# 综合实验1电子密码锁

## ****系统设计方案****

本系统采用以单片机为核心元件的控制方案。由于单片机种类繁多，各种型号都有其一定的应用环境，因此在选用时要多加比较，合理选择，以获得最佳的性价比。一般来说在选取单片机时从下面几个方面考虑：性能、存储器、运行速度、I/O口、定时/计数器、串行接口、模拟电路功能、工作电压、功耗、封装形式、抗干扰性、保密性，除了以上的一些方面外，还有一些最基本的条件，比如：中断源的数量和优先级、工作温度范围、有没有低电压检测功能、单片机内有无时钟振荡器、有无上电复位功能等。在开发过程中还要考虑开发工具、编程器、开发成本、开发人员的适应性、技术支持和服务等因素。基于以上因素本设计选用单片机STC89C516作为本设计的核心元件，利用单片机灵活的编程设计和丰富的I/O端口，及其控制的准确性，实现基本的密码锁功能。在单片机的外围电路外接输入键盘用于密码的输入和一些功能的控制，外接AT24C02芯片用于密码的存储，外接LCD1602显示器用于显示作用。

## 硬件设计

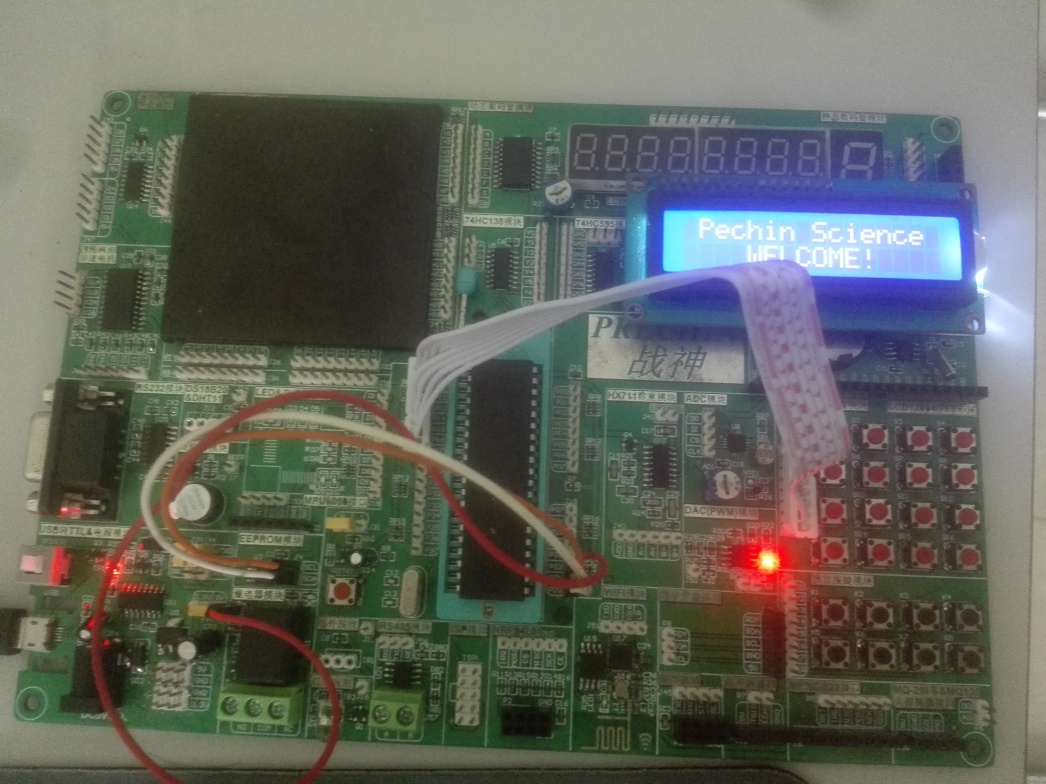
## 硬件连接如下：

P1 -->矩阵键盘

P20--> EEPROM模块SDA

P21--> EEPROM模块SCL

P22-->继电器模块J15



其他管脚说明：

P0 LCD1602\_DATAPINS

P2^7 LCD1602\_E;

P2^5 LCD1602\_RW;

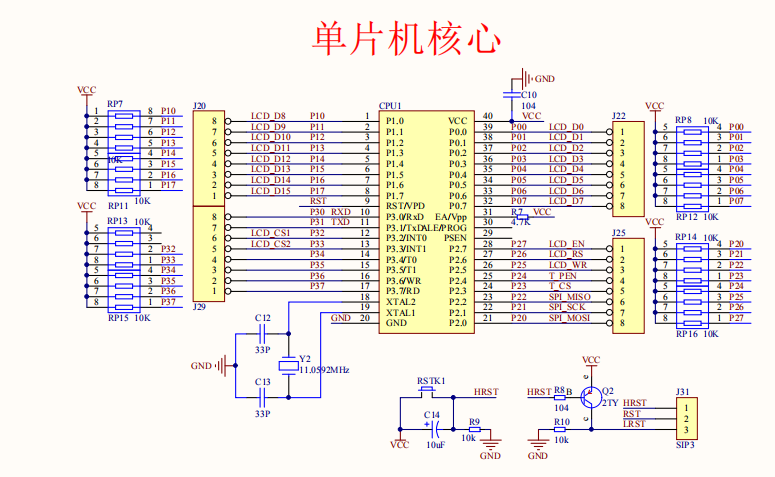
P2^6 LCD1602\_RS;

P1 矩阵键盘

P2^1; 存储模块 24C02 SCL

P2^0 存储模块 24C02 SDA

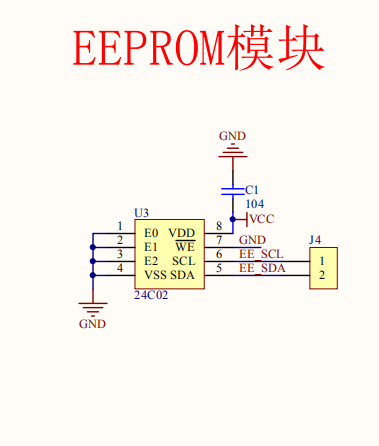
### 2.1 主控芯片



### 2.2存储模块24c02

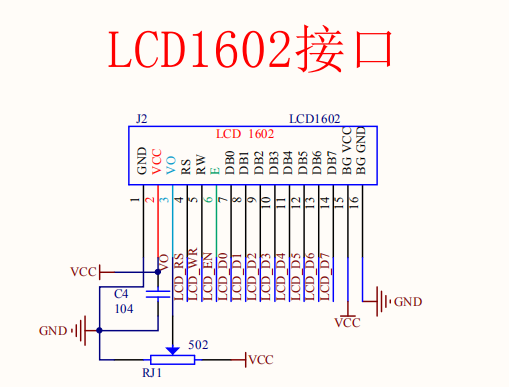
存储模块的设计是把WP引脚接到GND上因为要让器件进行正常的读/写操作，把SDA串行数据/地址与单片机的P2.0引脚使AT24C02与单片机进行所有数据的发送或接收，把SCL串行时钟引脚与单片机的P2.1引脚相连接，让单片机产生一个AT24C02工作的时钟，使其正常的工作。存储模块电路如图所示。

AT24C02的特点：数据线看门狗定时器；可编程复位门级；高数据传输速率400kHz速率和I2C总线；2.7V至7V电压；低功耗CMOS工艺兼容；16 bespate写缓冲区；反撇写保护芯片；高低电平复位信号输出；100万擦除周期；保存长达100年。



### 2.3显示模块LCD1602

如果想要了解系统的运行与工作状态显示器是不可缺少的，显示器是一个典型的输出设备并且它的实际应用也是极为广泛的，几乎所有的电子产品都会使用到显示器其差别仅在于显示器的结构类型不同而已。最简单的显示器可以是LED发光二极管或者数码管，它可以给出一个简单的开关信息或者数字显示，而需要显示复杂的完整的信息上述的器件就不能胜任，而液晶的出现就很好的解决了这一问题，它不仅显示内容丰富而且好节约了单片机的管脚资源。本设计采用的是液晶显示LCD1602。LCD1602内部的字符已经储存了不同的字符形式，每一个字符都有一个固定的代码，其代码与标准的ASCII字符代码一致。因此只要写入显示字符的ASCII码即可，这种标准化的设计给使用带来很大的方便。比如英文字母“C”的ASCII代码是01000011（43H），显示时单片机往液晶模块写入显示指令，模块就会把地址为43H中的点阵字符图形识别出，并会在液晶屏相应位置上看到字母“C”。



目前市场字符液晶绝大多数是基于HD44780的液晶芯片，控制原理是完全相同的，因此基于HD44780的写控制程序可以很容易地应用到市场上大部分的字符液晶。LCD1602液晶的几个特性：+5V电压、对比度可调、内含复位电路；提供各种控制命令，如：清屏、字符闪烁、光标闪烁、显示移位等多种功能；有80字节显示数据存储器DDRAM；内建有160个5X7点阵的字型的字符发生器CGROM 8个可由用户自定义的5X7的字符发生器CGRAM。1602的引脚如下所示：

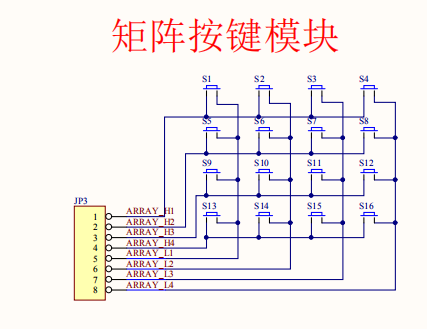


LCD1602采用标准的16脚接口，其中VSS为地电源，VDD接5V正电源，VEE为液晶显示器。RS为寄存器选择，高电平时选择数据寄存器、低电平时选择指令寄存器。RW为读写信号线，高电平1时进行读操作，低电平0时进行写操作。E端为使能端。D0~D7为8位双向数据线。

### 2.4键盘输入模块

键盘是单片机十分重要的输入设备，是实现人机对话的纽带。键盘是由一组规则排列的按键组成，一个按键实际上就是一个开关元件，即键盘是一组规则排列的开关。根据按键与单片机的连接方式不同，按键主要分为独立式按键和矩阵式按键 ，有了这些按键，对单片机的控制就方便多了。

本设计按键数量较多，所以采用矩阵式按键以节省I/O口线。将16个按键分为4排4列排列好，下图是矩阵键盘硬件结构。当有一个键按下时，通过某一边引脚赋低电平，扫描全部引脚看是否与最初的赋值一样，不一样则根据相应的算法（通过改变后的值与初始值相或，根据结果赋值）确定是哪个键按下。



### 2.5开锁电路

开锁电路的功能是当输入正确的密码后密码锁将被锁定，既开锁。当单片机P2.2引脚发出信号经三极管放大后，触动电磁阀即会把锁打开。一旦输入密码，单片机便会与初始密码进行比对，如果密码输入3次都与原始密码不相符即会报警。其原理图如图3-11所示

### 

## 软件设计

### 3.1 按键扫描

按键扫描程序如下：

#include "key.h"

u8 KeyValue=0;

void delay(u16 i)

{

while(i--);

}

u8 KeyDown(void)

{

char a=0;

GPIO\_KEY=0x0f;

if(GPIO\_KEY!=0x0f)//有按键按下

{

delay(1000);//延时10ms去抖

if(GPIO\_KEY!=0x0f)//有按键按下

{

GPIO\_KEY=0X0F;

switch(GPIO\_KEY)

{

case(0X07): KeyValue=0;break;

case(0X0b): KeyValue=1;break;

case(0X0d): KeyValue=2;break;

case(0X0e): KeyValue=3;break;

}

GPIO\_KEY=0XF0;

switch(GPIO\_KEY)

{

case(0X70): KeyValue=KeyValue;break;

case(0Xb0): KeyValue=KeyValue+4;break;

case(0Xd0): KeyValue=KeyValue+8;break;

case(0Xe0): KeyValue=KeyValue+12;break;

}

while((a<50)&&(GPIO\_KEY!=0xf0))

{

delay(1000);

a++;

}

}

}

else

{

KeyValue=0xff; //无按键按下

}

return KeyValue; //返回KeyValue

}

从前面硬件电路中可以看到，P1口的低四位接的4\*4矩阵键盘的行，高四位接的矩阵键盘的列。检测矩阵键盘是否有按键按下时：先将P1端口的低四位置1，高四位清零，检测P1端口的状态，如果高四位不为零，则表示有按键按下，并且可以知道是x0-x3哪一列有按键按下，比如P1=0x1f；则第一列有按键按下。此时我们在将P1口第四位清零，高四位置1；检测P1的状态，就知道y0-y3哪一行有按键按下。结合xy就可以知道具体是哪个按键按下。

### 3.2 1602初始化

16002初始化程序如下：

#include "lcd.h"

uchar i;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : Lcd1602\_Delay1ms

\* 函数功能 : 延时函数，延时1ms

\* 输 入 : c

\* 输 出 : 无

\* 说 名 : 该函数是在12MHZ晶振下，12分频单片机的延时。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Lcd1602\_Delay1ms(uint c) //误差 0us

{

uchar a,b;

for (; c>0; c--)

{

for (b=199;b>0;b--)

{

for(a=1;a>0;a--);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : LcdWriteCom

\* 函数功能 : 向LCD写入一个字节的命令

\* 输 入 : com

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef LCD1602\_4PINS //当没有定义这个LCD1602\_4PINS时

void LcdWriteCom(uchar com) //写入命令

{

LCD1602\_E = 0; //使能

LCD1602\_RS = 0; //选择发送命令

LCD1602\_RW = 0; //选择写入

LCD1602\_DATAPINS = com; //放入命令

Lcd1602\_Delay1ms(1); //等待数据稳定

LCD1602\_E = 1; //写入时序

Lcd1602\_Delay1ms(5); //保持时间

LCD1602\_E = 0;

}

#else

void LcdWriteCom(uchar com) //写入命令

{

LCD1602\_E = 0; //使能清零

LCD1602\_RS = 0; //选择写入命令

LCD1602\_RW = 0; //选择写入

LCD1602\_DATAPINS = com; //由于4位的接线是接到P0口的高四位，所以传送高四位不用改

Lcd1602\_Delay1ms(1);

LCD1602\_E = 1; //写入时序

Lcd1602\_Delay1ms(5);

LCD1602\_E = 0;

LCD1602\_DATAPINS = com << 4; //发送低四位

Lcd1602\_Delay1ms(1);

LCD1602\_E = 1; //写入时序

Lcd1602\_Delay1ms(5);

LCD1602\_E = 0;

}

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : LcdWriteData

\* 函数功能 : 向LCD写入一个字节的数据

\* 输 入 : dat

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef LCD1602\_4PINS

void LcdWriteData(uchar dat) //写入数据

{

LCD1602\_E = 0; //使能清零

LCD1602\_RS = 1; //选择输入数据

LCD1602\_RW = 0; //选择写入

LCD1602\_DATAPINS = dat; //写入数据

Lcd1602\_Delay1ms(1);

LCD1602\_E = 1; //写入时序

Lcd1602\_Delay1ms(5); //保持时间

LCD1602\_E = 0;

}

#else

void LcdWriteData(uchar dat) //写入数据

{

LCD1602\_E = 0; //使能清零

LCD1602\_RS = 1; //选择写入数据

LCD1602\_RW = 0; //选择写入

LCD1602\_DATAPINS = dat; //由于4位的接线是接到P0口的高四位，所以传送高四位不用改

Lcd1602\_Delay1ms(1);

LCD1602\_E = 1; //写入时序

Lcd1602\_Delay1ms(5);

LCD1602\_E = 0;

LCD1602\_DATAPINS = dat << 4; //写入低四位

Lcd1602\_Delay1ms(1);

LCD1602\_E = 1; //写入时序

Lcd1602\_Delay1ms(5);

LCD1602\_E = 0;

}

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名 ：

函数功能 ：显示\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

字符串显示函数

Coordinate 高四位显示坐标的X，第四位显示坐标的Y

如在第一行从第一位开始显示 hello： ShowString (0x01,“hello”);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ShowString (unsigned char Coordinate,char \*ptr)

{

if(Coordinate<=0x0f) //高四位为0，显示第一行

{

LcdWriteCom((Coordinate&0x0f)+0x80);

while(\*ptr!='\0')

{

LcdWriteData(\*ptr);

ptr++;

}

}

else //高四位为1，显示第2行

{

LcdWriteCom((Coordinate&0x0f)+0xc0);

while(\*ptr!='\0')

{

LcdWriteData(\*ptr);

ptr++;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : LcdInit()

\* 函数功能 : 初始化LCD屏

\* 输 入 : 无

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef LCD1602\_4PINS

void LcdInit() //LCD初始化子程序

{

LcdWriteCom(0x38); //开显示

LcdWriteCom(0x0c); //开显示不显示光标

LcdWriteCom(0x06); //写一个指针加1

LcdWriteCom(0x01); //清屏

LcdWriteCom(0x80); //设置数据指针起点

}

#else

void LcdInit() //LCD初始化子程序

{

LcdWriteCom(0x32); //将8位总线转为4位总线

LcdWriteCom(0x28); //在四位线下的初始化

LcdWriteCom(0x0c); //开显示不显示光标

LcdWriteCom(0x06); //写一个指针加1

LcdWriteCom(0x01); //清屏

LcdWriteCom(0x80); //设置数据指针起点

}

#endif

1602的初始化函数比较长，但其中大部分是1602驱动的底层协议，初学者不必深究，只需要学会怎么使用即可。这里简单介绍几个常用的函数。

1. LcdWriteCom()

1602的写命令函数，通过此函数向1602写命令。比如：清屏LcdWriteCom(0x01); LcdWriteCom(0x80); //设置数据指针起点

1. LcdWriteData()
2. 当需要在1602上显示时，就需要用到LcdWriteData()，不过在写数据之前需要通过LcdWriteCom()函数告诉要写数据的地址，1602的第一行的16个显示位地址是0x80到0x8f；第二行的地址是0xc0到0xcf。比如在1602的第一行第一位显示数字8：LcdWriteCom(0x80);

LcdWriteData(‘8’);

1. ShowString (unsigned char Coordinate,char \*ptr)

ShowString (首地址，字符串)函数在需要显示字符串时使用。比如在第二行第3位开始显示hello：

ShowString (0x13，”hello”);

其中首地址的高四位为0则表示在第一排显示，为“1”则在第二排显示。低四位为0则在第0位显示。

### 3.3 24C02初始化

24C02的初始化如下：

#include"i2c.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : Delay10us()

\* 函数功能 : 延时10us

\* 输入 : 无

\* 输出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay10us()

{

unsigned char a,b;

for(b=1;b>0;b--)

for(a=2;a>0;a--);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : I2cStart()

\* 函数功能 : 起始信号：在SCL时钟信号在高电平期间SDA信号产生一个下降沿

\* 输入 : 无

\* 输出 : 无

\* 备注 : 起始之后SDA和SCL都为0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void I2cStart()

{

SDA=1;

Delay10us();

SCL=1;

Delay10us();//建立时间是SDA保持时间>4.7us

SDA=0;

Delay10us();//保持时间是>4us

SCL=0;

Delay10us();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : I2cStop()

\* 函数功能 : 终止信号：在SCL时钟信号高电平期间SDA信号产生一个上升沿

\* 输入 : 无

\* 输出 : 无

\* 备注 : 结束之后保持SDA和SCL都为1；表示总线空闲

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void I2cStop()

{

SDA=0;

Delay10us();

SCL=1;

Delay10us();//建立时间大于4.7us

SDA=1;

Delay10us();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : I2cSendByte(unsigned char dat)

\* 函数功能 : 通过I2C发送一个字节。在SCL时钟信号高电平期间，保持发送信号SDA保持稳定

\* 输入 : num

\* 输出 : 0或1。发送成功返回1，发送失败返回0

\* 备注 : 发送完一个字节SCL=0,SDA=1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned char I2cSendByte(unsigned char dat)

{

unsigned char a=0,b=0;//最大255，一个机器周期为1us，最大延时255us。

for(a=0;a<8;a++)//要发送8位，从最高位开始

{

SDA=dat>>7; //起始信号之后SCL=0，所以可以直接改变SDA信号

dat=dat<<1;

Delay10us();

SCL=1;

Delay10us();//建立时间>4.7us

SCL=0;

Delay10us();//时间大于4us

}

SDA=1;

Delay10us();

SCL=1;

while(SDA)//等待应答，也就是等待从设备把SDA拉低

{

b++;

if(b>200) //如果超过2000us没有应答发送失败，或者为非应答，表示接收结束

{

SCL=0;

Delay10us();

return 0;

}

}

SCL=0;

Delay10us();

return 1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : I2cReadByte()

\* 函数功能 : 使用I2c读取一个字节

\* 输入 : 无

\* 输出 : dat

\* 备注 : 接收完一个字节SCL=0,SDA=1.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned char I2cReadByte()

{

unsigned char a=0,dat=0;

SDA=1; //起始和发送一个字节之后SCL都是0

Delay10us();

for(a=0;a<8;a++)//接收8个字节

{

SCL=1;

Delay10us();

dat<<=1;

dat|=SDA;

Delay10us();

SCL=0;

Delay10us();

}

return dat;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : void At24c02Write(unsigned char addr,unsigned char dat)

\* 函数功能 : 往24c02的一个地址写入一个数据

\* 输入 : 无

\* 输出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void At24c02Write(unsigned char addr,unsigned char dat)

{

I2cStart();

I2cSendByte(0xa0);//发送写器件地址

I2cSendByte(addr);//发送要写入内存地址

I2cSendByte(dat); //发送数据

I2cStop();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : unsigned char At24c02Read(unsigned char addr)

\* 函数功能 : 读取24c02的一个地址的一个数据

\* 输入 : 无

\* 输出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned char At24c02Read(unsigned char addr)

{

unsigned char num;

I2cStart();

I2cSendByte(0xa0); //发送写器件地址

I2cSendByte(addr); //发送要读取的地址

I2cStart();

I2cSendByte(0xa1); //发送读器件地址

num=I2cReadByte(); //读取数据

I2cStop();

return num;

}

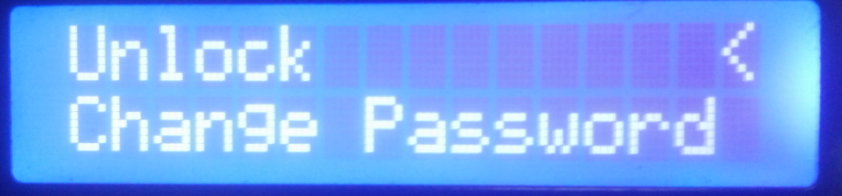
24c02采用IIC与单片机通信，其详细通信时序参考24C02数据手册。我们只需使用At24c02Read(unsigned char addr)和At24c02Write(unsigned char addr,unsigned char dat)即可。

## 实验现象

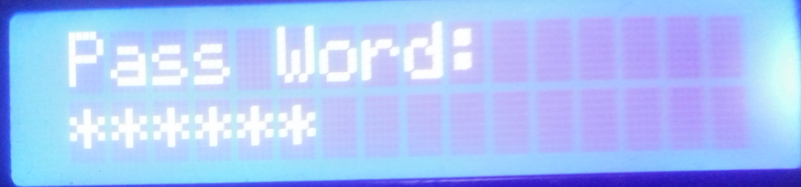
将线连接好并将程序下载后，可以看到1602上显示：



此时可以按下任意键继续



在这里我们可以“1”或“9”选择开锁或者是修改密码。如果这里选择的开锁，按下确定键，此时会提示输入密码，初始密码默认为888888，输入密码

然后按下确定键，如果密码正确则会显示：



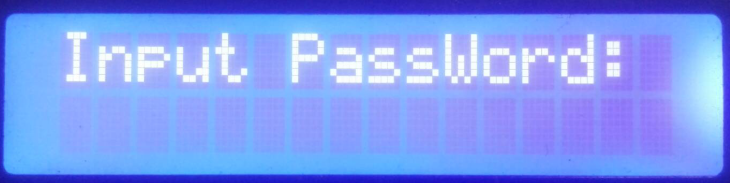
并且可以看到继电器模块指示灯亮并可以听到继电器闭合的声音。

如果密码错误则会提示：Error.

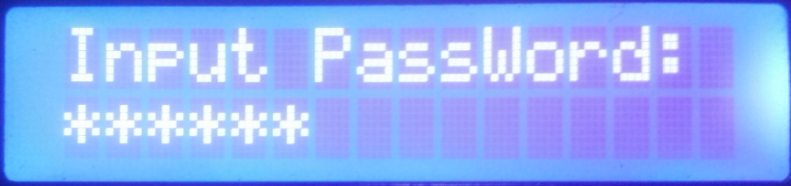
如果需要修改密码，则选择Change password



这里会提示输入密码：



输入原密码



按下确认键，如果密码正确会提示输入新密码，



输入密码后按下确认键，会提示OK并返回到待机画面。