

2022/2023学年第二学期

**《单片机技术应用》设计报告**

课题 电子密码锁的设计

班级 计算机2101

学号 3191803120

姓名 张康

2023年 5月 6日

**目 录**

目录

[1. 设计目的 3](#_Toc26446)

[1.1. 目的 3](#_Toc30520)

[2. 设计任务 3](#_Toc13390)

[2.1. 基本设计要求 3](#_Toc22401)

[2.1.1. 开机自检 3](#_Toc15287)

[2.1.2. 平时显示时间 3](#_Toc19990)

[2.1.3. 行列式键盘 3](#_Toc8354)

[2.1.4. 时钟设定 4](#_Toc30705)

[2.1.5. 密码开锁 4](#_Toc353)

[2.1.6. 密码设置 4](#_Toc619)

[2.1.7. 恢复出厂设置 4](#_Toc1257)

[2.2. 扩展功能（选做） 4](#_Toc5001)

[2.2.1. 显示、设定日期功能 4](#_Toc28814)

[2.2.2. 设定多组密码 5](#_Toc7488)

[2.2.3. 防偷窥密码的虚位密码功能 5](#_Toc28823)

[2.2.4. 其它自选的扩展功能 5](#_Toc15423)

[3. 系统设计 5](#_Toc21105)

[3.1. 硬件设计 5](#_Toc1484)

[3.2. 软件设计 6](#_Toc9617)

[3.2.1. 开机自检 7](#_Toc1455)

[3.2.2. 显示时间主界面 14](#_Toc8257)

[3.2.3. 行列式键盘 22](#_Toc29757)

[3.2.4. 时钟设定功能 25](#_Toc20453)

[3.2.5. 密码开锁功能 28](#_Toc11781)

[3.2.6. 密码设定 33](#_Toc6283)

[3.2.7. 恢复出厂设置 35](#_Toc437)

[3.3. 扩展功能 36](#_Toc26040)

[3.3.1. 可显示、设定日期功能 36](#_Toc19998)

[3.3.2. 防偷窥密码的虚位密码功能 37](#_Toc16076)

[4. 系统功能实现与调试 37](#_Toc31122)

[4.1. 各种功能展示与调试 37](#_Toc17280)

[5. 总结 41](#_Toc21738)

[5.1. 叙述 41](#_Toc31039)

[5.2. 参考文献 45](#_Toc20466)

[附录：程序清单及注释。 45](#_Toc22643)

[程序源代码展示 46](#_Toc21013)

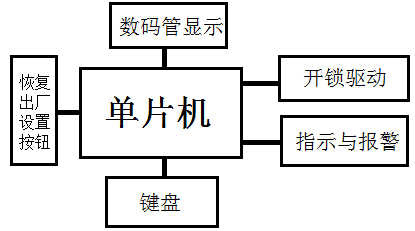
**电子密码锁的设计**

1. **设计目的**
   1. 目的

通过单片机应用产品的设计与调试过程，巩固《单片机技术应用》课程所学理论知识，初步了解单片机应用系统设计与调试的方法。

1. **设计任务**
   1. **基本设计要求**

设计一个以AT89S51单片机为核心的电子密码锁控制器，用键盘实现6位密码的输入开锁、密码设置、时间修改等操作；用8位数码管作为操作过程的状态显示，平时显示时钟；用蜂鸣器和LED给出相关状态指示与报警。系统结构框图如下：

图2-1 系统结构框图

查找资料，参照实际产品功能完成系统设计，并在单片机实验板上模拟调试实现控制器的功能。具体设计要求如下：

* + 1. **开机自检**

上电时通过程序控制检查相关接口及数码管显示器、指示灯、蜂鸣器等外设是否正常。

* + 1. **平时显示时间**

8位数码管显示器平时显示时间，可通过按键设定时间。

* + 1. **行列式键盘**

键盘用行列式键盘，按键定义如右图所示，键盘扫描还应采取消抖动处理。主要的按键有：数字键“0”~“9”，用于输入时间数值和密码数字，正常工作时无效。“\*”、“#”键用于显示切换、进入相应功能和设置状态。

* + 1. **时钟设定**

时钟设定功能：在显示时钟的基本显示界面状态下按1下“#”键则进入时钟设定界面，动态调取并只显示当前实时钟的小时和分钟值（随着走时变）；从高位开始可逐位巡回输入新的时、分值，当前数字位以2Hz频率闪烁显示的形式标示；一旦有输入新数字则停止动态调取实时钟的时、分值，即显示的时、分值不再跟随内部实时钟走时的变化；期间若按下“\*”键则撤销本次修改返回基本显示界面；若按下“#”键则确认本次修改，将新的时、分值更替到实时钟相应单元（时钟设定应当有超值域判别及处置），返回基本显示界面。

* + 1. **密码开锁**

密码开锁功能：在显示时钟的基本显示界面状态下按下“\*”键则进入密码开锁状态，输入密码不能直接显示出来，输入密码后按“#”键，正确可以开锁（假设开锁驱动为I/O口控制三极管驱动电磁执行机构的线圈，调试时用LED模拟，动作时间为0.5秒），输入错误，有相应提示；连续输入3次错误，警告并锁定，一定时间后才允许再输入。

* + 1. **密码设置**

密码设置功能：在显示时钟的基本显示界面状态下按2下“#”键则进入密码设定界面，输入原密码，按“#”键，正确的话可以输入新密码，按“#”键，二次确认输入密码，按“#”键。若过程成功或错误都有相应提示。过程中按“\*”键则放弃本次修改。

* + 1. **恢复出厂设置**

通过设置在内部隐蔽位置的按钮实现，按下后恢复初始的密码和时间。

* 1. **扩展功能（选做）**
     1. **显示、设定日期功能**
     2. **设定多组密码**
     3. **防偷窥密码的虚位密码功能**

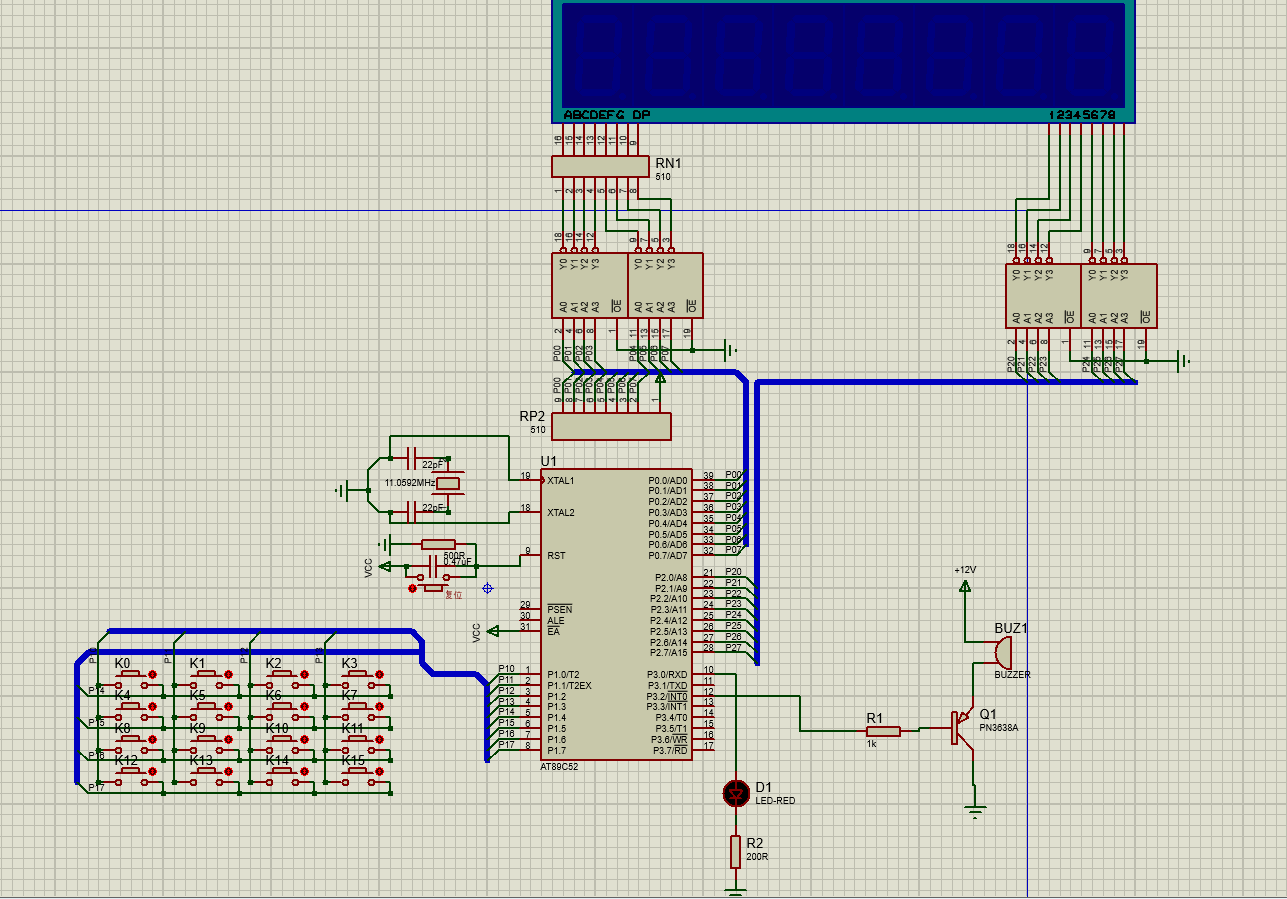
（虚位密码就是在正确的密码前面和后面加上任意位数的数字。也可以设置为正确密码前后随机输入同数量的数字。比如前面随机输入四个数字，则正确号码后，也需输入四位数字，不输数字或输入其他数量的数字，都无法开锁。）

* + 1. **其它自选的扩展功能**

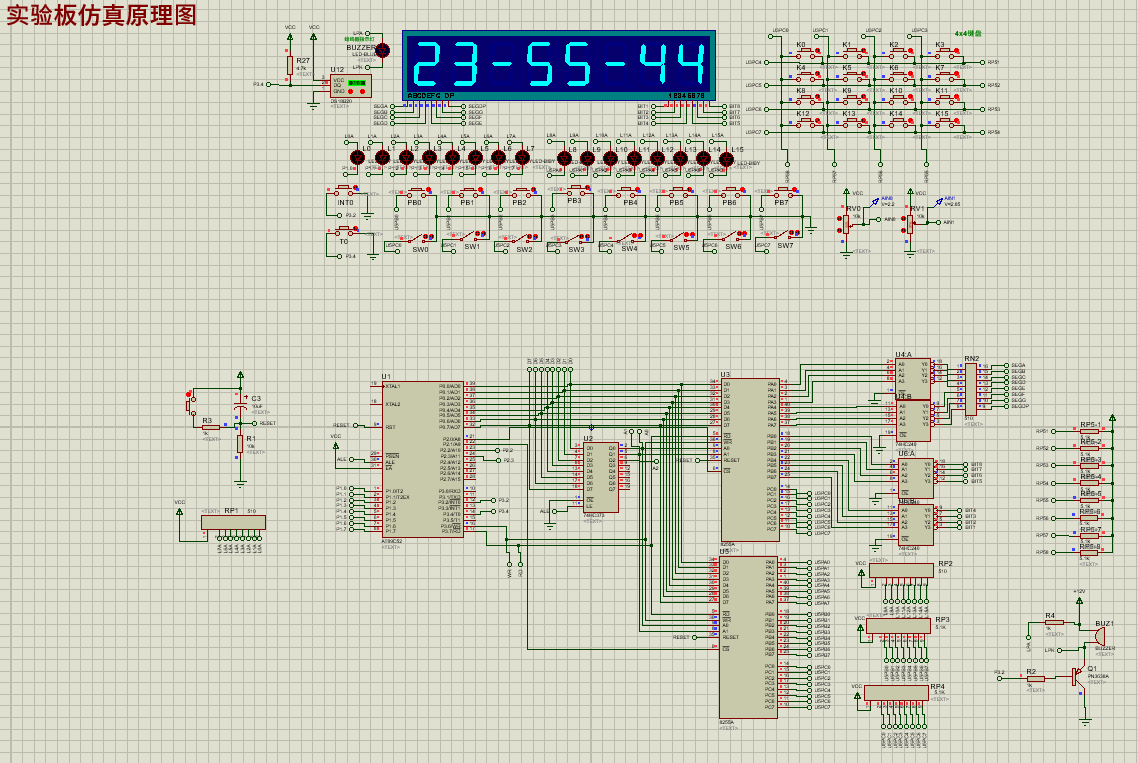
1. **系统设计**
   1. **硬件设计**

硬件在于仿真硬件和开发板硬件两个平台，在仿真硬件上，由一个块AT89C52、四块74HC240驱动液晶显示器、一块RX8排阻、一块RESPACK-8上拉排阻、自扫描矩阵按键模块、蜂鸣器模块、复位电路、晶振电路组成。在开发板上多了AT89C52IO扩展芯片的电路。

下面是仿真电路图展示：

图3-1 仿真电路原理图

下面是开发板适配的仿真电路展示（与开发板原理图适配）：

图3-2 开发板原理图

* 1. **软件设计**
     1. **开机自检**

设置菜单状态变量，并设置开机自检显示的位置变量zijianpos

*// 菜单*

unsigned char menu = MENU\_TIME;

unsigned char edit = 0;

unsigned char zijianpos = 0;

对于初始化部分，先初始化定时器，这将0号定时器设定为时间定时器，1号定时器没有使用，不用开启：

void **timer\_init**() {

    TMOD = 0x22; *// T1 方式2、定时、GATE=0；T0 方式2、定时，GATE=0*

    TH1 = 0x00;

    TL1 = 0x00; *//置T1计数初值0,计数256，每次中断定时256x12/11.0592 us*

    TH0 = 0x00;

    TL0 = 0x00; *//置T1计数初值0,计数256，每次中断定时256x12/11.0592 us*

    EA = 1; *//开中断总允许*

*//ET1 = 1;    // T1中断允许*

*//TR1 = 1;    //启动T1定时(时钟是否走时开关 1/0)！！！*

    ET0 = 1; *// T1中断允许*

    TR0 = 1; *//启动T1定时(时钟是否走时开关 1/0)！！！*

}

初始化8255A的U3CON，这里用宏来切换仿真电路上的代码编译和开发板上的代码编译：

#ifdef **U3CON**

U3CON = 0x89; *// 8255(U3)初始化，A、B口方式0输出，C口输入*

#endif

上一段的初始化，是基于定义外围接口的宏常量下才进行的，这里用宏来切换代码编译的版本：

#if 0

#define **DISP\_SEG** P0 *//扩展接口8255(U3)PA口地址,数码管字段输出口*

#define **DISP\_BIT** P2 *//扩展接口8255(U3)PB口地址,数码管字位输出口*

#define **VKEY\_RL** P1 *//扩展接口8255(U3)PC口地址,矩阵键盘行、列接口*

#else

#define **DISP\_SEG** XBYTE[0xfefc] *//扩展接口8255(U3)PA口地址,数码管字段输出口*

#define **DISP\_BIT** XBYTE[0xfefd] *//扩展接口8255(U3)PB口地址,数码管字位输出口*

#define **VKEY\_RL** XBYTE[0xfefe] *//扩展接口8255(U3)PC口地址,矩阵键盘行、列接口*

#define **U3CON** XBYTE[0xfeff] *//扩展接口8255(U3)控制口地址*

#endif

接着初始化时间和菜单，进入自检菜单，延迟60ms的原因是等待外围电路准备完毕：

**timer\_init**();

**delay\_ms**(60);

    hour = 23;

    min = 55;

    sec = 20;

    gear = 0;

    menu = MENU\_ZIJIAN;

在主循环里刷新显示、响应按键事件：

    while (1) {

*// 刷新显示*

**display\_buf**();

**display\_swbuf**();

*// 键盘检测*

**key\_scan**();

**key\_pro**();

**alarm**();

    }

对于在自检菜单显示自检符号-，由一个循环刷新每一个数码管，在拆字函数display\_buf里实现，用zijianpos控制位置：

    case MENU\_ZIJIAN:

        for (i=0; i<8; i++) {

            if (i == zijianpos) {

                dispBuf[i] = DK\_Del;

            } else dispBuf[i] = DK\_Space;

        }

        break;

对于移动zijianpos，在定时器中每0.25秒（900/3600）向右移动一个位置，下面代表了定时器做的所有工作，其中关键在zijianpos++所在的语句块内，其中play\*用于播放声音，hhFastTime用于设定一个变量计数器，用于计数到位触发语句的执行：

void **timer1\_isr**() interrupt 1 {

    static unsigned int j = 0;

*// 时、分快调计数*

    if (hhFastTime < 3600)

        hhFastTime++;

    blinkTime++;

    if (blinkTime >= 1800)

        blinkTime = 0;

    if(playType != 0) {

        playTimer++;

    }

    j++;

    if (menu == MENU\_ZIJIAN && hhFastTime >= 900) {

        playType = 1;

        playTime = 100;

        playTimer = 0;

        zijianpos++;

        if(zijianpos >= 8) menu = MENU\_TIME;

        hhFastTime = 0;

    }

    if (j >= 3600) {

        j = 0;

        if(gear) {

            if(gearTime <= 0) gear = 0;

            else gearTime--;

        }

        if(triedTimes >= 3) {

            if(triedTimer == 0) triedTimes = 0;

            else triedTimer--;

        }

*//秒、分、时修改\*如按日期规则修改，则可显示日期\**

        sec++;

**updateDateTime**(0, 0);

    }

}

对于显示函数，将拆好的内容从缓冲区显示到数码管，这里控制了位闪烁，由settime、blinkIndex和blinkTime来控制，由宏来切换仿真和开发板代码的编译，结尾再进行显示空格的原因是为了防止后面的数码管浮空输出出现乱码：

void **display\_swbuf**() {

    unsigned int i, j;

#ifdef **U3CON**

    unsigned int disp\_bit\_con = 0x80; *//字位控制码初值*

#else

    unsigned int disp\_bit\_con = 0x01; *//字位控制码初值*

#endif

    DISP\_SEG = 0xff; *//关显示，使各位亮度一致、避免频闪*

    for (i = 0, j = 0; i < 10 && dispBuf[i] != DK\_EOF; i++) {

*// 闪烁位*

        if (!(settime && blinkIndex == i && blinkTime < 900)) {

            if (dispBuf[i] == DK\_NUL) *// 空码忽略*

                continue;

            DISP\_BIT = disp\_bit\_con; *// 8255（U3）的B口输出字位控制码*

            DISP\_SEG = segtable[dispBuf[i]]; *//显缓内容查字段码表得到字段码;*

**delay\_ms**(1); *//延时1ms*

            DISP\_SEG = 0xff; *//关显示，使各位亮度一致、避免频闪*

        }

#ifdef **U3CON**

        disp\_bit\_con >>= 1; *//字位控制码左移一位*

#else

        disp\_bit\_con <<= 1; *//字位控制码左移一位*

#endif

        j++;

    }

    for (; i < 10; i++) {

        DISP\_BIT = disp\_bit\_con; *// 8255（U3）的B口输出字位控制码*

        DISP\_SEG = segtable[DK\_Space]; *//显缓内容查字段码表得到字段码;*

**delay\_ms**(1); *//延时1ms*

        DISP\_SEG = 0xff; *//关显示，使各位亮度一致、避免频闪*

#ifdef **U3CON**

        disp\_bit\_con >>= 1; *//字位控制码左移一位*

#else

        disp\_bit\_con <<= 1; *//字位控制码左移一位*

#endif

    }

}

显示缓存、字符字典、字符段码数据：

unsigned char dispBuf[10] = {0x10};

enum { MENU\_TIME, MENU\_DATE, MENU\_PWD, MENU\_PWD\_CHANGE1, MENU\_PWD\_CHANGE2, MENU\_PWD\_CHANGE3, MENU\_ZIJIAN };

enum {

    DK\_0,

    DK\_1,

    DK\_2,

    DK\_3,

    DK\_4,

    DK\_5,

    DK\_6,

    DK\_7,

    DK\_8,

    DK\_9,

    DK\_A,

    DK\_B,

    DK\_C,

    DK\_D,

    DK\_E,

    DK\_F,

    DK\_Space,

    DK\_Del,

    DK\_UdLine,

    DK\_Dot,

    DK\_D0,

    DK\_D1,

    DK\_D2,

    DK\_D3,

    DK\_D4,

    DK\_D5,

    DK\_D6,

    DK\_D7,

    DK\_D8,

    DK\_D9,

    DK\_DA,

    DK\_DB,

    DK\_DC,

    DK\_DD,

    DK\_DE,

    DK\_DF,

    DK\_NUL,

    DK\_EOF,

    DK\_P,

    DK\_Equal,

    DK\_Rectangle

};

unsigned char code segtable[] = {

    0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90, *// 0~9字段码*

    0x88, 0x83, 0xc6, 0xa1, 0x86, 0x8e, 0xff, 0xbf, 0xf7, 0x7f, *// A~F,空格,-,\_,.字段码*

    0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78, 0x00, 0x10, *// 0.~9.字段码*

    0x08, 0x03, 0x46, 0x21, 0x06, 0x0e, *// A.~F.字段码*

    0xcc, 0xee, 0x8c, 0xb7, 0xa3 *// 空码,结束码,P,=,正方形*

};

* + 1. **显示时间主界面**

8位数码管显示器平时显示时间，可通过按键设定时间。

首先在时间菜单状态对时间拆字，分为settime和非settime的原因是当设定时间的时候显示的是设置的值，当显示的时候显示实际的时间，有利于时间设定更独立，可以取消更改：

    case MENU\_TIME:

        if(settime == 1) {

*// 时*

            dispBuf[0] = tempA / 10;

            dispBuf[1] = tempA % 10;

            dispBuf[2] = DK\_Del;

*// 分*

            dispBuf[3] = tempB / 10;

            dispBuf[4] = tempB % 10;

            dispBuf[5] = DK\_Del;

*// 秒*

            dispBuf[6] = tempC / 10;

            dispBuf[7] = tempC % 10;

            dispBuf[8] = DK\_EOF;

        } else {

*// 时*

            dispBuf[0] = hour / 10;

            dispBuf[1] = hour % 10;

            dispBuf[2] = DK\_Del;

*// 分*

            dispBuf[3] = min / 10;

            dispBuf[4] = min % 10;

            dispBuf[5] = DK\_Del;

*// 秒*

            dispBuf[6] = sec / 10;

            dispBuf[7] = sec % 10;

            dispBuf[8] = DK\_EOF;

        }

        break;

拆字完就是显示，前面已经有显示的代码，这里不作展示。下面是设定时间，在按键扫描的执行函数里，由一个settime变量控制，绑定在K15按下事件内，如果是单击，在时间页面就设置时间，在日期页面就设置日期，如果是双击，进入设置密码菜单，单双击事件由一个sgClick是否单击过和hhFastTime计数器变量控制，这个计数器有自检计数器、单双击计数器两个作用。settime为1设置时间，为2设定日期，其中updateDateTime用于更新输入后的值显示settime为1或2时自动显示tempA/B/C对应的时间变量，设定完成后放回原时间变量。

int **key\_pro\_zixuan**() {

    unsigned char key;

*// 按下*

    if ((scan\_buf[0] == 16) && (scan\_buf[1] != 16)) {

        playType = 1;

        playTime = 200;

        playTimer = 0;

        switch (scan\_buf[1]) {

        case 13:

            if(settime) settime = 0;

            else {

                menu++;

                if (menu > MENU\_PWD)

                    menu = MENU\_TIME;

                if (menu == MENU\_PWD)

                    inputLen = 0;

            }

            return 1;

        case 15:

            if(settime == 1) { *// 更新时间*

                hour = tempA;

                min = tempB;

                sec = tempC;

                settime = 0;

            } else if(settime == 2) { *// 更新日期*

                year = tempA;

                month = tempB;

                day = tempC;

                settime = 0;

            } else if(hhFastTime < 1800) { *// 0.2s*

*// 双击事件*

                menu = MENU\_PWD\_CHANGE1;

                inputLen = 0;

                return 1;

            } else {

                if(menu == MENU\_TIME) {

                    settime = 1; *// 设置时间*

                    tempA = hour;

                    tempB = min;

                    tempC = sec;

                } else if(menu == MENU\_DATE) {

                    settime = 2; *// 设置日期*

                    tempA = year;

                    tempB = month;

                    tempC = day;

                }

            }

        }

        if(settime) {

            if (1 <= scan\_buf[1] && scan\_buf[1] <= 3) {

                key = scan\_buf[1];

            } else if (5 <= scan\_buf[1] && scan\_buf[1] <= 7) {

                key = scan\_buf[1] - 1;

            } else if (9 <= scan\_buf[1] && scan\_buf[1] <= 11) {

                key = scan\_buf[1] - 2;

            } else if (13 <= scan\_buf[1] && scan\_buf[1] <= 15) {

                key = scan\_buf[1] - 3;

            }

            if (key < 10 || key == 11) {

                if (key == 11)

                    key = 0;

                switch (blinkIndex) {

                case 0:

                    if(settime == 1) {

*// 设置时十位*

                        tempA = key \* 10 + tempA % 10;

                    } else {

*// 设置年千位*

                        tempA = tempA - (tempA / 1000 \* 1000) + key \* 1000;

                    }

**updateDateTime**(settime, blinkIndex);

                    blinkIndex++;

                    break;

                case 1:

                    if(settime == 1) {

*// 设置时个位*

                        tempA = (tempA / 10) \* 10 + key;

**updateDateTime**(settime, blinkIndex);

                        blinkIndex++;

                    } else {

*// 设置年百位*

                        tempA = (tempA / 1000 \* 10 + key) \* 100 + (tempA - tempA / 100 \* 100);

**updateDateTime**(settime, blinkIndex);

                    }

                    blinkIndex++;

                    break;

                case 2:

                    if(settime == 1) {

                        ;

                    } else {

*// 设置年十位*

                        tempA = (tempA / 100 \* 10 + key) \* 10 + tempA % 10;

                    }

**updateDateTime**(settime, blinkIndex);

                    blinkIndex++;

                    break;

                case 3:

                    if(settime == 1) {

*// 设置分十位*

                        tempB = key \* 10 + tempB % 10;

                    } else {

*// 设置年个位*

                        tempA = tempA / 10 \* 10 + key;

                    }

**updateDateTime**(settime, blinkIndex);

                    blinkIndex++;

                    break;

                case 4:

                    if(settime == 1) {

*// 设置分个位*

                        tempB = (tempB / 10) \* 10 + key;

**updateDateTime**(settime, blinkIndex);

                        blinkIndex++;

                    } else {

*// 设置月十位*

                        tempB = key \* 10 + tempB % 10;

**updateDateTime**(settime, blinkIndex);

                    }

                    blinkIndex++;

                    break;

                case 5:

                    if(settime == 1) {

                        ;

                    } else {

*// 设置月个位*

                        tempB = tempB / 10 \* 10 + key;

                    }

**updateDateTime**(settime, blinkIndex);

                    blinkIndex++;

                    break;

                case 6:

                    if(settime == 1) {

*// 设置秒十位*

                        tempC = key \* 10 + tempC % 10;

                    } else {

*// 设置日十位*

                        tempC = key \* 10 + tempC % 10;

                    }

**updateDateTime**(settime, blinkIndex);

                    blinkIndex++;

                    break;

                case 7:

                    if(settime == 1) {

*// 设置秒个位*

                        tempC = (tempC / 10) \* 10 + key;

                    } else {

*// 设置日个位*

                        tempC = tempC / 10 \* 10 + key;

                    }

**updateDateTime**(settime, blinkIndex);

                    blinkIndex = 0;

                    break;

                }

            }

            if (scan\_buf[1] == 0) {

*// 向左移动，设置时、十位*

                blinkIndex--;

                if (settime == 1) {

                    if (blinkIndex == 2 || blinkIndex == 5)

                        blinkIndex--;

                }

            }

            if (scan\_buf[1] == 4) {

*// 向右移动，设置个位、分*

                blinkIndex++;

                if (settime == 1) {

                    if (blinkIndex == 2 || blinkIndex == 5)

                        blinkIndex++;

                }

            }

            if (blinkIndex == 8)

*// 归位光标*

                blinkIndex = 0;

            if (blinkIndex == -1)

*// 光标移至末尾*

                blinkIndex = 7;

        }

    }

*// 松开*

    if((scan\_buf[0] != 0x10) && (scan\_buf[1] == 0x10)) {

        switch(scan\_buf[0]) {

            case 15:

*// PB2 单双击LED松开*

                if(!sgClick) {

                    sgClick = 1;

                    hhFastTime = 0; *// 计时清零*

                } else

                    sgClick = 0;

        }

    }

    return 0;

}

* + 1. **行列式键盘**

按键定义如右图所示，键盘扫描还应采取消抖动处理。主要的按键有：数字键“0”~“9”，用于输入时间数值和密码数字，正常工作时无效。“\*”、“#”键用于显示切换、进入相应功能和设置状态。

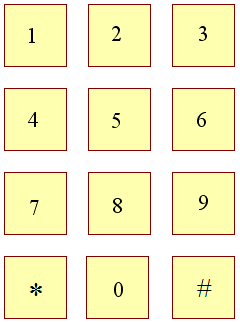
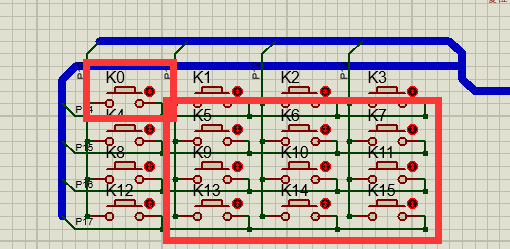
****

图3-1 按键定义

这里我用K5右下角部分作为按键区域，K13、K15对应\*和#来用于切换菜单、取消、设置时间日期、设置密码的功能。其他0~9按键作为数字按键一一对应，K0作为密码锁内部的恢复出厂模式按钮。

图3-2 电路上按键定义

矩阵按键的扫描部分，使用了滚动消抖技术、新态旧态存储技术来过滤杂波和记录按键按下、放开的事件，滚动消抖后还要滤波，对确定按下的按键进行筛选：

*// 矩阵键盘扫描*

void **key\_scan**() {

    unsigned int i;

    unsigned char line; *//行输入*

    unsigned char row; *//列输入*

*// 矩阵按键扫描*

*// 滚动存储*

    scan\_buf[0] = scan\_buf[1]; *//新态、旧态滚动存储*

    scan\_buf[2] = scan\_buf[3]; *//滤波单元滚动存储*

    scan\_buf[3] = scan\_buf[4];

#ifdef **U3CON**

*// 输入新键值*

*//DISP\_SEG = 0xff;          //关显示*

    U3CON = 0x81; *//行（PC7-PC4）为输出，列(PC3-PC0)为输入*

    VKEY\_RL = 0x00; *//行输出0*

    row = VKEY\_RL;

*//DISP\_SEG = 0xff;          //关显示*

    U3CON = 0x88; *//行（PC7-PC4）为输入，列(PC3-PC0)为输出*

    VKEY\_RL = 0x00; *//列输出0*

    line = VKEY\_RL;

#else

    DISP\_SEG = 0xff; *//关显示*

    VKEY\_RL = 0x0f; *//行输出0*

**delay\_ms**(1);

    row = VKEY\_RL & 0x0f;

    DISP\_SEG = 0xff; *//关显示*

    VKEY\_RL = 0xf0; *//列输出0*

**delay\_ms**(1);

    line = VKEY\_RL & 0xf0;

#endif

    row = row & 0x0f; *//屏蔽高4位*

    line = line & 0xf0; *// 屏蔽低4位*

    scan\_buf[5] = line | row; *//合并行列值*

*// 键盘编码-->键值存入scan\_buf[4]*

    for (i = 0; i < 16; i++) {

        if (scan\_buf[5] == Scan\_code[i]) {

            scan\_buf[4] = i; *//查键盘扫描码，转换成键值0-15*

            break;

        } else

            scan\_buf[4] = 16; *//其它情况为16*

    }

*// 滤波*

    if ((scan\_buf[2] == scan\_buf[3]) && (scan\_buf[2] == scan\_buf[4]))

        scan\_buf[1] = scan\_buf[4];

}

* + 1. **时钟设定功能**

均在上面矩阵按键扫描代码实现，按一次#进入事件、日期修改菜单，通过一个变量blinkIndex来控制巡回设置，同时blinkIndex和blinkTime控制数码管位的2Hz闪烁。设置时间时使用temp\*的时间，设置的时候显示不会走时，但时间正常计时。设置期间按下\*会取消设置时间，继续显示原来的值。按下#覆盖时间，确认修改，每更改一个位就判断超时域，并返回基本界面。

下面是判断超时域的代码，在更新时间日期函数内，一旦某个值超过正常值域，就会进位：

void **updateDateTime**(unsigned char set, unsigned char index)

{

    switch(set) {

        case 0:

        if (sec >= 60) {

            sec = 0;

            min++;

            if (min >= 60) {

                min = 0;

                hour++;

                if (hour >= 24) {

                    hour = 0;

                    day++;

                    if(day >= **getDayOfMonth**(year, month) + 1) {

                        day = 1;

                        month++;

                        if(month >= 13) {

                            month = 1;

                            year++;

                            if(year >= 9999) {

                                year = 2000;

                            }

                        }

                    }

                }

            }

        }

        break;

        case 1:

            switch(index) {

                case 7: case 6:

                    if (tempC >= 60) {

                        tempC = 0;

                        tempB++;

                    }

                case 4:case 3:

                    if (tempB >= 60) {

                        tempB = 0;

                        tempA++;

                    }

                case 1:case 0:

                    if (tempA >= 24) {

                        tempA = 0;

                    }

            }

        break;

        case 2:

            switch(index) {

                case 7:case 6:case 5:case 4:

                    if (tempC >= **getDayOfMonth**(tempA, tempB) + 1) {

                        tempC = 1;

                        tempB++;

                    }

                case 3:case 2:

                    if (tempB >= 13) {

                        tempB = 1;

                        tempA++;

                    }

                case 1:case 0:

                    if (tempA >= 9999) {

                        tempA = 0;

                    }

            }

        break;

    }

}

下面是得到某个月天数和判断闰年的方法：

*// 判断是否是闰年*

unsigned char **isLeapYear**(unsigned int y) {

    return y%400 == 0 || y % 4 == 0 && y %100 != 0;

}

*// 获取某个月的天数*

unsigned char **getDayOfMonth**(unsigned int y, unsigned char m) {

    switch(m) {

        case 1:case 3:case 5:case 7:case 8:case 10:case 12:

            return 31;

        case 2: return **isLeapYear**(y) ? 29:28;

        default:

            return 30;

    }

}

* + 1. **密码开锁功能**

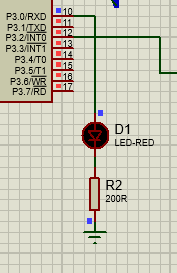
在显示时钟的基本显示界面状态下按下“\*”键则进入密码开锁状态，输入密码不能直接显示出来，输入密码后按“#”键，正确可以开锁（假设开锁驱动为I/O口控制三极管驱动电磁执行机构的线圈，调试时用LED模拟，动作时间为0.5秒），输入错误，有相应提示；连续输入3次错误，警告并锁定，一定时间后才允许再输入

进入开锁菜单，将会提示输入密码：

图3-3 输入密码

输入的密码不能直接显示出来，用方框表示，输入完6位密码，按#号键确认密码，判断密码正确后开锁。

锁模块用LED代替：

图3-4 锁模块

输入密码的代码在矩阵按键处理函数中写过了，这里不展示。

判断密码部分的代码，如果是开锁，正确就开锁，如果是修改密码，进入下一级菜单，输入新密码和确认密码，两次匹配相对才能修改成功：

            if (inputLen == 6) {

                if(menu == MENU\_PWD\_CHANGE1) {

                    for(i = 0; i < 6; i++)

                        if(pwd[i] != input[i]) break;

                    if(i==6) {

                        menu = MENU\_PWD\_CHANGE2;

                        inputLen = 0;

*// 播放输入成功声音*

                        playTime = 300;

                        playTimer = 0;

                        playType = 4;

                    } else {

*// 播放输入错误声音*

                        playTime = 700;

                        playTimer = 0;

                        playType = 3;

                        triedTimes++;

                        if(triedTimes >= 3) {

*// 30秒钟后再输入*

                            triedTimer = 30;

                        }

                    }

                } else if(menu == MENU\_PWD\_CHANGE2) {

                    for(i = 0; i < 6; i++)

                        pwdBack[i] = input[i];

                    menu = MENU\_PWD\_CHANGE3;

                    inputLen = 0;

*// 播放第一次输入成功声音*

                    playTime = 300;

                    playTimer = 0;

                    playType = 4;

                } else if(menu == MENU\_PWD\_CHANGE3) {

                    for(i = 0; i < 6; i++)

                        if(pwdBack[i] != input[i]) break;

                    if(i==6) {

                        for(i = 0; i < 6; i++)

                            pwd[i] = input[i];

                        menu = MENU\_TIME;

                        inputLen = 0;

*// 播放输入成功声音*

                        playTime = 400;

                        playTimer = 0;

                        playType = 4;

                    } else {

                        inputLen = 0;

*// 播放输入错误声音*

                        playTime = 700;

                        playTimer = 0;

                        playType = 3;

                    }

                } else {

                    for(i = 0; i < 6; i++)

                        if(pwd[i] != input[i]) break;

                    if(i == 6) {

                        gear = 1;

                        gearTime = 3; *// 开锁3秒*

                        inputLen = 0;

                        menu = MENU\_TIME;

*// 播放输入成功声音*

                        playTime = 300;

                        playTimer = 0;

                        playType = 4;

                        triedTimes = 0;

                    } else {

*// 播放输入错误声音*

                        playTime = 700;

                        playTimer = 0;

                        playType = 3;

                        triedTimes++;

                        if(triedTimes >= 3) {

*// 30秒钟后再输入*

                            triedTimer = 30;

                        }

                    }

                }

                inputLen = 0;

            }

正确的时候会提示声音，声音播放由以下变量控制：

unsigned char playType = 0; *// 播放类型：0不播放，1按键音，2失败，3成功*

unsigned int playTimer = 0; *// 蜂鸣器响一声计时器*

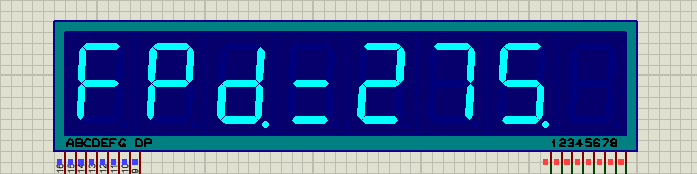
unsigned int playTime = 0;

unsigned char playSound = 0; *// 播放开关*

playTimer的更新在计时器里面完成，这里不展示代码。

播放成功的声音，其中playTime=300说明蜂鸣器声音间隔是300毫秒，playTimer=0初始化蜂鸣器计时器，开锁播放声音，playType=4定义响声次数。正确会有4次短促的声音，错误会有3次长暂的声音。

并且错误三次会警告锁定30秒：

图3-5 错误锁定

* + 1. **密码设定**

在显示时钟的基本显示界面状态下按2下“#”键则进入密码设定界面，输入原密码，按“#”键，正确的话可以输入新密码，按“#”键，二次确认输入密码，按“#”键。若过程成功或错误都有相应提示。过程中按“\*”键则放弃本次修改。

双击部分：

switch (scan\_buf[1]) {

        case 13:

            if(settime) settime = 0;

            else {

                menu++;

                if (menu > MENU\_PWD)

                    menu = MENU\_TIME;

                if (menu == MENU\_PWD)

                    inputLen = 0;

            }

            return 1;

        case 15:

            if(settime == 1) { *// 更新时间*

                hour = tempA;

                min = tempB;

                sec = tempC;

                settime = 0;

            } else if(settime == 2) { *// 更新日期*

                year = tempA;

                month = tempB;

                day = tempC;

                settime = 0;

            } else if(hhFastTime < 1800) { *// 0.2s*

*// 双击事件*

                menu = MENU\_PWD\_CHANGE1;

                inputLen = 0;

                return 1;

            } else {

                if(menu == MENU\_TIME) {

                    settime = 1; *// 设置时间*

                    tempA = hour;

                    tempB = min;

                    tempC = sec;

                } else if(menu == MENU\_DATE) {

                    settime = 2; *// 设置日期*

                    tempA = year;

                    tempB = month;

                    tempC = day;

                }

            }

        }

加上松开按键的部分：

*// 松开*

    if((scan\_buf[0] != 0x10) && (scan\_buf[1] == 0x10)) {

        switch(scan\_buf[0]) {

            case 15:

*// PB2 单双击LED松开*

                if(!sgClick) {

                    sgClick = 1;

                    hhFastTime = 0; *// 计时清零*

                } else

                    sgClick = 0;

        }

    }

修改过程中二次确认密码：



图3-6 二次确认密码

同样也有\*键放弃修改密码。

* + 1. **恢复出厂设置**

通过设置在内部隐蔽位置的按钮实现，按下后恢复初始的密码和时间。

按键K0用于初始化系统，重置各种变量的状态：

            case 0:

*// 初始化时间日期*

                hour = 23;

                min = 55;

                sec = 20;

                gear = 0;

*// 初始化密码*

                pwd[0] = 5;

                pwd[1] = 3;

                pwd[2] = 5;

                pwd[3] = 8;

                pwd[4] = 6;

                pwd[5] = 6;

*// 初始化状态*

                settime = 0;

                triedTimes = 0; *// 尝试连续输入密码的次数*

                triedTimer = 0; *// 输入密码计数器*

                zijianpos = 0;

                playSound = 0;

*// 进入自检菜单*

                menu = MENU\_ZIJIAN;

                break;

* 1. **扩展功能**
     1. **可显示、设定日期功能**

图3-7 显示日期

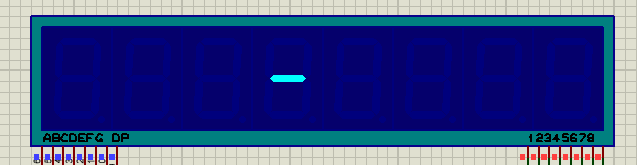
* + 1. **防偷窥密码的虚位密码功能**

（虚位密码就是在正确的密码前面和后面加上任意位数的数字。也可以设置为正确密码前后随机输入同数量的数字。比如前面随机输入四个数字，则正确号码后，也需输入四位数字，不输数字或输入其他数量的数字，都无法开锁。）

图3-8 防偷窥虚位密码

1. **系统功能实现与调试**
   1. **各种功能展示与调试**

开机进入自检状态：

图4-1 开机自检

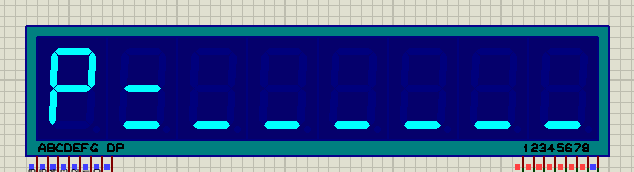
时间显示界面：

图4-2 时间显示主界面

按\*进入日期显示界面：

图4-3 日期显示界面

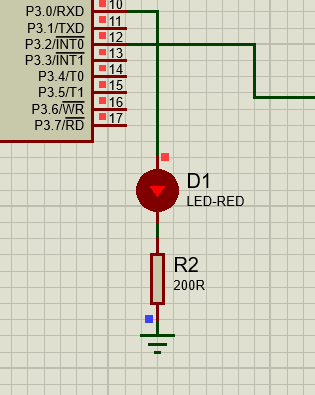
再按\*进入开锁界面，提示输入密码：

图4-4 输入密码开锁

输入失败，播放失败声音，三次失败会锁定30秒：

图4-5 输入错误三次锁定

输入成功，开锁灯亮：

图4-6 开锁灯亮

在时间界面按#修改时间，修改位会闪烁，自动巡回修改：

图4-7 巡回修改时间

按#确认修改，按\*取消：

图4-8 确认修改时间

进入修改日期：

图4-9 修改日期

确认退出：

图4-10 确认修改日期

1. **总结**
   1. **叙述**

总体上，单片机设计分为两部分的代码，一份在仿真器上运行，一份在开发板上运行。这只用通过一个宏#if 0来控制，非常方便。我用了很多的状态变量来完成对菜单的切换 、时间日期的修改、密码修改、错误锁定等不同显示。还添加了很多个计数器变量来完善单个定时器的分工。例如时间更新、按键音、提示音、锁定延时、闪烁位。

在键盘扫描部分，我使用老师的键盘扫描方法发现并不适用，由于我没有用8255A，这个8255A省去了部分的扫描功能，我把外部灌入电流修改为单片机单独输出扫描，同样分为两份扫描代码，一份用于仿真器、一份用于开发板：

#ifdef **U3CON**

*// 输入新键值*

*//DISP\_SEG = 0xff;          //关显示*

    U3CON = 0x81; *//行（PC7-PC4）为输出，列(PC3-PC0)为输入*

    VKEY\_RL = 0x00; *//行输出0*

    row = VKEY\_RL;

*//DISP\_SEG = 0xff;          //关显示*

    U3CON = 0x88; *//行（PC7-PC4）为输入，列(PC3-PC0)为输出*

    VKEY\_RL = 0x00; *//列输出0*

    line = VKEY\_RL;

#else

    DISP\_SEG = 0xff; *//关显示*

    VKEY\_RL = 0x0f; *//行输出0*

**delay\_ms**(1);

    row = VKEY\_RL & 0x0f;

    DISP\_SEG = 0xff; *//关显示*

    VKEY\_RL = 0xf0; *//列输出0*

**delay\_ms**(1);

    line = VKEY\_RL & 0xf0;

#endif

    row = row & 0x0f; *//屏蔽高4位*

    line = line & 0xf0; *// 屏蔽低4位*

    scan\_buf[5] = line | row; *//合并行列值*

对于硬件的描述，只有8255A被省去了，8255A主要是使用了51单片机的xcode部分的扩展地址用来访问扩展接口。通过一个XBYTE宏来访问：

#define **XBYTE** ((unsigned char volatile xdata \*) 0)

若在开发板上编译，则将会这样调用扩展地址：

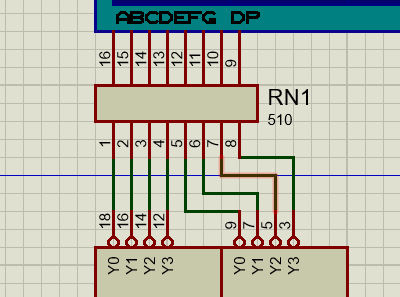
#define **DISP\_SEG** XBYTE[0xfefc] *//扩展接口8255(U3)PA口地址,数码管字段输出口*

#define **DISP\_BIT** XBYTE[0xfefd] *//扩展接口8255(U3)PB口地址,数码管字位输出口*

#define **VKEY\_RL** XBYTE[0xfefe] *//扩展接口8255(U3)PC口地址,矩阵键盘行、列接口*

#define **U3CON** XBYTE[0xfeff] *//扩展接口8255(U3)控制口地址*

仿真器上的74HC240是一种四路反相缓冲器/线路驱动器，这个芯片用来驱动数码管显示，将输入的TTL信号经过内置晶体管非门和三态门反向输出，具有电流微放大功能。高位的排阻作用限流驱动，防止因电路短路烧坏驱动芯片：

图5-1 排阻

对于上拉电阻，是一种提供默认值给外围IO的方法，在单片机输出悬空时可以给定一个确切的电平给外围电路。例如下面的P0就是单片机外部上拉的开漏输出：

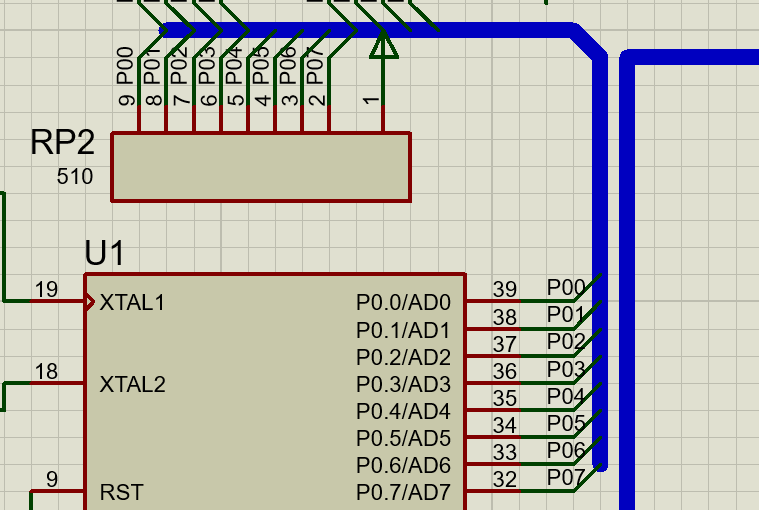


图5-2 外部上拉开漏输出

* 1. **参考文献**

1. Texas Instruments. SN74HC240 Octal Buffers and Line Drivers with 3-State Outputs[E]. 2019.

[2] 韩克、薛迎霄. 单片机应用技术——基于C51和Proteus 的项目设计与仿真[M]. 北京：清华大学出版社，2017.

[3] 林红、张士军、杨桦、杨凡、周鑫霞. 数字电路与逻辑设计[M]. 北京：科学出版社，2014.

[4] 秦雯. 电子技术基础[M]北京：机械工业出版社，2016.

[5] 张毅刚. 单片机原理及接口技术（第2版）[M]. 北京：人民邮电出版社，2015

**附录：程序清单及注释。**

**程序源代码展示**

下面是整个密码锁项目的源代码：

#include <absacc.h>     *// 扩展接口库*

#include <intrins.h>    *// 移位函数库*

#include <reg52.h>      *// 基本89C52库*

*// 此处切换仿真器代码和开发板代码*

#if 0

*// 仿真器代码*

#define **DISP\_SEG** P0 *//扩展接口8255(U3)PA口地址,数码管字段输出口*

#define **DISP\_BIT** P2 *//扩展接口8255(U3)PB口地址,数码管字位输出口*

#define **VKEY\_RL** P1 *//扩展接口8255(U3)PC口地址,矩阵键盘行、列接口*

#else

*// 开发板代码*

#define **DISP\_SEG** XBYTE[0xfefc] *//扩展接口8255(U3)PA口地址,数码管字段输出口*

#define **DISP\_BIT** XBYTE[0xfefd] *//扩展接口8255(U3)PB口地址,数码管字位输出口*

#define **VKEY\_RL** XBYTE[0xfefe] *//扩展接口8255(U3)PC口地址,矩阵键盘行、列接口*

#define **U3CON** XBYTE[0xfeff] *//扩展接口8255(U3)控制口地址*

#endif

unsigned char gearTime = 0; *// 门锁时间（秒）*

sbit gear = P3 ^ 0;

sbit beep = P3 ^ 2;

*// sbit key6 = P3^2;*

*// 显示缓存*

unsigned char dispBuf[10] = {0x10};

enum { MENU\_TIME, MENU\_DATE, MENU\_PWD, MENU\_PWD\_CHANGE1, MENU\_PWD\_CHANGE2, MENU\_PWD\_CHANGE3, MENU\_ZIJIAN };

*// 字符字典*

enum {

    DK\_0,

    DK\_1,

    DK\_2,

    DK\_3,

    DK\_4,

    DK\_5,

    DK\_6,

    DK\_7,

    DK\_8,

    DK\_9,

    DK\_A,

    DK\_B,

    DK\_C,

    DK\_D,

    DK\_E,

    DK\_F,

    DK\_Space,

    DK\_Del,

    DK\_UdLine,

    DK\_Dot,

    DK\_D0,

    DK\_D1,

    DK\_D2,

    DK\_D3,

    DK\_D4,

    DK\_D5,

    DK\_D6,

    DK\_D7,

    DK\_D8,

    DK\_D9,

    DK\_DA,

    DK\_DB,

    DK\_DC,

    DK\_DD,

    DK\_DE,

    DK\_DF,

    DK\_NUL,

    DK\_EOF,

    DK\_P,

    DK\_Equal,

    DK\_Rectangle

};

*// 字符字段*

unsigned char code segtable[] = {

    0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8, 0x80, 0x90, *// 0~9字段码*

    0x88, 0x83, 0xc6, 0xa1, 0x86, 0x8e, 0xff, 0xbf, 0xf7, 0x7f, *// A~F,空格,-,\_,.字段码*

    0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78, 0x00, 0x10, *// 0.~9.字段码*

    0x08, 0x03, 0x46, 0x21, 0x06, 0x0e, *// A.~F.字段码*

    0xcc, 0xee, 0x8c, 0xb7, 0xa3 *// 空码,结束码,P,=,正方形*

};

unsigned char Scan\_code[] = {0xee, 0xed, 0xeb, 0xe7, 0xde, 0xdd, 0xdb, 0xd7,

                             0xbe, 0xbd, 0xbb, 0xb7, 0x7e, 0x7d, 0x7b, 0x77}; *//键盘扫描码*

unsigned char scan\_buf[6] = {0x10, 0x10, 0x10, 0x10, 0x10, 0x10}; *//键盘滚动滤波缓冲单元*

*// 密码*

unsigned char pwd[6] = {5, 3, 5, 8, 6, 6}; *// 默认密码*

unsigned char input[6], pwdBack[6]; *// 输入密码、新密码*

unsigned char inputLen = 0; *// 输入密码的长度*

unsigned char triedTimes = 0; *// 尝试连续输入密码的次数*

unsigned char triedTimer = 0; *// 输入密码计数器*

unsigned int temp; *// 临时变量*

*// 菜单*

unsigned char menu = MENU\_TIME; *// 菜单状态变量*

unsigned char zijianpos = 0; *// 自检显示位置变量*

unsigned char settime = 0; *// 设置时间 ，0：不设置，1：设置时间，2：设置日期*

unsigned char sgClick = 0; *// 是否单击*

unsigned char hFast = 0; *// 时快调按下*

unsigned long hhFastTime = 0; *// 时快调计数*

unsigned int blinkTime = 0; *// 矩阵按键设置时间的光标闪烁计时器*

unsigned char blinkIndex = 0; *// 矩阵按键设置时间的光标闪烁位置*

unsigned char playType = 0; *// 播放类型：0不播放，1按键音，2失败，3成功*

unsigned int playTimer = 0; *// 蜂鸣器响一声计时器*

unsigned int playTime = 0; *// 播放次数*

unsigned char playSound = 0; *// 播放开关*

*// 时间日期*

unsigned int year = 2022;

unsigned char month = 1;

unsigned char day = 1;

unsigned char sec = 0;

unsigned char min = 0;

unsigned char hour = 0;

unsigned int tempA, tempB, tempC; *// 用来临时存储时间日期*

*// 延时函数，毫秒单位*

void **delay\_ms**(unsigned int j) {

    unsigned int k;

    for (; j > 0; j--)

        for (k = 0; k < 100; k++)

            ;

}

*// 拆字，在每个菜单分别放入显示缓存*

void **display\_buf**() {

    int i;

    switch (menu) {

    case MENU\_TIME:

        if(settime == 1) {

*// 时*

            dispBuf[0] = tempA / 10;

            dispBuf[1] = tempA % 10;

            dispBuf[2] = DK\_Del;

*// 分*

            dispBuf[3] = tempB / 10;

            dispBuf[4] = tempB % 10;

            dispBuf[5] = DK\_Del;

*// 秒*

            dispBuf[6] = tempC / 10;

            dispBuf[7] = tempC % 10;

            dispBuf[8] = DK\_EOF;

        } else {

*// 时*

            dispBuf[0] = hour / 10;

            dispBuf[1] = hour % 10;

            dispBuf[2] = DK\_Del;

*// 分*

            dispBuf[3] = min / 10;

            dispBuf[4] = min % 10;

            dispBuf[5] = DK\_Del;

*// 秒*

            dispBuf[6] = sec / 10;

            dispBuf[7] = sec % 10;

            dispBuf[8] = DK\_EOF;

        }

        break;

    case MENU\_DATE:

        if(settime == 2) {

*// 年*

            dispBuf[0] = tempA / 1000;

            dispBuf[1] = (tempA - dispBuf[0] \* 1000) / 100;

            dispBuf[2] = (tempA - tempA / 100 \* 100) / 10;

            dispBuf[3] = DK\_D0 + tempA % 10;

*// 月*

            dispBuf[4] = tempB / 10;

            dispBuf[5] = DK\_D0 + tempB % 10;

*// 日*

            dispBuf[6] = tempC / 10;

            dispBuf[7] = tempC % 10;

            dispBuf[8] = DK\_EOF;

        } else {

*// 年*

            dispBuf[0] = year / 1000;

            dispBuf[1] = (year - dispBuf[0] \* 1000) / 100;

            dispBuf[2] = (year - year / 100 \* 100) / 10;

            dispBuf[3] = DK\_D0 + year % 10;

*// 月*

            dispBuf[4] = month / 10;

            dispBuf[5] = DK\_D0 + month % 10;

*// 日*

            dispBuf[6] = day / 10;

            dispBuf[7] = day % 10;

            dispBuf[8] = DK\_EOF;

        }

        break;

    case MENU\_PWD: case MENU\_PWD\_CHANGE1: case MENU\_PWD\_CHANGE2: case MENU\_PWD\_CHANGE3:

        if(triedTimes >= 3) {

            dispBuf[0] = DK\_F;

            dispBuf[1] = DK\_P;

            dispBuf[2] = DK\_DD;

            dispBuf[3] = DK\_Equal;

            dispBuf[4] = triedTimer / 10;

            dispBuf[5] = triedTimer % 10;

            dispBuf[6] = DK\_D5;

            dispBuf[7] = DK\_Space;

        } else {

            if(menu == MENU\_PWD || menu == MENU\_PWD\_CHANGE1) {

                dispBuf[0] = DK\_P;

                dispBuf[1] = DK\_Equal;

            } else if(menu == MENU\_PWD\_CHANGE2) {

                dispBuf[0] = DK\_P;

                dispBuf[1] = DK\_D1;

            } else if(menu == MENU\_PWD\_CHANGE3) {

                dispBuf[0] = DK\_P;

                dispBuf[1] = DK\_D2;

            }

            for (i = 2; i < 8; i++) {

                if (i - 2 >= inputLen)

                   break;

                if (menu == MENU\_PWD\_CHANGE2 || menu == MENU\_PWD\_CHANGE3)

                    dispBuf[i] = input[i - 2];

                else

                    dispBuf[i] = DK\_Rectangle;

            }

            for (; i < 8; i++)

                dispBuf[i] = DK\_UdLine;

            dispBuf[8] = DK\_NUL;

        }

        break;

    case MENU\_ZIJIAN:

        for (i=0; i<8; i++) {

            if (i == zijianpos) {

                dispBuf[i] = DK\_Del;

            } else dispBuf[i] = DK\_Space;

        }

        break;

    }

}

*// 显示缓存到数码管*

void **display\_swbuf**() {

    unsigned int i, j;

#ifdef **U3CON**

    unsigned int disp\_bit\_con = 0x80; *//字位控制码初值*

#else

    unsigned int disp\_bit\_con = 0x01; *//字位控制码初值*

#endif

    DISP\_SEG = 0xff; *//关显示，使各位亮度一致、避免频闪*

    for (i = 0, j = 0; i < 10 && dispBuf[i] != DK\_EOF; i++) {

*// 闪烁位*

        if (!(settime && blinkIndex == i && blinkTime < 900)) {

            if (dispBuf[i] == DK\_NUL) *// 空码忽略*

                continue;

            DISP\_BIT = disp\_bit\_con; *// 8255（U3）的B口输出字位控制码*

            DISP\_SEG = segtable[dispBuf[i]]; *//显缓内容查字段码表得到字段码;*

**delay\_ms**(1); *//延时1ms*

            DISP\_SEG = 0xff; *//关显示，使各位亮度一致、避免频闪*

        }

#ifdef **U3CON**

        disp\_bit\_con >>= 1; *//字位控制码左移一位*

#else

        disp\_bit\_con <<= 1; *//字位控制码左移一位*

#endif

        j++;

    }

    for (; i < 10; i++) {

        DISP\_BIT = disp\_bit\_con; *// 8255（U3）的B口输出字位控制码*

        DISP\_SEG = segtable[DK\_Space]; *//显缓内容查字段码表得到字段码;*

**delay\_ms**(1); *//延时1ms*

        DISP\_SEG = 0xff; *//关显示，使各位亮度一致、避免频闪*

#ifdef **U3CON**

        disp\_bit\_con >>= 1; *//字位控制码左移一位*

#else

        disp\_bit\_con <<= 1; *//字位控制码左移一位*

#endif

    }

}

*// 计时器初始化*

void **timer\_init**() {

    TMOD = 0x22; *// T1 方式2、定时、GATE=0；T0 方式2、定时，GATE=0*

    TH1 = 0x00;

    TL1 = 0x00; *//置T1计数初值0,计数256，每次中断定时256x12/11.0592 us*

    TH0 = 0x00;

    TL0 = 0x00; *//置T1计数初值0,计数256，每次中断定时256x12/11.0592 us*

    EA = 1; *//开中断总允许*

*//ET1 = 1;    // T1中断允许*

*//TR1 = 1;    //启动T1定时(时钟是否走时开关 1/0)！！！*

    ET0 = 1; *// T1中断允许*

    TR0 = 1; *//启动T1定时(时钟是否走时开关 1/0)！！！*

}

*// 判断是否是闰年*

unsigned char **isLeapYear**(unsigned int y) {

    return y%400 == 0 || y % 4 == 0 && y %100 != 0;

}

*// 获取某个月的天数*

unsigned char **getDayOfMonth**(unsigned int y, unsigned char m) {

    switch(m) {

        case 1:case 3:case 5:case 7:case 8:case 10:case 12:

            return 31;

        case 2: return **isLeapYear**(y) ? 29:28;

        default:

            return 30;

    }

}

*// 更新时间处理函数*

void **updateDateTime**(unsigned char set, unsigned char index)

{

    switch(set) {

        case 0:

        if (sec >= 60) {

            sec = 0;

            min++;

            if (min >= 60) {

                min = 0;

                hour++;

                if (hour >= 24) {

                    hour = 0;

                    day++;

                    if(day >= **getDayOfMonth**(year, month) + 1) {

                        day = 1;

                        month++;

                        if(month >= 13) {

                            month = 1;

                            year++;

                            if(year >= 9999) {

                                year = 2000;

                            }

                        }

                    }

                }

            }

        }

        break;

        case 1:

            switch(index) {

                case 7: case 6:

                    if (tempC >= 60) {

                        tempC = 0;

                        tempB++;

                    }

                case 4:case 3:

                    if (tempB >= 60) {

                        tempB = 0;

                        tempA++;

                    }

                case 1:case 0:

                    if (tempA >= 24) {

                        tempA = 0;

                    }

            }

        break;

        case 2:

            switch(index) {

                case 7:case 6:case 5:case 4:

                    if (tempC >= **getDayOfMonth**(tempA, tempB) + 1) {

                        tempC = 1;

                        tempB++;

                    }

                case 3:case 2:

                    if (tempB >= 13) {

                        tempB = 1;

                        tempA++;

                    }

                case 1:case 0:

                    if (tempA >= 9999) {

                        tempA = 0;

                    }

            }

        break;

    }

}

*// 定时器0中断处理函数*

void **timer1\_isr**() interrupt 1 {

    static unsigned int j = 0;

*// 时、分快调计数*

    if (hhFastTime < 3600)

        hhFastTime++;

    blinkTime++;

    if (blinkTime >= 1800)

        blinkTime = 0;

    if(playType != 0) {

        playTimer++;

    }

    j++;

    if (menu == MENU\_ZIJIAN && hhFastTime >= 900) {

        playType = 1;

        playTime = 100;

        playTimer = 0;

        zijianpos++;

        if(zijianpos >= 8) menu = MENU\_TIME;

        hhFastTime = 0;

    }

    if (j >= 3600) {

        j = 0;

        if(gear) {

            if(gearTime <= 0) gear = 0;

            else gearTime--;

        }

        if(triedTimes >= 3) {

            if(triedTimer == 0) triedTimes = 0;

            else triedTimer--;

        }

*//秒、分、时修改\*如按日期规则修改，则可显示日期\**

        sec++;

**updateDateTime**(0, 0);

    }

}

*// 矩阵键盘扫描*

void **key\_scan**() {

    unsigned int i;

    unsigned char line; *//行输入*

    unsigned char row; *//列输入*

*// 矩阵按键扫描*

*// 滚动存储*

    scan\_buf[0] = scan\_buf[1]; *//新态、旧态滚动存储*

    scan\_buf[2] = scan\_buf[3]; *//滤波单元滚动存储*

    scan\_buf[3] = scan\_buf[4];

#ifdef **U3CON**

*// 输入新键值*

*//DISP\_SEG = 0xff;          //关显示*

    U3CON = 0x81; *//行（PC7-PC4）为输出，列(PC3-PC0)为输入*

    VKEY\_RL = 0x00; *//行输出0*

    row = VKEY\_RL;

*//DISP\_SEG = 0xff;          //关显示*

    U3CON = 0x88; *//行（PC7-PC4）为输入，列(PC3-PC0)为输出*

    VKEY\_RL = 0x00; *//列输出0*

    line = VKEY\_RL;

#else

    DISP\_SEG = 0xff; *//关显示*

    VKEY\_RL = 0x0f; *//行输出0*

**delay\_ms**(1);

    row = VKEY\_RL & 0x0f;

    DISP\_SEG = 0xff; *//关显示*

    VKEY\_RL = 0xf0; *//列输出0*

**delay\_ms**(1);

    line = VKEY\_RL & 0xf0;

#endif

    row = row & 0x0f; *//屏蔽高4位*

    line = line & 0xf0; *// 屏蔽低4位*

    scan\_buf[5] = line | row; *//合并行列值*

*// 键盘编码-->键值存入scan\_buf[4]*

    for (i = 0; i < 16; i++) {

        if (scan\_buf[5] == Scan\_code[i]) {

            scan\_buf[4] = i; *//查键盘扫描码，转换成键值0-15*

            break;

        } else

            scan\_buf[4] = 16; *//其它情况为16*

    }

*// 滤波*

    if ((scan\_buf[2] == scan\_buf[3]) && (scan\_buf[2] == scan\_buf[4]))

        scan\_buf[1] = scan\_buf[4];

}

*// 矩阵键盘四键加减功能，与自选实验冲突，只用其中一个*

int **key\_pro\_4**() {

    int i;

*// 矩阵按键事件*

*// 按下*

    if ((scan\_buf[0] == 0x10) && (scan\_buf[1] != 0x10)) {

*//在密码菜单输入密码，修改密码*

        if (menu == MENU\_PWD || menu == MENU\_PWD\_CHANGE1 || menu == MENU\_PWD\_CHANGE2 || menu == MENU\_PWD\_CHANGE3) {

            if(triedTimes >= 3) {

*// 播放无法操作声音*

                playTime = 700;

                playTimer = 0;

                playType = 3;

                return 1;

            }

            if (inputLen < 6) {

                if (scan\_buf[1] >= 1 && scan\_buf[1] <= 3)

                    input[inputLen++] = scan\_buf[1];

                else if (scan\_buf[1] >= 5 && scan\_buf[1] <= 7)

                    input[inputLen++] = scan\_buf[1] - 1;

                else if (scan\_buf[1] >= 9 && scan\_buf[1] <= 11)

                    input[inputLen++] = scan\_buf[1] - 2;

                else if (scan\_buf[1] == 14)

                    input[inputLen++] = 0;

            }

            if (inputLen == 6) {

                if(menu == MENU\_PWD\_CHANGE1) {

                    for(i = 0; i < 6; i++)

                        if(pwd[i] != input[i]) break;

                    if(i==6) {

                        menu = MENU\_PWD\_CHANGE2;

                        inputLen = 0;

*// 播放输入成功声音*

                        playTime = 300;

                        playTimer = 0;

                        playType = 4;

                    } else {

*// 播放输入错误声音*

                        playTime = 700;

                        playTimer = 0;

                        playType = 3;

                        triedTimes++;

                        if(triedTimes >= 3) {

*// 30秒钟后再输入*

                            triedTimer = 30;

                        }

                    }

                } else if(menu == MENU\_PWD\_CHANGE2) {

                    for(i = 0; i < 6; i++)

                        pwdBack[i] = input[i];

                    menu = MENU\_PWD\_CHANGE3;

                    inputLen = 0;

*// 播放第一次输入成功声音*

                    playTime = 300;

                    playTimer = 0;

                    playType = 4;

                } else if(menu == MENU\_PWD\_CHANGE3) {

                    for(i = 0; i < 6; i++)

                        if(pwdBack[i] != input[i]) break;

                    if(i==6) {

                        for(i = 0; i < 6; i++)

                            pwd[i] = input[i];

                        menu = MENU\_TIME;

                        inputLen = 0;

*// 播放输入成功声音*

                        playTime = 400;

                        playTimer = 0;

                        playType = 4;

                    } else {

                        inputLen = 0;

*// 播放输入错误声音*

                        playTime = 700;

                        playTimer = 0;

                        playType = 3;

                    }

                } else {

                    for(i = 0; i < 6; i++)

                        if(pwd[i] != input[i]) break;

                    if(i == 6) {

                        gear = 1;

                        gearTime = 3; *// 开锁3秒*

                        inputLen = 0;

                        menu = MENU\_TIME;

*// 播放输入成功声音*

                        playTime = 300;

                        playTimer = 0;

                        playType = 4;

                        triedTimes = 0;

                    } else {

*// 播放输入错误声音*

                        playTime = 700;

                        playTimer = 0;

                        playType = 3;

                        triedTimes++;

                        if(triedTimes >= 3) {

*// 30秒钟后再输入*

                            triedTimer = 30;

                        }

                    }

                }

                inputLen = 0;

            }

        }

    }

    return 0;

}

int **key\_pro\_zixuan**() {

    unsigned char key;

*// 按下*

    if ((scan\_buf[0] == 16) && (scan\_buf[1] != 16)) {

        playType = 1;

        playTime = 200;

        playTimer = 0;

        switch (scan\_buf[1]) {

        case 13:

            if(settime) settime = 0;

            else {

                menu++;

                if (menu > MENU\_PWD)

                    menu = MENU\_TIME;

                if (menu == MENU\_PWD)

                    inputLen = 0;

            }

            return 1;

        case 15:

            if(settime == 1) { *// 更新时间*

                hour = tempA;

                min = tempB;

                sec = tempC;

                settime = 0;

            } else if(settime == 2) { *// 更新日期*

                year = tempA;

                month = tempB;

                day = tempC;

                settime = 0;

            } else if(hhFastTime < 1800) { *// 0.2s*

*// 双击事件*

                menu = MENU\_PWD\_CHANGE1;

                inputLen = 0;

                return 1;

            } else {

                if(menu == MENU\_TIME) {

                    settime = 1; *// 设置时间*

                    tempA = hour;

                    tempB = min;

                    tempC = sec;

                } else if(menu == MENU\_DATE) {

                    settime = 2; *// 设置日期*

                    tempA = year;

                    tempB = month;

                    tempC = day;

                }

            }

        }

        if(settime) {

            if (1 <= scan\_buf[1] && scan\_buf[1] <= 3) {

                key = scan\_buf[1];

            } else if (5 <= scan\_buf[1] && scan\_buf[1] <= 7) {

                key = scan\_buf[1] - 1;

            } else if (9 <= scan\_buf[1] && scan\_buf[1] <= 11) {

                key = scan\_buf[1] - 2;

            } else if (13 <= scan\_buf[1] && scan\_buf[1] <= 15) {

                key = scan\_buf[1] - 3;

            }

            if (key < 10 || key == 11) {

                if (key == 11)

                    key = 0;

                switch (blinkIndex) {

                case 0:

                    if(settime == 1) {

*// 设置时十位*

                        tempA = key \* 10 + tempA % 10;

                    } else {

*// 设置年千位*

                        tempA = tempA - (tempA / 1000 \* 1000) + key \* 1000;

                    }

**updateDateTime**(settime, blinkIndex);

                    blinkIndex++;

                    break;

                case 1:

                    if(settime == 1) {

*// 设置时个位*

                        tempA = (tempA / 10) \* 10 + key;

**updateDateTime**(settime, blinkIndex);

                        blinkIndex++;

                    } else {

*// 设置年百位*

                        tempA = (tempA / 1000 \* 10 + key) \* 100 + (tempA - tempA / 100 \* 100);

**updateDateTime**(settime, blinkIndex);

                    }

                    blinkIndex++;

                    break;

                case 2:

                    if(settime == 1) {

                        ;

                    } else {

*// 设置年十位*

                        tempA = (tempA / 100 \* 10 + key) \* 10 + tempA % 10;

                    }

**updateDateTime**(settime, blinkIndex);

                    blinkIndex++;

                    break;

                case 3:

                    if(settime == 1) {

*// 设置分十位*

                        tempB = key \* 10 + tempB % 10;

                    } else {

*// 设置年个位*

                        tempA = tempA / 10 \* 10 + key;

                    }

**updateDateTime**(settime, blinkIndex);

                    blinkIndex++;

                    break;

                case 4:

                    if(settime == 1) {

*// 设置分个位*

                        tempB = (tempB / 10) \* 10 + key;

**updateDateTime**(settime, blinkIndex);

                        blinkIndex++;

                    } else {

*// 设置月十位*

                        tempB = key \* 10 + tempB % 10;

**updateDateTime**(settime, blinkIndex);

                    }

                    blinkIndex++;

                    break;

                case 5:

                    if(settime == 1) {

                        ;

                    } else {

*// 设置月个位*

                        tempB = tempB / 10 \* 10 + key;

                    }

**updateDateTime**(settime, blinkIndex);

                    blinkIndex++;

                    break;

                case 6:

                    if(settime == 1) {

*// 设置秒十位*

                        tempC = key \* 10 + tempC % 10;

                    } else {

*// 设置日十位*

                        tempC = key \* 10 + tempC % 10;

                    }

**updateDateTime**(settime, blinkIndex);

                    blinkIndex++;

                    break;

                case 7:

                    if(settime == 1) {

*// 设置秒个位*

                        tempC = (tempC / 10) \* 10 + key;

                    } else {

*// 设置日个位*

                        tempC = tempC / 10 \* 10 + key;

                    }

**updateDateTime**(settime, blinkIndex);

                    blinkIndex = 0;

                    break;

                }

            }

            if (scan\_buf[1] == 0) {

*// 向左移动，设置时、十位*

                blinkIndex--;

                if (settime == 1) {

                    if (blinkIndex == 2 || blinkIndex == 5)

                        blinkIndex--;

                }

            }

            if (scan\_buf[1] == 4) {

*// 向右移动，设置个位、分*

                blinkIndex++;

                if (settime == 1) {

                    if (blinkIndex == 2 || blinkIndex == 5)

                        blinkIndex++;

                }

            }

            if (blinkIndex == 8)

*// 归位光标*

                blinkIndex = 0;

            if (blinkIndex == -1)

*// 光标移至末尾*

                blinkIndex = 7;

        }

    }

*// 松开*

    if((scan\_buf[0] != 0x10) && (scan\_buf[1] == 0x10)) {

        switch(scan\_buf[0]) {

            case 0:

*// 初始化时间日期*

                hour = 23;

                min = 55;

                sec = 20;

                gear = 0;

*// 初始化密码*

                pwd[0] = 5;

                pwd[1] = 3;

                pwd[2] = 5;

                pwd[3] = 8;

                pwd[4] = 6;

                pwd[5] = 6;

*// 初始化状态*

                settime = 0;

                triedTimes = 0; *// 尝试连续输入密码的次数*

                triedTimer = 0; *// 输入密码计数器*

                zijianpos = 0;

                playSound = 0;

*// 进入自检菜单*

                menu = MENU\_ZIJIAN;

                break;

            case 15:

*// PB2 单双击LED松开*

                if(!sgClick) {

                    sgClick = 1;

                    hhFastTime = 0; *// 计时清零*

                } else

                    sgClick = 0;

        }

    }

    return 0;

}

*// 按键处理，分为两种*

void **key\_pro**() {

*// 设置时间日期*

    if(**key\_pro\_zixuan**()) return;

*// 输入密码*

    if(**key\_pro\_4**()) return;

}

*// 蜂鸣器*

void **alarm**() {

*// 蜂鸣器功能*

    if(playType != 0) {

        if(playTime == 0) temp = 1;

        else temp = playTimer / playTime;

        if(temp % 2 == 0) playSound = 1;

        else playSound = 0;

        playTimer++;

        if(temp >= (playType << 1)) {

            playType = 0;

            playTimer = 0;

            playSound = 0;

        }

    }

    beep = !playSound;

}

void **main**() {

#ifdef **U3CON**

    U3CON = 0x89; *// 8255(U3)初始化，A、B口方式0输出，C口输入*

#endif

**timer\_init**();

**delay\_ms**(60);

*// 初始化时间日期*

    hour = 23;

    min = 55;

    sec = 20;

    gear = 0;

*// 进入自检菜单*

    menu = MENU\_ZIJIAN;

*// 主循环*

    while (1) {

*// 刷新显示*

**display\_buf**();

**display\_swbuf**();

*// 键盘检测*

**key\_scan**();

**key\_pro**();

*// 蜂鸣器更新*

**alarm**();

    }

}