**重庆三峡学院**

**毕业设计（论文）**

|  |  |  |  |
| --- | --- | --- | --- |
| **题目** | **智能电子秤的设计与实现** | | |
| **学院** | **电子与信息工程** | | |
| **专业** | **电子信息工程** | | |
| **年级** | **2016级电子信息一班** | | |
| **姓名** | **肖增兵** | | |
| **学号** | **201607014103** | | |
| **指导教师** | **孙光壮** | **职称** | **讲师** |

**完成毕业设计（论文）时间 年 月**

目录

[摘要： I](#_Toc38634889)

[Abstract： II](#_Toc38634890)

[1 绪论 1](#_Toc38634891)

[1.1 背景及意义 1](#_Toc38634892)

[1.2 国内外电子秤的发展现状与趋势 1](#_Toc38634893)

[1.3 本设计研究的内容 2](#_Toc38634894)

[2 总设计原理及方案 1](#_Toc38634895)

[2.1 智能电子秤的设计要求 1](#_Toc38634896)

[2.2 智能电子秤的工作原理 1](#_Toc38634897)

[2.3 智能电子秤原理框图 1](#_Toc38634898)

[3 硬件电路设计 3](#_Toc38634899)

[3.1 硬件控制电路模块 3](#_Toc38634900)

[3.2 STC89C51 单片机模块的概述 3](#_Toc38634901)

[3.3 （A/D）转换模块 4](#_Toc38634902)

[3.4 称重传感器模块 6](#_Toc38634903)

[3.5 LCD12864 液晶显示模块 7](#_Toc38634904)

[3.6 超重报警模块 9](#_Toc38634905)

[3.7 复位电路 9](#_Toc38634906)

[3.8 矩阵键盘设计模块 9](#_Toc38634907)

[4 软件设计 11](#_Toc38634908)

[4.1 软件开发环境 11](#_Toc38634909)

[4.2 主程序流程图 11](#_Toc38634910)

[4.3 LCD12864显示函数流程图 13](#_Toc38634911)

[4.4 矩阵键盘检测函数的流程图 14](#_Toc38634912)

[4.5 超重报警部分流程图 16](#_Toc38634913)

[5 系统调试与调试结果 17](#_Toc38634914)

[5.1 硬件电路的制作 17](#_Toc38634915)

[5.2 硬件电路的调试和结果 17](#_Toc38634916)

[5.2.1 硬件电路的调试及结果 17](#_Toc38634917)

[5.3 软件系统的制作 19](#_Toc38634918)

[5.3.1 软件系统的调试及结果 19](#_Toc38634919)

[设计总结 20](#_Toc38634920)

[参考文献 21](#_Toc38634921)

[致谢 22](#_Toc38634922)

[附录 23](#_Toc38634923)

[附录一 智能电子秤原理图 23](#_Toc38634924)

[附录二 源程序 24](#_Toc38634925)

智能电子秤设计

肖增兵

重庆三峡学院电子与信息工程学院电子信息工程专业 2016 级 重庆万州 404100

摘要：本文设计是以 STC89C52 单片机为控制核心的智能电子秤，它主要由 STC89C52 单片机、HX711 模数转换器、LCD12864 显示器、应变式压力传感器、4 \* 4矩阵键盘五大部分组成。系统安装了矩阵键盘，用户可以通过矩阵键盘输入项目的价格，系统可以计算出总价格，并在液晶显示屏上实时显示出物体的重量和单价。为了防止系统负载，当被称重物超过10公斤时，添加了“超重提示机制”，系统为用户超重发送提示。

关键词：STC89C52 单片机，压力传感器，4\*4矩阵键盘，HX711模数转换器，LCD12864液晶显示

**Design of Portable intelligent electronic scale**

**Xiao zeng-bing**

Chongqing Three Gorges University School of Electronics and Information Engineering Electrical Electronic Information Engineering Professional 2016 Chongqing Wanzhou 404100

**Abstract：**The design of this article is based on STC89C52 single-chip microcomputer as the core of the control of intelligent electronic scale, which is mainly composed of five parts: STC89C52 single-chip microcomputer, HX711 analog-to-digital converter, LCD12864 display, strain gauge pressure sensor, 4 \* 4 matrix keyboard. The system is equipped with a matrix keyboard. The user can input the price of the item through the matrix keyboard. The system can calculate the total price and display the weight and unit price of the object on the LCD screen in real time. In order to prevent the system from loading, when the weight to be weighed exceeds 10 kg, an "overweight reminder mechanism" is added, and the system sends a reminder for the user to be overweight.

**Key words:** STC89C52 single chip microcomputer, pressure sensor, 4 \* 4 matrix keyboard, HX711 analog-to-digital converter, LCD12864 liquid crystal display

1 绪论

传统的机械秤具有许多缺点，例如精度低，结构复杂，易老化和成本高。随着社会的发展，对秤的市场需求越来越高，比如人体秤，厨房秤和其他便携式小秤。与一般古老的机械秤相比，现在的智能电子秤对比下，有了许多优势。它用压力传感器代替了机械秤的弹簧，从而极大地减小了秤的尺寸和制造难度。传统的表盘被LCD或LED显示屏取代，使外观更加美观。由于集成了微控制器和软件系统，电子秤也具有传统机械秤无与伦比的智能。他可以完成许多功能，例如过载警报，总价计算，数据通信等。

当前市场上的称重工具要么结构复杂，要么操作不可靠，并且成本高昂，并且总体水平不高。一些企业质量低劣，工艺不精，技术较弱，设备不齐全，资金缺乏，缺少产品开发能力，质量也难以得到保障。所以，制作开发出一套具有实用价值的电子秤实物，克服上述一些缺点，改善电子秤的应用的一些缺陷，是现在比较重要的意义。

* 1. 背景及意义

随着社会经济的飞速发展，商品贸易的种类和数量日益增加。长期以来，人们一直对使用传统的机械称重工具称量物品感到不满意。因此，随着科学技术的发展，我们已经开发出了性能更好，秤盘更方便的量具。中国电子秤的研发始于1960年代。原始的电子秤通过机电一体化实现了称重功能。后来，经过数十年的不断研究与开发和改进，伴随着计算机技术和电子技术的发展，电子秤的发展迅速，电子秤朝着数字化，全电子化，智能化，多功能化和高精度化的方向发展。以前的机械电子秤的测重方法已经从模拟测量逐渐发展为数字测量。由于电子称重工具比传统称重工具具有很大的优势，因此传统的机械称重工具逐渐被电子称重工具所取代。在过去的几十年中，中国在电子称重设备的研究和发展方面发展迅速，但就目前而言，中国电子称重仪表行业的发展水平还远远没有达到世界一流水平，差距还很大。体现在研发能力薄弱，制造工艺和技术比较落后，生产设备精度不够、测量仪器落后，产品稳定性差，功能不足，环境适应性差，产品类型不足，智能低，测量精度低以及工作可靠性高较差的。随着经济的快速发展，商品交易数量急剧增加。我们的对称称重工具需要易于操作，易于识别，可携带，直观显示，方便称重，使用简单，高精度和高度智能化。随着现代科技的飞速发展，电子称等工具精度也越来越高，更加智能化，使用也更方便快捷，性能也越来越可靠，运行与更加稳定，对环境适应性强等优点而被广泛使用。在人们的生活中。它不仅极大地促进了人们的生产生活，而且大大提高了人们的称重效率和商品交易效率。这也促进了社会经济的发展。

* 1. 本设计研究的内容

根据设计要求，智能电子秤的设计需要完成以下工作：首先，明确设计要求，根据设计要求，上网查询相关书籍，收集相关信息，阐明智能电子秤的基本原理。秤，使用知识，结合互联网上收集的书籍和相关信息，缩小设计思路，构建设计框架，设计智能电子秤的示意图，并使用Altium Designer软件进行布局。然后确定组件的类型和数量。根据便携式智能电子秤的要求，实现各种功能，查询相关信息，确定所需的组件，列出电子组件，在电子商务网站上购买相关组件，并使用学到的知识参考相关材料，根据智能电子秤各部分的功能，使用keil uvision4软件来完成系统软件的编译，校正，调试和刻录。最后，根据设计的示意图，将电子组件合理地布置在铜复合板上并进行焊接。使用万用表检查组件的功能是否正常以及每个电路中是否存在短路或断路。已使用原始电子秤软件和硬件系统正确测试。之后，使用电池为硬件电路供电，并对软件和硬件进行联合调试。在调试过程中，发现问题，并分析和分析问题的原因，直到智能电子秤的所有部分正常工作并符合设计要求为止。

本文的结构如下：

第一章 绪论，简单的概括，介绍了智能电子秤的背景及意义，研究的目的，意义和内容。

第二章 总设计原理及方案，这一章的内容是电子秤设计要求、工作原理、以及原理框图。首先是整体的设计要求，然后是每个硬件模块：压力传感器，HX711（A/D）转换模块，单片机最小系统，LCD12864液晶显示模块。以及软件方面的程序显示。

第三章 硬件电路设计，在扩展每个模块的方案中，它引入了简单的功能并将其应用于每种方案中使用的主芯片，并详细说明了此电路设计的具体电路图。

第四章 软件设计，本章主要介绍软件的电子设计，这被称为该设计的主程序流程图和某些模块的子程序图。

第五章 系统调试与调试结果。

最后，对本文的主要工作和结果进行了总结和讨论。

总设计原理及方案

* 1. 智能电子秤的设计要求

我们选择STC89C52作为本设计的主控制芯片，使用应变计压力传感器和HX711模块收集重量信息，转换为实际重量并由LCD12864实时显示，并增加了4个\* 4矩阵键盘，用户可以通过键盘输入单价，系统会根据单价和重量自动计算总价，还可以执行剥皮功能。 如果称量的重量超过10Kg的范围，显示屏将显示“ overweight”（超重），并且板上的指示灯将点亮以提示。本次设计实现的主要功能如下：

1）精度与量程：测量精度为 0.1%，测量误差为±0.010kg，量程范围为 0~10kg；

2）led警报灯：超出量程最大值 10kg 时光报警功能；

3) LCD液晶显示功能：开机显示动画，重量显示，单价显示，总价显示；

4）键盘功能：输入单价，重新输入，确认，退格功能。

* 1. 智能电子秤的工作原理

根据本设计的要求，进行如下设计：使用STC89C52 单片机作为整个系统的主要控制单元系统，按下电源开关，然后通过独立的按钮设置电子秤的量程（0〜10kg） 。当它在电子秤的称重平台上时，机械变形会导致电阻发生变化，从而产生压电效应。利用这一原理，称重传感器可以将称量物品的重量转换成可以用相关仪器测量的电压变化。电压信号进入放大电路，经过放大一系列操作后，HX711转换器将接收到的模拟信号转换成数字信号，然后将转换后的数字信号发送到单片机进行控制和解码。最后，被称量物品的重量以数字形式显示在LCD上。另外，当被称量物品的重量在（0〜10kg）范围内时，智能电子秤正常工作，超重报警系统不工作，即LED不闪烁；当被称量物品的重量超过电子秤的最大范围时。当该值是10kg时，超重警报系统起作用，即LED闪烁。

* 1. 智能电子秤原理框图

本次智能电子秤的基本原理是：将物品放在压力传感器称重平台上时，压力传感器将被秤物导致传感器的形变转换为电压的变化，之后进行通过相关仪器进行测量。通过放大器电路放大电压或电流后，HX711模块（A/D）将变化的电压从模拟信号转换成数字信号，然后将转换后的数字信号输送到到STC89C52单片机进行相应的处理。如果物品的重量大于电子秤的最大范围10kg，单片机将发出指令使超重报警系统发出报警； 如果物品的重量在电子秤的范围内（0〜10kg），则LCD显示屏会显示物品的重量。 根据上述智能电子秤的一般原理，结合所学知识并参考相关资料，设计了以下框图所示的方案。



图 2-1 智能电子秤原理图

硬件电路设计

* 1. 硬件控制电路模块

本次设计的智能电子秤有称重应变式力传感器传感器、HX711（A/D）转换芯片、STC89C52单片机系统、超重LED灯亮报警、矩阵按键处理、LCD12864液晶显示这六大模块，本章主要介绍该设计中电路各部分的设计原理。 通过每个模块的功能描述，了解其工作原理及其在设计中的作用。

* 1. STC89C51 单片机模块的概述

根据本次设计的要求，选用STC89C52单片机作为中央核心处理控制单元。下面是这个芯片的简介：STC89C52单片机是STC公司生产的单片机，单片机的生产过程中，采用了CMOS工艺和高密度、高速等相关技术。因此，STC89C52单片机具有低功耗的特点。STC89C52具有外部中断、定时器、时钟输出和计数器等功能。

STC89C52 单片机引脚如图 3-1 所示：

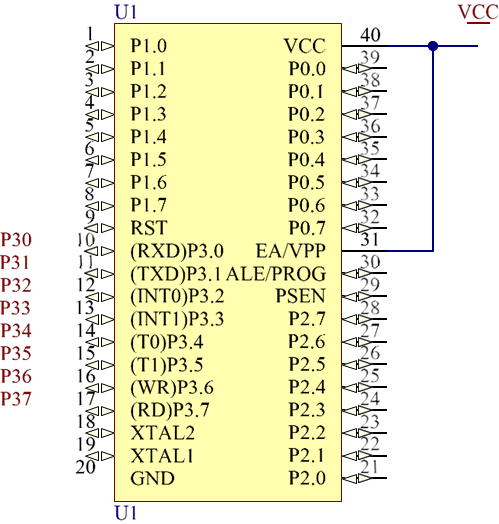


图 3-1 STC89C52 单片机引脚图

表 3-1 STC89C52 单片机引脚功能表

|  |  |
| --- | --- |
| 引脚符号 | 引脚功能说明 |
| VCC | 主电源输入引脚，接﹢5V 直流电源 |
| GND | 接地端 |
| RST | 复位输入引脚，保持两个机器周期的高电平时动作 |
| P0 口 | 共有 8 位双向 I/O 口，可作为低 8 位地址/数据复用 |
| P1 口 | 共有 8 位双向 I/O 口。作为输入口使用时，对 pP1 口写入“1” |
| P2 口 | 8 位双向 I/O 口，其内部接有上拉电阻。可作为高 8 位地址/数据复用 |
| P3 口 | 共有 8 位双向 I/O 口，其内部有上拉电阻。此端口可用作特殊功能 |
| XTAL1 | 反向振荡放大器的输入及内部时钟工作电路的输入端 |
| XTAL2 | 反向振荡输出端 |
| PSEN | 外部程序存储选通信号输出端 |
| EA/VPP | 外部程序访问允许端 |

* 1. HX711（A/D）转换模块

HX711（A/D）转换部分是比较关键的部分。如果这部分处理不好，整个设计就毫无意义。目前，世界上的adc有多种类型，包括传统的并行、逐次逼近和积分型adc，以及近年来新开发的sigma-delta和流水线adc。每种ADC都有各自的优缺点，可以满足不同的具体应用要求。A/D 转换的性能指标有：量化误差、分辨率、转换速度、偏移误差、满刻度误差、线性度。

下面是HX711的基本原理介绍：

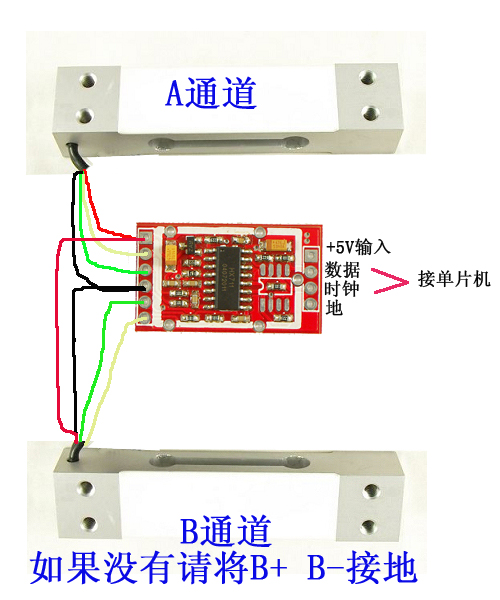


图 3-2 HX711 实物图

（一）模拟输入：

通道A(如图3-2)模拟差分输入可以直接连接到桥式传感器的差分输出。由于从电桥传感器输出的信号很小，该通道的可编程增益高达128或64，以最大化A/D转换器的输入动态范围。与这些增益相对应的满量程差分输入电压分别为±20mV或±40mV。

通道B(如图3-2)的固定增益为32，对应的满量程差分输入电压为±80mV。通道B用于检测系统参数，包括电池。

（二）电源

数字电源（DVDD）必须使用与MCU芯片相同的数字电源。 HX711芯片的稳压电路可同时向A / D转换器和外部传感器提供模拟电源。稳压电源电压（VSUP）可以与数字电源（DVDD）相同。稳压电源的输出电压值（VAVDD）由外部分压电阻R1和R2，芯片输出参考电压VBG（图1）和VAVDD = VBG（R1 + R2）/ R2确定。输出电压应至少选择为低于稳压电源的输入电压（VSUP）100mV。

如果不使用片上稳压器电路，则VSUP引脚必须连接到DVDD或AVDD高压引脚。 VBG引脚上不需要外部电容器。 VFB引脚必须接地，而BASE引脚未连接。时钟选择当XI引脚接地时，HX711自动选择使用内部时钟振荡器，并自动关闭外部时钟输入和晶体振荡器的相关电路。在这种情况下，典型的输出数据速率为10Hz或80Hz。如果您需要精确的输出数据速率，则可以通过20pF直流隔直电容将外部输入时钟连接至XI引脚，或者将晶体连接至XI和XO引脚。在这种情况下，芯片的时钟振荡器将自动关闭

闭合时使用晶体振荡器时钟或外部输入时钟电路。然后，如果晶振频率为11.0592MHz，则输出数据速率恰好为10Hz或80Hz。输出数据速率和晶体频率将与上述关系成比例地增加或减少。使用外部输入时钟时，外部时钟信号不必一定是方波。单片机晶体输出引脚上的时钟信号可以通过20pF直流隔离电容器作为外部时钟输入连接至XI引脚。外部时钟输入信号的幅度可以低至150mV。

（三）串行通讯

串行通信线路由引脚PD\_SCK和DOUT组成，用于数据输出，输入通道和增益选择。当数据输出引脚DOUT为高电平时，

A / D转换器尚未准备好输出数据。此时，串行时钟输入信号PD\_SCK必须为低电平。当DOUT从高电平变为低电平时，PD\_SCK需要输入25至27个时钟脉冲（图2）。在第一个时钟脉冲的上升沿读取24位数据的最高有效位（MSB），直到完成第24个时钟脉冲为止，然后逐位输出24位输出数据，最高有效位到最低将会完成。第25至第27个时钟脉冲用于选择输入通道和下一次A / D转换的增益，如下所示。

（四）HX711 芯片引脚图：

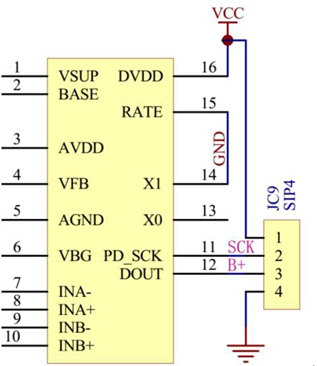


图 3-2.1 HX711 引脚功能图 如上图 3-2.1 所示，HX711AD 转换芯片的工作原理是：AD 转换芯片的外部电路将从称重传感器直流电桥输出的模拟信号进行滤波，再由 AD 转换芯片的模拟输入通道将模拟信号进行128 倍的增益，然后采样为 24bit 的数字信号输入到单片机。

表 3-2 HX711AD 转换器引脚功能表

|  |  |  |
| --- | --- | --- |
| 引脚标号 | 引脚符号 | 引脚功能说明 |
| 1 | VSUP | 稳压电路供电电源端口 稳压供电电压为 2.6 ~ 5.5V |
| 2 | BASE | 模拟输出端口 用作稳压电路控制输出 |
| 3 | AVDD | 模拟电源输入端，输入电压为: 2.6 ~ 5.5V |
| 4 | VFB | 模拟输入端 稳压电路控制输入（不用稳压电路时应接地） |
| 5 | AGND | 模拟地端 |
| 6 | VBG | 模拟输出端 参考电源输出 |
| 7 | INNA | 模拟输入端 通道 A 负输入端 |
| 8 | INPA | 模拟输入端、通道 A 正输入端 |
| 9 | INNB | 模拟输入端、通道 B 负输入端 |
| 10 | INPB | 模拟输入、通道 B 正输入端 |
| 11 | PD-SCK | 数字输入断电控制（高电平有效）和串口时钟输入 |
| 12 | DOUT | 数字输出、 串口数据输出 |
| 13 | XO | 数字输入、输出，晶振输入 |
| 14 | XI | 数字输入、外部时钟或晶振输入端 |
| 15 | RATE | 数字输入、输出数据速率控制端 |
| 16 | DVDD | 电源，供电标准为: 2.6 ~ 5.5V |

* 1. 称重传感器模块

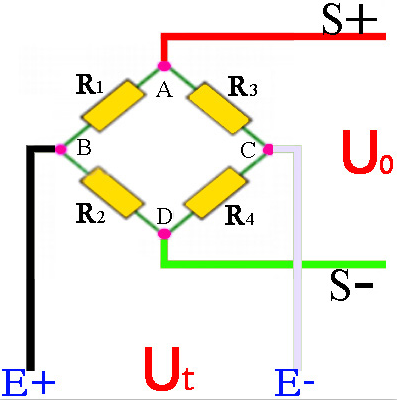


图3.3 电阻应变式压力传感器静态全桥

应变式力传感器中导体电阻随机械变形而变化的这种现象叫做电阻应变效应。用电阻应变计将机械应变信号转换成△R/R后，由于应变和相应电阻的变化很小，很难直接准确地测量，且不便于处理。因此，应变片△R/R的变化应通过转换电路转换为电压或电

流的变化。它的转换电路经常被用来测量电桥。

应变式力传感器具有以下特性：

（1） 应变式可以制成各种机械传感器。

（2） 高灵敏度、高精度。

（3） 量产化，使用便捷，可用于实现远程自动测量。

（4） 应变式力传感器的结构比较简易，对测试影响小，在复杂环境适中应性强，在高温、高压、强磁场等环境下，还能具有良好的响应。

* 1. LCD12864 液晶显示模块

显示模块采用LCD12864液晶显示器。在数字电路中，所有数据都保存为0和1，并且可以在LCD控制器上执行不同的数据操作以获得不同的结果。对于显示英文操作，由于英文字母的类型很少，因此仅需要8位（一个字节）。对于中文，它通常被使用，但是超过6000个，因此我们的前辈想到了一种表示ASCII表的128个字符的方法，而该表很少在两组中用来表示中文字符，即汉字的内部代码。其余的低128位保留给英文字符，即英文内部代码。

那么，获得汉字的内码后，它仍然只是一组数字，如何在屏幕上显示呢？这涉及文本的字体。尽管字体也是一组数字，但是其含义已从数字的含义根本上发生了变化。它使用数字信息记录英文或中文字符的形状。

12864是图形点矩阵液晶显示器，主要由行驱动器/列驱动器和128x64全点矩阵液晶显示器组成。 图形显示是可能的，并且还可以显示8 x 4（16 x 16点矩阵）汉字。

LCD12864液晶模块的引脚功能图如下图：



图 3-5 LCD1280 液晶模块引脚功能图

* 1. 超重报警模块

根据智能电子秤的设计需求，设计的这个模块的功能是：称重智能电子秤的重量时，如果被称量物品的重量超过智能电子秤范围的上限，即大于10kg，则发出命令进行 二极管闪烁，提醒用户避免超过称重范围的称重物品引起电子秤损坏或破坏。

* 1. 复位电路

智能电子秤每次使用时都需要复位（初始化）单片机，以使单片机系统的组件和电子秤的功能组件可以从初始状态开始工作。本设计使用了按键复位的方法，当然，也可以关机后再开机。如图3-7所示。

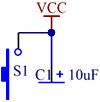




图 3-7 按键复位的电路图

如上所示，复位电路由一个独立的按钮，一个10kΩ电阻和一个10μF电容器组成。 复位电路工作位置的基本原理：使用按键复位时，复位电路中的电容器将向微控制器的RST引脚提供短时高电平信号。 它逐渐变弱，这表明RST端子上的高电平持续时间取决于电容器的充电时间。 因此，必须将RST端子上的高电平信号保持足够长的时间（两个机器周期或更长时间），以确保可以可靠地重置系统。复位信号必须在两个或两个以上的机器周期内处于高电平。 满足以上两点，微控制器可以响应并重置系统。

* 1. 矩阵键盘设计模块

当操作中需要较多的按键时，为了减少单片机的I/O口占用，通常采用矩阵式排列，即矩阵键盘。在矩阵键盘中，每一条水平线和垂直线在交点处不是直接连接的，而是通过一个键连接的。这样，一个端口（如P3端口）就可以形成4\*4=16个键，这是直接用端口线做键盘的两倍，而且线越多，区别就越明显。例如，添加另一行可以形成一个有20个键的键盘，而直接使用端口行只能再生成一个键（9个键）。因此，当键盘所需按键数较大时，采用矩阵法制作键盘是合理的。矩阵键盘的电路图如图3.8所示。



图 3-8 矩阵按键输入模块

每个按键相对应应的功能表如3-1所示。

表3-1 矩阵键盘对应功能表

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | 8 | 9 | 输入单价 |
| 4 | 5 | 6 | 去皮 |
| 1 | 2 | 3 | 重新输入 |
| . | 0 | 退格 | 确定 |

软件设计

* 1. 软件开发环境

该设计使用KeilμVision4进行编程。Keil C51是应用于PLC编程的软件开发工具，具有功能强大的仿真器，调试器，宏汇编器，单板计算机和符合生产标准的Keil C编译器。 基于嵌入式软件开发，详细的研发进度使您可以准确地模拟嵌入式开发的硬件配置，并使用详细的测试报告轻松合理地分析定量数据。

* 1. 主程序流程图

主函数main（）是程序的起始函数，完整的程序必须包含此函数。 在此功能开始时，需要先初始化MCU和某些外围设备，然后才能正常使用它们。 初始化并重新分配一些变量。 初始化后，进入死循环。 如果不进入死循环，程序将退出一次。 如果添加无限循环程序，它将继续循环以实现实时检测和执行的目的。 在设计主程序时，应注意，在主函数中放置过多代码是不合适的。 特定代码通常由函数封装，然后在主函数中调用，因此也可以轻松读取和修改。 具体的流程图如下面的4.1所示。



图 4-1 主程序流程图

* 1. LCD12864显示函数流程图

LCD12864液晶显示程序的模块介绍如下。本液晶是自带字库的，所以不需要对每个led点进行设置，我们几乎仅需要关注写入字符的起始地址就行了，其实地址设置好后后面的字符地址会自动加一。我们也可以在液晶屏上显示动画，本设计就设计了一个开机动画，函数命名为LCD12864\_image3216。

显示功能的流程图在4.2中显示。



图 4-2 显示部分流程图

* 1. 矩阵键盘检测函数的流程图

程序中矩阵键盘的具体测试方法如下（流程图中KEY代表P3口）。

（1） 首先，将键盘上所有的p3.0-p3.3线设置为低电平，然后检查p3.5-p3.7线是否有低电平现象。如果一列中有一个低电平，它将证明该列中的四个键之一已按下。如果线路中没有低电平，则没有按键。

（2） 当确认按下某个键时，将进一步确定按下哪个键。方法是：将四行p3.0-p3.3按顺序设置为低电平，即当一行低时，另一行保持在高电平。然后，通过确定当某条线是低电平时，如果第一步中获得的列是低电平，则可以确定该线与第一步中获得的列相交的键是按下的键。矩阵键盘检测功能流程图见4.4。



图 4-3 矩阵键盘按键部分流程图

* 1. 超重报警部分流程图

智能电子秤超重报警部分流程图如下图 4-4 所示

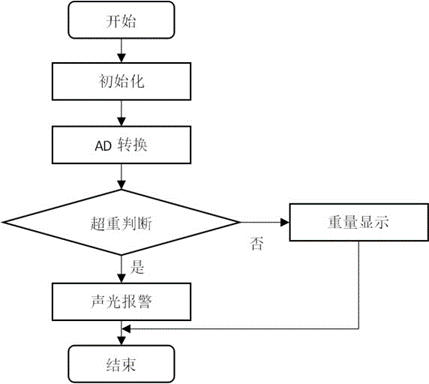


图 4-4 超重报警部分流程图

超重警报部分的工作方法是：称重传感器将物品的重量转换为可测量的电压变化，然后AD转换器将该模拟量转换为数字量并将其发送到微控制器。 比较范围。 如果物品的重量超过智能电子秤范围的最大值，单片机将调用超重警报子程序并发出指令，使LED灯亮起。

系统调试与调试结果

* 1. 硬件电路的制作

根据设计的原理图，将每个器件根据原理图合理的放置于电路板上，然后再一步一步的进行焊接。实物图如图 5-1 所示。

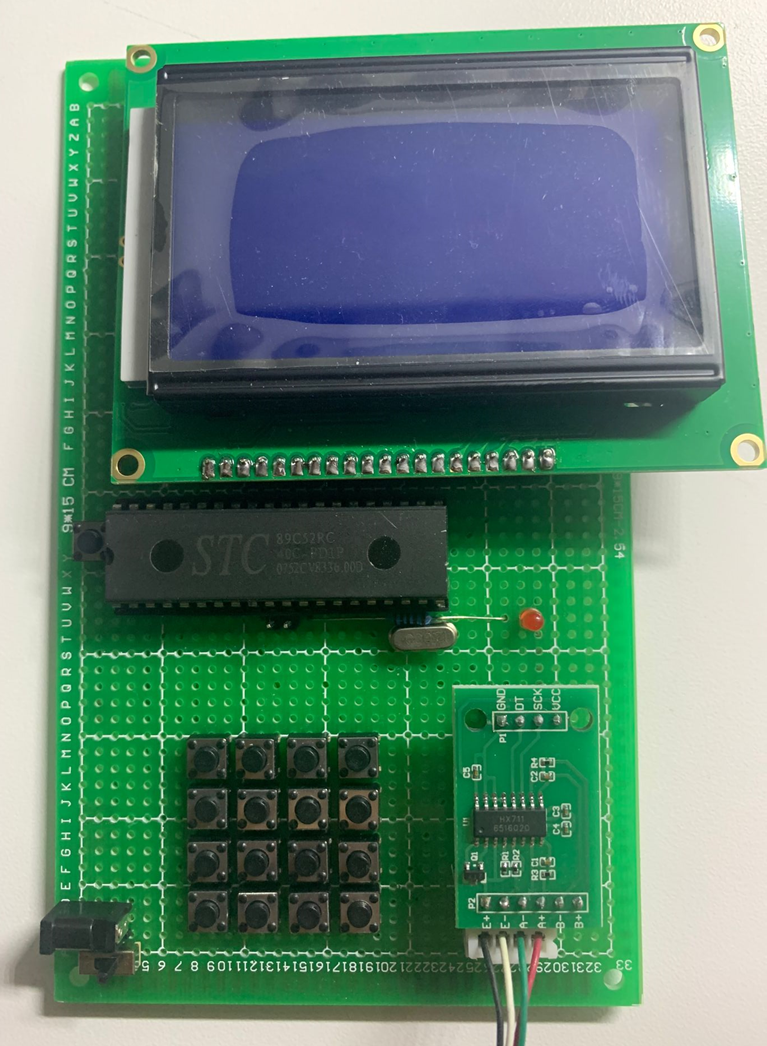


图 5-2 整体系统电路板实物图

* 1. 硬件电路的调试和结果
     1. 硬件电路的调试及结果

1.焊接完硬件部分的所有组件后，使用万用表分别测量每个组件和电路的每个部分，以检测组件是否有异常功能，以及电路是否存在短路和断路不符合设计要求。正常工作并修改错误电路的组件。

2.检查并修改编写的程序。最终检查正确后，将STC89C52 单片机安装到开发板上，将程序刻录到MCU中，然后将MCU移到电子秤的单片机基座中。

3.万用表检测电路完全正确后，连接电源并打开开关，LCD12864液晶显示屏上显示的参数正常。

4.操作矩阵按钮，等待片刻，归零，设置单价，然后将其放置在电子秤的称重平台上。此时，电子秤显示器的显示值与重量本身校准的重量值一致。然后按复位按钮，电子秤显示器的重量显示值返回到“0”。将估计重量大于10kg的物品缓慢放在称重平台上。当电子秤上被称重的物品的重量刚刚超过10kg时，led灯将亮起，电子秤的超重警报系统将正常工作。

硬件电路测试后，得出的结果记录如下表5-1 所示。

表 5-1 物品重量称量结果表

|  |  |  |
| --- | --- | --- |
| 物品名称 | 标准重量（g） | 智能电子秤示值（kg） |
| 5g 砝码 | 5 | 0.005 |
| 10g 砝码 | 10 | 0.010 |
| 20g 砝码 | 20 | 0.020 |
| 50g 砝码 | 50 | 0.050 |
| 一元硬币 | 6.05 | 0.006 |
| 5kg 秤砣 | 5000 | 4.993 |
| 10kg 秤砣 | 10000 | 9.991 |
| 20kg 秤砣 | 20000 | 超重报警 |

如表5-1所示，根据称重物品的数据，便携式智能电子秤的称重物品的测量精度，量程和测量误差均满足设计要求，其功能模块如下： 电子秤可以正常工作。 设计达到预期的效果。

* 1. 软件系统的制作
     1. 软件系统的调试及结果

测试所需的工具：KEIL软件，系统硬件，PL2303下载器等。

系统软件由KEIL软件编写，.HEX文件由编写的程序生成，然后通过PL2303下载器下载到微控制器。通过观察整个系统的运行状态，然后反复修改和调试该程序，最终获得了一个完美的程序。

系统软件调试主要遇到以下问题：

（1）LCD12864显示问题。

解决方案：本设计中使用LCD12864字体显示和图形显示。但是有些时候会显示花屏。首先，只要显示一个字体，它会自己清除掉显示的内容，也会清掉字体的内容。结果并非如此。未清除屏幕后，图形内容将重叠并导致出现花屏现象。通过检查LCD手册，可以发现字体显示和图形显示不同，以后可以修改程序。操作字体显示时，首先清除图形显示，反之亦然。 。

（2）有一种读取矩阵键盘按键的方法。在程序中，您需要将读取的每个键值与每个键对应起来，并赋予其特定的功能。如果直接直接手动计算键值，则似乎可以比较工作量。较大，可能会导致计算错误，从而导致大量时间浪费在调试上。

解决方案：因为此设计中有一个显示设备，所以可以将获得的键值直接显示在显示设备上，这样，当按下每个键时，相应的键值代码会显示在显示屏上，然后依次记录下来。即，每个按钮的设计功能以统一的方式执行。这样可以节省大量时间并确保正确性。

设计总结

随着集成电路和计算机技术的快速发展，现代的电子设备每隔几年，就会发生巨大的变化，传统的设备逐渐的被淘汰，智能机器的核心部件还是单片机，与高价格相比，实现了广泛的应用和发展，智能机加快发展，传感器作为对象信息，在测量控制系统的入口逐渐受到人们的重视，因此要充分掌握相关智能机、微型控制器、传感器和各部分之间的关系，才能满足要求。

虽然，在设计过程中遇到了一些困难，但是在查阅资料与指导老师的帮助下，问题都得到了解决，通过自己的实践增强实践能力，通过实际工程的设计，我可以看出书中所学的知识和实际运用的差异，也更进一步了解了单片机系统的设计。

参考文献

[1]胡向东. 传感器与检测技术[M]. 机械工业出版社，2013.9.

[2]谢维成. 单片机原理及 C51 程序设计[M]. 清华大学出版社，2014.1.

[3]王祁. 智能仪器设计基础[M]. 机械工业出版社，2015.2.

[4]谭浩强. C 语言设计教程[M]. 清华大学出版社，2013.8.

[5]朱巍. 微机原理及接口技术[M]. 人民邮电出版社，2016.1.

[6]马岚. 数字集成电路[M]. 电子工业出版社，2017.1.

[7]李阳. 电子与自动化类毕业设计指导[M]. 中国电力出版社，2016.5.

[8]韦建英.徐安静 模拟电子技术[M]. 华中科技大学出版社，2013.2.

[9]刘理云. 嵌入式单片机开发与应用[M]. 北京理工大学出版社，2016.1.

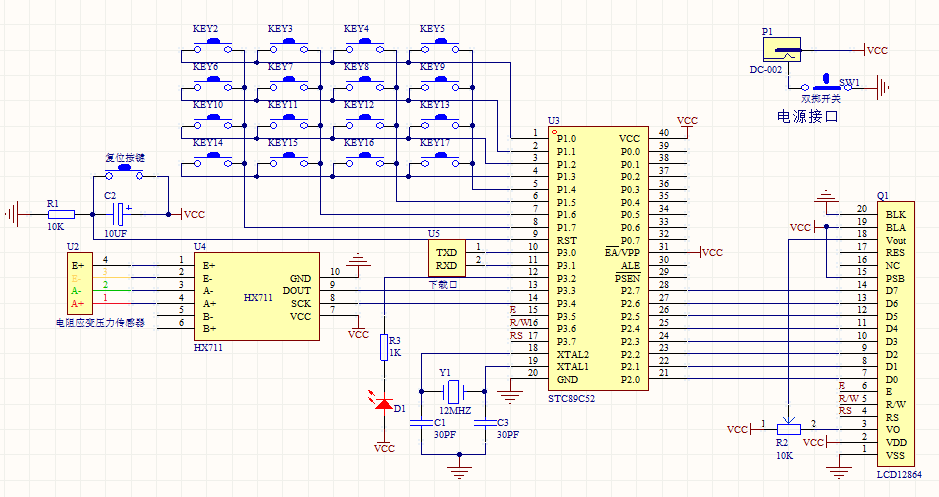
[10]魏芬. 基于 proteus 的单片机实验与课程设计[M]. 清华大学出版社，215.3.

致谢

本论文是在指导老师及相关材料的指导下完成的。本毕业设计是自己四年大学生涯的总结，是一项综合的应用实践。毕业设计也是过去四年对自己的考验。也可以将其视为填补空白的泄漏检查。不会的知识点会通过导师需求帮助，或者查阅相关的资料，也能检验自己的实践能力。在重庆三峡学院度过了四年的时光，在其期间，不仅结识了热情友好的同学与老师，还学习到了宝贵的知识，学到了做人、做事的道理。还有就是感谢我的学校，给了我一个学习的平台，让我接触到优秀的老师同学，老师渊博的知识与高度敬业的精神对我产生了重要的影响，也给我了深深的启迪。还有就是与同学的相处，让我学会了为人处世的道理。最好，再次对帮助我的老师同学表示衷心的感谢。

附录

附录一 智能电子秤原理图



附录二 源程序

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

HX711头文件

实现功能：HX711的控制显示

补充说明：

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

#ifndef \_HX711\_H\_

#define \_HX711\_H\_

#include <reg52.h>

#define uchar unsigned char

#define uint unsigned int

#define ulong unsigned long

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*HX711引脚定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

sbit ADDO = P3^3;

sbit ADSK = P3^4;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*HX711函数定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ulong ReadCount(void);

ulong fil1();

ulong fil2();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*HX711变量定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ulong init\_val; //存储零点重量

ulong value; //存储AD数值

//ulong weight; //存储实际重量

uint ii;

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

函数名称:ulong ReadCount(void)

函数作用:读取AD值

参数说明:

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

ulong ReadCount(void)

{

unsigned long Count;

unsigned char i;

ii=0;

ADSK=0; //使能AD（ PD\_SCK 置低）

ADDO=1;

Count=0;

while(ADDO&&ii<8000) //AD转换未结束则等待，否则开始读取

ii++;

for (i=0;i<24;i++)

{

ADSK=1; //PD\_SCK 置高（发送脉冲）

Count=Count<<1; //下降沿来时变量Count左移一位，右侧补零

ADSK=0; //PD\_SCK 置低

if(ADDO) Count++;

}

ADSK=1;

Count=Count^0x800000; //第25个脉冲下降沿来时，转换数据

ADSK=0;

return(Count);

}

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

函数名称:ulong fil()

函数作用:读取AD值

参数说明:

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

ulong fil()

{

uchar i;

ulong val=0; //记录采集值

for(i=0;i<3;i++) //循环采集5次

val+=ReadCount();

return val/3; //返回 中间数值 的数据

}

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

函数名称:ulong fil2()

函数作用:读取AD值。这个用于获取初值，增加采集次数和时间间隔使的更精确

初始都不准，那后面怎么测都会不准

参数说明:

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

ulong fil2()

{

uchar i,j;

ulong temp;

ulong val[9];

for(i=0;i<9;i++)

{

val[i]=ReadCount();

LCD12864\_delay(3);

}

for(i=0;i<9;i++)

{

for(j=0;j<9-i;j++)

{

if(val[j]>val[j+1])//把>改成<就是从小到大

{

temp=val[j];

val[j]=val[j+1];

val[j+1]=temp;

}

}

}

return val[4];

}

#endif

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

LCD12864头文件

实现功能：LCD12864的控制显示

补充说明：

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

#ifndef \_LCD12864\_H\_

#define \_LCD12864\_H\_

#include <reg52.h>

#include<intrins.h>

#define uchar unsigned char

#define uint unsigned int

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LCD12864引脚定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define LCD P2 //并行数据口D0~D7

sbit RS =P3^7; //数据/命令选择 引脚

sbit RW =P3^6; //读/写选择 引脚

sbit E =P3^5; //使能信号 引脚

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LCD12864函数定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD12864\_delay(uint x); //LCD12864延时

void write\_com(uchar com); //LCD12864写命令

void write\_data(uchar dat); //LCD12864写数据

void LCD12864\_display\_string(uchar x,uchar y,uchar \*s); //在第y行，x+1列开始显示字符串

void LCD12864\_image3216(uchar x,uchar y,uchar code \*pPicture);//LCD12864在横坐标x（0~7），纵坐标y（1~4）开始显示一个32\*16的图片

void LCD12864\_clear3216(uchar x,uchar y); //LCD12864在横坐标x（0~7），纵坐标y（1~4）开始清除一个32\*16的图片

void LCD12864\_clear12864(); //LCD12864清除整个屏幕的画板

void LCD12864\_init(void); //LCD12864初始化函数

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LCD12864变量定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uchar code num12864[]=

{

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFE,0x3F,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xFF,0x3E,0x73,0xE4,0xF8,0x03,0xE7,0xE0,0x08,0x01,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xE0,0x03,0x00,0x40,0x09,0x33,0x00,0x7F,0x8E,0x01,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xFC,0x13,0xC8,0x4C,0xF9,0x33,0x26,0x7F,0x18,0x03,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xE0,0x06,0x08,0x00,0x18,0x03,0x00,0x7E,0x3E,0x47,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xF1,0x9F,0x08,0x00,0x19,0x33,0x26,0x40,0x04,0x01,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xF9,0x8F,0x00,0x40,0xF8,0x03,0x26,0x7E,0x78,0x27,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xF0,0x0F,0x00,0x48,0xF9,0x33,0x00,0x3E,0x78,0x67,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xF2,0x27,0x39,0xC8,0x71,0x33,0x27,0x3E,0x7E,0x67,0x87,0xF0,0xFE,0x1F,0xFF,

0xFF,0xE4,0xF2,0x00,0x03,0x03,0x03,0xE0,0x78,0x7E,0x67,0x87,0xF0,0xFE,0x1F,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x87,0xF0,0xFE,0x1F,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

0xF4,0xBF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF8,0x3F,0xFF,0xFB,0xFF,0xFF,

0xF8,0x7F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF8,0x03,0xFF,0x79,0xFF,0xFF,

0xC0,0x05,0xFF,0xFF,0xFF,0xFC,0xFF,0xFF,0xFF,0xFF,0xFF,0xF3,0xEE,0x39,0xFF,0xFF,

0xC0,0x05,0xFF,0xFF,0xFF,0xFC,0xFF,0xFF,0xFF,0xFF,0xFF,0xF3,0xEE,0x39,0xFF,0xFF,

0xF8,0x7F,0xFF,0xFF,0xFF,0xE1,0xFC,0xFF,0xFF,0xFE,0x7F,0xF3,0xC8,0x09,0xFF,0xFF,

0xF4,0x3F,0xFF,0xFF,0xFF,0x81,0xE3,0xFB,0xFF,0xFF,0x8F,0xFF,0x00,0x81,0xFF,0xFF,

0xF4,0x3F,0xFF,0xFF,0xFF,0x81,0xE3,0xFB,0xFF,0xFF,0x8F,0xFF,0x00,0x81,0xFF,0xFF,

0xCC,0xDF,0xFF,0xFF,0xFC,0x03,0x03,0xC3,0xFE,0xFF,0xF3,0xFE,0xF1,0xC3,0xFF,0xFF,

0xFC,0xFE,0xFF,0x0F,0xF8,0x00,0x03,0x06,0x7F,0x0F,0xFC,0x7F,0xF1,0xFB,0xFF,0xFF,

0xFF,0xFF,0x3E,0x07,0xE0,0x00,0x00,0x04,0x5F,0x80,0xFE,0x1F,0xF7,0xFF,0xE7,0xFF,

0xFF,0xFF,0x3E,0x07,0xE0,0x00,0x00,0x04,0x5F,0x80,0xFE,0x1F,0xF7,0xFF,0xE7,0xFF,

0xFC,0xFD,0x3C,0xE3,0xC0,0x00,0x07,0x88,0x41,0xF0,0x0F,0x83,0xFF,0xFF,0xBF,0xFF,

0xFF,0xFE,0x1D,0x13,0xC0,0x00,0x00,0x40,0x40,0x1E,0x01,0xC1,0xFE,0x3F,0x9B,0xFF,

0xFF,0xFE,0x1D,0x13,0xC0,0x00,0x00,0x40,0x40,0x1E,0x01,0xC1,0xFE,0x3F,0x9B,0xFF,

0xFF,0xFA,0x1C,0x6B,0x80,0x00,0x00,0x01,0xC0,0x00,0x00,0x20,0xF8,0x0F,0x87,0xFF,

0xFF,0xFC,0x0C,0x68,0x00,0x00,0x00,0x01,0xC0,0x78,0x00,0x00,0x30,0xC6,0x1D,0xFF,

0xFF,0xE4,0x02,0x10,0x00,0x00,0x00,0x02,0x40,0x80,0x00,0x00,0x31,0x36,0x1B,0xFF,

0xFF,0xE4,0x02,0x10,0x00,0x00,0x00,0x02,0x40,0x80,0x00,0x00,0x31,0x36,0x1B,0xFF,

0xFF,0xF8,0x03,0x00,0x00,0x03,0xF8,0x02,0x20,0x00,0x00,0x00,0x16,0x84,0x02,0xFF,

0xFF,0xD8,0x03,0x80,0x00,0x1F,0xFC,0x04,0x20,0x00,0x00,0x00,0x06,0x84,0x01,0xFF,

0xFF,0xE0,0x0F,0xE0,0x00,0x00,0x1F,0x08,0x10,0x07,0xF8,0x00,0x01,0x08,0x00,0xFF,

0xFF,0xE0,0x0F,0xE0,0x00,0x00,0x1F,0x08,0x10,0x07,0xF8,0x00,0x01,0x08,0x00,0xFF,

0xFF,0xA0,0x1F,0x60,0x00,0x00,0x78,0x30,0x10,0x8F,0xFE,0x00,0x00,0x3E,0x01,0xFF,

0xFF,0xC0,0x00,0x60,0x00,0x00,0x70,0x40,0x0C,0x7F,0x00,0x00,0x00,0x7F,0x81,0x3F,

0xFF,0xC0,0x00,0x60,0x00,0x00,0x70,0x40,0x0C,0x7F,0x00,0x00,0x00,0x7F,0x81,0x3F,

0xFF,0x40,0x00,0x80,0x00,0x00,0x10,0x80,0x02,0x07,0x00,0x00,0x00,0x40,0x40,0xFF,

0xFF,0x80,0x1F,0x00,0x01,0x8C,0xE0,0x80,0x01,0x07,0x80,0x00,0x00,0x40,0x00,0xEF,

0xFF,0xC0,0xE1,0x00,0x00,0x42,0x03,0x00,0x00,0x81,0x80,0x00,0x00,0x40,0x00,0xC7,

0xFF,0xC0,0xE1,0x00,0x00,0x42,0x03,0x00,0x00,0x81,0x80,0x00,0x00,0x40,0x00,0xC7,

0xFF,0x41,0x20,0x80,0x01,0xA1,0x03,0x00,0x00,0x61,0x00,0x00,0x00,0x70,0x00,0xEF,

0xFC,0x02,0x10,0x80,0x00,0x40,0x03,0x00,0x00,0x60,0xC4,0x20,0x00,0x48,0x00,0xFF,

0xFC,0x02,0x10,0x80,0x00,0x40,0x03,0x00,0x00,0x60,0xC4,0x20,0x00,0x48,0x00,0xFF,

0xFF,0x02,0x0C,0x60,0xE0,0x00,0x03,0x00,0x00,0x10,0x08,0x40,0x1F,0x80,0x00,0x2F,

0xFF,0x84,0x0C,0x10,0x9C,0x00,0x04,0x00,0x00,0x10,0x31,0xA0,0x28,0xC2,0x00,0x3F,

0xFB,0x04,0x02,0x0F,0x1B,0xC0,0x18,0x00,0x00,0x10,0x00,0x40,0xC6,0x31,0x80,0x2F,

0xFB,0x04,0x02,0x0F,0x1B,0xC0,0x18,0x00,0x00,0x10,0x00,0x40,0xC6,0x31,0x80,0x2F,

0xFF,0x04,0x02,0x00,0x18,0x7F,0xE0,0x00,0x00,0x10,0x00,0x01,0xC1,0x00,0x40,0x1F,

0xFB,0x84,0x02,0x00,0x18,0x40,0x00,0x00,0x00,0x08,0x00,0x0E,0xC1,0x00,0x00,0x37,

0xFB,0x84,0x02,0x00,0x18,0x40,0x00,0x00,0x00,0x08,0x00,0x0E,0xC1,0x00,0x00,0x37,

0xFF,0x04,0x02,0x00,0x18,0x40,0x00,0x00,

};

uchar code ASI[]=

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0xE0,0x40,0x00,

0x00,0x00,0x40,0x00,0x00,0x00,0x5D,0x80,0x7F,0xF8,0x89,0x00,0x08,0x80,0x8A,0x1B,//

0x08,0x80,0x8A,0x26,0x08,0x81,0x0C,0x26,0x08,0x81,0x0A,0x26,0x08,0x81,0x09,0x1E,

0x10,0x8A,0x09,0x06,0x20,0x8A,0x1C,0xA6,0xC0,0x78,0x00,0x1C,0x00,0x00,0x00,0x00,//单价.BMP0

};

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

函数名称:void LCD12864\_delay(uint x)

函数作用:LCD12864延时函数

参数说明:

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

void LCD12864\_delay(uint x)

{

uint j,i;

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

{

for(i=0;i<120;i++);

}

}

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

函数名称:uchar Lcd\_CheckBusy(void)

函数作用:LCD12864读忙函数

参数说明:

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

uchar Lcd\_CheckBusy(void)

{

unsigned char Busy;

LCD=0xff;

RS=0;

RW=1;

E=1;

\_nop\_();

Busy=LCD&0x80;

E=0;

return Busy;

}

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

函数名称:void write\_com(uchar com)

函数作用:LCD12864写命令

参数说明:com为LCD12864指令，参考手册

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

void write\_com(uchar com)

{

while(Lcd\_CheckBusy());

RS=0;

RW=0;

E=0;

\_nop\_();

\_nop\_();

LCD=com;

\_nop\_();

\_nop\_();

E=1;

\_nop\_();

\_nop\_();

E=0;

}

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

函数名称:void write\_data(uchar dat)

函数作用:LCD12864写数据

参数说明:dat为所写内容

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

void write\_data(uchar dat)

{

while(Lcd\_CheckBusy());

RS=1;

RW=0;

E=0;

\_nop\_();

\_nop\_();

LCD=dat;

E=1;

\_nop\_();

\_nop\_();

E=0;

}

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

函数名称:void LCD12864\_display\_string(uchar x,uchar y,uchar \*s)

函数作用:LCD12864显示字符

参数说明:在横坐标x（0~7），纵坐标y（1~4）显示字符串\*s

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

void LCD12864\_display\_string(uchar x,uchar y,uchar \*s)

{

uchar add; //存储显示位置

if(y==1) //在第1行显示

add=0x80+x;

else

if(y==2) //在第2行显示

add=0x90+x;

else

if(y==3) //在第3行显示

add=0x88+x;

else

if(y==4) //在第4行显示

add=0x98+x;

write\_com(add); //先写显示地址

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

{

write\_data(\*s);

s++;

LCD12864\_delay(1);

}

}

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

函数名称:void LCD12864\_image3216(uchar x,uchar y,uchar code \*pPicture)

函数作用:LCD12864显示一张32\*16像素图片

参数说明:在横坐标x（0~7），纵坐标y（1~4）开始显示一个32\*16的图片

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

void LCD12864\_image3216(uchar x,uchar y,uchar code \*pPicture)

{

uchar add,i,j;

if(y%2==1)

add=0x80;

else

if(y%2==0)

add=0x90;

write\_com( 0x34 ) ;

write\_com( 0x36 ) ;

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

{

write\_com(add+i) ;

if(y>2)

write\_com(0x88+x);

else

write\_com(0x80+x);

for(j=0;j<2;j++)

{

write\_data(pPicture[i\*4+j\*2] );

write\_data(pPicture[i\*4+j\*2+1] );

}

}

write\_com( 0x30 ) ;

}

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

函数名称:void LCD12864\_clear3216(uchar x,uchar y)

函数作用:LCD12864清除一张32\*16像素图片

参数说明:在横坐标x（0~7），纵坐标y（1~4）开始清除一个32\*16的图片

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

void LCD12864\_clear3216(uchar x,uchar y)

{

uchar add,i,j;

if(y%2==1)

add=0x80;

else

if(y%2==0)

add=0x90;

write\_com( 0x34 ) ;

write\_com( 0x36 ) ;

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

{

write\_com(add+i) ;

if(y>2)

write\_com(0x88+x);

else

write\_com(0x80+x);

for(j=0;j<2;j++)

{

write\_data(0x00);

write\_data(0x00);

}

}

write\_com( 0x30 ) ;

}

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

函数名称:void LCD12864\_image12864( uchar code \*pPicture )

函数作用:LCD12864显示一张128\*64的图片

参数说明:输入参数为字模数组，可以用图片取模软件获得

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

void LCD12864\_image12864( uchar code \*pPicture )

{

unsigned char i,j;

write\_com( 0x34 ) ;

write\_com( 0x36 ) ;

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

{

write\_com( 0x80+i);

write\_com( 0x80 );

for(j=0;j<8;j++)

{

write\_data( ~pPicture[i\*16+j\*2] );

write\_data( ~pPicture[i\*16+j\*2+1] );

}

}

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

{

write\_com( 0x80+i) ;

write\_com( 0x88 );

for(j=0;j<8;j++)

{

write\_data( ~pPicture[32\*16+i\*16+j\*2] );

write\_data( ~pPicture[32\*16+i\*16+j\*2+1] );

}

}

write\_com( 0x30 ) ;

}

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

函数名称:void LCD12864\_clear12864()

函数作用:LCD12864清除整个屏幕的画板

参数说明:

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

void LCD12864\_clear12864()

{

unsigned char i,j;

write\_com( 0x34 ) ;

write\_com( 0x36 ) ;

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

{

write\_com( 0x80+i) ;

write\_com( 0x80 );

for(j=0;j<8;j++)

{

write\_data(0x00 );

write\_data(0x00 );

}

}

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

{

write\_com( 0x80+i) ;

write\_com( 0x88 );

for(j=0;j<8;j++)

{

write\_data(0x00 );

write\_data(0x00 );

}

}

write\_com( 0x30 ) ;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LCD12864初始化\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD12864\_init(void)

{

write\_com(0x30); //选择基本指令集

write\_com(0x30); //选择8bit数据流

write\_com(0x0c); //开显示(无游标、不反白)

write\_com(0x01); //清除显示，并且设定地址指针为00H

write\_com(0x06); //指定在资料的读取及写入时，设定游标的移动方向及指定显示的移位，光标从右向左加1位移动

}

#endif

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

矩阵键盘头文件

实现功能：矩阵键盘的控制

补充说明：

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

#ifndef \_KEY\_H\_

#define \_KEY\_H\_

#include<reg52.h>

#define uchar unsigned char

#define uint unsigned int

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*矩阵键盘引脚定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define Key P1

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*矩阵键盘函数声明\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uchar jiema(unsigned char key); //解码函数，输入按键编码，返回按键位置

void delay(); //延时函数

uchar keyscan(void); //按键查询函数，返回矩阵键盘位置

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

函数名称:uchar jiema(unsigned char key)

函数作用:转换按键码为1~16的数字

参数说明:返回按下的按键位置

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

uchar jiema(unsigned char key)

{

uchar n;

switch(key)

{

case 0x11: n= '.'; break;

case 0x21: n= '0'; break;

case 0x41: n= 'T'; break;

case 0x81: n= '='; break;

case 0x12: n= '1'; break;

case 0x22: n= '2'; break;

case 0x42: n= '3'; break;

case 0x82: n= 'C'; break;

case 0x14: n= '4'; break;

case 0x24: n= '5'; break;

case 0x44: n= '6'; break;

case 0x84: n= 'Q'; break;

case 0x18: n= '7'; break;

case 0x28: n= '8'; break;

case 0x48: n= '9'; break;

case 0x88: n= 'D'; break;

default: break;

}

return n;

}

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

函数名称:void delay()

函数作用:延时函数

参数说明:

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

void delay() //延时子程序

{

uchar m;

for (m = 5; m > 0; m--);

}

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

函数名称:uchar Keyscan(void)

函数作用:进行按键扫描

参数说明:返回按键值，=0时表示没有按键按下

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

uchar keyscan(void) //按键扫描程序 P1.0--P1.3为行线 P1.4--P1.7为列线

{

unsigned char rcode, ccode;

Key = 0xF0; // 发全0行扫描码，列线输入

if((Key&0xF0) != 0xF0) // 若有键按下

{

delay();// 延时去抖动

if((Key&0xF0) != 0xF0)

{

rcode = 0xFE; // 逐行扫描初值

while((rcode&0x10) != 0)

{

Key = rcode; // 输出行扫描码

if((Key&0xF0) != 0xF0) // 本行有键按下

{

ccode = (Key&0xF0)|0x0F;

do{;}

while((Key&0xF0) != 0xF0); //等待键释放

return jiema((~rcode) + (~ccode)); // 返回键编码

}

else

rcode = (rcode<<1)|0x01; // 行扫描码左移一位

}

}

}

return 0xff; // 无键按下，返回值为0

}

#endif

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

电子秤

补充说明：

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

#include<reg52.h>//头文件

#include<lcd12864.H>

#include<HX711.h>

#include<KEY.h>

#include<math.h>

#include<stdio.h>

#define MAX 3 //宏定义，单价最大输入3位数

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*引脚定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

sbit led=P3^2; //超重指示灯

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*变量定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uchar key; //存储按键值

double price=0; //存储零时单价

double prices=0; //存储最终单价

bit price\_f=0; //记录输入单价标志位

bit price\_w=0; //记录输入单价完成标志位

uchar price\_z=0; //存储总价

bit dian\_f=0; //小数单价标志位

bit dian\_w=0; //小数单价标志位

uchar n=0; //记录当前输入了几位数

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

函数名称:void fixed\_display()

函数作用:固定显示函数

参数说明:

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

void fixed\_display()

{

LCD12864\_display\_string(2,1,"电子秤"); //2：表示第3列，1：表示第1行，所有关于显示都一样

LCD12864\_display\_string(0,2,"重量:"); //0：表示第1列，2：表示第2行，所有关于显示都一样

LCD12864\_display\_string(0,3,"单价: 0");

LCD12864\_display\_string(0,4,"总价:");

LCD12864\_display\_string(7,2,"g");

LCD12864\_image3216(6,3,ASI); //6：表示第7列，3：表示第3行，显示【元/Kg】

LCD12864\_display\_string(7,4,"元");

}

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

函数名称:void chuli\_num(uchar keys)

函数作用:在输入单价的时候按下数字按键处理函数

参数说明:

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

void chuli\_num(uchar keys)

{

if((price\_f==1&&price\_w==0&&n<MAX)||(dian\_f==1&&dian\_w==0))//判断是否为输入单价状态，并且输入未满最大整数位数或者小数未满一位，才可以继续输入

{

if(dian\_f==0) //判断是否为整数

{

price=price\*10+keys;//单价计算

n++; //输入的位数加1

if(n==MAX) //判断是否输入完成

price\_w=1; //是的话标记输入完成

}

else //否者为。小数

{

price=price+(float)keys/10;//单价计算 12+ 1/10=12.1

dian\_w=1; //标记小数输入完成。因为单价最低也就0.1元，所以只能输入一位小数

price\_w=1; //标记单价输入完成

}

}

}

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

函数名称:void chuli()

函数作用:按键处理函数

参数说明:

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

void chuli()

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*按下数字键1~9键\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(key=='1'||key=='2'||key=='3'||key=='4'||key=='5'||key=='6'||key=='7'||key=='8'||key=='9')//判断当前按下是否为0~9的数字键

chuli\_num(key-0x30);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*按下‘0’键\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(key=='0')

{

if((price\_f==1&&price!=0&&price\_w==0&&n<MAX)||(dian\_f==1&&dian\_w==0))//判断是否为输入单价状态，并且输入未满最大整数位或者小数未满一位，才可以继续输入

{

if(dian\_f==0) //判断是否为整数

{

price=price\*10+0; //单价计算

n++; //输入位数加1

if(n==MAX) //判断是否输入四位完成

price\_w=1; //是，标记输入完成

}

else //小数

{

price=price+0.0; //单价计算

dian\_w=1; //标记小数输入完成。

price\_w=1; //标记单价输入完成

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*按下‘.’键\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(key=='.')

{

if(price\_f==1&&dian\_f==0) //标记位当前单价是具有小数

dian\_f=1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*按下‘去皮’键\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(key=='Q')

{

if(price\_f!=1) //重新获取初值，0kg对应的AD值

init\_val=fil();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*按下‘单价’键\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(key=='D')

{

price\_f=1; //标记位单价输入状态

LCD12864\_display\_string(3,3," ");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*按下‘重输’键\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(key=='C')

{

if(price\_f==1)

{

price\_f=1; //清除所以数据，重新输入单价

price=0;

n=0;

price\_w=0;

dian\_f=0;

dian\_w=0;

LCD12864\_display\_string(3,3," ");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*按下‘退格’键\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(key=='T') //13

{

if(price\_f==1&&price!=0) //判断是否为输入状态，并且当前输入的单价不为0采集必要进行退格处理

{

if(dian\_f==1) //小数

{

price\_w=0;

dian\_w=0;

dian\_f=0;

price=(ulong)price;

LCD12864\_display\_string(3,3," ");

}

else //整数

{

n--; //输入的位数减1

price\_w=0;

price=(ulong)price/10;//单价计算

LCD12864\_display\_string(3,3," ");

}

}

else

if(price\_f==0)

{

prices=0;

//清空显示合计后的价格

LCD12864\_display\_string(3,3," ");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*按下‘确认’键\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(key=='=')

{

if(price\_f==1) //按下确定键后，将输入的单价记录下来

{

prices=price; //记录单价

price\_f=0; //以下清除所有变量

price=0;

n=0;

price\_w=0;

dian\_f=0;

dian\_w=0;

}

}

}

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

函数名称:void main()

函数作用:主函数

参数说明:

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

void main()

{

uchar weight\_s[8]; //存储重量转换为字符串

uchar price\_s[4]="0"; //存储单价转换为字符串

uchar z\_s[8];

uchar wei,i;

LCD12864\_init();

/\*LCD12864\_display\_string(1,2,"初始化.");

init\_val=fil2();

LCD12864\_display\_string(1,2,"初始化..");

init\_val=(init\_val+fil2())/2;

LCD12864\_display\_string(1,2,"初始化...");

init\_val=(init\_val+fil2())/2;\*/

write\_com(0x01); //清除屏幕显示

LCD12864\_image12864(num12864); //开机显示启动画面

init\_val=fil2(); //读取0kg初始值

while(ii>=8000) //判断模块是否插接好

{

LCD12864\_clear12864(); //清除开机画面

LCD12864\_display\_string(2,1,"☆警告☆");

LCD12864\_display\_string(1,2,"未检测到模块");

LCD12864\_display\_string(0,3,"关闭电源后检测下");

LCD12864\_display\_string(0,4,"HX711 是否插接好");

}

init\_val=(init\_val+fil2())/2; //读取0kg初始值，这里多次采集取平均值，为了提高测量精度

init\_val=(init\_val+fil2())/2;

init\_val=(init\_val+fil2())/2;

init\_val=fil2();

init\_val=(init\_val+fil2())/2;

init\_val=(init\_val+fil2())/2;

init\_val=fil2();

init\_val=(init\_val+fil2())/2;

LCD12864\_clear12864(); //清除开机画面

fixed\_display(); //显示固定内容

while(1) //死循环

{

if(price\_f==0) //非单价输入状态

{

value=fil(); //采集实际AD

if(value<init\_val)

value=(init\_val-value)/400.03;//转换成实际重量，其中最后一个数字是指1g所占的AD值

//value=(init\_val-value)/41.220;//转换成实际重量，其中最后一个数字是指1g所占的AD值

else

{

value=0; //重量=0

led=1; //关闭蜂鸣器

}

if(value<=10000) //判断是否超重

//if(value<=100000)

{

led=1; //关闭超重警示

//显示实际重量

wei=sprintf(weight\_s,"%ld",(ulong)value);

//wei=sprintf(weight\_s,"%0.1f",(double)value/10);

for(i=wei/2+3;i<7;i++)

LCD12864\_display\_string(i,2," ");

LCD12864\_display\_string(3,2,weight\_s);

//显示总价

wei=sprintf(z\_s,"%0.1f",(double)value/1000\*prices);

//wei=sprintf(z\_s,"%0.1f",(double)value/10000\*prices);

for(i=wei/2+3;i<7;i++)

LCD12864\_display\_string(i,4," ");

LCD12864\_display\_string(3,4,z\_s);

}

else //超重

{

led=0; //指示灯警示

LCD12864\_display\_string(3,2,"超重 ");//显示“超重”提示

}

}

key=keyscan();//获取按键返回值

if(key!=0xff) //判断是否有按键按下

{

chuli(); //按键处理

if(price\_f==1) //单价输入

{

if(dian\_f==1)

{

sprintf(price\_s,"%0.1f",price); //将价格转换成字符，有小数

LCD12864\_display\_string(3,3,price\_s); //显示价格

LCD12864\_clear3216(6,3);

}

else

{

sprintf(price\_s,"%ld",(ulong)price); //将价格转换成字符，整数

LCD12864\_display\_string(3,3,price\_s); //显示价格

LCD12864\_clear3216(6,3);

}

}

else

{

sprintf(price\_s,"%0.1f",prices); //将价格转换成字符，最终的价格

LCD12864\_display\_string(3,3,price\_s); //显示价格

LCD12864\_image3216(6,3,ASI); //显示【元/Kg】

}

}

}

}