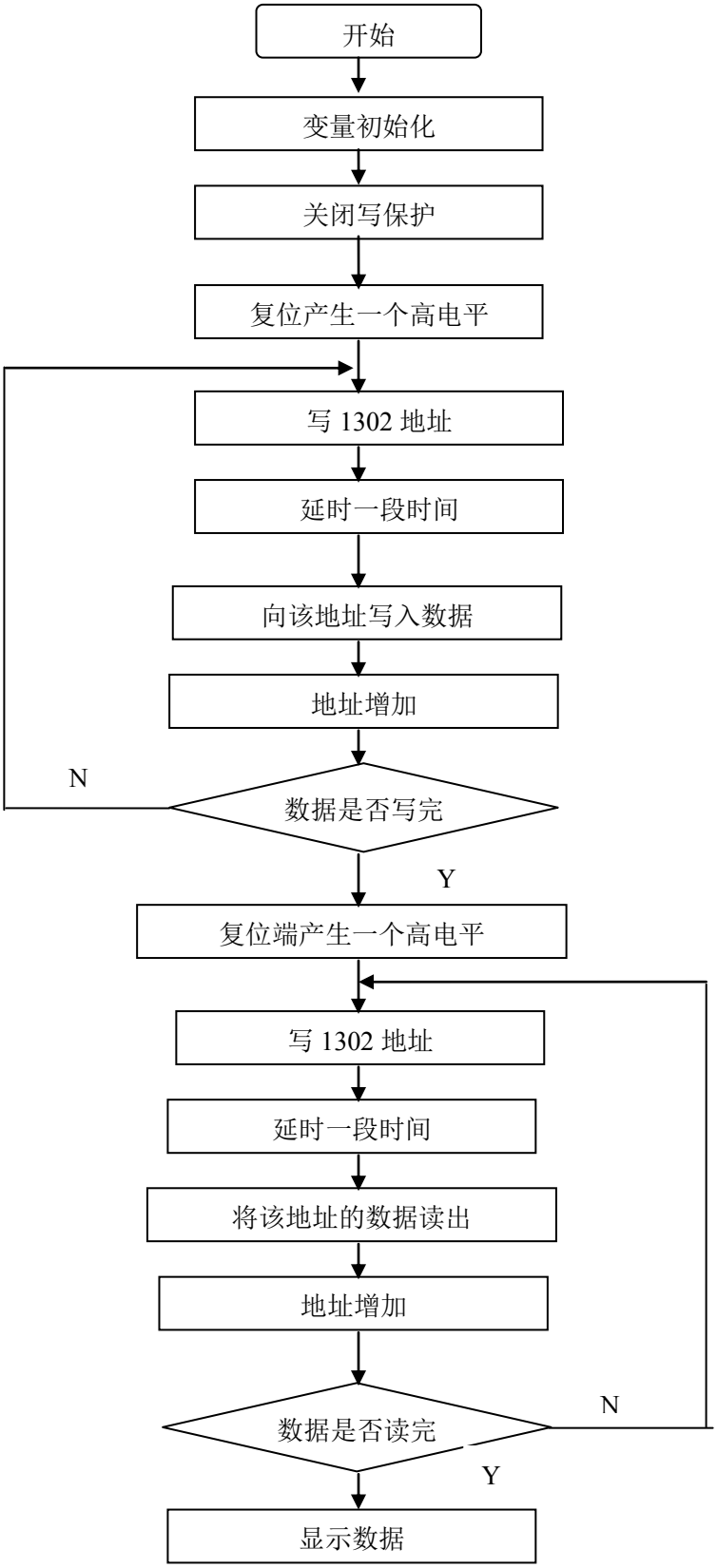

开始
变量初始化
关闭写保护
复位产生一个高电平
写 1302 地址
延时一段时间
向该地址写入数据
地址增加
N
数据是否写完
Y
复位端产生一个高电平
写 1302 地址
延时一段时间
将该地址的数据读出
地址增加
数据是否读完
N
Y
显示数据

图 5、定时模块逻辑流程图

temp_circle)

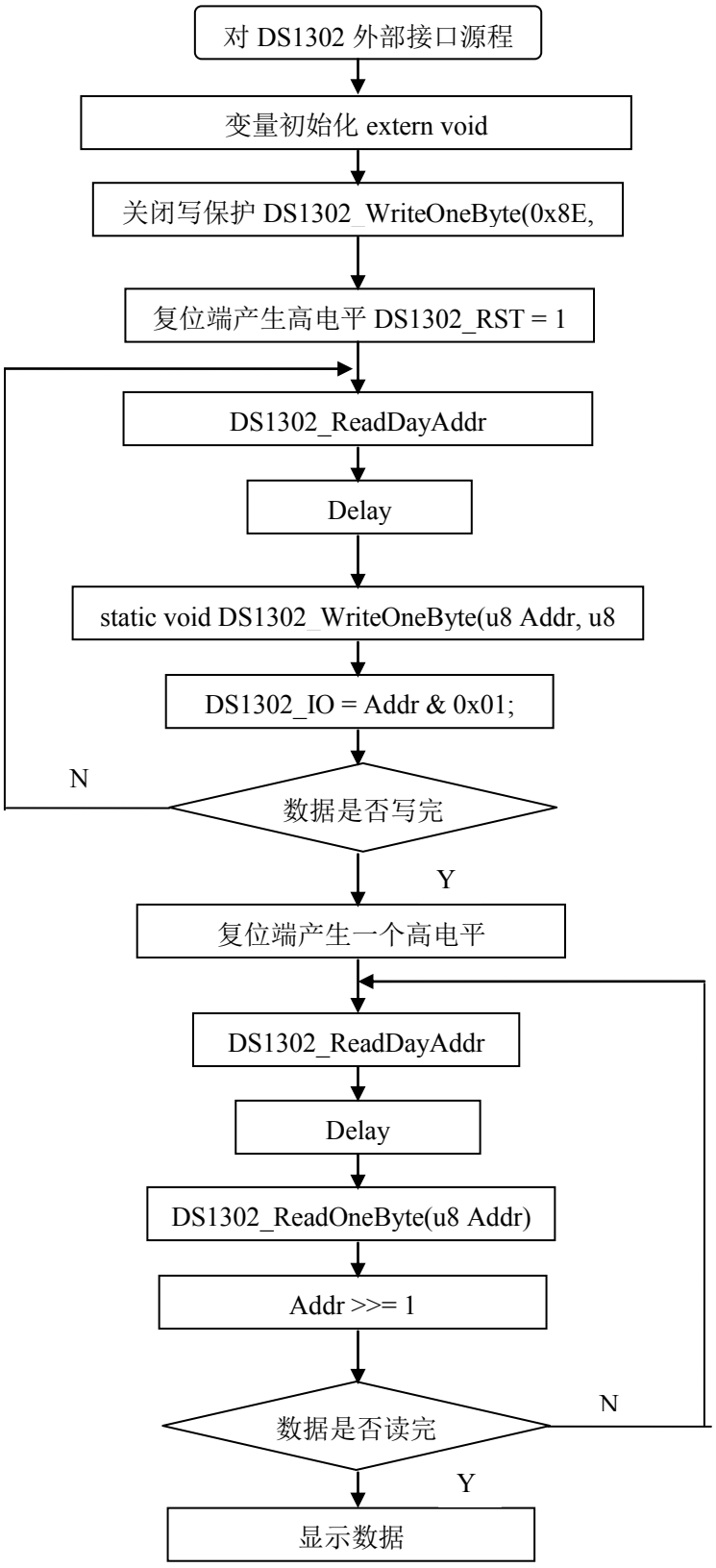


图 6、定时模块程序流程图

以上是 DS1302 实时时间程序流程图，其实现的功能就是通过定时时间芯片 DS1302 完成整个系统的定时和实时时间显示。其主要步骤如下：首先，对 DS1302 的端口变量初始化，同时关闭芯片的写保护程序。当复位端产生个高电平时，开始写 DS1302 的地址，然后延时一段时间，再向写入的地址写数据，同时地址依次增加，当地址写完时，让复位端产生一个高电平。再次向 DS1302 写入地址，并且在延时后读出地址，地址读完后显示数据，如果没有读完，程序则返回至再次写入 DS1302 地址处。

No. 5

电子密码锁

软件代码及分析

题目： 电子密码锁的设计

小组成员：

谷林海

顾友峰

兰顺福

葛振涛

引言

本文档主要是对电子密码锁设计的软件代码进行分析，解释代码的作用，说明编写的逻辑思想。全文一共分为主程序和子程序两大板块，其中子程序分为六个部分。这五个部分是依据编程中对密码锁不同组成部分区分开来。它们分别是：时钟芯片 DS1302 相关程序；键盘 KEY 相关程序；屏显 LCD12864 相关程序；时钟 LOCK 部分程序和显示 DISPLAY 部分程序。所有程序都用 C 语言编写，统一被主函数调用。同时，五个子程序部分也相互调用。每个子程序都有自己的头文件，头文件中主要是对驱动程序的声明。

一、电子密码锁主程序

```
/*********************************************************************
* 文 件 名   : main.c
* 作       者   : 第九组
* 日       期   : 10/20/2013
* 描       述   : 电子密码锁主程序
*********************************************************************/
/* 头文件包含 ----------------------------------------------------------*/
#include "STC51_lib.h"
#include "DISPLAY.h"
#include "LOCK.h"
#include "LCD12864.h"
#include "AT24C02.h"
#include "fontlibrary.h"
#include "DS1302.h"
#include "KEY.h"
/* ---------------------------全局变量  ------------------------------------*/
static bit Time_500msFlag = 0;            /* 500MS 定时控制标志位  */
u8 Time_1s = 0;                  /*  开锁时间 1s 计时  */
u8 Time_1000Ms = 0;                      /*  自锁时间 1s 计时  */
int main()
{
      unsigned char i;
     u8 KEY_MatrixValue = 0;
```

```
    TMOD = 0x01;
    TH0 = (65536-50000)/256;
    TL0 = (65536-50000)%256;       /* 50ms 初始值  */
    EA = 1;
    ET0 = 1;
    TR0 = 1;
    LCD_Init();
    LCD_ShowStandby(1);   /*  显示待机界面  */
    while(1)
    {
        KEY_MatrixValue = KEY_MatrixScanf();
        if (OpenLockFlag)       //锁没有打开的情况下进入
        {
            if (OpenLockStatusFlag)
            {
                if (KEY_MENU == KEY_MatrixValue)
                {
                    OpenLockStatusFlag = 0;//用户可能要开锁
                }
            }
            if (!OpenLockStatusFlag)
            {
                LOCK_OpenProcess(Time_500MsFlag,            KEY_MatrixValue,
&Time_1000Ms);   //用户开锁状态
            }
```

```
            else
            {
                DS1302_TimeProcess();
                LCD_ShowNumber(0,14,' ');
                LCD_ShowNumber(0,15,' ');
            }
        }
        else
        {
            if (KEY_MenuFlag)   /* 进入菜单 */
            {
                KEY_MenuProcess(Time_500MsFlag, KEY_MatrixValue);

            }
            else  /* 待机状态 */
            {
                DS1302_TimeProcess();
            }
            if (KEY_MatrixValue != KEY_NULL)  //有按键按下 时就重新初始化，开始计数
            {
                LockTime = SetLockTime;
            }
            if (Time_1s)
            {
```

```
                    Time_1s = 0;
                    if (--LockTime == 0)
                    {
                        LockTime = 0;
                        LOCK = 1;   //锁
                        OpenLockFlag = 1;   /*  锁的状态为关  */
                        LCD_ShowStandby(1);   //
                    }
                    LCD_ShowNumber(0,14, LockTime/10);
                    LCD_ShowNumber(0,15, LockTime%10);   /*  显示到计时  */
                }
            }
        }
        return (0);
}
void time1() interrupt 1
{
        static u8 Count_500 = 0;
        static u8 Count_1000 = 0;
        TH0 = (65536-50000)/256;
        TL0 = (65536-50000)%256;
        if (++Count_500 >= 10)     /* 500ms 定时  */
        {
            Count_500 = 0;
            Time_500MsFlag = ~Time_500MsFlag;
```

```
    }
    if (++Count_1000 >= 20)   /* 1000ms 定时  */
    {
        Count_1000 = 0;
        Time_1000Ms = 1;
        Time_1s = 1;
    }
}
/*************************文件结束*****************************/
```

二、键盘驱动程序

```
/*********************************************************************
* 文 件 名   : KEY.c
* 作      者  : 第九组
* 日      期  : 15/10/2013
* 描      述  : 键盘驱动程序
*********************************************************************/
/*******************头文件包含  ******************************/
#include "KEY.h"
#include "LCD12864.h"
#include "fontlibrary.h"
#include "DS1302.h"
#include "DISPLAY.h"
#include "LOCK.h"
/***********************全局变量定义  ****************************/
bit KEY_MenuFlag = 0;        /*  菜单选项  */
```

```
bit Adj_StatusFlag = 0;      /*  菜单状态  */
bit Adj_MenuFlag = 0;       /*  菜单有效标志位  */
extern u8 KEY_MatrixScanf(void)
{
    u8 num_key = 16;     /*  按键号  */
    u8 temp = 0x00;        /*  读取 P2 口线数据  */
    static u8 temp_code = 0;  /*  用于保存按键值  */
    static u8 temp_circle = 0xFE;     /*线上的循环扫描值  */
    static u8 num_check = 0;  /*  低电平计数  */
    static bit key_flag = 0;     /*  按键有效标识  */
    KEY_MATRIXPORT = temp_circle;   /* 0xFX */
    temp = KEY_MATRIXPORT;              /*  读取 keyport 数据  */
    if(temp != temp_circle)                /*    有按键动作  */
    {
        if(++num_check >= 5)     /*  连续 5 次(5ms)低电平有效    */
        {
            num_check--;
            key_flag = TRUE;     /*  按键有效标识置 1 */
            temp_code = temp;    /*  保存按键值  */
        }
    }
    else /*  松手 OR 无按键动作,此时应该改变扫描线  */
    {
        num_check = 0;
        if(key_flag)  /*  按键有效判断  */
```

```
{
    key_flag = FALSE;
    switch(temp_code) /*  读取按键号  */
    {
        //0 线
        case 0xEE:   num_key = 1;          break;
        case 0xDE:   num_key = 2;          break;
        case 0xBE:   num_key = 3;          break;
        case 0x7E:   num_key = 10;         break;      /*  上  */
        //1 线
        case 0xED:   num_key = 4;          break;
        case 0xDD:   num_key = 5;          break;
        case 0xBD:   num_key = 6;          break;
        case 0x7D:   num_key = 11;         break;     /*  左  */
        //2 线
        case 0xEB:   num_key = 7;          break;
        case 0xDB:   num_key = 8;          break;
        case 0xBB:   num_key = 9;          break;
        case 0x7B:   num_key = 12;         break;   /*  右  */
        //3 线
        case 0xE7:                                  /* * */
        {
                num_key = 13;
                KEY_MenuFlag = 1;                   /*  菜单有效  */
        }
```

```
                    break;
                    case 0xD7:   num_key = 0;          break;    /* 0 */
                    case 0xB7:   num_key = 14;         break;    /* # */

                    case 0x77:   num_key = 15;         break;    /*  下  */
                    default : num_key = 16;          break;
                }
            }
            temp_circle = crol(temp_circle, 1);/*  改变扫描线  */
            if(0xEF == temp_circle)
            {
                temp_circle = 0xFE;
            }
        }
        return (num_key);/*  返回按键号    */
}
extern void KEY_MenuProcess(bit Time_500MsFlag, u8 KEY_MatrixValue)
{
        static u8 Adj_FirstMenuPos = 0;    /*  菜单项索引位置  */
        if (!Adj_StatusFlag)    /*  进入菜单  */
        {
            if (KEY_MENU == KEY_MatrixValue)
            {
                Adj_MenuFlag = 1;
                Adj_StatusFlag = 1;    /*  自锁  */
```

```
            LCD_ShowFirstMenu(0, Adj_FirstMenuPos);
            KEY_MatrixValue = KEY_NULL;
        }
    }
    else
    {
        if (Adj_MenuFlag)
        {
            switch(KEY_MatrixValue)          //移动光标选择要调整项
            {
                case KEY_UP:
                {
                    if (--Adj_FirstMenuPos == 255)
                    {
                        Adj_FirstMenuPos = 2;
                    }
                    LCD_ShowFirstMenu(1, Adj_FirstMenuPos);     /*显示*/
                }
                break;
                case KEY_DOWN:
                {
                    if (++Adj_FirstMenuPos > 2)
                    {
                        Adj_FirstMenuPos = 0;
                    }
```

```
            LCD_ShowFirstMenu(1, Adj_FirstMenuPos);
        }
        break;
        case KEY_MENU:     /*  确定  */
        {
            Adj_MenuFlag = 0;   /*  允许进入菜单子程序  */

        }
        break;
        case KEY_ESC:   /*  退出  */
        {
            Adj_StatusFlag = 0;   /*  为下次进入菜单做准备  */
            Adj_MenuFlag   = 0;

            Adj_FirstMenuPos = 0;
            KEY_MenuFlag = 0;     /*    退出菜单  */
            LCD_ShowStandby(0);   /*  显示待机界面  */
            KEY_MatrixValue = KEY_NULL;
        }
        break;
        default : break;
    }

}
if (!Adj_MenuFlag)     //
```

```
        {
            switch(Adj_FirstMenuPos)    //菜单
            {
                case 0:     //时间调整
                {
                    DS1302_AdjParameter(Time_500MsFlag, KEY_MatrixValue);
                }
                break;
                case 1:
                {
                    LOCK_ChangePassword(Time_500MsFlag,
KEY_MatrixValue);
                }
                break;
                case 2:
                {
                    LOCK_Administrator(Time_500MsFlag, KEY_MatrixValue);
                }
                break;
            }
        }
    }
}
```

三、LOCK 的驱动源程序

```
/*********************************************************************
```

```
* 文 件 名   : LOCK.c
* 作      者   : 第九组
* 日      期   : 15/10/2013
* 描      述   : LOCK 的驱动源程序
**********************************************************************/
/* 头文件包含 ----------------------------------------------------------*/
#include "locking.h"
#include "at24c02.h"
#include "key.h"
#include "lcd12864.h"
#include "fontlibrary.h"
#include "display.h"
/* 本文件用的全局变量 ----------------------------------------------------*/
u32 LOCK_SetPassword = 891202;   /*设置密码，默认为 891202*/
/* 本程序用的全局变量 ----------------------------------------------------*/
static void LOCK_ShowAdjParameter(bit Time_500MsFlag, u8 yPos, u8 xPos[3])
{
    u8 i = 0, j = 0;
    static u8 yPosTemp = 4;

    if (Time_500MsFlag)
    {
        LCD_ShowNumber(yPos, xPos[yPos-1], '|');
    }
    else
```

```
    {
        LCD_ShowNumber(yPos, xPos[yPos-1], ' ');
    }
    if (yPosTemp != yPos)    //当行坐标发生改变时
    {
        yPosTemp = yPos;      //
        for (i = 1; i < 4; i++)     //除调整行外的其他未使用部分不显示
        {
            if (i != yPos)    //不是当前要调整行
            {
                for (j = xPos[i-1]; j <= 15; j++)
                {
                    LCD_ShowNumber(i, j, ' ');
                }
            }
        }
    }
    for (j = 9; j < 15; j++)
    {
        if (j < xPos[yPos-1])
        {
            LCD_ShowNumber(yPos, xPos[yPos-1]-1, '*');      //输入一个字符就显
示一个*号
        }
        else
```

```
            {
                LCD_ShowNumber(yPos, j+1, ' ');    //将后面部分不显示
            }
        }
    }


static void LOCK_ShowPasswordInit(void)
{
    LCD_ClearScreen(0, 3);
    LCD_ShowChinese(0, 2, Table_Xiu);     //修改密码
    LCD_ShowChinese(0, 3, Table_Gai);
    LCD_ShowChinese(0, 4, Table_Mi);
    LCD_ShowChinese(0, 5, Table_Ma);

    LCD_ShowChinese(1, 0, Table_Chu1);     //初始密码
    LCD_ShowChinese(1, 1, Table_Shi1);
    LCD_ShowChinese(1, 2, Table_Mi);
    LCD_ShowChinese(1, 3, Table_Ma);
    LCD_ShowNumber(1, 8, ':');    //:

    LCD_ShowChinese(2, 0, Table_Shu);     //输入密码
    LCD_ShowChinese(2, 1, Table_Ru);
    LCD_ShowChinese(2, 2, Table_Mi);
    LCD_ShowChinese(2, 3, Table_Ma);
```

```
    LCD_ShowNumber(2, 8, ':');      //:

    LCD_ShowChinese(3, 0, Table_Que);     //确认密码
    LCD_ShowChinese(3, 1, Table_Ren1);
    LCD_ShowChinese(3, 2, Table_Mi);
    LCD_ShowChinese(3, 3, Table_Ma);
    LCD_ShowNumber(3, 8, ':');      //:
}


static void Delay_xms(u16 xms)
{
    u16 y;
    for (xms; xms > 0; xms--)
    {
        for (y = 110; y > 0; y--)
        {
            ;
        }
    }
}


static u32 CP_InputCode[3] = {0};                /* 输入密码 */
static u8 CP_InputPosX[3] = {9, 9, 9};       /* 参数输入的 X 位置 */
```

```
extern void LOCK_ChangePassword(bit Time_500MsFlag, u8 KEY_MatrixValue)
{
    static u8 CP_StatusNum = 0;
    static u8 CP_InputPosY = 1;                    /* 参数输入的 Y 位置 */
    if ((KEY_MENU == KEY_MatrixValue) || (KEY_ESC == KEY_MatrixValue))
    {
        if (KEY_MENU == KEY_MatrixValue)     //确定
        {
            CP_StatusNum++;
            if (1 == CP_StatusNum)
            {
                /* 在此加入关于密码修改的显示界面 */
                LOCK_ShowPasswordInit();
            }
            else if (2 == CP_StatusNum)
            {
                /* 此处加入密码验证 完全正确 就保存 */
                if (LOCK_SetPassword == CP_InputCode[0])  //输入的原密码与设置密码相同时
                {
                    if (CP_InputCode[1] == CP_InputCode[2])  //两次输入的新密码相同时
                    {
                        LOCK_SetPassword = CP_InputCode[1];  //设置修改后的密码
                        LCD_ClearScreen(0, 3);       //   显示修改成功
```

```
                LCD_ShowChinese(1, 3, Table_Mi);     //密码修改成功
                LCD_ShowChinese(1, 4, Table_Ma);
                LCD_ShowChinese(2, 2, Table_Xiu);
                LCD_ShowChinese(2, 3, Table_Gai);
                LCD_ShowChinese(2, 4, Table_Cheng);
                LCD_ShowChinese(2, 5, Table_Gong);
                Delay_xms(3000);
                CP_InputCode[0] = 0;
                CP_InputCode[1] = 0;
                CP_InputCode[2] = 0;
                CP_InputPosX[0] = 9;
                CP_InputPosX[1]     = 9;
                CP_InputPosX[2]     = 9;
                CP_StatusNum = 0;
                CP_InputPosY = 1;//清零 Adj_MenuFlag= 1;   /*  一级菜
单解锁  */
                LCD_ShowFirstMenu(0, 1);

            }
            else
            {
                if ((CP_InputPosX[1] < 12) || (CP_InputPosX[2] < 12))
                {         //输入密码少于 4 位
                    LCD_ClearScreen(0, 3);
                    LCD_ShowChinese(1, 2, Table_Shu);
```

```
        LCD_ShowChinese(1, 3, Table_Ru);
        LCD_ShowChinese(1, 4, Table_Mi);
        LCD_ShowChinese(2, 1, Table_Ma);
        LCD_ShowChinese(2, 2, Table_Di);
        LCD_ShowChinese(2, 3, Table_Yu);
        LCD_ShowChinese(2, 4, Table_Si);
        LCD_ShowChinese(2, 5, Table_Wei);
    }
    else
    {    //加入两次输入的密码不一致

        LCD_ClearScreen(0, 3);
        LCD_ShowChinese(1, 2, Table_Liang);
        LCD_ShowChinese(1, 3, Table_Ci);
        LCD_ShowChinese(1, 4, Table_Shu);
        LCD_ShowChinese(1, 5, Table_Ru);
        LCD_ShowChinese(2, 1, Table_De);
        LCD_ShowChinese(2, 2, Table_Mi);
        LCD_ShowChinese(2, 3, Table_Ma);
        LCD_ShowChinese(2, 4, Table_Bu);
        LCD_ShowChinese(2, 5, Table_Yi);
        LCD_ShowChinese(2, 6, Table_Zhi);

    }
    Delay_xms(3000);
```

```
                LOCK_ShowPasswordInit();    //从新显示修改密码界面
                CP_InputCode[0] = 0;
                CP_InputCode[1] = 0;
                CP_InputCode[2] = 0;
                CP_InputPosX[0] = 9;
                CP_InputPosX[1]     = 9;
                CP_InputPosX[2]     = 9;
                CP_StatusNum = 1;
                CP_InputPosY = 1;   //清零操
作
                }
            }
            else
            {     //加入显示  输入原密码不正确
                LCD_ClearScreen(0, 3);
                LCD_ShowChinese(1, 2, Table_Shu);
                LCD_ShowChinese(1, 3, Table_Ru);
                LCD_ShowChinese(1, 4, Table_Yuan1);
                LCD_ShowChinese(2, 1, Table_Mi);
                LCD_ShowChinese(2, 2, Table_Ma);
                LCD_ShowChinese(2, 3, Table_Bu);
                LCD_ShowChinese(2, 4, Table_Zheng1);
                LCD_ShowChinese(2, 5, Table_Que);
                Delay_xms(3000);
                LOCK_ShowPasswordInit();    //从新显示修改密码界面
```

```
                    CP_InputCode[0] = 0;
                    CP_InputCode[1] = 0;
                    CP_InputCode[2] = 0;
                    CP_InputPosX[0] = 9;
                    CP_InputPosX[1]     = 9;
                    CP_InputPosX[2]     = 9;
                    CP_StatusNum = 1;
                    CP_InputPosY = 1;   //清零操
                }
            }
        }
        else if (KEY_ESC == KEY_MatrixValue)
        {
            CP_InputCode[0] = 0;
            CP_InputCode[1] = 0;
            CP_InputCode[2] = 0;
            CP_InputPosX[0] = 9;
            CP_InputPosX[1]     = 9;
            CP_InputPosX[2]     = 9;
            CP_StatusNum = 0;
            CP_InputPosY = 1;   //清零操作
            Adj_MenuFlag   = 1;/*  一级菜单解锁  */
            LCD_ShowFirstMenu(0, 1);
        }
    }
```

```
    else
    {
        if (1 == CP_StatusNum)
        {
            switch(KEY_MatrixValue)
            {
                case 0:
                case 1:
                case 2:
                case 3:
                case 4:
                case 5:
                case 6:
                case 7:
                case 8:
                case 9:   /* 注以上无 break */
                {
                    if ((CP_InputPosX[CP_InputPosY-1] >= 9) && (CP_InputPosX[CP_InputPosY-1] <= 15))
                    {
                        CP_InputCode[CP_InputPosY-1] = CP_InputCode[CP_InputPosY-1]*10+KEY_MatrixValue;   /* 输入密码 */
                    }
                    if (++CP_InputPosX[CP_InputPosY-1] > 15)
                    {
```

```
            CP_InputPosX[CP_InputPosY-1] = 15;
        }
    }
    break;
    case KEY_DOWN:
    {
        if (++CP_InputPosY > 3)
        {
            CP_InputPosY = 1;
        }
    }
    break;
    case KEY_UP:
    {
        if (--CP_InputPosY == 0)
        {
            CP_InputPosY = 3;
        }
    }
    break;
    case KEY_LEFT:        /*  删除一位密码  */
    {
       if (--CP_InputPosX[CP_InputPosY-1] < 9)
       {
            CP_InputPosX[CP_InputPosY-1] = 9;
```

```
                    }
                    CP_InputCode[CP_InputPosY-1] /= 10;     //删除一位
                }
                break;
            }
            LOCK_ShowAdjParameter(Time_500MsFlag, CP_InputPosY, CP_InputPosX); /*  显示调整参数  */
        }
    }
}
u8 SetAlarmTime = 10;          /*  报警时间.秒  */       //10-90
u8 SetOpenLockTime = 30;    /*  输入时间限制.秒  */ //5-60
u8 SetOpenLockNum = 3;          /*  开锁次数限制.次  */ //2-6
u8 SetLockTime = 30;           /*  自锁时间.秒  */    //10-90
static u8 yPosTemp = 4;        //调整管理员权限时显示控制用

static void LOCK_ShowAdjAdminPar(bit Time_500MsFlag, u8 yPos, u8 ShowData[4])
{
    u8 i = 0;
    if (yPosTemp != yPos)     //下次数据与上次数据是否不同？
    {
        yPosTemp = yPos;    //备份本次数据
        for (i = 0; i < 4; i++)
        {
```

```
            if (i != yPos)
            {
                LCD_ShowNumber(i, 11, ShowData[i]/10);
                LCD_ShowNumber(i, 12, ShowData[i]%10);     //刷新非调整数据
                LCD_ShowNumber(i, 0, ' ');
            }
            else
            {
                LCD_ShowNumber(yPos, 0, '*');    //调整项光标指示
            }
        }
    }
    else
    {
        if (Time_500MsFlag)      //闪灯显示正在调整项
        {
            LCD_ShowNumber(yPos, 11, ShowData[yPos]/10);
            LCD_ShowNumber(yPos, 12, ShowData[yPos]%10);
        }
        else
        {
            LCD_ShowNumber(yPos, 11, ' ');
            LCD_ShowNumber(yPos, 12, ' ');
        }
    }
```

```
}

static u8 LOCK_AdjDataTemp[4]={0};

extern void LOCK_Administrator(bit Time_500MsFlag, u8 KEY_MatrixValue)
{
    static u8 LOCK_AdjStatus = 0;   //菜单状态
    static u8 LOCK_AdjPosY = 0;         //行坐标控制
    static u32 x = 0;     //输入数据缓存
    if ((KEY_MENU == KEY_MatrixValue) || (KEY_ESC == KEY_MatrixValue))
    {
        if (KEY_MENU == KEY_MatrixValue)      //enter
        {
            LOCK_AdjStatus++;
            if (1 == LOCK_AdjStatus)     /*  第一次按下菜单按钮时  */
            {
                LCD_ClearScreen(0, 3);
                LCD_ShowNumber(0, 0, '*');
                LCD_ShowChinese(0, 1, Table_Bao);     //报警时间
                LCD_ShowChinese(0, 2, Table_Jing);
                LCD_ShowChinese(0, 3, Table_Shi);
                LCD_ShowChinese(0, 4, Table_Jian);
                LCD_ShowNumber(0, 10, ':');
                LCD_ShowChinese(0, 7, Table_Miao);     //秒
```

```
LCD_ShowChinese(1, 1, Table_Chao);     //超时时间
LCD_ShowChinese(1, 2, Table_Shi);
LCD_ShowChinese(1, 3, Table_Shi);
LCD_ShowChinese(1, 4, Table_Jian);
LCD_ShowNumber(1, 10, ':');    //:
LCD_ShowChinese(1, 7, Table_Miao); //秒

LCD_ShowChinese(2, 1, Table_Shu);     //输入次数
LCD_ShowChinese(2, 2, Table_Ru);
LCD_ShowChinese(2, 3, Table_Ci);
LCD_ShowChinese(2, 4, Table_Shu1);
LCD_ShowNumber(2, 10, ':');     //:
LCD_ShowChinese(2, 7, Table_Ci);    //次

LCD_ShowChinese(3, 1, Table_Zi);     //自锁时间
LCD_ShowChinese(3, 2, Table_Suo);
LCD_ShowChinese(3, 3, Table_Shi);
LCD_ShowChinese(3, 4, Table_Jian);
LCD_ShowNumber(3, 10, ':');     //:
LCD_ShowChinese(3, 7, Table_Miao);    //秒

LOCK_AdjDataTemp[0] = SetAlarmTime;    //读取系统设置
LOCK_AdjDataTemp[1] = SetOpenLockTime;
LOCK_AdjDataTemp[2] = SetOpenLockNum;
LOCK_AdjDataTemp[3] = SetLockTime;
```

```
            }
            if (2 == LOCK_AdjStatus)
            {
                Adj_MenuFlag   = 1;/* 一级菜单解锁 */
                LCD_ShowFirstMenu(0, 2);

                SetAlarmTime = LOCK_AdjDataTemp[0];   //设置数据保存
                SetOpenLockTime = LOCK_AdjDataTemp[1];
                SetOpenLockNum = LOCK_AdjDataTemp[2];
                SetLockTime = LOCK_AdjDataTemp[3];

                x = 0;
                yPosTemp = 4;
                LOCK_AdjStatus = 0;
                LOCK_AdjPosY = 0;
            }
        }
        else if (KEY_ESC == KEY_MatrixValue)   //返回但不保存已经调整好的
数据
        {
            Adj_MenuFlag   = 1;/* 一级菜单解锁 */
            LCD_ShowFirstMenu(0, 2);

            x = 0;
            yPosTemp = 4;
```

```
            LOCK_AdjStatus = 0;
            LOCK_AdjPosY = 0;    /*  清零操作  */
        }
    }
    else
    {
        if (1 == LOCK_AdjStatus)
        {
            switch(KEY_MatrixValue)
            {
                case KEY_UP:      /*调整光标选择相应的设置项  */
                {
                    if (--LOCK_AdjPosY == 255)
                    {
                        LOCK_AdjPosY = 3;
                    }
                    x = 0;
                }
                break;
                case KEY_DOWN:
                {
                    if (++LOCK_AdjPosY > 3)
                    {
                        LOCK_AdjPosY = 0;
                    }
```

```
            x = 0;
        }
        break;
    }
    switch(LOCK_AdjPosY)
    {
        case 0:   //调整第一项
        {
            if (KEY_MatrixValue <= 9)
            {
                x = x*10+KEY_MatrixValue;
                if (x > 99)
                {
                    x %= 100;
                }
                if (x < 10)      //报警最小时间 10s
                {
                    LOCK_AdjDataTemp[0] = 10;
                }
                else if (x > 90)
                {
                    LOCK_AdjDataTemp[0] = 90;     //超时报警最大时
间 99s
                }
                else
```

```
            {
                LOCK_AdjDataTemp[0] = x;
            }
        }
    }
    break;
    case 1:
    {
        if (KEY_MatrixValue <= 9)
        {
            x = x*10+KEY_MatrixValue;
            if (x > 99)
            {
                x %= 100;
            }
            if (x < 5)            //超时最小时间 5s
            {
                LOCK_AdjDataTemp[1] = 5;
            }
            else if (x > 60)      //超时最大时间 60s
            {
                LOCK_AdjDataTemp[1] = 60;
            }
            else
            {
```

```
                LOCK_AdjDataTemp[1] = x;
            }
        }
    }
    break;
    case 2:
    {
        if (KEY_MatrixValue <= 9)
        {
            x = x*10+KEY_MatrixValue;
            if (x > 99)
            {
                x %= 100;
            }

            if (x < 2)              //超次次数最少 2 次
            {
                LOCK_AdjDataTemp[2]  = 2;
            }
            else if (x > 6)      //超次次数最少 6 次
            {
                LOCK_AdjDataTemp[2]  = 6;
            }
            else
            {
```

```
                LOCK_AdjDataTemp[2] = x;
            }
        }
    }
    break;
    case 3:
    {
        if (KEY_MatrixValue <= 9)
        {
            x = x*10+KEY_MatrixValue;
            if (x > 99)
            {
                x %= 100;
            }

            if (x < 10)    //自锁时间最少 10
            {
                LOCK_AdjDataTemp[3] = 10;
            }
            else if (x > 90)    //自锁时间最大 90
            {
                LOCK_AdjDataTemp[3] = 90;
            }
            else
            {
```

```
                                LOCK_AdjDataTemp[3] = x;
                            }
                        }
                    }
                    break;
                }
                LOCK_ShowAdjAdminPar(Time_500MsFlag, LOCK_AdjPosY, &LOCK_AdjDataTemp);
            }
        }
    }


static void LOCK_OpenShowPar(bit Time_500MsFlag, u8 xPos, u8 InputTime)
{
    u8 i = 0;
    if (Time_500MsFlag)
    {
        LCD_ShowNumber(1, xPos, '|');
    }
    else
    {
        LCD_ShowNumber(1, xPos, ' ');
    }
    for (i = 5; i < 11; i++)
```

```
    {
        if (i < xPos)
        {
            LCD_ShowNumber(1, xPos-1, '*');          //输入一个字符就显示一个*
号
        }
        else
        {
            LCD_ShowNumber(1, i+1, ' ');   //将后面部分清显示
        }
    }
    LCD_ShowNumber(2, 10, InputTime/10);
    LCD_ShowNumber(2, 11, InputTime%10);
}



static void LCOK_Meun(void)
{
    LCD_ClearScreen(0, 3);
    LCD_ShowChinese(0, 2, Table_Qing);     //请输入密码
    LCD_ShowChinese(0, 3, Table_Shu);
    LCD_ShowChinese(0, 4, Table_Ru);
    LCD_ShowChinese(0, 5, Table_Mi);
    LCD_ShowChinese(0, 6, Table_Ma);
```

```
    LCD_ShowChinese(1, 0, Table_Mi);
    LCD_ShowChinese(1, 1, Table_Ma);   //密码
    LCD_ShowNumber(1, 4, ':');    //:

    LCD_ShowChinese(2, 0, Table_Shi);
    LCD_ShowChinese(2, 1, Table_Jian);
    LCD_ShowChinese(2, 2, Table_Xian);
    LCD_ShowChinese(2, 3, Table_Zhi);   //时间限制
    LCD_ShowNumber(2, 8, ':');    //:
    LCD_ShowChinese(2, 6, Table_Miao);

    LCD_ShowChinese(3, 0, Table_Kai);
    LCD_ShowChinese(3, 1, Table_Suo);    //开锁
    LCD_ShowChinese(3, 6, Table_Fan);
    LCD_ShowChinese(3, 7, Table_Hui);    //返回
}

bit OpenLockFlag = 1;    //开锁标志  0 已经开，1 为关闭状态
bit OpenLockStatusFlag = 1;    //
u8 LockTime = 0;    //开锁后的计数器


extern void LOCK_OpenProcess(bit Time_500MsFlag, u8 KEY_MatrixValue, u8 *
pTime_1s)
{
```

```
u8 * pTable = (void *)0;
static bit LOCK_MenuFlag = 0; //菜单自锁
static u8 LOCK_InputPosX = 5; //行基坐标
static u8 LOCK_InputNum = 3;    //输入最大次数
static u8 LOCK_InputTime = 0;    //输入最大时间
static u32 x = 0;
if (!LOCK_MenuFlag)
{
    if (KEY_MENU == KEY_MatrixValue)
    {
        LOCK_MenuFlag = 1;    //自锁
        KEY_MatrixValue = KEY_NULL;
        LCOK_Meun();
        LOCK_InputNum = SetOpenLockNum;
        LOCK_InputTime = SetOpenLockTime; //得到设置的最大时间限制
    }
}
else
{
    if (KEY_NULL != KEY_MatrixValue)
    {
        LOCK_InputTime  =  SetOpenLockTime;  //得到设置的最大时间限制

    }
    if (*pTime_1s)
```

```
{
    (*pTime_1s) = 0;
    LOCK_InputTime--;
    if (LOCK_InputTime == 0)    //用户输入已经到最大时间限制
    {
        LOCK_MenuFlag = 0;    //解锁
        LOCK_InputPosX = 5;
        LOCK_InputNum = 3;
        LOCK_InputTime = 0;
        x = 0;
        OpenLockStatusFlag = 1; //退出用户输入密码状态
        KEY_MenuFlag = 0;
        LCD_ShowStandby(1);          //返回到待机界面
        KEY_MatrixValue = KEY_NULL;
        return ;    //
    }
}
if (KEY_MatrixValue <= 9)
{
    if ((LOCK_InputPosX >= 5) && (LOCK_InputPosX <= 10))
    {
        x = x*10+KEY_MatrixValue;
    }
    if (++LOCK_InputPosX > 10)
    {
```

```
            LOCK_InputPosX = 11;
        }
    }
    else if (KEY_LEFT == KEY_MatrixValue)
    {
        if (--LOCK_InputPosX < 5)          //删除一位
        {
            LOCK_InputPosX = 5;
        }
        x /= 10;    //
    }
    else if (KEY_MENU == KEY_MatrixValue)     //开锁验证
    {
        KEY_MatrixValue = KEY_NULL;
        LOCK_InputPosX = 5;    //输入行坐标回到基址
        LCD_ClearScreen(0, 3);
        if (LOCK_SetPassword == x)      //密码正确
        {
            x = 0;        //缓存清零
            LOCK_InputNum = 3;
            LOCK = 0;    //开锁
            OpenLockFlag = 0;//指示开锁状态为开！
            LOCK_MenuFlag = 0;    //解锁

            LCD_ShowChinese(1, 2, Table_Kai);      //开锁成功
```

```
        LCD_ShowChinese(1, 3, Table_Suo);
        LCD_ShowChinese(1, 4, Table_Cheng);
        LCD_ShowChinese(1, 5, Table_Gong);
        LCD_ShowChinese(2, 1, Table_Huan);     //     欢迎你进入
        LCD_ShowChinese(2, 2, Table_Ying);
        LCD_ShowChinese(2, 3, Table_Nin);
        LCD_ShowChinese(2, 4, Table_Jin);
        LCD_ShowChinese(2, 5, Table_Ru);
        Delay_xms(2000);
        OpenLockStatusFlag = 1;   //退出用户输入密码状态
        KEY_MenuFlag = 0;
        LockTime = SetLockTime; //开锁后开始计数
        LCD_ShowStandby(0);//  返回到待机界面
        return ;  //退出
    }
    else//密码不正确
    {
        x = 0;        //缓存清零
        if (--LOCK_InputNum < 1)     //记录输入密码错误的次数
        {
            LOCK_InputNum = 0;
        }
        LOCK = 1; //不开锁
        OpenLockFlag = 1;   //指示开锁状态  为关闭
        LOCK_MenuFlag = 1;   //不解锁
```

```
ALARM = 0;   //开报警
if (0 != LOCK_InputNum)          //错误次数少于三次
{

    LCD_ShowChinese(1, 1, Table_Mi);     //密码错误
    LCD_ShowChinese(1, 2, Table_Ma);
    LCD_ShowChinese(1, 3, Table_Cuo);
    LCD_ShowChinese(1, 4, Table_Wu1);
    LCD_ShowNumber(1, 10, '!');
    LCD_ShowNumber(1, 11, '!');  //!
    LCD_ShowChinese(2, 1, Table_Hai);     //还有 x 次机会
    LCD_ShowChinese(2, 2, Table_You);
    switch(LOCK_InputNum)
    {
        case 1:   pTable = Table_Yi;    break;
        case 2:   pTable = Table_Er;    break;
        case 3:   pTable = Table_San;  break;
        case 4:   pTable = Table_Si;    break;
        case 5:   pTable = Table_Wu;   break;
        case 6:   pTable = Table_Liu;   break;
        default:  break;
    }
    LCD_ShowChinese(2, 3, pTable);       //
    LCD_ShowChinese(2, 4, Table_Ci);
    LCD_ShowChinese(2, 5, Table_Ji);
```

```
        LCD_ShowChinese(2, 6, Table_Hui);
        Delay_xms(3000); //等待 3s
        ALARM = 1; //关报警
        LCOK_Meun(); //重新显示输入密码
    }
    else       //
    {
        LOCK_MenuFlag = 0;    //解锁
        LOCK_InputNum = 3;
        //加入非法入侵，键盘已经锁定 30s 报警，返回待机界面
        LCD_ShowChinese(1, 2, Table_Fei);     //非法入侵
        LCD_ShowChinese(1, 3, Table_Fa);
        LCD_ShowChinese(1, 4, Table_Ru);
        LCD_ShowChinese(1, 5, Table_Qin1);
        LCD_ShowChinese(2, 1, Table_Jian1);     //键盘已经锁定
        LCD_ShowChinese(2, 2, Table_Pan);
        LCD_ShowChinese(2, 3, Table_Yi1);
        LCD_ShowChinese(2, 4, Table_Jing1);
        LCD_ShowChinese(2, 5, Table_Suo);
        LCD_ShowChinese(2, 6, Table_Ding);
        KEY_LED = 0;  //指示键盘已经锁定
        if (SetAlarmTime < 60)
        {
            Delay_xms(SetAlarmTime*1000);   //  报警延时
        }
```

```
                    else
                    {
                        Delay_xms(60000);   // 报警延时
                        Delay_xms((SetAlarmTime-60)*1000);   // 报警延时
                    }
                    ALARM = 1; //关报警
                    KEY_LED = 1;   //键盘已经解锁
                    //加入退出密码输入状态的程序
                    OpenLockStatusFlag = 1;   //退出用户输入密码状态
                    KEY_MenuFlag = 0;
                    LCD_ShowStandby(1);// 返回到待机界面

                    return ;
                }
            }
        }
        LOCK_OpenShowPar(Time_500MsFlag, LOCK_InputPosX, LOCK_InputTime);
        if (KEY_ESC == KEY_MatrixValue)     //退出开锁状态
        {
            LOCK_MenuFlag = 0;   //解锁
            LOCK_InputPosX = 5;
            LOCK_InputNum = 3;
            x = 0;
            OpenLockStatusFlag = 1;   //退出用户输入密码状态
```

```
                KEY_MenuFlag = 0;
                LCD_ShowStandby(1);// 返回到待机界面
                KEY_MatrixValue = KEY_NULL;
            }
        }
    }
```

四、DS1302 实时时钟的驱动源程序

```
/*********************************************************************
* 文 件 名   : DS1302.c
* 作      者  : 第九组成员
* 日      期  : 10/25/2013
* 描      述  : DS1302 实时时钟的驱动源程序
*********************************************************************/
/* 头文件包含 ----------------------------------------------------------*/
#include "ds1302.h"
#include "KEY.H"
#include "lcd12864.h"
#include "fontlibrary.h"
#include "display.h"
/* 操作地址常量定义 ----------------------------------------------------*/
#define DS1302_ReadYearAddr             ((u8)0x8D)
#define DS1302_ReadWeekAddr             ((u8)0x8B)
#define DS1302_ReadMonthAddr      ((u8)0x89)
#define DS1302_ReadDayAddr              ((u8)0x87)
#define DS1302_ReadHourAddr             ((u8)0x85)
```

```
#define DS1302_ReadMinuteAddr           ((u8)0x83)
#define DS1302_ReadSecondAddr           ((u8)0x81)        /* 读地址 */
#define DS1302_WriteYearAddr        ((u8)0x8C)
#define DS1302_WriteWeekAddr        ((u8)0x8A)
#define DS1302_WriteMonthAddr       ((u8)0x88)
#define DS1302_WriteDayAddr             ((u8)0x86)
#define DS1302_WriteHourAddr        ((u8)0x84)
#define DS1302_WriteMinuteAddr          ((u8)0x82)
#define DS1302_WriteSecondAddr          ((u8)0x80)        /* 写地址 */
/* 内部函数申明 ------------------------------------------------------------*/
static void DS1302_WriteOneByte(u8 Addr, u8 Value);
static u8 DS1302_ReadOneByte(u8 Addr);
DS1302_TypeDef CurrentTime;          /* 定义时间结构体 */
static void DS1302_WriteOneByte(u8 Addr, u8 Value)
{
    u8 i = 0;
    Value = ((Value/10)<<4) + (Value%10); /* 10 进制数转换成 BCD 码 */
    DS1302_RST = 0;
    DS1302_SCLK = 0;
    DS1302_RST = 1;          /* 首先复位 */
    for (i = 0; i < 8; i++)     /* 先写入地址 */
    {
        DS1302_IO = Addr & 0x01;       /* 每次取低位 */
        DS1302_SCLK = 0;
        DS1302_SCLK = 1;    /*上升沿有效 */
```

```
        Addr >>= 1;
    }
    for (i = 0; i < 8; i++)        /*  在写入数据  */
    {
        DS1302_IO = Value & 0x01;
        DS1302_SCLK = 0;
        DS1302_SCLK = 1;
        Value >>= 1;
    }
    DS1302_RST = 0;
}
static u8 DS1302_ReadOneByte(u8 Addr)
{
    u8 i = 0;
    u8 GetTimeValue = 0;
    DS1302_RST = 0;
    DS1302_SCLK = 0;
    DS1302_RST = 1;
    for (i = 0; i < 8; i++)     /*  先写入地址  */
    {
        DS1302_IO = Addr & 0x01;
        DS1302_SCLK = 0;
        DS1302_SCLK = 1;       /*上升沿有效  */
        Addr >>= 1;
    }
```

```
    for (i = 0; i < 8; i++)     /*在读取数据  */
    {
        if (1 == DS1302_IO)
        {
            GetTimeValue |= 0x80;     /*每次取高位  */
        }
        DS1302_SCLK = 1;
        DS1302_SCLK = 0;     /*下降沿有效  */
        GetTimeValue >>= 1;
    }
    DS1302_RST = 0;
    return    (((GetTimeValue>>4)*10) + GetTimeValue%16);    /*    BCD  码转换成
10 进制数  */
}

extern void DS1302_WriteCurrentTime(DS1302_TypeDef * pCurrentTime)
{
    DS1302_WriteOneByte(0x8E, 0x00);    /*  关闭写保护  */
    DS1302_WriteOneByte(0x84, 0x00);    /* 24 小时模式  */
    DS1302_WriteOneByte(DS1302_WriteYearAddr, pCurrentTime->Year);
    DS1302_WriteOneByte(DS1302_WriteWeekAddr, pCurrentTime->Week);
    DS1302_WriteOneByte(DS1302_WriteMonthAddr, pCurrentTime->Month);
    DS1302_WriteOneByte(DS1302_WriteDayAddr, pCurrentTime->Day);
    DS1302_WriteOneByte(DS1302_WriteHourAddr, pCurrentTime->Hour);
    DS1302_WriteOneByte(DS1302_WriteMinuteAddr, pCurrentTime->Minute);
```

```
    DS1302_WriteOneByte(DS1302_WriteSecondAddr, pCurrentTime->Second);
//  DS1302_WriteOneByte(DS1302_WriteSecondAddr, 0x00);     //启动时间
    DS1302_WriteOneByte(0x8E, 0x80);        /* 开写保护   防数据出错   */
}

extern void DS1302_ReadCurrentTime(void)
{
    CurrentTime.Year = DS1302_ReadOneByte(DS1302_ReadYearAddr);
    CurrentTime.Week = DS1302_ReadOneByte(DS1302_ReadWeekAddr);
    CurrentTime.Month = DS1302_ReadOneByte(DS1302_ReadMonthAddr);
    CurrentTime.Day = DS1302_ReadOneByte(DS1302_ReadDayAddr);
    CurrentTime.Hour = DS1302_ReadOneByte(DS1302_ReadHourAddr);
    CurrentTime.Minute = DS1302_ReadOneByte(DS1302_ReadMinuteAddr);
    CurrentTime.Second = DS1302_ReadOneByte(DS1302_ReadSecondAddr);
}

static void DS1302_ShowAdjParameter(bit Time_500MsFlag, u8 yStar, u8 xStar, u8
SetValue, bit Week)
{
    if (Time_500MsFlag)
    {
        if (Week)   //调整星期时专用
        {
            LCD_ShowChinese(yStar, xStar, Table0);   //
        }
```

```
            else
            {
                LCD_ShowNumber(yStar, xStar, 12);  //显示空白
                LCD_ShowNumber(yStar, xStar+1, 12);    //显示空白
            }
        }
        else
        {
            if (Week)   //星期专用
            {
            #ifdef __DS1302_DATE__
                LCD_ShowWeek(yStar, xStar, SetValue%10);   //
            #endif
            }
            else
            {
                LCD_ShowNumber(yStar, xStar, SetValue/10);     //显示高位
                LCD_ShowNumber(yStar, xStar+1, SetValue%10);     //显示低位

            }

        }
}

extern void DS1302_AdjParameter(bit Time_500MsFlag, u8 KEY_MatrixValue)
```

```
{
    static u8 AdjFlag = 0;
#ifdef __DS1302_DATE__
    static u8 AdjPosY = 0;
#endif

#ifndef __DS1302_DATE__
    u8 AdjPosY = 1;
#endif
    static u8 AdjPosX = 0;
    static u32 x = 0;
    if ((KEY_MENU == KEY_MatrixValue) || (KEY_ESC == KEY_MatrixValue))
    {
        if (KEY_MENU == KEY_MatrixValue)
        {
            AdjFlag++;
            if (1 == AdjFlag)
            {
                LCD_ClearScreen(0, 3);      //清屏
                LCD_ShowChinese(0, 0, Table0);
                LCD_ShowChinese(0, 1, Table0);
                LCD_ShowChinese(0, 2, Table_Shi);
                LCD_ShowChinese(0, 3, Table_Jian);
                LCD_ShowChinese(0, 4, Table_Tiao);
                LCD_ShowChinese(0, 5, Table_Zheng);
```

```
#ifdef __DS1302_DATE__
    LCD_ShowChinese(1, 2, Table_Nian);     //年
    LCD_ShowChinese(1, 4, Table_Yue);
    LCD_ShowChinese(1, 6, Table_Ri);
    LCD_ShowChinese(3, 0, Table_Xing);  //星期
    LCD_ShowChinese(3, 1, Table_Qi);
#endif
    LCD_ShowChinese(2, 3, Table_Shi);   //时
    LCD_ShowChinese(2, 5, Table_Fen);
    LCD_ShowChinese(2, 7, Table_Miao);

#ifdef __DS1302_DATE__
    LCD_ShowNumber(1, 0, 2);
    LCD_ShowNumber(1, 1, 0);     //年的 20 固定显示，
    DS1302_ShowAdjParameter(0, 1, 2, CurrentTime.Year, 0);
    DS1302_ShowAdjParameter(0, 1, 6, CurrentTime.Month, 0);
    DS1302_ShowAdjParameter(0, 1, 10, CurrentTime.Day, 0);

    DS1302_ShowAdjParameter(0, 3, 2, CurrentTime.Week, 1);     /*显
示参数*/
#endif

    DS1302_ShowAdjParameter(0, 2, 4, CurrentTime.Hour, 0);

    DS1302_ShowAdjParameter(0, 2, 8, CurrentTime.Minute, 0);
    DS1302_ShowAdjParameter(0, 2, 12, CurrentTime.Second, 0);
```

```
        }
        else if (2 == AdjFlag)
        {
            AdjFlag = 0;
            x = 0;
            AdjPosX = 0;
            AdjPosY = 0;
            /* 在此处加入调整后的数据保存，并写入 DS1302 中*/
            DS1302_WriteCurrentTime(&CurrentTime);

            Adj_MenuFlag   = 1;   /*返回一级菜单 */
            LCD_ShowFirstMenu(0, 0);
        }
    }
    else if (KEY_ESC == KEY_MatrixValue)
    {
        AdjFlag = 0;
        x = 0;
        AdjPosX = 0;
        AdjPosY = 0;

        Adj_MenuFlag   = 1;/* 一级菜单解锁 */
        LCD_ShowFirstMenu(0, 0);
    }
    KEY_MatrixValue = KEY_NULL;
```

```
    }
    else
    {
        if (1 == AdjFlag)
        {
            switch(KEY_MatrixValue)
            {
            #ifdef __DS1302_DATE__
                case  KEY_DOWN:   //选择光标到要调整的相应行  即纵向光标选择
                {
                    if (++AdjPosY > 2)
                    {
                        AdjPosY = 0;
                    }
                    AdjPosX = 0;   //每行光标变化  时，将列光标指向  当前行的首位要设置的参数
                }
                break;
                case KEY_UP:   //选择光标到要调整的相应行
                {
                    if (--AdjPosY == 255)
                    {
                        AdjPosY = 2;
                    }
```

```
                AdjPosX = 0;
            }
            break;
        #endif
            case KEY_RIGHT:          //选择光标到要调整的相应列，即横向光
标选择
            {
                if (++AdjPosX > 2)
                {
                    AdjPosX = 0;
                }
                x = 0;
            }
            break;
            case KEY_LEFT:
            {
                if (--AdjPosX == 255)
                {
                    AdjPosX = 2;
                }
                x = 0;
            }
            break;
        }
        switch (AdjPosY)     //根据光标对应行，来确定列光标位置
```

```
{
#ifdef __DS1302_DATE__
    case 0:   //调整年月日
    {
        switch(AdjPosX)     //相光标来进行相应位置的显示调整
        {
            case 0:     //调年
            {
                if (KEY_MatrixValue <= 9)
                {
                    x = x * 10 + KEY_MatrixValue;
                    if (x>99)
                    {
                        x %= 100;
                    }
                    CurrentTime.Year = x;
                }
                if (Time_500MsFlag)     //闪烁显示  数字 20
                {
                    LCD_ShowNumber(1, 0, 12);     //空白
                    LCD_ShowNumber(1, 1, 12);     //
                }
                else
                {
                    LCD_ShowNumber(1, 0, 2);     //显示高位
```

```
                LCD_ShowNumber(1, 1, 0);     //显示低位

            }
            DS1302_ShowAdjParameter(0, 3, 2, CurrentTime.Week, 1);
            DS1302_ShowAdjParameter(0, 1, 6, CurrentTime.Month, 0);
            DS1302_ShowAdjParameter(0, 1, 10, CurrentTime.Day, 0); //当调整选项光标发生变化时，刷新前面显示设置的内容
            DS1302_ShowAdjParameter(Time_500MsFlag, 1, 2, CurrentTime.Year, 0); //闪烁显示当前光标处内容

        }
        break;
        case 1:    //调整月
        {

            if (KEY_MatrixValue <= 9)
            {
                x = x * 10 + KEY_MatrixValue;
                if (x > 99)
                {
                    x %= 100;
                }
                if (x <= 0)
                    CurrentTime.Month = 1;
                else if (x > 12)
```

```
                CurrentTime.Month = 12;
            else
                CurrentTime.Month = x;
        }
        LCD_ShowNumber(1, 0, 2);
        LCD_ShowNumber(1, 1, 0);    //年的 20 固定显示，

        DS1302_ShowAdjParameter(0, 1, 2, CurrentTime.Year, 0);
        DS1302_ShowAdjParameter(0, 1, 10, CurrentTime.Day, 0); //当调整选项光标发生变化时，刷新前面显示设置的内容
        DS1302_ShowAdjParameter(Time_500MsFlag, 1, 6, CurrentTime.Month, 0);    //闪烁显示当前光标处内容
    }
    break;
    case 2:    //调整日
    {
        if (KEY_MatrixValue <= 9)
        {
            x = x * 10 + KEY_MatrixValue;
            if (x>99)
            {
                x %= 100;
            }
            if (x > 31)
                CurrentTime.Day = 31;
```

```
                    else if (x <= 0)
                        CurrentTime.Day = 1;
                    else
                        CurrentTime.Day = x;

                }
                LCD_ShowNumber(1, 0, 2);
                LCD_ShowNumber(1, 1, 0);     //年的 20 固定显示，

                DS1302_ShowAdjParameter(0, 1, 2, CurrentTime.Year, 0);
                DS1302_ShowAdjParameter(0, 1, 6, CurrentTime.Month, 0); //当调整选项光标发生变化时，刷新前面显示设置的内容
                DS1302_ShowAdjParameter(Time_500MsFlag, 1, 10, CurrentTime.Day, 0); //闪烁显示当前光标处内容
                }
                break;
            }
        }
        break;
    #endif
        case 1:      //时分秒
        {
            switch(AdjPosX)     //相光标来进行相应位置的显示调整
            {
                case 0:     //调时
```

```
                {
                    if (KEY_MatrixValue <= 9)
                    {
                        x = x * 10 + KEY_MatrixValue;
                        if (x>99)
                        {
                            x %= 100;
                        }
                        if (x > 23)
                            CurrentTime.Hour = 23;
                        else
                            CurrentTime.Hour = x;
                    }
                #ifdef __DS1302_DATE__
                    LCD_ShowNumber(1, 0, 2);
                    LCD_ShowNumber(1, 1, 0);     //年的 20 固定显示，

                   DS1302_ShowAdjParameter(0, 1, 2, CurrentTime.Year,
0);
                    DS1302_ShowAdjParameter(0, 1, 6,
CurrentTime.Month, 0); //当调整选项光标发生变化时，刷新前面显示设置的内容
                    DS1302_ShowAdjParameter(0, 1, 10,
CurrentTime.Day, 0);
                    /*以上为刷新显示第一（即年月日行）数据内容*/

                #endif
```

```
                    DS1302_ShowAdjParameter(Time_500MsFlag, 2, 4,
CurrentTime.Hour, 0);     //闪烁显示当前调整项

                    DS1302_ShowAdjParameter(0, 2, 8,
CurrentTime.Minute, 0);
                    DS1302_ShowAdjParameter(0, 2, 12,
CurrentTime.Second, 0);
                }
                break;
                case 1:    //调整分
                {
                    if (KEY_MatrixValue <= 9)
                    {
                        x = x * 10 + KEY_MatrixValue;
                        if (x > 99)
                        {
                            x %= 100;
                        }
                        if (x > 59)
                            CurrentTime.Minute = 59;
                        else
                            CurrentTime.Minute = x;
                    }
                    DS1302_ShowAdjParameter(0, 2, 4, CurrentTime.Hour,
0);
                    DS1302_ShowAdjParameter(Time_500MsFlag, 2, 8,
CurrentTime.Minute, 0);
```

```
                DS1302_ShowAdjParameter(0, 2, 12,
CurrentTime.Second, 0);
            }
            break;
            case 2:    //调整秒
            {
                if (KEY_MatrixValue <= 9)
                {
                    x = x * 10 + KEY_MatrixValue;
                    if (x>99)
                    {
                        x %= 100;
                    }
                    if (x > 59)
                        CurrentTime.Second = 59;
                    else
                        CurrentTime.Second = x;

                }
            #ifdef __DS1302_DATE__
                LCD_ShowNumber(1, 0, 2);
                LCD_ShowNumber(1, 1, 0);    //年的 20 固定显示，
            #endif
                DS1302_ShowAdjParameter(0, 2, 4, CurrentTime.Hour,
0);
                DS1302_ShowAdjParameter(0, 2, 8,
```

```
CurrentTime.Minute, 0);
                        DS1302_ShowAdjParameter(Time_500MsFlag, 2, 12,
CurrentTime.Second, 0);
                    }
                    break;
                }
            }
            break;
        #ifdef __DS1302_DATE__
            case 2:    //星期
            {
                if (KEY_MatrixValue <= 9)
                {
                    x =   KEY_MatrixValue;
                    if (x > 7)
                        CurrentTime.Week = 7;
                    else if (x <= 0)
                        CurrentTime.Week = 1;
                    else
                        CurrentTime.Week = x;

                }
                LCD_ShowNumber(1, 0, 2);
                LCD_ShowNumber(1, 1, 0);     //年的 20 固定显示，
                DS1302_ShowAdjParameter(0, 1, 2, CurrentTime.Year, 0);
```

```
                    DS1302_ShowAdjParameter(0, 1, 6, CurrentTime.Month, 0); //
当调整选项光标发生变化时，刷新前面显示设置的内容
                    DS1302_ShowAdjParameter(0, 1, 10, CurrentTime.Day, 0);
                    DS1302_ShowAdjParameter(0, 2, 4, CurrentTime.Hour, 0);

                    DS1302_ShowAdjParameter(0, 2, 8, CurrentTime.Minute, 0);
                    DS1302_ShowAdjParameter(0, 2, 12, CurrentTime.Second, 0);
                    DS1302_ShowAdjParameter(Time_500MsFlag, 3, 2,
CurrentTime.Week, 1);
                }
                break;
            #endif
            }
        }
    }
}


extern void DS1302_TimeProcess(void)
{
    DS1302_ReadCurrentTime();
    LCD_ShowCurrentTime();
}

#ifdef __DS1302_INIT__
extern void DS1302_TimeInit(void)
```

```
{
    DS1302_TypeDef CurrentTime;
    CurrentTime.Year = 15;
    CurrentTime.Week = 7;
    CurrentTime.Month = 3;
    CurrentTime.Day = 17;
    CurrentTime.Hour = 23;
    CurrentTime.Minute = 59;
    CurrentTime.Second = 47;
    DS1302_WriteCurrentTime(&CurrentTime);
}
#endif   /* __DS1302_INIT__ */
```

五、LCD12864 液晶驱动程序

```
/*********************************************************************
* 文 件 名   : LCD12864.c
* 作      者   : 谷林海
* 日      期   : 10/15/2013
* 描      述   : LCD12864 液晶驱动程序
*********************************************************************/

/*****************************头文件包含 *************************/

#include "FONTLIBRARY.h"              /* 汉字库 */
#include "LCD12864.H"
```

```
static void LCD_WriteLeftByte(u8 WriteType, u8 Value)//写数据指令.左
{
    LCD_EN = 1;
    LCD_RW = 1;
    LCD_RS = (bit)(WriteType&0x01);
    LCD_RW = 0;
    LCD_CS2 = 1;
    LCD_CS1 = 0;      /*  使能左半屏  */
    LCD_DATAPORT = Value;
    LCD_EN = 0;
}

static void LCD_WriteRightByte(u8 WriteType, u8 Value)//写数据指令.右
{
    LCD_EN = 1;
    LCD_RW = 1;
    LCD_RS = (bit)(WriteType&0x01);
    LCD_RW = 0;
    LCD_CS2 = 0;
    LCD_CS1 = 1; /*  右半屏  */
    LCD_DATAPORT = Value;
    LCD_EN = 0;
}
extern void LCD_Init(void)
```

```
{
    /* 左半屏初始化操作 */
    LCD_WriteLeftByte(COMMAND, 0x3F);   /* 显示开 */
    LCD_WriteLeftByte(COMMAND, 0xC0);   /* 初始化 */
    LCD_WriteLeftByte(COMMAND, 0xB8);   /* 初始化页 */
    LCD_WriteLeftByte(COMMAND, 0x40);   /* 初始化行 */

    /* 右半屏初始化操作 */
    LCD_WriteRightByte(COMMAND, 0x3F);   /* 显示开 */
    LCD_WriteRightByte(COMMAND, 0xC0);    /* 初始化 */
    LCD_WriteRightByte(COMMAND, 0xB8);    /* 初始化页     */
    LCD_WriteRightByte(COMMAND, 0x40);   /* 初始化行 */
}
static void LCD_GetAddr(u8 * pAddr)
{
    switch(*pAddr)
    {
        case 0:  *pAddr = 0xB8;  break;     /* 1 行 */
        case 1:  *pAddr = 0xBA; break;
        case 2:  *pAddr = 0xBC; break;
        case 3:  *pAddr = 0xBE; break;     /* 4 行 */
        default :                   break;
    }
}
```

```
extern void LCD_ShowNumber(u8 yAddr, u8 xAddr, u8 Num)
{
    u8 i = 0;
    void (* pLCD_WriteByte)(u8, u8);        /*  指定函数  */
    switch(Num)    /*  判断是否为字符  */
    {
        case '-':  Num = 10;    break;
        case ':':  Num = 11;    break;
        case ' ':  Num = 12;    break;
        case '*':  Num = 13;    break;
        case '|':  Num = 14;    break;
        case '!':  Num = 15;    break;
        default:  break;
    }
    LCD_GetAddr(&yAddr);    /*  显示行物理地址  */
    if (xAddr < 8)    /*  左半屏  */
    {
        pLCD_WriteByte = LCD_WriteLeftByte;
    }
    else                    /*  右半屏  */
    {
        pLCD_WriteByte = LCD_WriteRightByte;
        xAddr -= 8;     /*  列地址抵消 8 位  */
    }
```

```
    (*pLCD_WriteByte)(COMMAND,  yAddr);     /*  写入屏显示行上部分基地址
*/
    (*pLCD_WriteByte)(COMMAND, 0x40+(xAddr<<3));
    for(i = Num<<4; i < ((Num<<4)+8); i++)       /*    扫描显示数据  */
    {
        (*pLCD_WriteByte)(DATA, ~TableNumber[i]);
    }

    (*pLCD_WriteByte)(COMMAND, yAddr+0x01);     /*  写入屏显行下部分基地
址  */
    (*pLCD_WriteByte)(COMMAND, 0x40+(xAddr<<3));
    for(i = Num<<4; i < ((Num<<4)+8); i++)
    {
        (*pLCD_WriteByte)(DATA, ~TableNumber[i+8]);
    }
}


extern void LCD_ShowChinese(u8 yAddr, u8 xAddr, u8 * pTable)
{
    u8 i = 0;
    void (* pLCD_WriteByte)(u8, u8);
    LCD_GetAddr(&yAddr);
    if (xAddr < 4)    /*  左半屏  */
    {
```

```
            pLCD_WriteByte = LCD_WriteLeftByte;
        }
        else                    /*  右半屏  */
        {
            pLCD_WriteByte = LCD_WriteRightByte;
            xAddr -= 4;
        }
        (*pLCD_WriteByte)(COMMAND, yAddr);
        (*pLCD_WriteByte)(COMMAND, 0x40+(xAddr<<4));
        for (i = 0; i < 16; i++)        /*  上半部分  */
        {
            (*pLCD_WriteByte)(DATA, ~pTable[i]);
        }
        (*pLCD_WriteByte)(COMMAND, yAddr+0x01);
        (*pLCD_WriteByte)(COMMAND, 0x40+(xAddr<<4));
        for (i = 0; i < 16; i++)        /*  下半部分  */
        {
            (*pLCD_WriteByte)(DATA, ~pTable[i+16]);
        }
    }
    extern void LCD_ClearScreen(u8 yLineStar0, u8 yLineStar1)
    {
        u8 i = 0;
        for (; yLineStar0 <= yLineStar1; yLineStar0++)
        {
```

```
            for (i = 0; i < 8; i++)
            {
                LCD_ShowChinese(yLineStar0, i, Table0);
            }
        }
    }
```

六、显示函数

```
/*****************************************************************************
* 文 件 名   : DISPLAY.c
* 作       者   : 第九小组
* 日       期   : 10/20/2013
* 描       述   : 显示函数
*****************************************************************************/
/* 头文件包含 ------------------------------------------------------------*/
#include "DISPLAY.h"
#include "LCD12864.h"
#include "DS1302.h"
#include "FontLibrary.h"
extern void LCD_ShowStandby(bit vType)
{
    LCD_ClearScreen(0, 3);     /* 清屏 */
    LCD_ShowChinese(0, 0, Table0);
    LCD_ShowChinese(0, 1, Table_Huan);
    LCD_ShowChinese(0, 2, Table_Ying);
    LCD_ShowChinese(0, 3, Table_Nin);
```

```
    LCD_ShowChinese(0, 4, Table_Hui);
    LCD_ShowChinese(0, 5, Table_Lai);   /* 欢迎你回来 */
    #ifdef __DS1302_DATE__
    LCD_ShowChinese(1, 2, Table_Nian);
    LCD_ShowChinese(1, 4, Table_Yue);
    LCD_ShowChinese(1, 6, Table_Ri);    /* 年月日 */
    #endif
    LCD_ShowChinese(2, 3, Table_Shi);
    LCD_ShowChinese(2, 5, Table_Fen);
    LCD_ShowChinese(2, 7, Table_Miao);  /* 时分秒 */
    if (vType)
    {
        LCD_ShowChinese(3, 0, Table_Kai);
        LCD_ShowChinese(3, 1, Table_Suo);   /* 开锁 */
    }
    else
    {
        LCD_ShowChinese(3, 0, Table_Cai);
        LCD_ShowChinese(3, 1, Table_Dan);   /* 菜单 */
    }
#ifdef __DS1302_DATE__
    LCD_ShowChinese(3, 5, Table_Xing);
    LCD_ShowChinese(3, 6, Table_Qi);      /* 星期 */
#endif
}
```

```
#ifdef __DS1302_DATE__

extern void LCD_ShowWeek(u8 yAddr, u8 xAddr,u8 Week)
{
    u8 * pTable = (void *)0;
    switch(Week)
    {
        case 1:   pTable = Table_Yi;    break;
        case 2:   pTable = Table_Er;    break;
        case 3:   pTable = Table_San;  break;
        case 4:   pTable = Table_Si;    break;
        case 5:   pTable = Table_Wu;   break;
        case 6:   pTable = Table_Liu;   break;
        case 7:   pTable = Table_Ri;    break;
    }
    LCD_ShowChinese(yAddr, xAddr, pTable);       //显示星期
}
#endif


extern void LCD_ShowCurrentTime(void)
{
#ifdef __DS1302_DATE__
    LCD_ShowNumber(1,0, 2);    //2
```

```
    LCD_ShowNumber(1,1, 0);      //0
    LCD_ShowNumber(1,2, CurrentTime.Year/10);
    LCD_ShowNumber(1,3, CurrentTime.Year%10);      //年
    LCD_ShowNumber(1,6, CurrentTime.Month/10);
    LCD_ShowNumber(1,7, CurrentTime.Month%10);    //月
    LCD_ShowNumber(1,10, CurrentTime.Day/10);
    LCD_ShowNumber(1,11, CurrentTime.Day%10);    //日
    LCD_ShowWeek(3, 7, CurrentTime.Week%10-1);   //  星期
#endif

    LCD_ShowNumber(2,4, CurrentTime.Hour/10);
    LCD_ShowNumber(2,5, CurrentTime.Hour%10);    //时
    LCD_ShowNumber(2,8, CurrentTime.Minute/10);
    LCD_ShowNumber(2,9, CurrentTime.Minute%10);   //分
    LCD_ShowNumber(2,12, CurrentTime.Second/10);
    LCD_ShowNumber(2,13, CurrentTime.Second%10); //秒
}
extern void LCD_ShowFirstMenu(bit ShowStatus, u8 ShowData)
{
    u8 i = 0;
    /*  以下为菜单内容显示  */
    if (!ShowStatus)
    {
        LCD_ClearScreen(0, 3);
        LCD_ShowChinese(0, 1, Table_Shi);      //时间调整
```

```
    LCD_ShowChinese(0, 2, Table_Jian);
    LCD_ShowChinese(0, 3, Table_Tiao);
    LCD_ShowChinese(0, 4, Table_Zheng);
    LCD_ShowChinese(1, 1, Table_Mi);     //密码修改
    LCD_ShowChinese(1, 2, Table_Ma);
    LCD_ShowChinese(1, 3, Table_Xiu);
    LCD_ShowChinese(1, 4, Table_Gai);
    LCD_ShowChinese(2, 1, Table_Guan);     //管理员权限
    LCD_ShowChinese(2, 2, Table_Li);
    LCD_ShowChinese(2, 3, Table_Yuan);
    LCD_ShowChinese(2, 4, Table_Quan);
    LCD_ShowChinese(2, 5, Table_Xian);
}
/*  以下为显示‘*’  */
for (i = 0; i < 4; i++)
{
    if (i == ShowData)
    {
        LCD_ShowNumber(ShowData, 0, '*');    //    '*'
    }
    else
    {
        LCD_ShowNumber(i, 0, ' ');    //' '
    }
}
```

```
}
/**********************文件结束*****************/
```

七、对 AT24C02 功能的驱动程序

```
/*************************************************************************
*  文 件 名  : AT24C02.h
*  作      者  : 第九组成员
*  日      期  : 30/10/2013
*  描      述  : 对 AT24C02 功能的驱动程序
*************************************************************************/
#include "AT24C02.h"
unsigned char iic_send_byte(unsigned char wdata)
{
    unsigned char bit_cnt;
    for(bit_cnt = 0;bit_cnt < 8;bit_cnt ++)
    {
        if(wdata & 0x80) SDA = 1;
        else SDA = 0;
        wdata <<= 1;

        _delay_10uS();
        SCL = 1;
        _delay_10uS();
        SCL = 0;
    }
```

```
    _delay_10uS();

    SDA = 1;
    SCL = 1;
    _delay_10uS();
    if(SDA == 1) bit_cnt = 0;//返回 0,失败的写入
    else bit_cnt = 1;//返回 1,成功的写入
    SCL = 0;
    return bit_cnt;
}

unsigned char iic_recv_byte(void)
{
    unsigned char bit_cnt;
    unsigned char rdata;

    SDA = 1;
    for(bit_cnt = 0;bit_cnt < 8;bit_cnt ++)
    {
        SCL = 0;
        _delay_10uS();
        SCL = 1;
        _delay_10uS();
        rdata <<= 1;
        if(SDA == 1) rdata |= 1;
```

```
        _delay_10uS();
    }
    SCL = 0;
    return rdata;
}

unsigned char iic_page_write(unsigned char sla,\
    unsigned char suba,unsigned char *s,unsigned char len)
{
    unsigned char i;

    iic_start();//启动总线
    if(iic_send_byte(sla) == 0) return 0xff;//发送器件地址,失败返回 0xff
    if(iic_send_byte(suba) == 0) return 0xff;//发送数据地址，失败返回 0xff

    for(i = 0;i < len;)
    {
        if(iic_send_byte(*s ++) == 0) return 0xff;
        i ++;

        suba ++;
        if((suba & 0x07) == 0) break;//页越界，跳出
    }
    iic_stop();//结束总线
    return(len - i);//写入 i 字节成功
```

```
}

unsigned char iic_send_str(unsigned char sla,\
    unsigned char suba,unsigned char *s,unsigned char len)
{
    unsigned char rm_cnt = len;
    unsigned int i;

    do
    {
        rm_cnt = iic_page_write(sla,suba + (len - rm_cnt),s + (len - rm_cnt),rm_cnt);
        if(rm_cnt == 0xff) return 0;
        for(i = 0;i < 1000;i ++) _delay_10uS();//10mS 延时
    }
    while(rm_cnt);
    return 1;//写入多字节成功
}

unsigned char iic_recv_str(unsigned char sla,\
    unsigned char suba,unsigned char *s,unsigned char len)
{
    unsigned char i;

    iic_start();//启动总线
    if(iic_send_byte(sla) == 0) return 0;//发送器件地址,失败返回 0
```

```
    if(iic_send_byte(suba) == 0) return 0;//发送数据地址，失败返回 0


    iic_start();//重新启动总线
    if(iic_send_byte(sla | 1) == 0) return 0;//发送器件地址,失败返回 0
    for(i = 0;i < len - 1;i ++)
    {
        *s ++ = iic_recv_byte();//接收一字节数据
        iic_ack(0);//发送有效应答位
    }
    *s = iic_recv_byte();//接收最后一字节数据
    iic_ack(1);//发送非应答位
    iic_stop();
    return(1);
}
```

No. 6

电子密码锁
调试过程故障分析报告

题目：电子密码锁的设计

小组成员：

谷林海

顾友峰

兰顺福

葛振涛

目　录

项目基本信息

项目名称	多功能电子密码锁
开发软件	Keil 、Proteus、Protel
所用语言	C 语言
项目测试时间	8 周
测试参与人员分工	程序编写　　谷林海
	编译仿真　　顾友峰　兰顺福
	文档编写　　顾友峰　葛振涛

第一章 引言

一、编写目的

本报告目的在于总结测试阶段的测试以及分析测试结果，描述系统是否符合需求功能目标。对于测试出来的故障进行合理的分析，及时修正错误。具体体现在以下几个方面：

1.分析测试的过程，产品，资源，信息，为以后制定测试计划提供参考；

2.评估测试结果与功能要求是否一致；

3.分析系统存在的缺陷，为修复和预防提供建议和指导思想；

4.通过对测试结果的分析，得到对该多功能电子密码锁质量的评价。

二、项目背景

随着科学技术的不断发展，人们对日常生活中的安全保险器件的要求越来越高。为满足人们对锁的使用要求，增加其安全性，用密码代替钥匙的密码锁应运而生。在安全技术防范领域，具有防盗报警功能的电子密码锁逐渐代替传统的机械式密码锁，克服了机械式密码锁密码量少、安全性能差的缺点，使密码锁无论在技术上还是在性能上都大大提高一步。

并且伴随着大规模集成电路技术的发展，特别是单片机的问世，出现了带微处理器的智能密码锁，它除具有电子密码锁的功能外，还引入了智能化管理、专家分析系统等功能，从而使密码锁具有很高的安全性、可靠性，应用日益广泛。

三、用户群

主要读者：电子密码锁项目设计开发人员。

其他读者：密码锁项目相关人员

四、调试对象

多功能电子密码锁系统程序

五、调试工具

Keil 编译软件和 Proteus 仿真软件

第二章 调试

一、调试过程

1、主要故障

(1)Keil 与 Proteus 两个独立的软件，每次测试必须链接；

(2)独立按键与矩阵键盘的差别，识别矩阵键盘的按键；

(3)键盘的复用；

(4)while(1)循环嵌套 while 如何处理(1)；

(5)按键去抖动延时时间不够长；

(6)输完密码后，扬声器长鸣不止；

(7)只有第一次输入正确的密码才能开锁。

2、分析

(1)Keil 与 Proteus 的链接：

一定要把 Keil 的工程和 Proteus 的文件放到同一个目录下(这里所说的 Keil 的工程指工程的目录,即 Proteus 的工程 Design 文件（后缀名.DSN）要和包含了 Keil 工程文件的那个文件夹在同一层目录下)。

(2)独立按键与矩阵键盘的差别，识别矩阵键盘的按键；

独立按键：一个按键占用单独的一个 I/O 口；

矩阵键盘：为了节省 I/O 口，通常将按键排列成矩阵形式，每条水平线和垂直线在交叉处不直接连通，而是通过一个按键加以连接。

4×4 矩阵键盘识别处理,每个按键有它的行值和列值,行值和列值的组合就是识别这个按键的编码。矩阵的行线和列线分别通过两并行接口和 CPU 通信。每个按键的状态同样需变成数字量“0”和“1”，开关的一端（列线）通过电阻接 VCC，而接地是通过程序输出数字“0”实现的。键盘处理程序的任务是：确定有无键按下，判断哪一个键按下，键的功能是什么；还要消除按键在闭合或断开时的抖动。两个并行口中，一个输出扫描码，使按键逐行动态接地，另一个并行口输入按键状态，由行扫描值和回馈信号共同形成键编码而识别按键，通过软件查表，查出该键的功能。

(3)键盘的复用可以有效的节约资源，提高键盘的利用率。单键盘在显示密码锁的状态下的作用于在显示时间上的区别。

(4)while(1)是死循环，一般用 break；退出，项目中的时钟显示就是这个问题，因为需要时刻记录变换时间。但是如何退出呢，就需要在 while(1)中加退出的条件。

(5)按键去抖动延时时间不够长

在按键操作时发现按键一次却产生了 2~3 次的键值输入，显然按键的去抖动延时时间不够长，源程序的去抖动延时时间占 5084 个指令周期，晶体的频率值为 12MHz，经计算延时时间只有 5ms 左右，由于延时过短，除非以很快的速度按键，否则因按键抖动现象的出现，会导致可靠性降低。将源程序的去抖动延时时间改为 50ms 即可有效消除抖动现象。

(6)输完密码后，扬声器长鸣不止

原程序在判断密码正误后，都会循环调用提示音子程序发出提示音，以提示密码的对与错，而这种循环都是减数循环，即对某一寄存器置初值，在循环时不断地使该寄存器的值递减，当寄存器的值等于 0 的时候就跳出循环，但原程序却把置初值语句放在循环体之内，使这个减数循环成了死循环，从而导致输入密码后扬声器长鸣不止，除非单片机断电，将置初值语句放到减数循环之外，上述故障消失。

二、原程序的优化

原程序没有对用户的输入进行监视和限时，例如有人输入了 4 位密码后有事离开了，程序就一直处于等待状态，而下一个人输入密码时就会造成密码输到一半就报警提示密码错误。

从提高程序的抗干扰，提高程序结果的正确性，监视用户操作方面对原程序进行优化。

1、在抗干扰能力方面

原程序监测到按键按下后延时，延时后却没有重新检测按键是否维持按下状态，这样会造成如果按键的引脚受到干扰而出现低电平时（按键按下后，引脚由高电平变为低电平），程序在延时过后会继续执行下去的现象。应改为延时过后重新检测按键的动作情况。

2、监视用户输入

在程序加入了用户输入监视系统，对用户输入进行监视，当程序等待用户输入时，监视程序会自动启动定时器 T0，当在 30s 内用户无输入时，使程序自动返回起始处，等待下一个用户输入密码。

No. 7

电子密码锁测试报告

题目：电子密码锁的设计

小组成员：

谷林海

顾友峰

兰顺福

葛振涛

目　录

项目基本信息

项目名称	多功能电子密码锁
开发软件	Keil 、Proteus、Protel
所用语言	C 语言
项目测试时间	8 周
测试参与人员分工	程序编写　　谷林海
	编译仿真　　顾友峰　兰顺福
	文档编写　　顾友峰　葛振涛

第一章 引言

一、编写目的

本报告目的在于总结测试阶段的测试以及分析测试结果，描述系统是否符合需求功能目标。对于测试出来的故障进行合理的分析，及时修正错误。具体体现在以下几个方面：

1.分析测试的过程，产品，资源，信息，为以后制定测试计划提供参考；

2.评估测试结果与功能要求是否一致；

3.分析系统存在的缺陷，为修复和预防提供建议和指导思想；

4.通过对测试结果的分析，得到对该多功能电子密码锁质量的评价。

二、项目背景

随着科学技术的不断发展，人们对日常生活中的安全保险器件的要求越来越高。为满足人们对锁的使用要求，增加其安全性，用密码代替钥匙的密码锁应运而生。在安全技术防范领域，具有防盗报警功能的电子密码锁逐渐代替传统的机械式密码锁，克服了机械式密码锁密码量少、安全性能差的缺点，使密码锁无论在技术上还是在性能上都大大提高一步。

并且伴随着大规模集成电路技术的发展，特别是单片机的问世，出现了带微处理器的智能密码锁，它除具有电子密码锁的功能外，还引入了智能化管理、专家分析系统等功能，从而使密码锁具有很高的安全性、可靠性，应用日益广泛。

三、用户群

主要读者：电子密码锁项目设计开发人员。

其他读者：密码锁项目相关人员

四、测试对象

多功能电子密码锁系统程序

五、测试工具

Keil 编译软件和 Proteus 仿真软件

第二章 测试

一、测试工具

Keil C51 是美国 Keil Software 公司出品的 51 系列兼容单片机 C 语言软件开发系统，与汇编相比，C 语言在功能上、结构性、可读性、可维护性上有明显的优势，因而易学易用。Keil 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）将这些部分组合在一起。运行 Keil 软件需要 WIN98、NT、WIN2000、WINXP 等操作系统。如果你使用 C 语言编程，那么 Keil 几乎就是你的不二之选，即使不使用 C 语言而仅用汇编语言编程，其方便易用的集成环境、强大的软件仿真调试工具也会令你事半功倍。

C51 工具包的整体结构，uVision 与 Ishell 分别是 C51 for Windows 和 for Dos 的集成开发环境(IDE)，可以完成编辑、编译、连接、调试、仿真等整个开发流程。开发人员可用 IDE 本身或其它编辑器编辑 C 或汇编源文件。然后分别由 C51 及 C51 编译器编译生成目标文件（.OBJ）。目标文件可由 LIB51 创建生成库文件，也可以与库文件一起经 L51 连接定位生成绝对目标文件(.ABS）。ABS 文件由 OH51 转换成标准的 Hex 文件，以供调试器 dScope51 或 tScope51 使用进行源代码级调试，也可由仿真器使用直接对目标板进行调试，也可以直接写入程序存贮器如 EPROM 中。

二、测试方案

1、通过键盘输入 6 位初始密码，若密码正确，则将锁打开。若密码不正确，则重新输入，若三次否没有输入正确，则键盘自动锁定。

2、验证是否具有开锁、超时报警、超次锁定、管理员解密、设置密码、修改用户密码基本的密码锁的功能以及是否可以对报警时间、超时时间、输入次数、解锁时间进行设置。

3、验证密码可以由用户自己修改设定，锁打开后才能修改密码。修改密码之前必须再次输入密码，在输入新密码时候需要二次确认，以防止误操作。

4、测试密码的输入、清除、更改功能：（a）密码输入功能：按下一个数字

键，一个“*”就显示在最右边的数码管上，同时将先前输入的所有“*”向左移动一位。（b）密码更改功能：将输入的值作为新的密码。

5、测试密码锁的开锁功能：当按下开锁键，系统将输入与密码进行检查核对，如果正确锁打开，否则不打开，并发出报警提示。如果键入完密码后不按确认键系统会当做放弃开锁处理。

6、系统能否设置超时时间与自锁时间；

7、系统是否具有掉电存储、光提示等功能。

8、系统是否具有显示时钟的功能，并且可以修改时间。

9、验证系统的各个操作，不管成功、失败都是否有指示作为交互。

三、测试方法

为了更好的对系统进行测试，我们可以采取以下两种方法：

(1) (1) 按照测试方案上的要求，逐项测试各个功能是否实现

(2) 不按照顺序，随机测试某项功能，看系统是否运作正常

按照这种结合的测试方法进行测试，可以更详细更精确的得到系统的具体性能参数，并可以相互对比，指导进一步的修改和测试方案、。

四、测试结果

功能测试结果概要

功能	基本要求	测试情况	测试通过	
			是	否
密码登录	输入正确的用户名和密码可以登录。输入错误的用户名和密码系统给出明确提示	8 次	是	
密码修改成功	密码修改后可以开锁	5 次	是	
输入次数	输入次数必须小于 3	12 次	是	
时间	能准确显示系统时间	2 次	是	
显示屏	能显示所有操作内容	3 次	是	

最后阶段可以不按照顺序，随机测试某项功能，看系统是否运作正常。

五、测试分析

通过对系统的详尽测试，可以知道密码锁系统基本上实现了本次设计的主要功能。诸如开锁，重置密码，管理员解密，超次锁定报警的功能。不仅如此本系统还实现了掉电保护、灯光提示等功能。当然了本系统也有不足之处，例如界面不美观、交互性不太好、也没有实现串行通信和日志记录等功能。很多功能也并不完善，这些都是我们可以该进的地方

第三章 测试总结

通过本实验课的学习，让我们复习了很多学过的知识并学到了许多新知识，同时锻炼了自己的动手和查阅资料的能力。尤其是提高了在实际中解决问题的能力。

(1) 通过这次的设计，我们熟悉了使用 Protel99SE 和 Proteus，学会了软件仿真和制作 PCB 板图的一些技巧。

(2) 在设计中，我学到了如何使用 C 语言对单片机进行编写程序，熟悉了使用 Keil C 软件，并且加深了对单片机的编程技巧。

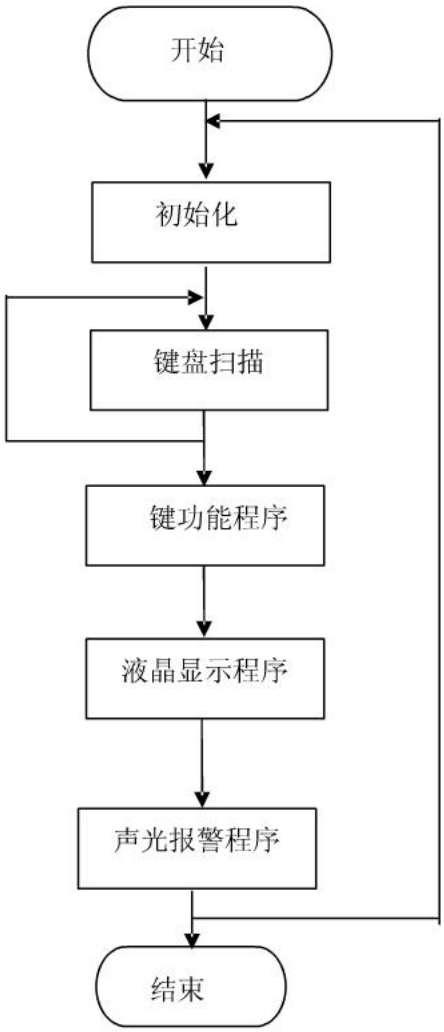
(3) 在做这个设计的过程中，遇到了不少的问题。这锻炼了我们筛选、查阅资料，并将理论结合到自己的设计中的能力。一步一步的排除故障原因，找到故障的原因并解决故障。

(4) 这次的设计使我对模拟电子、数字电子知识加深了了解，尤其是在对电路进行调试的时候，出现了很多这些方面的问题，通过对以往知识的复习巩固和查阅资料，将问题解决。

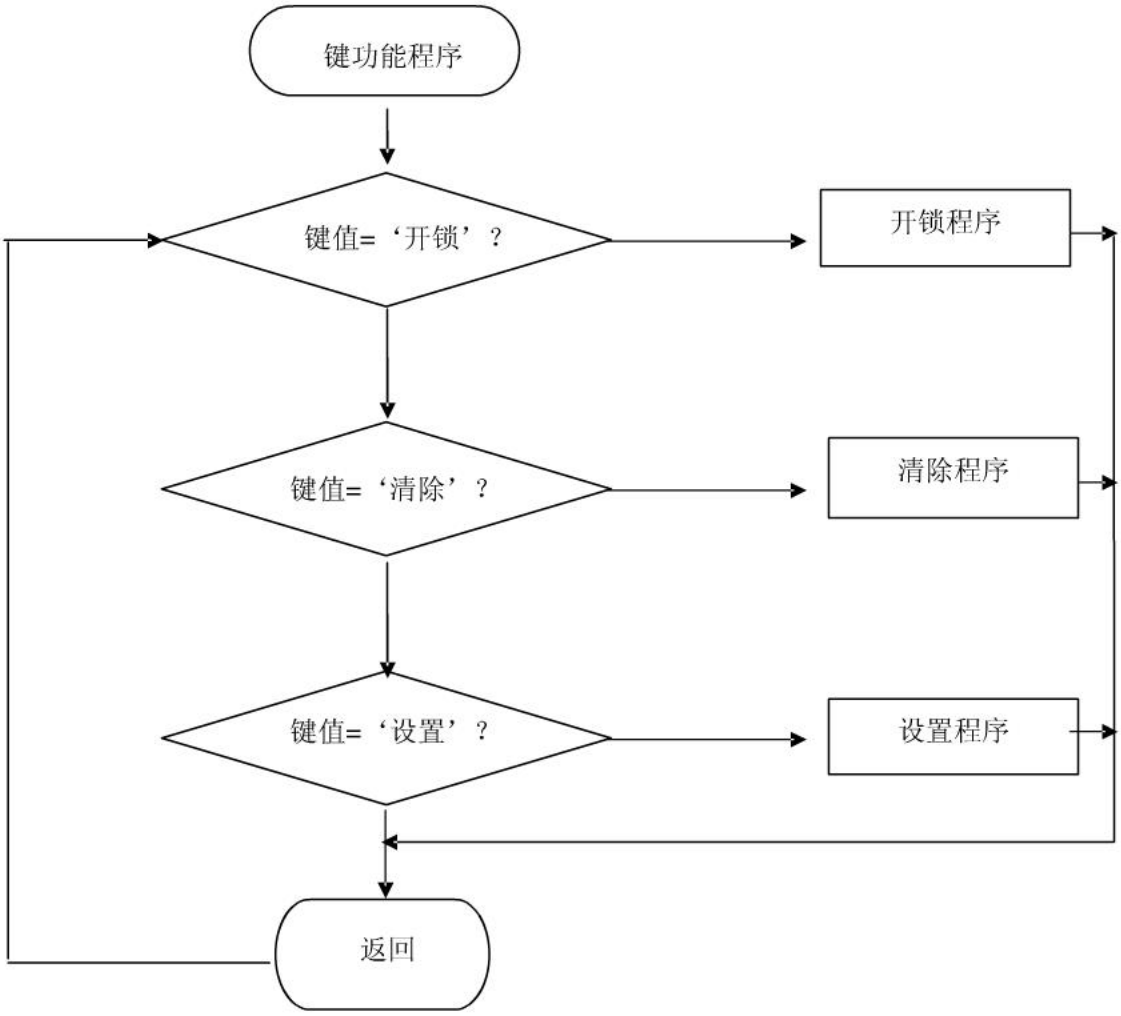
附件1

电子密码锁C程序流程图

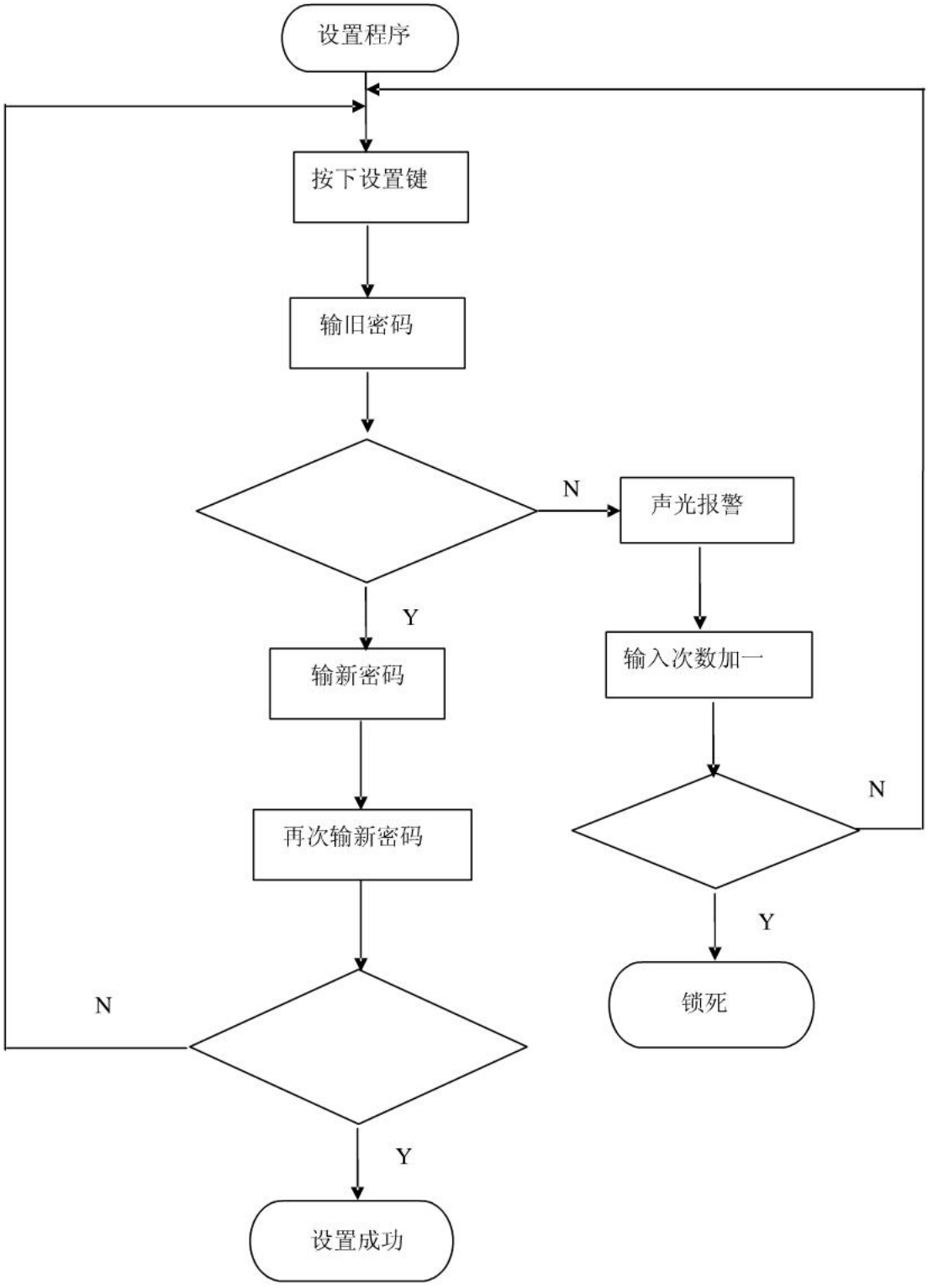
主程序流程图

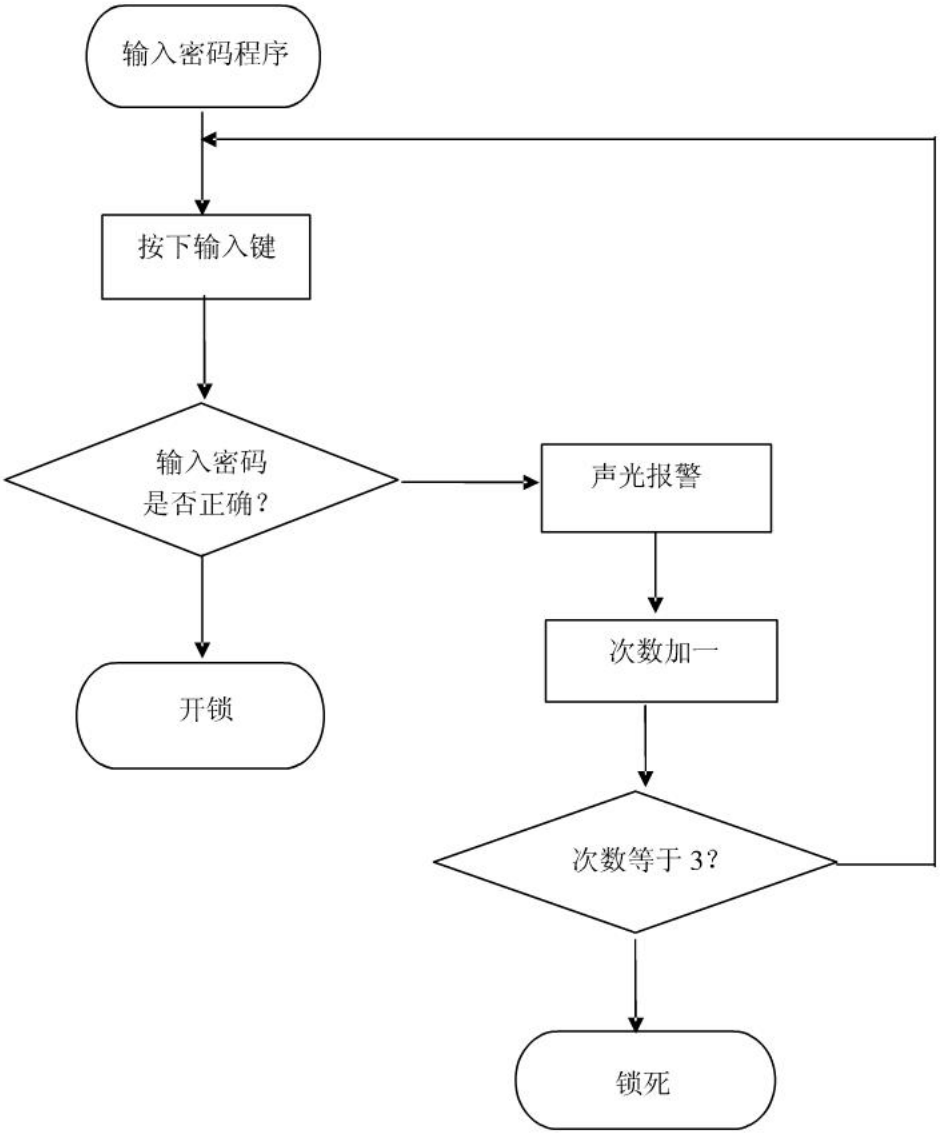


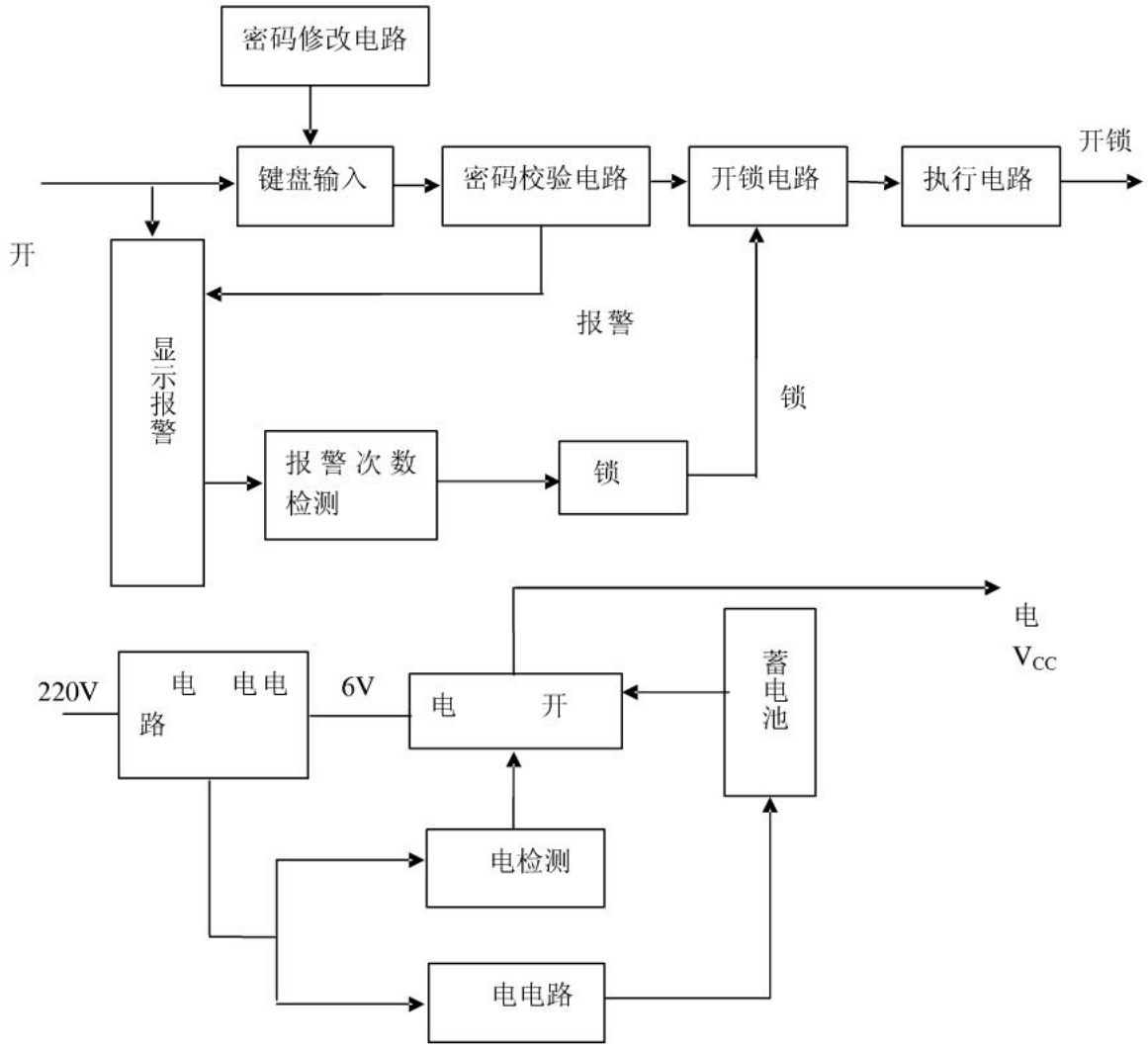
键功能流程图



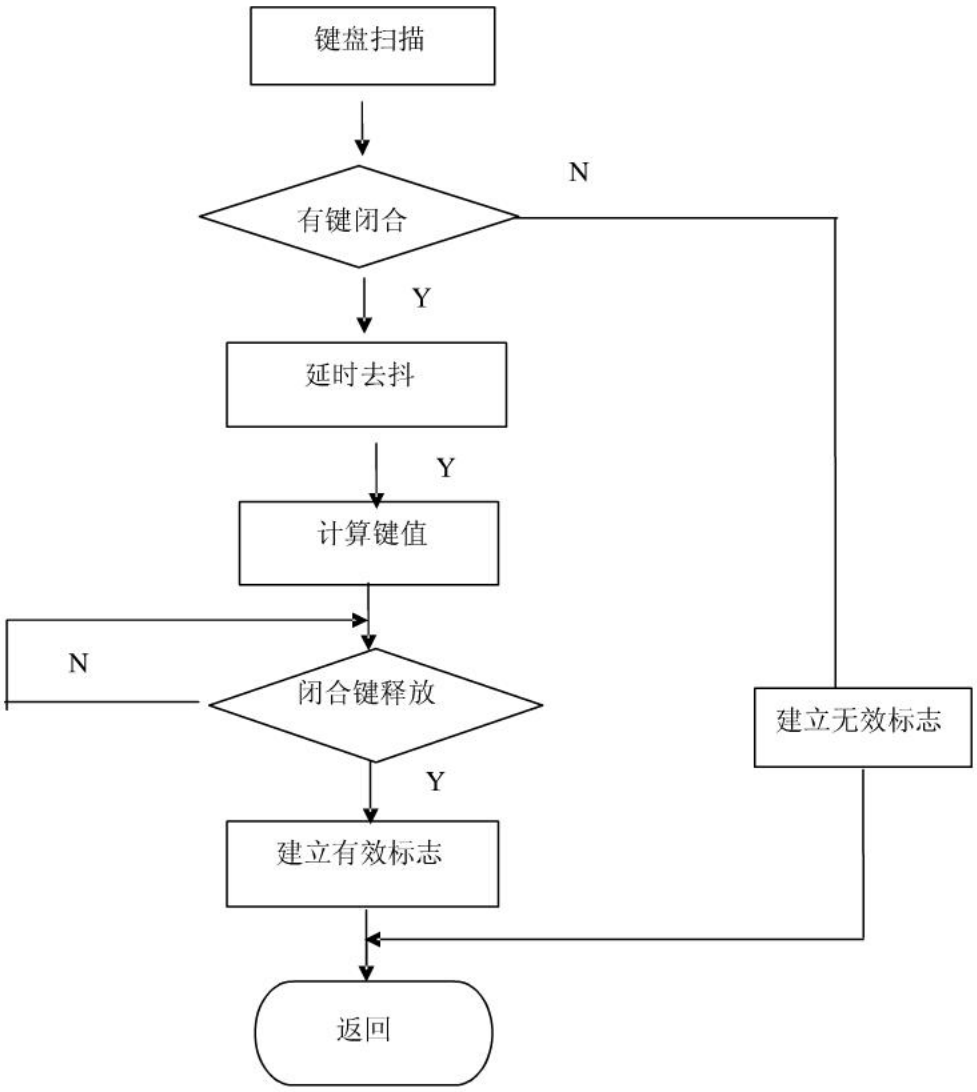
密码设置流程图




输入密码程序
按下输入键
输入密码
是否正确?
声光报警
次数加一
开锁
次数等于 3?
锁死


密码修改电路
键盘输入
密码校验电路
开锁电路
执行电路
开锁
开
显示报警
报警
锁
报警次数检测
锁
电
V_{CC}
220V
电 电电路
6V
电 开
蓄电池
电检测
电电路

键盘扫描流程图



附件 2

PCB 等文件清单

★ 电子密码锁.PCB

★ 电子密码锁.Sch

★ 电子密码锁.DSN

★ 电子密码锁程序清单

☆ 本项目接收实战学员，掌握本项目的完整技术，培养实际动手开发能力。随报随学，全程手把手指导。

☆ 项目完成通过验证合格后，提供参与项目研发合格证明。

☆ 如有需求，请联系：ilynchpin@lynchpin.com.cn