**2016届毕业生**

**毕业论文**

**题 目: 超市储物柜**

**院系名称：**

**专业班级：**

**学生姓名： 学 号：**

**指导教师： 教师职称：**

**20 年 月 日**

**摘 要**

本设计采用美国Atmel公司的51单片机与无字库的LCD12864显示模块构成，利用C语言编程，通过软件控制。对于外部按键电路得到的数据，通过主控芯片进行读取、处理、传送，使得数据得以显示。

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了传统的机械式密码锁，电子密码锁具有安全性高、成本低、功耗低、易操作等优点。本文从经济实用的角度出发，用C语言编写的主控芯片控制程序与AT24C02读写程序相结合，并用Keil软件进行编译，设计了一款可以设置密码的超市储物柜。

关键字：储物柜 密码锁

**ABSTRACT**

This design adopts the 51 single-chip microcomputer and the untyped LCD12864 display module of Atmel company in the United States, which is programmed by C language and controlled by software.The data obtained from the external key circuit is read, processed and transmitted through the main control chip, enabling the data to be displayed.

In the field of security technology to guard against, with security alarm function of electronic combination lock gradually replace the traditional mechanical combination lock, electronic combination lock with high safety, low cost, low power consumption, easy operation, etc.This article from the perspective of economic and practical, and the main control chip control program written in C language to read and write AT24C02 procedures, the combination of using Keil software compiling, designed a supermarket store content ark can set the password.

**Key words:** locker lock.

目录

**[一、设计的背景目的及意义 5](#_Toc22588)**

[1.1、设计的背景 5](#_Toc8804)

[1.2、设计的意义 5](#_Toc19918)

[1.3、设计的目的 5](#_Toc28964)

**[二、 系统主要器件的选择 7](#_Toc22225)**

[2.1、主控器件 7](#_Toc13675)

[2.2、12864液晶 13](#_Toc11947)

**[三、 硬件电路设计及其原理 19](#_Toc30702)**

[3.1、最小系统的原理图 19](#_Toc24719)

[3.2、报警电路的设计 22](#_Toc28447)

[3.3、24c02原理图 22](#_Toc26887)

[3.4、12864原理图 23](#_Toc19896)

[3.6、箱号指示灯原理图 24](#_Toc7892)

**[四、 程序设计 25](#_Toc2723)**

**[参考文献 27](#_Toc21440)**

**[致 谢 28](#_Toc5368)**

**[附录 29](#_Toc29970)**

**[原理图： 29](#_Toc15665)**

**[源程序： 30](#_Toc28799)**

# 一、设计的背景目的及意义

## 1.1、设计的背景

随着人们的生活水平的不断提高，超市越来越成为人们购物地方的首选，传统的机械锁，由于其构造的简单，安全性能低，开锁麻烦，被撬的事件屡见不鲜。在科学技术不断发展的今天，电子密码锁作为临时储存的作用显得日趋重要。电子密码锁是由电子电路控制锁体的新型的锁具，本设计采用51单片机作为核心元件，设计了一款具有键盘输入、液晶显示、开锁、报警、设置密码等功能的电子密码锁储物柜。即简单又适用。

## 1.2、设计的意义

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，所以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。目前使用的电子密码锁大部分是基于单片机技术，以单片机为主要器件，其编码器与解码器的生成为软件方式。相比传统的机械式钥匙开锁携带不方便、安全性能差等特点，电子密码锁有易操作、功耗低等优势，使其越来越成为市场上的主流产品。

## 1.3、设计的目的

为了对单片机有更深刻的了解，同时也在电子方面有更深刻的认识，通过学习，选择做一个通过矩阵按键控制单片机的超市储物柜。核心部件单片机STC89C54RD+，它在整个人类史上的地位已经不容置疑地确立了，相信它会越来越深入地浸透到人们的生活中，并且将在一定程度上影响人们对生活的理解和诠释。储物柜被越来越多的超市所使用，密码锁可靠性极高，使用方便，将两者结合将更便于消费者的使用。用单片机制作电子产品也会越来越多，也是众多领域实施编程开发不可缺少的一部分，这将必然成为一种趋势。为更多的了解单片机的原理和应用，故做此设计来进行巩固。

本设计以MCS-51系列单片机为核心设计的超市储物柜系统，并通过密码锁控制储物箱的开关。对于不同型号的单片机只需要相应的改变一下地址即可。

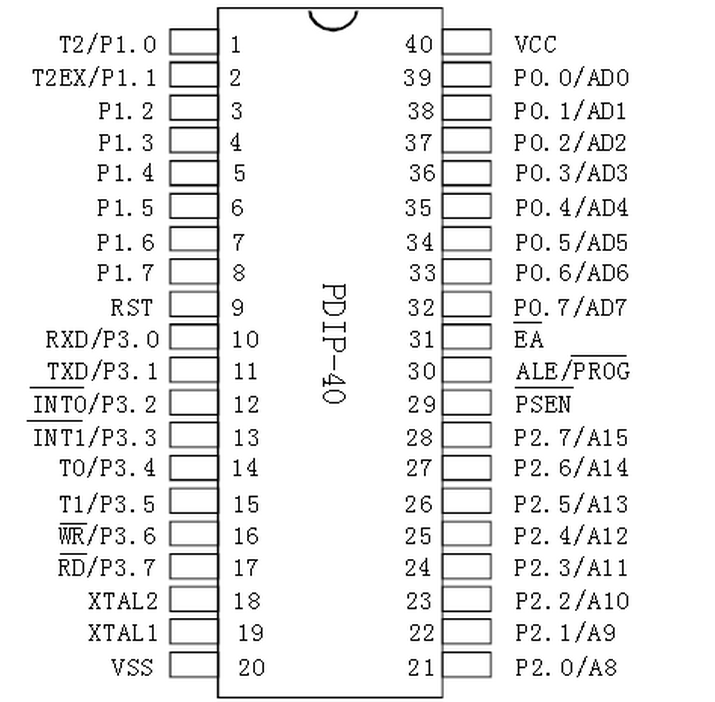
# 系统主要器件的选择

系统主要有单片机最小系统和24C02串行EEPROM组成。

## 2.1、主控器件

本设计以STC89C54RD+单片机为控制核心。

单片机是STC公司最新推出的一种新型51内核的单片机。片内含有Flash程序存储器、SRAM、UART、SPI、PWM等模块。封装图如下图：



### 2.1.1、主要特性

1. 增强型8051单片机，6时钟/机器周期和12时钟/机器周期可以任意选择，指令代码完全兼容传统8051.

2. 工作电压：5.5V～3.3V（5V单片机）/3.8V～2.0V（3V单片机）

3. 工作频率范围：0～40MHz，相当于普通8051的0～80MHz，实际工作频率可达48MHz

4. 用户应用程序空间为8K字节 5. 片上集成512字节RAM

6. 通用I/O口（32个），复位后为：P1/P2/P3/P4是准双向口/弱上拉，P0口是漏极开路输出，作为总线扩展用时，不用加上拉电阻，作为I/O口用时，需加上拉电阻。

1. ISP（在系统可编程）/IAP（在应用可编程），无需专用编程器，无需专用仿真器，可通过串口（RxD/P3.0,TxD/P3.1）直接下载用户程序，数秒即可完成一片
2. 具有EEPROM功能
3. 具有看门狗功能

10. 共3个16位定时器/计数器。即定时器T0、T1、T2

11. 外部中断4路，下降沿中断或低电平触发电路，Power Down模式可由外部中断低电平触发中断方式唤醒

12. 通用异步串行口（UART），还可用定时器软件实现多个UART

13. 工作温度范围：-40～+85℃（工业级）/0～75℃（商业级）

### 2.1.2、引脚功能

VCC（40引脚）：电源电压

VSS（20引脚）：接地

P0端口（P0.0～P0.7，39～32引脚）：P0口是一个漏极开路的8位双向I/O口。作为输出端口，每个引脚能驱动8个TTL负载，对端口P0写入“1”时，可以作为高阻抗输入。在访问外部程序和数据存储器时，P0口也可以提供低8位地址和8位数据的复用总线。此时，P0口内部上拉电阻有效。在Flash ROM编程时，P0端口接收指令字节；而在校验程序时，则输出指令字节。验证时要求外接上拉电阻。

P1端口（P1.0～P1.7，1～8引脚）：P1口是一个带内部上拉电阻的8位双向I/O口。P1的输出缓冲器可驱动（吸收或者输出电流方式）4个TTL输入。对端口写入1时，通过内部的上拉电阻把端口拉到高电位，这是可用作输入口。P1口作输入口使用时，因为有内部上拉电阻，那些被外部拉低的引脚会输出一个电流。

P2端口（P2.0～P2.7，21～28引脚）：P2口是一个带内部上拉电阻的8位双向I/O端口。P2的输出缓冲器可以驱动（吸收或输出电流方式）4个TTL输入。对端口写入1时，通过内部的上拉电阻把端口拉到高电平，这时可用作输入口。P2作为输入口使用时，因为有内部的上拉电阻，那些被外部信号拉低的引脚会输出一个电流

P3口引脚复用功能 引脚号 复用功能

P3.0 RXD（串行输入口）

P3.1 TXD（串行输出口）

P3.2 （外部中断0）

P3.3 （外部中断1）

P3.4 T0（定时器0的外部输入）

P3.5 T1（定时器1的外部输入）

P3.6 （外部数据存储器写选通）

P3.7 （外部数据存储器读选通）

RST（9引脚）：复位输入。当输入连续两个机器周期以上高电平时为有效，用来完成单片机单片机的复位初始化操作。看门狗计时完成后，RST引脚输出96个晶振周期的高电平。特殊寄存器AUXR（地址8EH）上的DISRTO位可以使此功能无效。DISRTO默认状态下，复位高电平有效。

ALE/（30引脚）：地址锁存控制信号（ALE）是访问外部程序存储器时，锁存低8位地址的输出脉冲。在Flash编程时，此引脚（）也用作编程输入脉冲。 在一般情况下，ALE以晶振六分之一的固定频率输出脉冲，可用来作为外部定时器或时钟使用。然而，特别强调，在每次访问外部数据存储器时，ALE脉冲将会跳过。如果需要，通过将地址位8EH的SFR的第0位置“1”，ALE操作将无效。这一位置“1”，ALE仅在执行MOVX或MOV指令时有效。否则，ALE将被微弱拉高。这个ALE使能标志位（地址位8EH的SFR的第0位）的设置对微控制器处于外部执行模式下无效。

VPP（31引脚）：访问外部程序存储器控制信号。为使能从0000H到FFFFH的外部程序存储器读取指令，必须接GND。注意加密方式1时，将内部锁定位RESET。为了执行内部程序指令，应该接VCC。在Flash编程期间，也接收12伏VPP电压。

XTAL1（19引脚）：振荡器反相放大器和内部时钟发生电路的输入端。

XTAL2（18引脚）：振荡器反相放大器的输入端。

### 2.1.3、串口通信

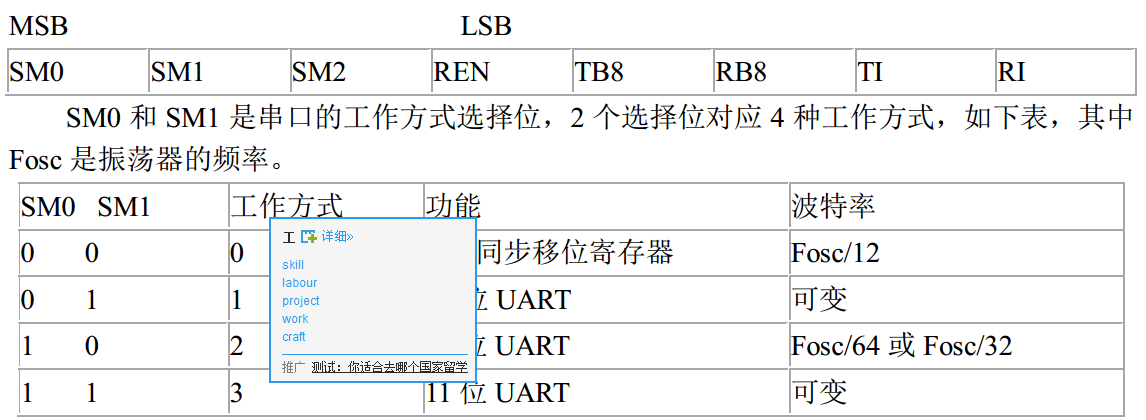
一条信息的各位数据被逐位顺序传送的通信方式成为串行通信。根据信息的传送方向，串行通信可以可以进一步划分为单工、半双工和全双工 3 种。信息只能单方向传送为单工；信息能双向传送但不能同时双向传送为半双工；信息能够同时双向传送则成为全双工。 8051 系列单片机有一个全双工串行口， 全双工的串行通信只需要一根输出线和输入线。

串行通信又有异步通信和同步通信这两种方式。异步通信用起始位“0”表示字符的开始，然后从低位到高位逐位传送数据，最后用停止位“1”表示字符结束。一个字符又称作一帧信息，一帧信息包括 1 位起始位、8 位数据位、1 位停止位，若数据位增加到第 9 位， 在 8051 系列单片机中， 第九位数据可以用作奇偶校验位， 也可以用作地址/数据帧标志。

8051 系列单片机串行 I/O 接口的工作原理就是： 当要发送数据时， 单片机自动将 SBUF 内的 8 位并行数据转换为一定格式的串行数据， 从 TXD 引脚按规定的波特率来输出； 当要接收数据时， 要监视 RXD 引脚，一旦出现起始位“0” ，按规定的波特率将外围设备送来的一定格式的串行数据转换成 8 位并行数据，等待用户读取 SBUF 寄存器，若不及时读取，SBUF 中的数据有可能被刷新。8051 系列单片机上有通用异步接收/发送器用于串行通信，发送时数据由 TXD 引脚输出，接收时数据从 RXD 引脚输入。有两个缓冲器（Serial Buffer） ，一个作发送缓冲器，另外一个作为接收缓冲器。UART 是可编程的全双工的串行口。

SBUF是可以直接寻址的专用寄存器。物理上，它对应着两个寄存器，即一个发送寄存器一个接收寄存器，CPU写SBUF就是修改发送寄存器；读SBUF就是读接收寄存器。接收器是双缓冲的，以避免在接收下一帧数据之前，CPU未能及时的响应接收器的中断，没有把上一帧的数据读走而产生两帧数据重叠的问题。对于发送器，为了保持最大的传输速率，一般不需要双缓冲，因为发送时CPU是主动的，不会产生重叠问题。

SCON是一个逐位定义的8位寄存器，用于控制串行通信的方式选择、接收和发送，指示串口的状态，SCON即可以字节寻址也可以位寻址，字节地址98H，地址位为98H~9FH。它的各个位定义如下：



SM2在工作方式2和3中是多机通信的使能位。在工作方式0中，SM2必须为0。在工作方式1中，若SM2=1且没有接收到有效的停止位，则接收中断标志位RI不会被激活。在工作方式2和3中若SM2=1且接收到的第9位数据（RB8）为0，则接收中断标志RB8不会被激活，若接收到的第9位数据（RB8）为1，则RI置位。此功能可用于多处理机通信。

REN为允许串行接收位，由软件置位或清除。置位时允许串行接收，清除时禁止串行接收。

TB8是工作方式2和3要发送的第9位数据。在许多通信协议中该位是奇偶位，可以按需要由软件置位或清除。在多处理机通信中，该位用于表示是地址帧还是数据帧。

RB8是工作方式2和3中接收到的第9位数据（例如是奇偶位或者地址/数据标识位），在工作方式1中若SM2=0，则RB8是已接收的停止位。在工作方式0中RB8不使用。

TI 为发送中断标志位，由硬件置位，软件清除。工作方式0中在发送第8位末尾由硬件置位；在其他工作方式时，在发送停止位开始时由硬件置位。TI=1时，申请中断。CPU响应中断后，发送下一帧数据。在任何工作方式中都必须由软件清除TI。

RI为接收中断标志位，由硬件置位，软件清除。工作方式0中在接收第8位末尾由硬件置位；在其他工作方式时，在接收停止位的中间由硬件置位。RI=1时，申请中断，要求CPU取走数据。但在工作方式1中，SM2=1且未接收到有效的停止位时，不会对RI置位。

在任何工作方式中都必须由软件清除RI。

系统复位时，SCON的所有位都被清除。

（1） 工作方式0

SM0=0且SM1=0时，串口选择工作方式0，实质这是一种同步移位寄存器模式。其数据传输的波特率固定为Fosc/12，数据由RXD引脚输入或输出，同步时钟由TXD引脚输出。接收/发送的是8位数据，传输是低位在前，帧格式如下：

…….. D0 D1 D2 D3 D4 D5 D6 D7 …….

（2）工作方式1

当SM0=0且SM1=1时，串口选择工作方式1，其数据传输的波特率由定时/计数器T1、T2的溢出速率决定，可通过程序设定。当T2CON寄存器中的RCLK和TCLK置位时，用T2作为发送和接收波特率发生器，而RCLK=TCLK=0时，用T1作为波特率发生器，两者还可以交叉使用，即发送和接收采用不同的波特率。数据由TXD引脚发送，由RXD引脚接收。发送或接收一帧的数据为10位，即1位起始位（0）、8位数据位（低位在先）和1位停止位（1）。帧格式如下：

起始位0 D0 D1 D2 D3 D4 D5 D6 D7 停止位1

类似于工作方式0，当执行任一条SBUF指令时，就启动串行数据的发送。在执行写入SBUF的指令时，也将“1”写入发送移位寄存器的第9位，并通知发送控制器有发送请求。实际上，发送过程始于内部的16分频计数器下次满度翻转（全“1”变全“0”）后的那几个机器周期的开始。所以，每位的发送过程与16分频计数器同步，而不是与“写SBUF”同步。

1. 方式 2 和方式 3

这两种方式都是 11 位异步接收/发送方式。他们的操作过程都是完全一样

的，所不同的是波特率而已。方式 3 波特率同方式 1（定时器 1 作为波特率时钟发生器） 。

方式 2 和方式 3 的发送起始于任何一条 SBUF 数据装载指令。当第 9 位数据（TB8）输出之后，TI 将被置位（TI=1） 。

方式 2 和方式 3 的接收数据前提条件也是 REN 被编程为 1。 在第 9 位数据接收到后， 如果下列条件同时满足，即 RI=0 且 SM2=0 或者接收到的第 9 位为 1，则将已接受的数据装入 SBUF 缓冲器和 RB8，并将RI 置位（RI=1）否则接收数据无效。

8051 串行口的不同寻常的特征是包括第 9 位方式。它允许把在串行口通信增加的第 9 位用于标志特殊字节的接收。用这种方式，一个单片机可以和大量的其他单片机对话而不打扰不寻址的单片机，这种多机通信方式必须工作在严格的主从方式，由软件进行分析。

## 2.2、12864液晶

### 2.2.1、12864液晶的概述

HS12864-15 系列中文图形液晶模块的特性主要由其控制器 ST7920 决定。ST7920 同时作为控制器和驱动器，它可提供 33 路 com 输出和 64 路 seg 输出。 在驱动器 ST7921 的配合下，最多可以驱动 256×32 点阵液晶。

HS12864-15 系列产品硬件特性如下：

◆提供 8 位，4 位并行接口及串行接口可选

◆并行接口适配 M6800 时序 ◆自动电源启动复位功能

◆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）

◆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）

HS12864-15 系列产品软件特性如下：

◆文字与图形混合显示功能

◆画面清除功能 光标归位功能 显示开/关功能

◆光标显示/隐藏功能

◆显示字体闪烁功能 光标移位功能功能

◆显示移位功能

◆垂直画面旋转功能 反白显示功能

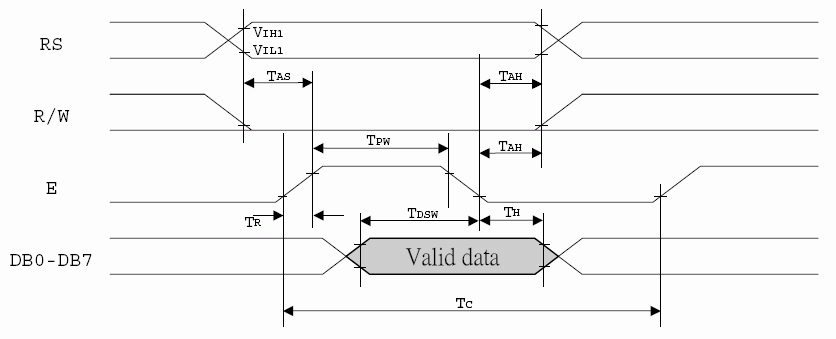
◆休眠模式

HS12864产片与单片机连接接口

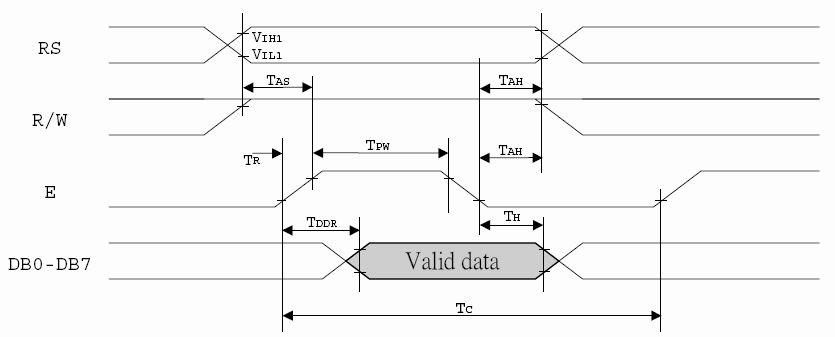


### 2.2.2、12864读写操作时序

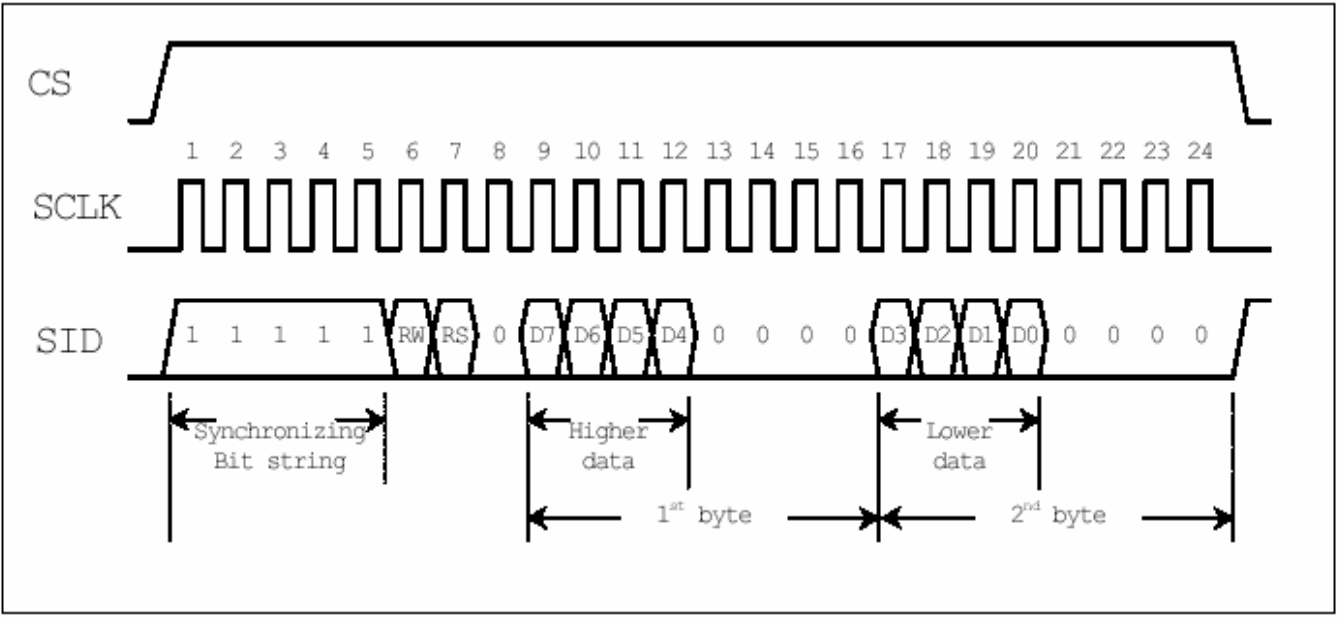
1. 、八位并口写操作时序



1. 、八位滨口读操作时序



1. 、串行连接操作时序



串行数据传送共分三个字节完成：

第一字节：串口控制—格式 11111ABC

A 为数据传送方向控制：H 表示数据从 LCD 到 MCU，L 表示数据从MCU到LCD

B 为数据类型选择：H 表示数据是显示数据，L 表示数据是控制指令

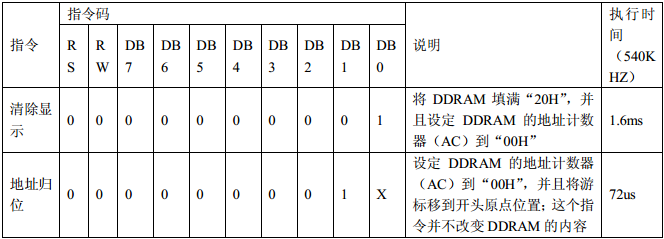
C 固定为 0

第二字节：(并行)8 位数据的高 4 位—格式 DDDD0000

第三字节：(并行)8 位数据的低 4 位—格式 0000DDDD

### 2.2.3、12864的基本指令

指令表 1：（RE=0：基本指令集）





1、清除显示

CODE：

RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L L L L L L L L L H

功能：清除显示屏幕，把 DDRAM 位址计数器调整为“00H”

2、位址归位

CODE：

RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L L L L L L L L H X

功能：把 DDRAM 位址计数器调整为“00H”，游标回原点，该功能不影响显示 DDRAM

3、位址归位

CODE：

RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L L L L L L L H I/D S

功能：把 DDRAM 位址计数器调整为“00H”，游标回原点，该功能不影响显示 DDRAM 功能：执行该命令后，所设置的行将显示在屏幕的第一行。显示起始行是由 Z 地址计数器控制的，该命令自动将 A0-A5 位地址送入 Z 地址计数器，起始地址可以是 0-63 范围内任意一行。Z 地址计数器具有循环计数功能，用于显示行扫描同步，当扫描完一行后自动加一。

4、显示状态 开/关

CODE：

RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L L L L L L H D C B

功能： D=1；整体显示 ON C=1；游标 ON B=1；游标位置 ON

5、游标或显示移位控制

CODE：

RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L L L L L H S/C R/L X X

功能：设定游标的移动与显示的移位控制位：这个指令并不改变 DDRAM 的内容

6、功能设定

CODE：

RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L L L L H DL X RE X X

功能：DL=1（必须设为 1） RE=1；扩充指令集动作 RE=0：基本指令集动作

7、设定 CGRAM 位址

CODE：

RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L L L H AC5 AC4 AC3 AC2 AC1 AC0

功能：设定 CGRAM 位址到位址计数器（AC）

8、设定 DDRAM 位址

CODE：

RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L L H AC6 AC5 AC4 AC3 AC2 AC1 AC0

功能：设定 DDRAM 位址到位址计数器（AC）

9、读取忙碌状态（BF）和位址

CODE：

RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

L H BF AC6 AC5 AC4 AC3 AC2 AC1 AC0

功能：读取忙碌状态（BF）可以确认内部动作是否完成，同时可以读出位址计数器（AC）的值

10、写资料到 RAM

CODE：

RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

H L D7 D6 D5 D4 D3 D2 D1 D0

功能：写入资料到内部的 RAM（DDRAM/CGRAM/TRAM/GDRAM）

11、读出 RAM 的值

CODE：

RS RW DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

H H D7 D6 D5 D4 D3 D2 D1 D0

功能：从内部 RAM 读取资料（DDRAM/CGRAM/TRAM/GDRAM）

# 硬件电路设计及其原理

超市储物柜硬件的设计有单片机电路、24C02存储电路，矩阵键盘电路等；单片机超市储物柜实现存储多个密码，管理员密码等。

硬件框图如下：

电源电路

单 片 机 控 制 电 路

液晶显示电路

存储电路

晶振电路

蜂鸣器报警电路

复位电路

按键矩阵电路

LED指示灯电路

系统总设计框图

## 3.1、最小系统的原理图

单片机最小系统由电源接口，开关，时钟电路，下载接口，复位电路和STC89C54RD+单片机组成。

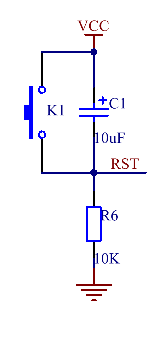


复位电路：当在89C51单片机的RST引脚引入高电平并保持2个机器周期时，单片机内部就执行复位操作（若该引脚持续保持高电平，单片机就处于循环复位状态）

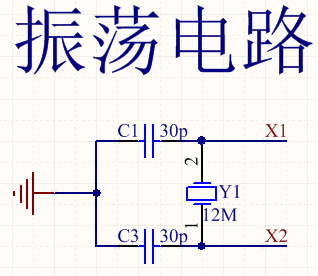
复位电路通常采用上电自动复位和按钮复位两种方式。

最简单的上电自动复位电路中上电自动复位是通过外部复位电路的电容充放电来实现的。只要Vcc的上升时间不超过1ms,就可以实现自动上电复位。

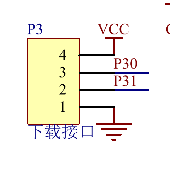
除了上电复位外，有时还需要按键手动复位。本设计就是用的按键手动复位。按键手动复位有电平方式和脉冲方式两种。其中电平复位是通过RST（9）端与电源Vcc接通而实现的。按键手动复位电路见下图。时钟频率用12MHZ时C取10uF,R取10kΩ。



### 时钟电路：STC89C52RC单片机的时钟信号通常有两种方式产生：一是内部时钟方式，二是外部时钟方式。内部时钟方式如图3所示。在89S51单片机内部有一振荡电路，只要在单片机的XTAL1（18）和XTAL2（19）引脚外接石英晶体（简称晶振），就构成了自激振荡器并在单片机内部产生时钟脉冲信号。图中电容C1和C2的作用是稳定频率和快速起振，电容值在5~30pF，典型值为30pF。晶振CYS的振荡频率范围在1.2~12MHz间选择，典型值为12MHz和6MHz。



下载接口：STC89C54系列单片机具有在系统可编程（ISP）特性，ISP的好处是：省去购买通用编程器，单片机在用户系统上即可下载／烧录用户程序，而无须将单片机从已生产好的产品上拆下，再用通用编程器将程序代码烧录进单片机内部。有些程序尚未定型的产品可以一边生产，一边完善，加快了产品进入市场的速度，减小了新产品由于软件缺陷带来的风险。由于可以在用户的目标系统上将程序直接下载进单片机看运行结果对错，故无须仿真器。单片机出厂时就已完全加密。需要单片机内部的电放光后上电复位（冷起动）才运行系统ISP程序，如从P3.0／RｘD检测到合法的下载命令流就下载用户程序，如检测不到就系统复位到用户程序区。具体如下图所示：



## 3.2、报警电路的设计

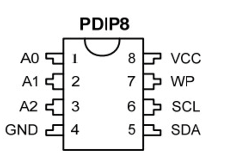
蜂鸣器的正极接电源正极，根据单片机的电流特性，不足以直接驱动蜂鸣器。有三极管的输出特性，利用晶体管的饱和和截止特性构成开关电路。P20为高电平时基极与集电极之间反向偏置，不导通；当P20为低电平时，基极与集电极之间正向偏置，导通。电路工作，蜂鸣器工作，单片机只需要吸收娇小的输入电流。



## 3.3、24c02原理图

串行时钟信号引脚行时钟信号引脚(SCL)(SCL)：在SCL输入时钟信号的上升沿将数据送入EEPROM器件,并在时钟的下降沿将数据读出。

串行数据输入串行数据输入//输出引脚输出引脚(SDA)(SDA)：SDA引脚可实现双向串行数据传输。该引脚为开漏输出，可与其它多个开漏输出器件或开集电极器件线或连接.



## 3.4、12864原理图

选用并行读写模式，故根据引脚定义，通过 Vout 脚引出，通过电位器调节后， 从接口的 V0 脚引入模块用来驱动 LCD。直接驱动 LCD 的是 V0，V0 电压越高， 对比度越深。其原理图如下：



## 3.6、箱号指示灯原理图

用于指示箱子的开关状态。



# 程序设计

超市储物柜系统主要由单片机控制,超市储物柜系统程序需要完成以下的功能：

1、用户存物、用户取物、满箱提示；

2、管理员取物、管理员修改密码；

3、存物时用户密码通过液晶显示；

4、存物箱状态指示、自动开/关箱、有效按键声音提示、密码输入提示、密码正确提示、密码错误提示等功能；

程序流程图如下：



# 参考文献

[1] L.Coetzee, et al. Fingerprint Recognition in Low Quality Images. Pattern Recognition.1993,vol.26:1441-1460.

[2] TIAN Jie, CHEN Xinjian, et al. Recent progress in fingerprint recognition. Progress in Natural Science. 2006, vol.03:231-240.

[3] 郭天祥．新概念51单片机C语言教程[M]．北京:电子工业出版社，2009:60-400．

[4] 张毅刚．单片机原理与应用[M]．北京:高等教育出版社，2006:56-234．

[5] 赵建领．51单片机开发与应用技术详解[M]．北京:电子工业出版社，2009:75-124．

[6] 陈光新．自动指纹识别技术及其应用[J]．江苏船舶．2004，24（03）:32-35．

[7] Kazuharu Yamato, et al. Problems of Fingerprint Lookup Devices and How to Resolve Them. Gazo Denshi Gakkaishi.1995,vol.24:382-391.

[8] 郑春来，韩团军，李鑫．编译软件Keil在单片机教学中的应用[J]．高教论坛．2009，15（12）:96-97．

[9] Prata．S．C Primer Plus中文版[M]．北京:人民邮电出版社，2005:74-99．

[10] 吴凌燕．基于Proteus的单片机仿真设计[J]．仪表技术．2011，17（07）:31-37．

[11] 唐钱辉，丁明亮．Proteus ISIS 和Keil在单片机教学中的应用[J]．重庆电力高等专科学校学报．2006，8（04）:46-49．

[12] 林三宝．高效焊接[M]．北京:机械工业出版社，2012:45-78.

# 致 谢

这篇论文的制作历时了两个月的时间，在论文的写作过程中遇到了无数的困难和障碍，都在同学和老师的帮助下度过了。尤其要强烈感谢同学们，在整个论文实践和论文写作的过程中，他们都对我进行了无私的指导和帮助，有的同学手把手教我焊电路板，由于我不熟练，焊坏了好几块板子，可他们还是鼓励我。还有的同学帮我调试程序，有时一个程序要连续调好几个小时，他们很有耐心，一边调试一边给我讲程序的道理，我获益匪浅。另外，在校图书馆查找资料的时候，图书馆的老师也给我提供了很多方面的支持与帮助。还要感谢指导老师关学忠教授及学长不厌其烦的帮助我进行论文的修改和改进。还有我大学四年来教导过我的所有老师，为我打下了专业知识的基础，在此向所有帮助和指导过我的各位老师表示最衷心的感谢！

感谢这篇论文所涉及到的各位学者。本文引用了许多学者的研究文献，如果没有各位学者的研究成果的帮助和启发，我将很难完成本篇论文的写作。

最后，衷心的感谢在百忙之中评阅论文和参加答辩的各位老师，请您们对论文的不足进行批评和指正。

# 附录

## 原理图：



## 源程序：

#include <STC89C5xRC.H>

#include <intrins.h>

#include "LCD12864\_No\_font\_library.h"

#include "zi\_ku.h"

#include "24C02.h"

uchar system\_time=0;

bit s0=0;

uchar beep\_time=0; //蜂鸣器端口定义

sbit beep=P0^5; //蜂鸣器端口定义

uchar key\_back=16; //按键返回值变量

uchar ms=0; //按键返回值变量

bit key\_flag=0;

uchar Box\_number=0;

uchar xdata password[6] ={0,0,0,0,0,0}; //存储密码的数组

uchar xdata password\_flsh[6]={0,0,0,0,0,0}; //存储密码的数组

uchar password\_count=0; //存储密码的数组计数变量

uchar xdata Password\_flsh2[6]; //这个存储的是，要修的密码密码的时候，再次输入的密码

bit Modify=0; //这个标志位是修改密码用的

uchar Box\_state[8];

uchar Surplus=0;

uchar admin[6];

uchar state=0;

uchar state\_clear=0;

bit clear\_flag=1;

sbit led0=P3^0;

sbit led1=P3^1;

sbit led2=P3^2;

sbit led3=P3^3;

sbit led4=P3^4;

sbit led5=P3^5;

sbit led6=P3^6;

sbit led7=P3^7;

uchar led0\_time=0;

uchar led1\_time=0;

uchar led2\_time=0;

uchar led3\_time=0;

uchar led4\_time=0;

uchar led5\_time=0;

uchar led6\_time=0;

uchar led7\_time=0;

void Uart1Data(uchar dat) //串口数据发送

{

SBUF=dat;

while(!TI);

TI=0;

}

void delay(unsigned int T)

{

while(T--);

}

void memory\_Pass(uchar dat1,dat2,dat3,dat4,dat5,dat6,Num) //保存密码

{

unsigned char tab[6];

tab[0]=dat1;

tab[1]=dat2;

tab[2]=dat3;

tab[3]=dat4;

tab[4]=dat5;

tab[5]=dat6;

Write\_dat\_24C02(tab,Num\*6,6);

}

void read\_Pass(Num) //读取密码

{

Read\_dat\_24C02(password\_flsh,Num\*6,6);

}

void memory\_Box() //保存箱子状态

{

Write\_dat\_24C02(Box\_state,100,8);

}

void read\_memory\_Box() //读箱子状态

{

Read\_dat\_24C02(Box\_state,100,8);

}

void memory\_admin() //保存箱子状态

{

Write\_dat\_24C02(admin,120,6);

}

void read\_memory\_admin() //读箱子状态

{

Read\_dat\_24C02(admin,120,6);

}

void Chinese\_characters\_8X16(unsigned char X\_dat,unsigned char Y\_dat,unsigned char Box\_dat[])

{

unsigned char i;

unsigned char dis\_Box[16];

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

{

dis\_Box[i]=Box\_dat[i];

}

LCD\_display\_byte(X\_dat ,Y\_dat,dis\_Box[0]);

LCD\_display\_byte(X\_dat+1,Y\_dat,dis\_Box[1]);

LCD\_display\_byte(X\_dat+2,Y\_dat,dis\_Box[2]);

LCD\_display\_byte(X\_dat+3,Y\_dat,dis\_Box[3]);

LCD\_display\_byte(X\_dat+4,Y\_dat,dis\_Box[4]);

LCD\_display\_byte(X\_dat+5,Y\_dat,dis\_Box[5]);

LCD\_display\_byte(X\_dat+6,Y\_dat,dis\_Box[6]);

LCD\_display\_byte(X\_dat+7,Y\_dat,dis\_Box[7]);

LCD\_display\_byte(X\_dat ,Y\_dat+1,dis\_Box[8]);

LCD\_display\_byte(X\_dat+1,Y\_dat+1,dis\_Box[9]);

LCD\_display\_byte(X\_dat+2,Y\_dat+1,dis\_Box[10]);

LCD\_display\_byte(X\_dat+3,Y\_dat+1,dis\_Box[11]);

LCD\_display\_byte(X\_dat+4,Y\_dat+1,dis\_Box[12]);

LCD\_display\_byte(X\_dat+5,Y\_dat+1,dis\_Box[13]);

LCD\_display\_byte(X\_dat+6,Y\_dat+1,dis\_Box[14]);

LCD\_display\_byte(X\_dat+7,Y\_dat+1,dis\_Box[15]);

}

void Chinese\_characters\_16X16(unsigned char characters\_X,unsigned char characters\_Y,unsigned char mu[])

{

unsigned char x=0,y=0;

unsigned char characters\_flsh[32];

for(x=0;x<32;x++)

{

characters\_flsh[x]= mu[x];

}

LCD\_display\_byte(characters\_X,characters\_Y,mu[0]);

LCD\_display\_byte(characters\_X+1,characters\_Y,mu[1]);

LCD\_display\_byte(characters\_X+2,characters\_Y,mu[2]);

LCD\_display\_byte(characters\_X+3,characters\_Y,mu[3]);

LCD\_display\_byte(characters\_X+4,characters\_Y,mu[4]);

LCD\_display\_byte(characters\_X+5,characters\_Y,mu[5]);

LCD\_display\_byte(characters\_X+6,characters\_Y,mu[6]);

LCD\_display\_byte(characters\_X+7,characters\_Y,mu[7]);

LCD\_display\_byte(characters\_X+8,characters\_Y,mu[8]);

LCD\_display\_byte(characters\_X+9,characters\_Y,mu[9]);

LCD\_display\_byte(characters\_X+10,characters\_Y,mu[10]);

LCD\_display\_byte(characters\_X+11,characters\_Y,mu[11]);

LCD\_display\_byte(characters\_X+12,characters\_Y,mu[12]);

LCD\_display\_byte(characters\_X+13,characters\_Y,mu[13]);

LCD\_display\_byte(characters\_X+14,characters\_Y,mu[14]);

LCD\_display\_byte(characters\_X+15,characters\_Y,mu[15]);

LCD\_display\_byte(characters\_X,characters\_Y+1,mu[16]);

LCD\_display\_byte(characters\_X+1,characters\_Y+1,mu[17]);

LCD\_display\_byte(characters\_X+2,characters\_Y+1,mu[18]);

LCD\_display\_byte(characters\_X+3,characters\_Y+1,mu[19]);

LCD\_display\_byte(characters\_X+4,characters\_Y+1,mu[20]);

LCD\_display\_byte(characters\_X+5,characters\_Y+1,mu[21]);

LCD\_display\_byte(characters\_X+6,characters\_Y+1,mu[22]);

LCD\_display\_byte(characters\_X+7,characters\_Y+1,mu[23]);

LCD\_display\_byte(characters\_X+8,characters\_Y+1,mu[24]);

LCD\_display\_byte(characters\_X+9,characters\_Y+1,mu[25]);

LCD\_display\_byte(characters\_X+10,characters\_Y+1,mu[26]);

LCD\_display\_byte(characters\_X+11,characters\_Y+1,mu[27]);

LCD\_display\_byte(characters\_X+12,characters\_Y+1,mu[28]);

LCD\_display\_byte(characters\_X+13,characters\_Y+1,mu[29]);

LCD\_display\_byte(characters\_X+14,characters\_Y+1,mu[30]);

LCD\_display\_byte(characters\_X+15,characters\_Y+1,mu[31]);

}

void dispaly()

{

if(state\_clear!=state)

{

state\_clear=state;

LCD\_clear(); password\_count=0;

}

if(state==0) //待机界面

{

password\_count=0;

//显示箱子状态

if(Box\_state[0]==0) Chinese\_characters\_16X16(0,0,num1);

else Chinese\_characters\_16X16(0,0,numEnd);

if(Box\_state[1]==0) Chinese\_characters\_16X16(0,2,num2);

else Chinese\_characters\_16X16(0,2,numEnd);

if(Box\_state[2]==0) Chinese\_characters\_16X16(0,4,num3);

else Chinese\_characters\_16X16(0,4,numEnd);

if(Box\_state[3]==0) Chinese\_characters\_16X16(0,6,num4);

else Chinese\_characters\_16X16(0,6,numEnd);

if(Box\_state[4]==0) Chinese\_characters\_16X16(112,0,num5);

else Chinese\_characters\_16X16(112,0,numEnd);

if(Box\_state[5]==0) Chinese\_characters\_16X16(112,2,num6);

else Chinese\_characters\_16X16(112,2,numEnd);

if(Box\_state[6]==0) Chinese\_characters\_16X16(112,4,num7);

else Chinese\_characters\_16X16(112,4,numEnd);

if(Box\_state[7]==0) Chinese\_characters\_16X16(112,6,num8);

else Chinese\_characters\_16X16(112,6,numEnd);

//显示欢迎界面

Chinese\_characters\_16X16(32,0,huan);

Chinese\_characters\_16X16(48,0,ying);

Chinese\_characters\_16X16(64,0,guang);

Chinese\_characters\_16X16(80,0,lin);

//显示剩余

Surplus=8-(Box\_state[0]+Box\_state[1]+Box\_state[2]+Box\_state[3]+Box\_state[4]+Box\_state[5]+Box\_state[6]+Box\_state[7]);

if(Surplus!=0)

{

Chinese\_characters\_16X16(32,2,sheng);

Chinese\_characters\_16X16(48,2,yu);

Chinese\_characters\_16X16(64,2,xiang);

Chinese\_characters\_8X16(80,2,Ascii\_mao\_hao);

switch(Surplus)

{

case 1 : Chinese\_characters\_8X16(88,2,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(88,2,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(88,2,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(88,2,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(88,2,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(88,2,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(88,2,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(88,2,Ascii\_8); break;

}

//显示提示语

Chinese\_characters\_16X16(24,4,cun);

Chinese\_characters\_16X16(40,4,wu);

Chinese\_characters\_16X16(56,4,qing3);

Chinese\_characters\_16X16(72,4,an);

Chinese\_characters\_16X16(88,4,cun);

Chinese\_characters\_16X16(24,6,qu);

Chinese\_characters\_16X16(40,6,wu);

Chinese\_characters\_16X16(56,6,qing3);

Chinese\_characters\_16X16(72,6,an);

Chinese\_characters\_16X16(88,6,qu);

}

else

{

Chinese\_characters\_16X16(32,2,ci);

Chinese\_characters\_16X16(48,2,gui);

Chinese\_characters\_16X16(64,2,yi);

Chinese\_characters\_16X16(80,2,man);

}

}

else if(state==1) //存物界面

{

Chinese\_characters\_16X16(8,0,qing3);

Chinese\_characters\_16X16(24,0,she);

Chinese\_characters\_16X16(40,0,zhi);

Chinese\_characters\_16X16(56,0,shu\_zi\_liu);

Chinese\_characters\_16X16(72,0,wei);

Chinese\_characters\_16X16(88,0,mi);

Chinese\_characters\_16X16(104,0,ma); ///提示语，请输入六位密码

Chinese\_characters\_16X16(0,2,bing);

Chinese\_characters\_16X16(16,2,an);

Chinese\_characters\_16X16(32,2,xia);

Chinese\_characters\_16X16(48,2,que);

Chinese\_characters\_16X16(64,2,ren);

Chinese\_characters\_16X16(80,2,jian);

Chinese\_characters\_16X16(96,2,que);

Chinese\_characters\_16X16(112,2,ding);

switch(system\_time/10%10)

{

case 0 : Chinese\_characters\_8X16(56,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(56,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(56,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(56,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(56,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(56,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(56,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(56,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(56,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(56,4,Ascii\_9); break;

}

switch(system\_time%10)

{

case 0 : Chinese\_characters\_8X16(64,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(64,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(64,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(64,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(64,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(64,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(64,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(64,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(64,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(64,4,Ascii\_9); break;

}

Chinese\_characters\_16X16(16,6,mi);

Chinese\_characters\_16X16(32,6,ma);

Chinese\_characters\_8X16(48,6,Ascii\_mao\_hao);

if(password\_count>=1) Chinese\_characters\_8X16(56,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(56,6,Ascii\_kong);

if(password\_count>=2) Chinese\_characters\_8X16(64,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(64,6,Ascii\_kong);

if(password\_count>=3) Chinese\_characters\_8X16(72,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(72,6,Ascii\_kong);

if(password\_count>=4) Chinese\_characters\_8X16(80,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(80,6,Ascii\_kong);

if(password\_count>=5) Chinese\_characters\_8X16(88,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(88,6,Ascii\_kong);

if(password\_count>=6) Chinese\_characters\_8X16(96,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(96,6,Ascii\_kong);

}

else if(state==2) //取物界面-输入箱号

{

Chinese\_characters\_16X16(8,0,qing3);

Chinese\_characters\_16X16(24,0,shu);

Chinese\_characters\_16X16(40,0,ru);

Chinese\_characters\_16X16(56,0,qu);

Chinese\_characters\_16X16(72,0,wu);

Chinese\_characters\_16X16(88,0,xiang);

Chinese\_characters\_16X16(104,0,hao);

Chinese\_characters\_16X16(0,2,bing);

Chinese\_characters\_16X16(16,2,an);

Chinese\_characters\_16X16(32,2,xia);

Chinese\_characters\_16X16(48,2,que);

Chinese\_characters\_16X16(64,2,ren);

Chinese\_characters\_16X16(80,2,jian);

Chinese\_characters\_16X16(96,2,jixu\_ji);

Chinese\_characters\_16X16(112,2,xu);

switch(system\_time/10%10)

{

case 0 : Chinese\_characters\_8X16(56,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(56,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(56,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(56,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(56,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(56,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(56,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(56,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(56,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(56,4,Ascii\_9); break;

}

switch(system\_time%10)

{

case 0 : Chinese\_characters\_8X16(64,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(64,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(64,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(64,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(64,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(64,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(64,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(64,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(64,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(64,4,Ascii\_9); break;

}

Chinese\_characters\_16X16(40,6,xiang);

Chinese\_characters\_16X16(56,6,hao);

Chinese\_characters\_8X16(72,6,Ascii\_mao\_hao);

if(password\_count==0)

{

if(s0)

{

Chinese\_characters\_8X16(80,6,Ascii\_xia\_hua\_xian);

}

else

{

Chinese\_characters\_8X16(80,6,Ascii\_kong);

}

}

else

{

switch(Box\_number%10)

{

case 0 : Chinese\_characters\_8X16(80,6,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(80,6,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(80,6,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(80,6,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(80,6,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(80,6,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(80,6,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(80,6,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(80,6,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(80,6,Ascii\_9); break;

}

}

}

else if(state==3) //取物界面-输入密码

{

Chinese\_characters\_16X16(8,0,qing3);

Chinese\_characters\_16X16(24,0,shu);

Chinese\_characters\_16X16(40,0,ru);

Chinese\_characters\_16X16(56,0,qu);

Chinese\_characters\_16X16(72,0,wu);

Chinese\_characters\_16X16(88,0,mi);

Chinese\_characters\_16X16(104,0,ma);

Chinese\_characters\_16X16(0,2,bing);

Chinese\_characters\_16X16(16,2,an);

Chinese\_characters\_16X16(32,2,xia);

Chinese\_characters\_16X16(48,2,que);

Chinese\_characters\_16X16(64,2,ren);

Chinese\_characters\_16X16(80,2,jian);

Chinese\_characters\_16X16(96,2,que);

Chinese\_characters\_16X16(112,2,ding);

switch(system\_time/10%10)

{

case 0 : Chinese\_characters\_8X16(56,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(56,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(56,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(56,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(56,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(56,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(56,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(56,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(56,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(56,4,Ascii\_9); break;

}

switch(system\_time%10)

{

case 0 : Chinese\_characters\_8X16(64,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(64,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(64,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(64,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(64,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(64,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(64,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(64,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(64,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(64,4,Ascii\_9); break;

}

Chinese\_characters\_16X16(16,6,mi);

Chinese\_characters\_16X16(32,6,ma);

Chinese\_characters\_8X16(48,6,Ascii\_mao\_hao);

if(password\_count>=1) Chinese\_characters\_8X16(56,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(56,6,Ascii\_kong);

if(password\_count>=2) Chinese\_characters\_8X16(64,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(64,6,Ascii\_kong);

if(password\_count>=3) Chinese\_characters\_8X16(72,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(72,6,Ascii\_kong);

if(password\_count>=4) Chinese\_characters\_8X16(80,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(80,6,Ascii\_kong);

if(password\_count>=5) Chinese\_characters\_8X16(88,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(88,6,Ascii\_kong);

if(password\_count>=6) Chinese\_characters\_8X16(96,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(96,6,Ascii\_kong);

}

else if(state==4) //存物成功界面，存物成功，箱号XX 请您牢记密码

{

Chinese\_characters\_16X16(32,0,cun);

Chinese\_characters\_16X16(48,0,wu);

Chinese\_characters\_16X16(64,0,cheng);

Chinese\_characters\_16X16(80,0,gong);

Chinese\_characters\_16X16(16,2,qing3);

Chinese\_characters\_16X16(32,2,nin);

Chinese\_characters\_16X16(48,2,lao);

Chinese\_characters\_16X16(64,2,ji);

Chinese\_characters\_16X16(80,2,mi);

Chinese\_characters\_16X16(96,2,ma);

switch(system\_time/10%10)

{

case 0 : Chinese\_characters\_8X16(56,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(56,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(56,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(56,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(56,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(56,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(56,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(56,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(56,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(56,4,Ascii\_9); break;

}

switch(system\_time%10)

{

case 0 : Chinese\_characters\_8X16(64,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(64,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(64,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(64,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(64,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(64,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(64,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(64,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(64,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(64,4,Ascii\_9); break;

}

Chinese\_characters\_16X16(40,6,xiang);

Chinese\_characters\_16X16(56,6,hao);

Chinese\_characters\_8X16(72,6,Ascii\_mao\_hao);

switch(Box\_number%10)

{

case 0 : Chinese\_characters\_8X16(80,6,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(80,6,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(80,6,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(80,6,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(80,6,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(80,6,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(80,6,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(80,6,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(80,6,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(80,6,Ascii\_9); break;

}

}

else if(state==5) //取物界面-取物成功界面 取物成功， 欢迎再次光临

{

Chinese\_characters\_16X16(32,0,qu);

Chinese\_characters\_16X16(48,0,wu);

Chinese\_characters\_16X16(64,0,cheng);

Chinese\_characters\_16X16(80,0,gong);

switch(Box\_number%10)

{

case 0 : Chinese\_characters\_8X16(24,2,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(24,2,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(24,2,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(24,2,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(24,2,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(24,2,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(24,2,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(24,2,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(24,2,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(24,2,Ascii\_9); break;

}

Chinese\_characters\_16X16(32,2,xiang);

Chinese\_characters\_16X16(48,2,men);

Chinese\_characters\_16X16(64,2,yi);

Chinese\_characters\_16X16(80,2,kai);

Chinese\_characters\_16X16(16,6,huan);

Chinese\_characters\_16X16(32,6,ying);

Chinese\_characters\_16X16(48,6,zai);

Chinese\_characters\_16X16(64,6,xia\_ci\_ci);

Chinese\_characters\_16X16(80,6,guang);

Chinese\_characters\_16X16(96,6,lin);

}

else if(state==6) //存物 密码

{

Chinese\_characters\_16X16(32,0,cun);

Chinese\_characters\_16X16(48,0,wu);

Chinese\_characters\_16X16(64,0,shi);

Chinese\_characters\_16X16(80,0,bai);

Chinese\_characters\_8X16(24,2,Ascii\_1);

Chinese\_characters\_16X16(32,2,xiang);

Chinese\_characters\_16X16(48,2,men);

Chinese\_characters\_16X16(64,2,yi);

Chinese\_characters\_16X16(80,2,kai);

Chinese\_characters\_16X16(16,6,huan);

Chinese\_characters\_16X16(32,6,ying);

Chinese\_characters\_16X16(48,6,zai);

Chinese\_characters\_16X16(64,6,xia\_ci\_ci);

Chinese\_characters\_16X16(80,6,guang);

Chinese\_characters\_16X16(96,6,lin);

}

else if(state==7) //管理员登录界面

{

Chinese\_characters\_16X16(24,0,guan);

Chinese\_characters\_16X16(40,0,li);

Chinese\_characters\_16X16(56,0,yuan);

Chinese\_characters\_16X16(72,0,deng);

Chinese\_characters\_16X16(88,0,lu);

switch(system\_time/10%10)

{

case 0 : Chinese\_characters\_8X16(56,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(56,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(56,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(56,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(56,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(56,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(56,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(56,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(56,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(56,4,Ascii\_9); break;

}

switch(system\_time%10)

{

case 0 : Chinese\_characters\_8X16(64,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(64,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(64,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(64,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(64,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(64,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(64,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(64,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(64,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(64,4,Ascii\_9); break;

}

Chinese\_characters\_16X16(16,6,mi);

Chinese\_characters\_16X16(32,6,ma);

Chinese\_characters\_8X16(48,6,Ascii\_mao\_hao);

if(password\_count>=1) Chinese\_characters\_8X16(56,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(56,6,Ascii\_kong);

if(password\_count>=2) Chinese\_characters\_8X16(64,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(64,6,Ascii\_kong);

if(password\_count>=3) Chinese\_characters\_8X16(72,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(72,6,Ascii\_kong);

if(password\_count>=4) Chinese\_characters\_8X16(80,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(80,6,Ascii\_kong);

if(password\_count>=5) Chinese\_characters\_8X16(88,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(88,6,Ascii\_kong);

if(password\_count>=6) Chinese\_characters\_8X16(96,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(96,6,Ascii\_kong);

}

else if(state==8) //管理员登录界面,箭头指向第一个选项“管理员取物”

{

Chinese\_characters\_16X16(24,0,guan);

Chinese\_characters\_16X16(40,0,li);

Chinese\_characters\_16X16(56,0,yuan);

Chinese\_characters\_16X16(72,0,jie);

Chinese\_characters\_16X16(88,0,mian);

switch(system\_time/10%10)

{

case 0 : Chinese\_characters\_8X16(56,2,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(56,2,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(56,2,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(56,2,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(56,2,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(56,2,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(56,2,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(56,2,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(56,2,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(56,2,Ascii\_9); break;

}

switch(system\_time%10)

{

case 0 : Chinese\_characters\_8X16(64,2,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(64,2,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(64,2,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(64,2,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(64,2,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(64,2,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(64,2,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(64,2,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(64,2,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(64,2,Ascii\_9); break;

}

Chinese\_characters\_8X16(8,4,Ascii\_1);

Chinese\_characters\_8X16(16,4,Ascii\_mao\_hao);

Chinese\_characters\_16X16(32,4,guan);

Chinese\_characters\_16X16(48,4,li);

Chinese\_characters\_16X16(64,4,yuan);

Chinese\_characters\_16X16(80,4,qu);

Chinese\_characters\_16X16(96,4,wu);

Chinese\_characters\_8X16(112,4,Ascii\_zuo\_jiantou);

Chinese\_characters\_8X16(8,6,Ascii\_2);

Chinese\_characters\_8X16(16,6,Ascii\_mao\_hao);

Chinese\_characters\_16X16(32,6,geng);

Chinese\_characters\_16X16(48,6,gai);

Chinese\_characters\_16X16(64,6,mi);

Chinese\_characters\_16X16(80,6,ma);

}

else if(state==9) //管理员界面,箭头指向第2个选项“更改密码”

{

Chinese\_characters\_16X16(24,0,guan);

Chinese\_characters\_16X16(40,0,li);

Chinese\_characters\_16X16(56,0,yuan);

Chinese\_characters\_16X16(72,0,jie);

Chinese\_characters\_16X16(88,0,mian);

switch(system\_time/10%10)

{

case 0 : Chinese\_characters\_8X16(56,2,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(56,2,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(56,2,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(56,2,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(56,2,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(56,2,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(56,2,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(56,2,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(56,2,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(56,2,Ascii\_9); break;

}

switch(system\_time%10)

{

case 0 : Chinese\_characters\_8X16(64,2,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(64,2,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(64,2,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(64,2,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(64,2,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(64,2,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(64,2,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(64,2,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(64,2,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(64,2,Ascii\_9); break;

}

Chinese\_characters\_8X16(8,4,Ascii\_1);

Chinese\_characters\_8X16(16,4,Ascii\_mao\_hao);

Chinese\_characters\_16X16(32,4,guan);

Chinese\_characters\_16X16(48,4,li);

Chinese\_characters\_16X16(64,4,yuan);

Chinese\_characters\_16X16(80,4,qu);

Chinese\_characters\_16X16(96,4,wu);

Chinese\_characters\_8X16(8,6,Ascii\_2);

Chinese\_characters\_8X16(16,6,Ascii\_mao\_hao);

Chinese\_characters\_16X16(32,6,geng);

Chinese\_characters\_16X16(48,6,gai);

Chinese\_characters\_16X16(64,6,mi);

Chinese\_characters\_16X16(80,6,ma);

Chinese\_characters\_8X16(112,6,Ascii\_zuo\_jiantou);

}

else if(state==10) //管理员取物界面

{

Chinese\_characters\_16X16(24,0,guan);

Chinese\_characters\_16X16(40,0,li);

Chinese\_characters\_16X16(56,0,yuan);

Chinese\_characters\_16X16(72,0,qu);

Chinese\_characters\_16X16(88,0,wu);

Chinese\_characters\_16X16(0,2,an);

Chinese\_characters\_16X16(16,2,xia);

Chinese\_characters\_16X16(32,2,dui);

Chinese\_characters\_16X16(48,2,ying4);

Chinese\_characters\_16X16(64,2,xiang);

Chinese\_characters\_16X16(80,2,hao);

Chinese\_characters\_16X16(96,2,kai);

Chinese\_characters\_16X16(112,2,xiang);

switch(system\_time/10%10)

{

case 0 : Chinese\_characters\_8X16(56,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(56,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(56,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(56,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(56,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(56,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(56,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(56,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(56,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(56,4,Ascii\_9); break;

}

switch(system\_time%10)

{

case 0 : Chinese\_characters\_8X16(64,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(64,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(64,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(64,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(64,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(64,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(64,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(64,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(64,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(64,4,Ascii\_9); break;

}

if(Box\_state[0]==0) Chinese\_characters\_16X16(0,6,num1);

else Chinese\_characters\_16X16(0,6,numEnd);

if(Box\_state[1]==0) Chinese\_characters\_16X16(16,6,num2);

else Chinese\_characters\_16X16(16,6,numEnd);

if(Box\_state[2]==0) Chinese\_characters\_16X16(32,6,num3);

else Chinese\_characters\_16X16(32,6,numEnd);

if(Box\_state[3]==0) Chinese\_characters\_16X16(48,6,num4);

else Chinese\_characters\_16X16(48,6,numEnd);

if(Box\_state[4]==0) Chinese\_characters\_16X16(64,6,num5);

else Chinese\_characters\_16X16(64,6,numEnd);

if(Box\_state[5]==0) Chinese\_characters\_16X16(80,6,num6);

else Chinese\_characters\_16X16(80,6,numEnd);

if(Box\_state[6]==0) Chinese\_characters\_16X16(96,6,num7);

else Chinese\_characters\_16X16(96,6,numEnd);

if(Box\_state[7]==0) Chinese\_characters\_16X16(112,6,num8);

else Chinese\_characters\_16X16(112,6,numEnd);

}

else if(state==11) //管理员取物界面

{

Chinese\_characters\_16X16(16,0,geng);

Chinese\_characters\_16X16(32,0,gai);

Chinese\_characters\_16X16(48,0,guan);

Chinese\_characters\_16X16(64,0,li);

Chinese\_characters\_16X16(80,0,mi);

Chinese\_characters\_16X16(96,0,ma);

switch(system\_time/10%10)

{

case 0 : Chinese\_characters\_8X16(56,2,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(56,2,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(56,2,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(56,2,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(56,2,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(56,2,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(56,2,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(56,2,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(56,2,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(56,2,Ascii\_9); break;

}

switch(system\_time%10)

{

case 0 : Chinese\_characters\_8X16(64,2,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(64,2,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(64,2,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(64,2,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(64,2,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(64,2,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(64,2,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(64,2,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(64,2,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(64,2,Ascii\_9); break;

}

Chinese\_characters\_16X16(0,4,shu);

Chinese\_characters\_16X16(16,4,ru);

Chinese\_characters\_16X16(32,4,mi);

Chinese\_characters\_16X16(48,4,ma);

Chinese\_characters\_8X16(64,4,Ascii\_mao\_hao);

if(password\_count>=1) Chinese\_characters\_8X16(72,4,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(72,4,Ascii\_kong);

if(password\_count>=2) Chinese\_characters\_8X16(80,4,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(80,4,Ascii\_kong);

if(password\_count>=3) Chinese\_characters\_8X16(88,4,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(88,4,Ascii\_kong);

if(password\_count>=4) Chinese\_characters\_8X16(96,4,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(96,4,Ascii\_kong);

if(password\_count>=5) Chinese\_characters\_8X16(104,4,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(104,4,Ascii\_kong);

if(password\_count>=6) Chinese\_characters\_8X16(112,4,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(112,4,Ascii\_kong);

Chinese\_characters\_16X16(0,6,zai);

Chinese\_characters\_16X16(16,6,ci);

Chinese\_characters\_16X16(32,6,shu);

Chinese\_characters\_16X16(48,6,ru);

Chinese\_characters\_8X16(64,6,Ascii\_mao\_hao);

if(password\_count>=7) Chinese\_characters\_8X16(72,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(72,6,Ascii\_kong);

if(password\_count>=8) Chinese\_characters\_8X16(80,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(80,6,Ascii\_kong);

if(password\_count>=9) Chinese\_characters\_8X16(88,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(88,6,Ascii\_kong);

if(password\_count>=10) Chinese\_characters\_8X16(96,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(96,6,Ascii\_kong);

if(password\_count>=11) Chinese\_characters\_8X16(104,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(104,6,Ascii\_kong);

if(password\_count>=12) Chinese\_characters\_8X16(112,6,Ascii\_xing\_hao);

else Chinese\_characters\_8X16(112,6,Ascii\_kong);

}

else if(state==12) //提示界面，输入的箱号，没有存物界面

{

Chinese\_characters\_16X16(32,0,ti);

Chinese\_characters\_16X16(48,0,ti\_shi\_shi);

Chinese\_characters\_16X16(64,0,jie);

Chinese\_characters\_16X16(80,0,mian);

switch(system\_time/10%10)

{

case 0 : Chinese\_characters\_8X16(56,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(56,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(56,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(56,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(56,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(56,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(56,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(56,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(56,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(56,4,Ascii\_9); break;

}

switch(system\_time%10)

{

case 0 : Chinese\_characters\_8X16(64,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(64,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(64,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(64,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(64,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(64,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(64,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(64,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(64,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(64,4,Ascii\_9); break;

}

Chinese\_characters\_16X16(16,6,ci);

Chinese\_characters\_16X16(32,6,xiang);

Chinese\_characters\_16X16(48,6,mei);

Chinese\_characters\_16X16(64,6,you);

Chinese\_characters\_16X16(80,6,cun);

Chinese\_characters\_16X16(96,6,wu);

}

else if(state==13) //提示界面，取物失败，密码错误

{

Chinese\_characters\_16X16(32,0,qu);

Chinese\_characters\_16X16(48,0,wu);

Chinese\_characters\_16X16(64,0,shi);

Chinese\_characters\_16X16(80,0,bai);

switch(system\_time/10%10)

{

case 0 : Chinese\_characters\_8X16(56,2,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(56,2,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(56,2,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(56,2,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(56,2,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(56,2,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(56,2,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(56,2,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(56,2,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(56,2,Ascii\_9); break;

}

switch(system\_time%10)

{

case 0 : Chinese\_characters\_8X16(64,2,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(64,2,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(64,2,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(64,2,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(64,2,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(64,2,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(64,2,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(64,2,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(64,2,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(64,2,Ascii\_9); break;

}

Chinese\_characters\_16X16(32,4,mi);

Chinese\_characters\_16X16(48,4,ma);

Chinese\_characters\_16X16(64,4,cuo\_wu\_cuo);

Chinese\_characters\_16X16(80,4,cuo\_wu\_wu);

Chinese\_characters\_16X16(96,4,dou\_hao);

Chinese\_characters\_16X16(112,4,yi\_wang\_yi);

Chinese\_characters\_16X16(0,6,yi\_wang\_wang);

Chinese\_characters\_16X16(16,6,mi);

Chinese\_characters\_16X16(32,6,ma);

Chinese\_characters\_16X16(48,6,lian);

Chinese\_characters\_16X16(64,6,xi);

Chinese\_characters\_16X16(80,6,guan);

Chinese\_characters\_16X16(96,6,li);

Chinese\_characters\_16X16(112,6,yuan);

}

else if(state==14) //管理员更改密码成功界面 ying

{

Chinese\_characters\_16X16(16,0,geng);

Chinese\_characters\_16X16(32,0,gai);

Chinese\_characters\_16X16(48,0,guan);

Chinese\_characters\_16X16(64,0,li);

Chinese\_characters\_16X16(80,0,mi);

Chinese\_characters\_16X16(96,0,ma);

switch(system\_time/10%10)

{

case 0 : Chinese\_characters\_8X16(56,2,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(56,2,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(56,2,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(56,2,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(56,2,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(56,2,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(56,2,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(56,2,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(56,2,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(56,2,Ascii\_9); break;

}

switch(system\_time%10)

{

case 0 : Chinese\_characters\_8X16(64,2,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(64,2,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(64,2,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(64,2,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(64,2,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(64,2,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(64,2,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(64,2,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(64,2,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(64,2,Ascii\_9); break;

}

Chinese\_characters\_16X16(32,6,geng);

Chinese\_characters\_16X16(48,6,gai);

Chinese\_characters\_16X16(64,6,cheng);

Chinese\_characters\_16X16(80,6,gong);

}

else if(state==15) //管理员更改密码失败界面

{

Chinese\_characters\_16X16(16,0,geng);

Chinese\_characters\_16X16(32,0,gai);

Chinese\_characters\_16X16(48,0,guan);

Chinese\_characters\_16X16(64,0,li);

Chinese\_characters\_16X16(80,0,mi);

Chinese\_characters\_16X16(96,0,ma);

switch(system\_time/10%10)

{

case 0 : Chinese\_characters\_8X16(56,2,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(56,2,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(56,2,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(56,2,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(56,2,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(56,2,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(56,2,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(56,2,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(56,2,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(56,2,Ascii\_9); break;

}

switch(system\_time%10)

{

case 0 : Chinese\_characters\_8X16(64,2,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(64,2,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(64,2,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(64,2,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(64,2,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(64,2,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(64,2,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(64,2,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(64,2,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(64,2,Ascii\_9); break;

}

Chinese\_characters\_16X16(32,6,geng);

Chinese\_characters\_16X16(48,6,gai);

Chinese\_characters\_16X16(64,6,shi);

Chinese\_characters\_16X16(80,6,bai);

}

else if(state==16) //管理员登录界面

{

Chinese\_characters\_16X16(24,0,guan);

Chinese\_characters\_16X16(40,0,li);

Chinese\_characters\_16X16(56,0,yuan);

Chinese\_characters\_16X16(72,0,deng);

Chinese\_characters\_16X16(88,0,lu);

switch(system\_time/10%10)

{

case 0 : Chinese\_characters\_8X16(56,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(56,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(56,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(56,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(56,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(56,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(56,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(56,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(56,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(56,4,Ascii\_9); break;

}

switch(system\_time%10)

{

case 0 : Chinese\_characters\_8X16(64,4,Ascii\_0); break;

case 1 : Chinese\_characters\_8X16(64,4,Ascii\_1); break;

case 2 : Chinese\_characters\_8X16(64,4,Ascii\_2); break;

case 3 : Chinese\_characters\_8X16(64,4,Ascii\_3); break;

case 4 : Chinese\_characters\_8X16(64,4,Ascii\_4); break;

case 5 : Chinese\_characters\_8X16(64,4,Ascii\_5); break;

case 6 : Chinese\_characters\_8X16(64,4,Ascii\_6); break;

case 7 : Chinese\_characters\_8X16(64,4,Ascii\_7); break;

case 8 : Chinese\_characters\_8X16(64,4,Ascii\_8); break;

case 9 : Chinese\_characters\_8X16(64,4,Ascii\_9); break;

}

Chinese\_characters\_16X16(24,6,mi);

Chinese\_characters\_16X16(40,6,ma);

Chinese\_characters\_16X16(56,6,cuo\_wu\_cuo);

Chinese\_characters\_16X16(72,6,cuo\_wu\_wu);

}

}

void Time0\_int()

{

TMOD = 0x01; //设置定时器模式

TL0 = 0x00; //设置定时初值

TH0 = 0x4C; //设置定时初值

TR0 = 1; //定时器0开始计时

ET0 = 1;

}

void Uart\_int()

{

PCON &= 0x7F; //波特率不倍速

SCON = 0x50; //8位数据,可变波特率

AUXR &= 0xBF; //定时器1时钟为Fosc/12,即12T

AUXR &= 0xFE; //串口1选择定时器1为波特率发生器

TMOD &= 0x0F; //清除定时器1模式位

TMOD |= 0x20; //设定定时器1为8位自动重装方式

TL1 = 0xFD; //设定定时初值

TH1 = 0xFD; //设定定时器重装值

ET1 = 0; //禁止定时器1中断

TR1 = 1; //启动定时器1

}

/\*

4\*4矩阵键盘处理函数

对于当前程序与硬件来说的键值：

1→0xE7 2→0xD7 3→0xB7 存 →0x77

4→0xEB 5→0xDB 6→0xBB 取 →0x7B

7→0xED 8→0xDD 9→0xBD 管理→0x7D

1→0xEE 1→0xDE 1→0xBE 选择→0x7E

\*/

void key\_back\_dispose()

{

unsigned char Key\_Flsh1=0,Key\_Flsh2=0;

P1=0x0F;

delay(55);

if(P1!=0x0F)

{

if(key\_flag)

{

key\_flag=0;

Key\_Flsh1=P1;

P1=0xf0;

if(P1!=0xf0) //

{

Key\_Flsh2=P1|Key\_Flsh1;

switch(Key\_Flsh2)

{

case 0xE7 : key\_back=1; break;

case 0xD7 : key\_back=2; break;

case 0xB7 : key\_back=3; break;

case 0x77 : key\_back=12; break;

case 0xEB : key\_back=4; break;

case 0xDB : key\_back=5; break;

case 0xBB : key\_back=6; break;

case 0x7B : key\_back=13; break;

case 0xED : key\_back=7; break;

case 0xDD : key\_back=8; break;

case 0xBD : key\_back=9; break;

case 0x7D : key\_back=14; break;

case 0xEE : key\_back=10; break;

case 0xDE : key\_back=0; break;

case 0xBE : key\_back=11; break;

case 0x7E : key\_back=15; break;

}

beep\_time=1;

// while(P1!=0x0F) //按键死循环释放

// {

// P1=0x0f; //先赋值，赋值之后短延时稳定一下数据，然后再判断数据

// delay(333);

// };

if(state!=0)system\_time=10;

}

}

}

else

{

key\_flag=1;

}

}

void key\_dispose()

{

if(key\_back!=16)

{

if(state==0) //如果说是待机界面 ,只有存取按键和管理界面可用

{

if(key\_back==12) //存按键按下

{

key\_back=16; //清除按键值

system\_time=10;

state=1;

password\_count=0;

}

else if(key\_back==13) //取按键按下

{

key\_back=16; //清除按键值

system\_time=10;

state=2;

}

else if(key\_back==14) //管理按键按下

{

key\_back=16; //清除按键值

system\_time=10;

state=7;

}

}

else if(state==1) //跳转到存物界面，按键部分只有0-9，删除/退出 确定 按键可用

{

if(key\_back>=0&&key\_back<=9) //按键按下0-9，说明是输入密码

{

if(password\_count<6) //判断密码输入位数够不够六位

{

password[password\_count]=key\_back;

password\_count++;

}

else

{

beep\_time=4;

}

key\_back=16; //清除按键值

}

else if(key\_back==10) //退格按键按下，只要密码不为空，就退一位

{

key\_back=16; //清除按键值

if(password\_count>0) password\_count--;

}

else if(key\_back==11) //确认按键按下

{ key\_back=16; //清除按键值

if(password\_count==6)

{

if(Box\_state[0]==0)

{ Box\_state[0]=1;

memory\_Box(); Box\_number=1;

memory\_Pass(password[0],password[1],password[2],password[3],password[4],password[5],Box\_number);

led0\_time=40; led0=0;

}

else if(Box\_state[1]==0)

{ Box\_state[1]=1;

memory\_Box(); Box\_number=2;

memory\_Pass(password[0],password[1],password[2],password[3],password[4],password[5],Box\_number);

led1\_time=40; led1=0;

}

else if(Box\_state[2]==0)

{ Box\_state[2]=1;

memory\_Box(); Box\_number=3;

memory\_Pass(password[0],password[1],password[2],password[3],password[4],password[5],Box\_number);

led2\_time=40; led2=0;

}

else if(Box\_state[3]==0)

{Box\_state[3]=1;

memory\_Box(); Box\_number=4;

memory\_Pass(password[0],password[1],password[2],password[3],password[4],password[5],Box\_number);

led3\_time=40; led3=0;

}

else if(Box\_state[4]==0)

{ Box\_state[4]=1;

memory\_Box(); Box\_number=5;

memory\_Pass(password[0],password[1],password[2],password[3],password[4],password[5],Box\_number);

led4\_time=40; led4=0;

}

else if(Box\_state[5]==0)

{ Box\_state[5]=1;

memory\_Box(); Box\_number=6;

memory\_Pass(password[0],password[1],password[2],password[3],password[4],password[5],Box\_number);

led5\_time=40; led5=0;

}

else if(Box\_state[6]==0)

{ Box\_state[6]=1;

memory\_Box(); Box\_number=7;

memory\_Pass(password[0],password[1],password[2],password[3],password[4],password[5],Box\_number);

led6\_time=40; led6=0;

}

else if(Box\_state[7]==0)

{ Box\_state[7]=1;

memory\_Box(); Box\_number=8;

memory\_Pass(password[0],password[1],password[2],password[3],password[4],password[5],Box\_number);

led7\_time=40; led7=0;

}

system\_time=3;

state=4;

}

else

{

beep\_time=4;

}

}

else if(key\_back==15) //回到主界面按键按下

{ key\_back=16; //清除按键值

password\_count=0; //计数变量也清零

state=0; //回到主界面变量

}

}

else if(state==2) //跳转到取物界面，按键部分只有0-9，删除/退出 按键可用 这里是取物箱号选定的数据处理部分

{

if(key\_back>=0&&key\_back<=9) //按键按下0-9，说明是输入密箱子号

{

if(password\_count<=1) //判断密码输入位数够不够一位

{

Box\_number=key\_back;

if(password\_count!=1) password\_count++;

}

else

{

beep\_time=4;

}

key\_back=16; //清除按键值

}

else if(key\_back==10) //退格按键按下，只要密码不为空，就退一位

{

key\_back=16; //清除按键值

if(password\_count>0) password\_count--;

}

else if(key\_back==15) //回到主界面按键按下

{ key\_back=16; //清除按键值

password\_count=0; //计数变量也清零

state=0; //回到主界面变量

}

else if(key\_back==11) //确认按键按下

{ key\_back=16; //清除按键值

if(password\_count==1&&Box\_number!=0&&Box\_number!=9) //先判断是否有正确的箱号输入

{

if(Box\_state[Box\_number-1]==1) //确定一下这个箱子里面有东西

{

state=3;

password\_count=0;

}

else //这里欠缺一个界面，显示出箱子原本是空的界面处理

{

beep\_time=4; system\_time=1; state=12;

}

}

else //这里还可以添加一个界面，显示出，填写的密码不正确的界面

{

beep\_time=10;

}

}

}

else if(state==3)

{

if(key\_back>=0&&key\_back<=9) //按键按下0-9，说明是输入密码

{

if(password\_count<6) //判断密码输入位数够不够六位

{

password[password\_count]=key\_back;

password\_count++;

}

else

{

beep\_time=4;

}

key\_back=16; //清除按键值

}

else if(key\_back==10) //退格按键按下，只要密码不为空，就退一位

{

key\_back=16; //清除按键值

if(password\_count>0) password\_count--;

}

else if(key\_back==11) //确认按键按下

{ key\_back=16; //清除按键值

if(password\_count==6)

{

read\_Pass(Box\_number);

if(password\_flsh[0]==password[0]&&password\_flsh[1]==password[1]&&password\_flsh[2]==password[2]&&password\_flsh[3]==password[3]&&password\_flsh[4]==password[4]&&password\_flsh[5]==password[5])

{

memory\_Pass(0,0,0,0,0,0,Box\_number); //密码确认正确，清空密码

Box\_state[Box\_number-1]=0;

state=5;

system\_time=1;

memory\_Box();

switch(Box\_number)

{

case 1 : led0\_time=40; led0=0; break;

case 2 : led1\_time=40; led1=0; break;

case 3 : led2\_time=40; led2=0; break;

case 4 : led3\_time=40; led3=0; break;

case 5 : led4\_time=40; led4=0; break;

case 6 : led5\_time=40; led5=0; break;

case 7 : led6\_time=40; led6=0; break;

case 8 : led7\_time=40; led7=0; break;

}

}

else

{

state=13; system\_time=2;

}

}

else

{

beep\_time=4;

}

}

else if(key\_back==15) //回到主界面按键按下

{ key\_back=16; //清除按键值

password\_count=0; //计数变量也清零

state=0; //回到主界面变量

}

}

else if(state==4||state==15||state==14||state==16)

{

if(key\_back!=16)

{

if(state==15||state==14)

{

state=8;

}

else

{

state=0;

}

key\_back=16;

}

}

else if(state==7) //跳转到管理界面, 按键部分只有0-9，删除/退出 还有选项按键可用

{

if(key\_back>=0&&key\_back<=9) //按键按下0-9，说明是输入密码

{

if(password\_count<6) //判断密码输入位数够不够六位

{

password[password\_count]=key\_back;

password\_count++;

}

else

{

beep\_time=4;

}

key\_back=16; //清除按键值

}

else if(key\_back==10) //退格按键按下，只要密码不为空，就退一位

{

key\_back=16; //清除按键值

if(password\_count>0) password\_count--;

}

else if(key\_back==11) //确认按键按下

{ key\_back=16; //清除按键值

if(password\_count==6)

{

if((password[0]==1&&password[1]==3&&password[2]==1&&password[3]==4&&password[4]==2&&password[5]==0)||(password[0]==admin[0]&&password[1]==admin[1]&&password[2]==admin[2]&&password[3]==admin[3]&&password[4]==admin[4]&&password[5]==admin[5]))

{

password\_count=0;

state=8;

}

else

{

state=16; system\_time=2;

}

}

}

else if(key\_back==15) //回到主界面按键按下

{ key\_back=16; //清除按键值

password\_count=0; //计数变量也清零

state=0; //回到主界面变量

}

}

else if(state==8||state==9)

{

if(key\_back==1) //选择按键，选择第一个子菜单

{

state=8;

key\_back=16;

}

else if(key\_back==2) //选择按键，选择第二个子菜单

{

state=9;

key\_back=16;

}

else if(key\_back==11) //确认按键按下

{ key\_back=16;

if(state==8) //如果处于第一个子菜单，就跳转到，指定开箱的界面

{

state=10;

}

else

{

state=11; Modify=0;

}

}

else if(key\_back==15) //回到主界面按键按下

{ key\_back=16; //清除按键值

password\_count=0; //计数变量也清零

state=0; //回到主界面变量

}

}

else if(state==10)

{

if(key\_back>=1&&key\_back<=8)

{

Box\_number=key\_back;

memory\_Pass(0,0,0,0,0,0,Box\_number); //密码确认正确，清空密码

Box\_state[Box\_number-1]=0;

memory\_Box();

switch(Box\_number)

{

case 1 : led0\_time=40; led0=0; break;

case 2 : led1\_time=40; led1=0; break;

case 3 : led2\_time=40; led2=0; break;

case 4 : led3\_time=40; led3=0; break;

case 5 : led4\_time=40; led4=0; break;

case 6 : led5\_time=40; led5=0; break;

case 7 : led6\_time=40; led6=0; break;

case 8 : led7\_time=40; led7=0; break;

}

key\_back=16;

}

else if(key\_back==11) //确认按键按下

{ key\_back=16;

state=8;

}

else if(key\_back==15) //回到主界面按键按下

{ key\_back=16; //清除按键值

password\_count=0; //计数变量也清零

state=0; //回到主界面变量

}

else key\_back=16;

}

else if(state==11)

{

if(key\_back>=0&&key\_back<=9) //按键按下0-9，说明是输入密箱子号

{

if(Modify==0) //修改数据标志位为0，说明正在填写要修改的密码

{

if(password\_count<6) //这里对记录第一次输入的密码记录

{

password[password\_count]=key\_back;

password\_count++;

}

}

else //这里对记录第二次输入的密码记录

{

if(password\_count>=6&&password\_count<12)

{

Password\_flsh2[password\_count-6]=key\_back;

password\_count++;

}

}

key\_back=16; //清除按键值

}

else if(key\_back==10) //退格按键按下，只要密码不为空，就退一位

{

key\_back=16; //清除按键值

if(password\_count>0) password\_count--;

if(password\_count<=5) Modify=0;

}

else if(key\_back==15) //回到主界面按键按下

{ key\_back=16; //清除按键值

password\_count=0; //计数变量也清零

state=0; //回到主界面变量

}

else if(key\_back==11) //确认按键按下

{

key\_back=16; //清除按键值

if(password\_count==6)

{

Modify=1;

}

if(password\_count==12)

{

if(password[0]==Password\_flsh2[0]&&password[1]==Password\_flsh2[1]&&password[2]==Password\_flsh2[2]&&password[3]==Password\_flsh2[3]&&password[4]==Password\_flsh2[4]&&password[5]==Password\_flsh2[5])

{

admin[0]=password[0];

admin[1]=password[1];

admin[2]=password[2];

admin[3]=password[3];

admin[4]=password[4];

admin[5]=password[5];

memory\_admin();

state=14; system\_time=2;

}

else

{

state=15; system\_time=2;

}

}

}

}

}

}

void led\_dispose()

{

if(led0\_time!=0) led0\_time--;

else led0=1;

if(led1\_time!=0) led1\_time--;

else led1=1;

if(led2\_time!=0) led2\_time--;

else led2=1;

if(led3\_time!=0) led3\_time--;

else led3=1;

if(led4\_time!=0) led4\_time--;

else led4=1;

if(led5\_time!=0) led5\_time--;

else led5=1;

if(led6\_time!=0) led6\_time--;

else led6=1;

if(led7\_time!=0) led7\_time--;

else led7=1;

}

void main()

{

Time0\_int();

// Uart\_int();

EA=1;

LCD\_initialize();

LCD\_clear();

read\_memory\_Box();

read\_memory\_admin();

while(1)

{

key\_back\_dispose();

dispaly();

key\_dispose();

}

}

void time0() interrupt 1

{

TH0=0x3C;

TL0=0xb0;

ms++;

led\_dispose();

if(beep\_time!=0)

{

if(beep\_time!=0)

{

beep\_time--;

beep=~beep;

}

}else beep=1;

if(ms%10==0) s0=~s0;

if(ms>=20)

{

ms=0;

if(system\_time!=0) system\_time--;

else

{

if(state==15||state==14)

{

system\_time=10; state=8;

}

else state=0;

}

}

}

//void uart() interrupt 4

// {

// if(RI)

// {

// RI=0;

// }

// }