毕业设计(论文)

题 目 基于STM32单片机的电子秤系统设计

学生姓名

学 号

专业班级

指导教师

学 院

答辩日期

目 录

[摘 要 I](#_Toc7572570)

[ABSTRACT II](#_Toc7572571)

[第一章 绪论 1](#_Toc7572572)

[1.1 设计背景 1](#_Toc7572573)

[1.2 设计依据 1](#_Toc7572574)

[1.3 设计目的与主要功能 2](#_Toc7572575)

[第二章 方案设计 3](#_Toc7572589)

[2.1 设计思路 3](#_Toc7572590)

[2.2 系统构成 3](#_Toc7572591)

[第三章 硬件电路设计 5](#_Toc7572592)

[3.1 系统硬件原理图 5](#_Toc7572593)

[3.2 STM32概述 5](#_Toc7572594)

[3.2.1 STM32引脚 6](#_Toc7572595)

[3.2.2 STM32内部原理 7](#_Toc7572596)

[3.3 矩阵键盘电路 8](#_Toc7572597)

[3.4 电源电路 9](#_Toc7572598)

[3.5 重量监测电路 10](#_Toc7572599)

[3.6 LCD1602液晶显示电路 10](#_Toc7572600)

[第四章 系统软件设计 13](#_Toc7572603)

[4.1 软件主程序 13](#_Toc7572604)

[4.2 LCD1602显示子程序 14](#_Toc7572605)

[4.3 串口初始化子程序 14](#_Toc7572606)

[4.4 重量检测子程序 15](#_Toc7572607)

[4.5 按键子程序 16](#_Toc7572608)

[第五章 系统调试 18](#_Toc7572611)

[5.1 焊接调试 18](#_Toc7572612)

[5.2 功能调试 18](#_Toc7572613)

[第六章 总结 19](#_Toc7572614)

[致 谢 20](#_Toc7572615)

[参 考 文 献 21](#_Toc7572616)

基于STM32单片机的电子秤系统

摘 要：本次设计的目的是设计一款基于STM32单片机的电子秤系统，随着科学技术的不断发展，传统的机械秤逐渐被电子秤所取代，相比于计量不精确，费时费力的机械秤，电子秤采用在方便快捷的同时，又保证了结果的准确性。本次设计采用STM32单片机作为处理器，控制各个传感器和外设，采用LCD1602液晶屏作为显示器，显示检测到的重量，压力传感器用于称重，通过检测物体对传感器造成的压力，通过公式计算得出物体的重量，此外通过16个按键组成矩阵键盘，可以输入数字，并进行加减乘除操作，通过重量和单价计算出总价

关键词：电子秤；STM32单片机；LCD1602液晶屏；压力传感器；

Electronic scale system based on STM32 single chip microcomputer

XXX Xxx-xxx, XXX Xxx-xxx

(*Electrical Engineering College, Longdong University, Qingyang* 745000*, Gansu, China*)

**Abstract:** The purpose of this design is to design an electronic scale system based on STM32 single-chip microcomputer. With the continuous development of science and technology, the traditional mechanical scale is gradually replaced by electronic scales. Compared with the inaccurate measurement, time-consuming and laborious mechanical scales, electronics The scale is convenient and fast, and the accuracy of the result is guaranteed. This design uses STM32 single-chip microcomputer as the processor, LCD1602 LCD screen as the display, showing the detected weight, pressure sensor for weighing, by detecting the pressure of the object on the sensor, the weight of the object is calculated by the formula, in addition to 16 The buttons form a matrix keyboard, which can input numbers, perform addition, subtraction, multiplication and division operations, calculate the total price by weight and unit price, and broadcast the weight and price through the voice chip.

**Key words:** electronic scale; STM32 single chip microcomputer; LCD1602 liquid crystal screen; pressure sensor; voice broadcast

# 1 引言

## 设计背景

随着科学技术的发展和生产力的提高，电子产品逐渐取代了一些传统的机器，在以前，人们往往采用机械式的秤砣来进行货物的承重，这种传统的机械秤，虽然能够解决大部分的称重问题，给人们生活带来极大的方便，但是由于其实现方法是以杠杆原理为基础，这种方式称重的结果不是很精确，时间久了，秤砣损坏，还会导致数据的误差进一步加大，更有黑心商人靠着作弊的手段，依靠机械秤做偷斤少量的买卖，造成了很恶劣的影响。随着电子秤的问世，机械秤逐渐淡出了市场，电子秤的种类繁多，从上万吨的大型电子秤到几克的小型电子秤，都是基于电子秤的原理实现的。电子秤的实现原理很简单，将货物放置在称重面上，压力传感器监测货物对其的压力，受到的压力和重量成正比，按照一定的公式，单片机将接收到的压力通过计算得出重量，并显示在液晶屏上进行展示，同时语音芯片进行语音播报。另一个创新点就是电子秤具有计算功能，这一创新功能大大提高了秤的功能性，将计算器和秤结合在一起，通过16个按键组成按键矩阵，进行加减乘除操作，根据重量和单价，很快计算出总价，对于货物的出售带来了极大的方便。为了更好地研究电子秤的工作原理和实现方式，本次设计了一款基于STM32单片机的电子秤系统。

## 设计依据

在单片机选项这一款，相比传统的AT89C52芯片，这次采用更加强大的STM32单片机，集成度更高，价格便宜，功能丰富，通过I/O口操作单片机上的外设，比较简单。重量监测功能采用压力传感器和HX711AD模块实现，本次设计采用5KG压力传感器和一个HX711AD转换模块共同组成重量监测，首先压力传感器进行压力监测，根据压力和4个压力电阻的变化正比关系，将压力转换成电流进行输出给HX711芯片，HX711AD模块将模拟量转换成数字量并发数给单片机。继电器就是采用我们通常使用的5v继电器，这种继电器安全性也非常高，电路设计也相对比较简单。在引用的时候在需要确定继电器的耐压值是多少就可以了。 一般确定耐压值就是电源电路的整体供电电压。比如单片机是5V供电，继电器选择耐压值只要大于或等于5V就可以了。这样就大大的节约了资源。给人们带来了更好的效益。所以就可以更好的来利用资源。显示部分用LCD1602液晶屏来显示当前的信息，因为传统的数码管显示的缺陷是信息量比较少，能显示的内容有限，而LCD1602显示就弥补这样的缺陷，它可以显示当前的重量和计算的结果及特殊的符号，也可以显示数字。而且可以来调节背光亮度。节约资源。语音播报部分采用.语音芯片和扬声器组成，该语音芯片采用3V供电，自带循环播放功能，而且控制方式简单，单片机通过I/O线进行操作，将检测到的重量和计算出的结果进行播报。计算器功能则是通过键盘矩阵实现，采用16个按键组成键盘矩阵，包括0~9共10位数字和加减乘除、小数点和等于功能。

## 1.3 设计目的与主要功能

本次设计以STM32单片机为核心，利用压力传感器监测货物的压力并转换成电流输出给AD芯片，HX711转换模块将模拟量转换成数字量并输出给单片机，STM32单片机通过公式计算出具体的货物重量，采用按键矩阵对结果进行加减乘除，得出总价或者其他的结果。LCD1602液晶屏上显示检测到的货物重量、单价和总价。下面是本次设计的主要功能：

1.最小系统的硬件设计：

（1）上电复位电路。

（2）晶体振荡器电路。

（3）单片机电源。

2.重量监测系统

（1）压力传感器监测货物对其的压力，转换成电流变化并输出给hx711芯片

（2）HX711AD模块将

4. LCD1602液晶显示屏

（1）显示检测到的货物重量

（2）显示单价和总价

5、 按键设置电路

（1）设置货物的单价

（2）通过重量和单价计算出总价

# 2 方案设计

## 2.1 设计思路

以STM32为中央处理器，通过压力传感器监测货物对其造成的压力，并以电流变化输出给HX711AD模块，hx711AD芯片将模拟量转换成数字量并发送给单片机，经单片机计算后，得到当前的重量。通过安检矩阵设置货物的单价，通过加减乘除操作，算的货物的总价。LCD1602液晶显示屏上显示检测到的货物重量，设置的商品单价以及最后算出的货物总价。

## 2.2 系统构成

本次系统构成分为硬件系统和软件部分。硬件系统就是单片机的选型，传感器的选型，电子元器件的选择。当所有材料选型完成后，进行电子电路设计，确保每个电路的正确和畅通。之后按照设计好的电路图，将所有材料焊接起来，组成一个单片机系统。当硬件部分完成后，进行软件设计，用来控制整个单片机。这里采用KEIL软件进行软件的编程，KEIL软件集成链接器，，编译器，宏汇编，管理库和仿真调试器。采用C++语言进行整体设计，编写完成后进行测试，没有问题将生成的hex文件烧录进单片机。

STM32单片机作为整个设计的核心处理器，整体分为检测部分、显示部分、计算部分和语音播报。检测部分由压力传感器和AD转换模块组成，压力传感器将货物的压力转换成电流输出给HX711AD芯片，AD模块将模拟量转换成数字量并发送给单片机，单片机计算得出货物重量。显示部分采用LCD1602液晶显示屏，将采集到的重量和设置的单价以及计算出的总价。计算部分按键用于对重量结果进行计算，设置单价，计算出总价。

# 3 硬件电路设计

## 3.1 系统硬件原理图

本次设计采用LED1602液晶显示器显示采集到的重量、单价和总价，单片机型号为STM32单片机，压力传感器和HX711AD芯片组成重量监测部分，将压力转换成数字并发送给单片机进行计算。16个按键组成按键矩阵，通过引脚进行并联，扫描矩阵，设置单价并计算出总价。本次单片机系统采用5V直流电源进行供电，图3.1是其硬件电路原理图



图3.1 系统硬件原理图

## 3.2 STM32概述

STM32系列是为要求高性能、低成本、低功耗的嵌入式应用专门设计的ARM Cortex-M3内核。按性能分成两个不同的系列：“增强型”STM32F103系列和“基本型”STM32F101系列。增强型系列的时钟频率能达到72MHz，是同类产品中频率最高的；基本型的时钟频率为36MHz，用16位产品一样的价格得到比16位产品更大的性能，是16位产品的最好选择。两个系列都有内置的32K到128K的闪存，不同的是SRAM的最大容量和外设接口的组合。时钟频率为72MHz时，从闪存执行代码来看，STM32功耗36mA，是32位市场上功耗最低的产品，相当于0.5mA/MHz[6]。

以STM32103C8T6单片机为处理器，该单片机为32位处理器。内核为Cortex-M3，其并行总线结构，嵌套中断向量控制单元，调试系统和它标准的存储映射。嵌套中断向量控制器（Nested Vector Interrupt Controller,简称NVIC）是Cortex-M3处理器中一个比较关键的组件，它为基于Cortex-M3的微控制器提供了标准的中断架构和优秀的中断响应能力，为超过240个中断源提供专门的中断入口，而且可以为每个中断源赋予单独的优先级。采用NVIC可以达到非常快的中断响应速度，从收到中断请求到执行中断服务的第一条指令仅需12个周期。这种极快的响应速度一方面得益于Cortex-M3内核对堆栈的自动处理机制，这种机制是通过固化在CPU内部的微代码实现的。另一方面，在中断请求连续出现的情况下，NVIC使用一种称为“尾链”的技术，使连续而来的中断可以在6个时钟周期内得到服务。在中断的压栈阶段，更高优先级的中断可以不耗费任何额外的CPU周期就能完成嵌入低优先级中断的动作。具体的细节后面我会继续总结的。用户可以通过设置CPU自动进入低功耗状态，而使用中断来将其唤醒，CPU在中断时间来临之前会一直保持睡眠状态[7]。  
　　Cortex-M3的CPU支持两种运行模式：线程模式（Thread Mode）与处理模式（Handler Mode）并且需要注意的是，这两种模式都拥有各自独立的堆栈。这种设计使得开发人员可以进行更为精密的程序设计，对实时操作系统的支持也就更好了。Cortex-M3处理器还包含了一个24位可自动重装载定的定时器，可以为内核（RTOS）提供一个周期中断。

在指令集方面，ARM7和ARM9都有两种指令集（32位指令集和16位指令集），而Cortex-M3系列处理器支持Thumb-2指令集。由于Thumb-2指令集融合了Thumb指令集和ARM指令集，使得32位指令集的性能和16位指令集的代码密度之间取得了平衡[5]。 专业嵌入式、单片机技术实训。而且，ARM Thumb-2 专门为C/C++编译器设计，这就意味着Cortex-M3系列处理器的开发应用可以全部在C语言环境中完成。  
　　STM32微控制器的推出标志着ST公司在两条产品主线（低价位主线和高性能主线）上迈出了重大一步。STM32最初发布时有14个不同型号，分为两个版本：最高CPU时钟为72MHZ的“增强型”和最高CPU时钟为36MHZ的“基本型”。这些不同STM32型号里内置的Flash最大可达128KB，SRAM最大为20KB，在STM32发布之初，配置更大Flash,RAM和更复杂外设的版本就已经在规划之中了。不管是什么版本，什么型号的STM32器件，它们在引脚功能和应用软件上是兼容的。这就使得开发人员在使用STM32系列微控制器时，不必改动PCB就可以根据需要随意更换器件型号。乍一看STM32的设备配备，与往日熟悉的51单片机倒有几分相似。一般，STM32都会配备常见外设，诸如多通道ADC,通用定时器，I2C总线接口，SPI总线接口，CAN总线接口，USB控制器，实时时钟RTC等。但是，它的每一个外部设备都具有独特之处。例如，12位精度的ADC具备多种转换模式，并带有一个内部温度传感器，带有双ADC的STM32器件，还可以使两个ADC同时工作，从而衍生出了更为高级的9种转换模式；STM32的每一个定时器都具备4个捕获比较单元，而且每个定时器都可以和另外的定时器联合工作以生成更为精密的时序；STM32有专门为电机控制而设的高级定时器，带有6个死区时间可编程的PWM输出通道，同时其带有的紧急制动通道可以在异常情况出现时，强迫PWM信号输出保持在一个预订好的安全状态；SPI接口含有一个硬件CC8单元，支持8位字节和16位半字数据的CC8计算。在对SD或MMC等存储介质进行数据存取时相当有用。而且，STM32还包含了7个DMA通道。没恶搞通道都可以用来在设备与内存之间进行8位，16位，32位数据的传输。每个设备都可以向DMA控制器请求发送或者接收数据。STM32内部总线仲裁器和总线矩阵将CPU数据接口和DMA通道之间的连接大大的简化了，这就意味着DMA通道单元是很灵活的其使用方法简单，足以应付微控制器应用中常见的数据传输要求。

### 2.2.2 STM32的主要优点

* 使用ARM最新、先进的构架Cortex-M3内核
* 出色的实时性能
* 出色的功率控制
* 出众和创新的外设
* 最大程度的集成整合
* 易于开发，可使产品
* 快速进入市场

STM32硬件的特色接口：

* **I/O:**输入/输出口
* 低功耗模式、定时器/计数器、输入捕获
* **PWM:**脉宽调变
* **A/D:**模/数转换
* **DMA：**直接存储器存取
* **USART、SPI:**单个程序启动
* **BOOT:**

### 2.2.3 STM32开发板

STM32开发板如图2.2所示[4]

* STM32F103C8T6,TQFP64,FLASH:256K,SRAM:40K
* 1个JTAG/SWD调试的下载口
* 1个电源指示灯（蓝色）
* 2个状态指示灯（DS0：红色，DS1：绿色）
* 1个红外接收头
* 1个IIC接口的EEPROM芯片，24C02，容量256字节
* 一个SPIFLASH芯片，W25X16，容量2M字节
* 1个DS18B20温度传感器预留接口
* 一个标准的2.4 / 2.8英寸液晶屏接口，支持触摸屏
* 1个OLED模块的接口
* 1个USB SLAVE接口，用于USB通信
* 1个插SD卡的接口
* 1个PS/2接口，外接鼠标、键盘等
* 1组5V电源供应/接入口
* 1个启动模式的配置选择接口
* 2个2.4G的无线通信接口（24L01和JF24C）
* 1个复位按键，用来对MCU和LCD进行复位
* 3个功能按键，其中WK\_UP兼具唤醒功能
* 1个电源开关，控制整个板的电源
* 除晶振占有的IO口外，其余所以得IO全部引出，其中GPIOA和GPIOB按顺序引出

2.3 最小系统设计

单片机要正常工作必须电源电路提供电源，通过震荡电路产生时钟周期，同时为了防止系统异常还需加上复位电路，可手动让系统重新工作。

（1）震荡电路

震荡电路用的是8M晶振，因为STM32内部可以通过锁相环可以进行倍频，变为最高72MHZ的频率，所以外部接8MHZ晶振，经过倍频可达到72MHZ[1]。其电路如图2.3所示。其电路主要有晶振、电容和电阻组成。其中OSDIN 和OSDOUT为STM32外部时钟电路引脚。通过震荡电路产生时钟从该管脚输入作为处理器时钟源。

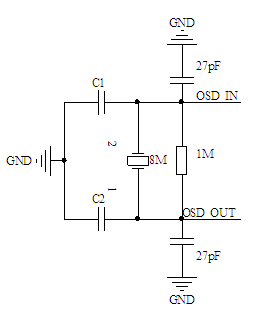


图2.3 振荡电路

（2）复位电路

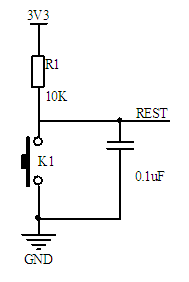


图2.4 复位电路

Stm32是低电平复位并且每次上电是会复位一次，所以系统上电之后默认情况下其RST脚应该为高电平。其电路如图2.4所示。R1位上拉电阻，REST为单片机复位引脚，当按键K1按下RST引脚输出低电平，此时系统复位；当复位按键K1没按下时，由于RST直接接在上拉电阻上，默认为高电平，所以系统正常工作。

## 3.3 矩阵键盘电路

单片机的矩阵键盘电路是由4行4列共16个微动开关并联组成的电路，按键的一段连接单片机的双向I/O口，另一端接地。当按键开关处于张开状态时，I/O引脚和地断开，此时该I/O引脚处于高电平状态。当按下按键时，I/O引脚接地，此时高电平置低电平，并返回一个低电平信号给单片机。按键开关的检测其原理就是高低电平的检测。按键开关由于其拨动时间，难免产生误差，所以在编写按键函数时，要额外写一个去抖函数，提高精确性。这些按键分别表示0-9共10位数字、加减乘除功能、小数点和等于。图3.4为矩阵键盘电路原理图：

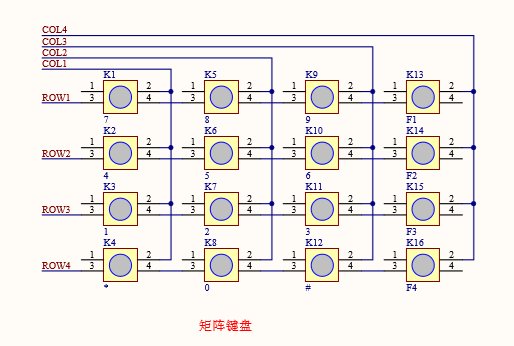


图3.3 矩阵键盘电路

## 3.4 电源电路

单片机的电源采用直流5V供电，电源模块包括一个3脚的电源座子和6脚的电源开关。电源座子用于连接外部的电源插头，电源开关用于控制整个单片机的电路开和关。电源座子的2口引脚接地，3口引脚仅仅起到固定的作用，没有特殊的用处，1口引脚连接到电源开关的3口引脚，电源开关的1、3口引脚和4、6口引脚的作用相同，用于电源的正极输出。电源开关的2、5口引脚作为单片机的接地引脚，在使用时采取相对的选择，即选择1、3口引脚作为输出，那么就要选择5口引脚作为接地引脚，选择4、6引脚作为输出端口，2口引脚则作为接地引脚。本次单片机的传感器和无线传输芯片的电压都在5V内，所以5V的电压足够满足。若有12V或者其他电压的传感器，则可以采取升压模块将5V提升到更高的电压，进行供电。本次设计的电源电路如图3.5所示：

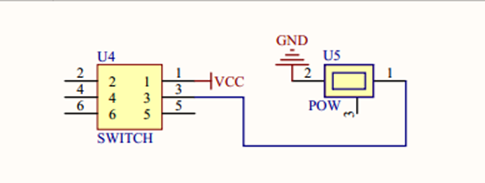


图3.4 电源电路

## 3.5 重量监测电路

本次设计的重量监测部分由压力传感器和HX711AD模块组成，压力传感器的量程范围是0~5KG，将货物放置在压力传感器上面，压力传感器的监测原理是采用4个监测电阻，当受到压强是，其阻值变大，电路变小，通过监测电流和压力的线性比，按照一定的计算公式，监测重量。HX711AD模块将压力传感器输出引脚发送来的模拟量转换成数字量，并发送给单片机进行计算，通过公式计算出重量。HX711ad芯片是专门为电子秤设计的24位A/D转换器芯片，包括稳压电源，片内时钟振荡器等外围电路，集成度高、响应速度快、抗干扰性强，成本低，提高了整体的性能和可靠性。图3.6是重量监测电路：



图3.5 重量监测电路

## 3.6 LCD1602液晶显示电路

本次设计采用LCD1602液晶显示器作为输出显示器，LCD1602液晶显示器是一种工业字符型液晶，从其名称中可以得知，显示屏具有2行输出，每行16个字符，能够同时显示16\*2即32个字符。LCD1602液晶屏显示利用液晶的特性，通过改变电压达到控制区域显示，最终显示出想要的图形或者数字。LCD1602液晶显示屏不仅可以显示数字，还可以显示符号和字母。图3.7是LCD1602液晶屏的引脚图，从图中可以看出，引脚1是电源接地引脚，和单片机的GND联通。引脚2是VCC电源输入引脚，其工作电压为5V。引脚3—VO引脚可以调节显示屏的显示亮度，引脚4—RS引脚用于寄存器的选择，高电平时作为数据寄存器，低电平时作为指令寄存器，引脚5是RW读写信号，用于数据的读写。引脚6—EN引脚为使能端，高电平读取信息，低电平执行指令。引脚7~14是8位双向数据段口，引脚15和引脚16是背景灯的电源引脚。在和单片机进行通信时，和P0口进行连接，中间要有上拉电阻进行连接。本次设计LCD1602液晶显示屏上展示检测到的货物重量、单价和总价。其引脚连接图如图3.7所示：

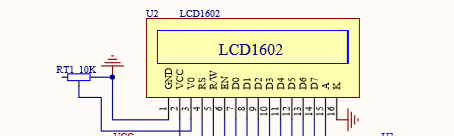


图3.6 LCD1602液晶显示电路图

# 4 系统软件设计

## 4.1 软件主程序

当整个系统在运行时，各个硬件开始初始化，当初始化完成会进入显示主界面，启动的步骤为：串口初始化，LCD1602初始化，定时器初始化，压力传感器开始检测货物对其压力，HX711A/D转换模块将压力模拟量转换成数字量发送给单片机，单片机通过计算得出货物重量。扫描按键矩阵，通过按键设置货物单间，根据单价和货物重量，计算出货物总价。在LCD1602液晶显示器上显示监测到的重量、单价和总价。同时单片机通过引脚控制.语音芯片通过扬声器播报检测到的重量和总价。主程序流程图如图4.1所示：

开始

LCD1602初始化

HX711初始化

语音芯片初始化

LCD1602显示当前信息

否

计算当前总价

检测当前重量

是

按键设置单价

是否按下按键

图4.1 主程序流程图

## 

## 4.2 LCD1602显示子程序

LCD1602液晶显示屏上电后，首先进行初始化操作，屏幕清零，内部存储清空。之后和单片机进行双向通信，单片机控制显示屏亮度和显示内容，延时一定时间后，将检测到的货物重量显示在液晶屏上，扫描监测矩阵键盘，显示单价和计算出的总价。单片机操控写入引脚写入要显示内容的命令。之后显示屏执行指令，将内容显示出来。始化显示内容为uchar code Init1[]="WELCOME: ";lcd延迟void LCDdelay(uint z)；写入命令void write\_com(uchar com)；写数据void write\_data(uchar date)；初始化void Init1602(){write\_com(0x38)读取字节for(i=0;i<14;i++) {write\_data(Init1[i]);}LCD1602显示流程图如图4.2所示：



图4.2 LCD1602初始化流程图

## 4.4 重量检测子程序

当单片机上电后，各个传感器开始工作，单片机给压力传感器发送指令，进入重量监测子程序，首先进行传感器初始化，然后设置引脚输出，通过压力传感器监侧货物对其造成的压力并转换成电流模拟量发送给HX711a/d转换模块，HX711将模拟量通过相应的公式转换成数字量，并发送给单片机进行处理。其流程图如图4.4所示：



图4.4 重量检测子程序

## 4.5 按键子程序

本次设计按键电路功能不多，主要包括0-9共10个数字和加减乘除、小数点和等于功能。单片机的按键具有4个引脚，相同的一侧是连接的，16个按键并联在一起，只要将相同的引脚连接一起即可。按键的工作原理就是对低电平信号的检测，在主程序中，循环执行检测，一旦检测到按键低电平信号，单片机产生信号中断，进入按键子程序中。16个按键分别表示数字选择，小数点选择，加减乘除计算和等于功能。分别用16个函数执行不同的操作，注意进行延时操作，防止误差。判断按键是否按下if(K1==0){while(K1==0)去延时，数值选择1;if(K2==0){delay\_key()延迟，然后while(K2==0);是否按下如果按下，数选择2。if(K3==0){while(K3==0)数值选择3……，如果按下，则设置成功。按键可以用来设置货物单价，并和重量进行计算得出总价。流程图如下：流程图如4-5所示：



图4-5 按键流程图

5 系统调试

## **5.1** 焊接调试

准备好所有的材料和电烙铁，按照设计好的电路板原理图，开始单片机电路板的焊接。首先将插排焊接上去，之后焊接单片机最小系统的晶振和复位电路。确定好LCD1602液晶显示屏位置，将上拉电阻焊接在P0口，之后通过导线连接显示屏。后面分别焊接各个传感器模块，最后用导线将各个模块按照电路图连接起来，确保没有出现短路现象。STM32单片机用烧录器将编译好的软件烧录进去，最后插入到插排上。用5V直流电源供电，按下开关，观察LCD1602液晶显示屏是否正常显示，正常显示后，说明显示电路正常，之后观察其他传感器是否正常工作，显示屏上是否有输出，如果正常显示，则一切都没问题，当出现问题时，就要找出具体出问题的部分，逐一解决。

## **5.2** 功能调试

单片机焊接测试没问题后，进行功能调试，测试软件是否正确。首先给单片机重新上电，这时LCD1602液晶屏首先进行初始化操作，LCD1602液晶屏上应该出现“INIT”，这一步没问题后，接下来测试各个传感器功能。 当供电后，进入主页面，主页面分为两行，上面显示测得的货物重量，下面显示按键矩阵设置的单价和计算出来的总价。将货物放置在压力传感器上，就可以在LCD1602液晶显示屏上看到货物的重量。按下矩阵按键，可以在LCD1602液晶显示屏上显示设置的货物单价，通过按键矩阵计算得出货物总价并显示。同时.语音芯片控制扬声器播报此时的货物重量和总价。至此，所有功能测试完成。

# 6 总结

经过这几个月的努力，毕业设计终于完成了。根据以往的经验，较小的硬件开销需要相对复杂的软件进行补偿，由于压力传感器模块与微机处理器之间采用串行数据传送，因此，在对压力传感器进行程序编写时，必须严格保重读写时序，否则将无法读取重量结果。在此次毕业设计中，老师的指导对我而言是及其重要的，从论文选题到整体思路，从材料购买到电路设计，从硬件焊接到软件编写，其中的复杂度超出自己的预想，能够顺利的进行下来，这里还要感谢老师的指导，

本设计论文系统描述了基于STM32单片机的电子秤系统功能和思路以及对各个芯片的相关使用方法。这次设计中，系统的原理，芯片的选型、引脚功能、使用的优缺点，硬件的设计，电路的连接，程序编写等。对于这些问题的解决，需要相关的专业知识，在本次设计中极大的提高了自己的专业知识，对各个知识点综合理解。通过这次的毕业设计使我对单片机有了更深刻的认识，使我在大学课堂上学的理论知识通过实践变的更加深刻，为我以后在接触单片机方面的工作打下了扎实的基础，其次在这次毕业设计中也锻炼了我独立解决困难问题的能力，培养了我的团队合作精神。

针对基于STM32单片机的电子秤系统的现状和所要研究问题的分析，提出了一种采用压力传感器和矩阵键盘，并运用STM32作为控制芯片的电子秤系统，接着从单片机的结构，原理分析起，再熟悉了STM32的工作原理及使用方法后，详细的分析了压力传感器模块的结构、工作原理和检测原理的方法，最后又简要了解下语音播报的工作原理。在这个过程中我查阅了大量的相关资料，综合所分析所有内容，再结合单片机基本知识设计了系统电路，通过仿真确定了功能的可实现性，最后做出实物。通过这次学习，并亲自动手制作焊接，学到了很多，也加强了很多大学学习的基本技能，是理论与实际得到很好的结合，在这个过程中，也和同学进行了很多沟通，交流，体会到了团队合作的乐趣，经过最后的调试，系统成功的实现了设计功能，由于时间有限和本身知识水平的欠缺，所以本系统可能还有需要改进和提高的地方。

# 致 谢

在大学度过了整整四年的学习生涯，在这紧张并且有趣的学校过程中我不仅系统的学习了单片机方面的知识要点，并且深深的感谢各位老师无私奉献，任劳任怨的解答我的问题，传授给我知识。

经过本次毕业设计，让我把在学校学习的专业建立了一个体系，让不同的知识点融会贯通，在通过老师的不断帮助，不断完善知识体系，最后变为为自己的知识，这就是毕业设计的意义。让我学会了很多以前没有接触到的知识点。软件部分每一个模块的使用，如何去查询相应的资料。对程序代码格式的要求，子程序如何导入到主函数中。硬件部分应该如何考虑到设计布局的合理性。以及同一种芯片不同的规格如何去分辨，画图软件ALTIUM DESIGNED 如何使用。这些都让我受益良多。大大的增强了我对单片机这一块如何去更好的为系统的处理实物，接触新事物。

在几个月的毕设设计制作过程中，受到了导师的辛勤指导，在导师的耐心指导下，顺利的完成了本次毕业设计，在此由衷的向导师表示敬意，也同样非常感谢在毕业设计过程中给我提供帮助的同学和朋友。本次毕业设计的内容是基于单片机的水体监测系统设计，从选题到构思到最后的实现，期间遇到了很多问题，在导师的指导下，一点点去查资料，去不断地尝试实验，期间学到了很多知识，难题逐个击破。在导师身上，不仅学到了深厚的专业知识，而且学到了做人的道理，做事的态度和方法。导师无私奉献的精神，严谨的教学态度，渊博的知识让我深受启发。在此我要深深的向我的导师致以深深的敬意和衷心的感谢！

# 参 考 文 献

[1] 沙占友主编．集成化智能传感器原理与应用．北京： 电子工业出版社， 2004，36-56．

[2] 赵德安．单片机原理与应用[M]．北京：机械工业出版社，2009．

[3] 清源计算机工作室．Protel 99SE 原理图与PCB及仿真．北京：机械工业出版社，2001．

[4] 李广第．单片机基础．北京：北京航空航天大学出版社，1995，33-64．

[5] 徐恕宏．传感器原理及其设计基础．北京：机械工业出版社，1988，30-45．

[6] 彭军．传感器与检测技术[M]．西安：西安电子科技大学出版社，2003．

[7] 周航慈．单片机应用程序设计技术．北京： 北京航空航天大学出版社，1991，18-27．

[8] 任向民．卢惠林．汇编语言程序设计使用教程[M]．北京：清华大学出版社，2009．

[9] 王晓音，聂裕平，庞伟正．DDS 输出频谱杂散的抑制[J]．电子对抗技术．2003．

[10] 杨扬. 基于TC35i的智能传输系统[J]. 电子测试, 2016(16).

[11] 赵欣. 基于SIM300/TC35i无线GSM芯片实现短信远程控制TRT的研究[J]. 消费导刊, 2017(30).

[12] 钱钧, 惠王伟, 高莹,等. RC滤波电路实验设计与研究[J]. 大学物理实验, 2017, 30(5):58-62.

[13] 刘文楷, 苏宁宁, 戴澜. 霍尔传感器读出电路的低噪声运算放大器设计[J]. 电子世界, 2016(17):79-80.

[14] 许瑞庆. 脉搏检测系统设计及其信号处理算法研究[D]. 南京信息工程大学, 2008.

[15] 仝兆景, 时俊岭, 李月,等. 基于无线通讯技术脉搏检测仪的设计与实现[J]. 计算机测量与控制, 2017, 25(1):242-244.

[16] Jian W U, Hou W, Zheng B. A Temperature Control System Based on STM32 MCU[J]. Computer Knowledge & Technology, 2011.

[17] Xiao-Qing Y U, Zhang Z L. Remote Monitoring System for Irrigation Based on STM32 and GSM[J]. Water Saving Irrigation, 2011.

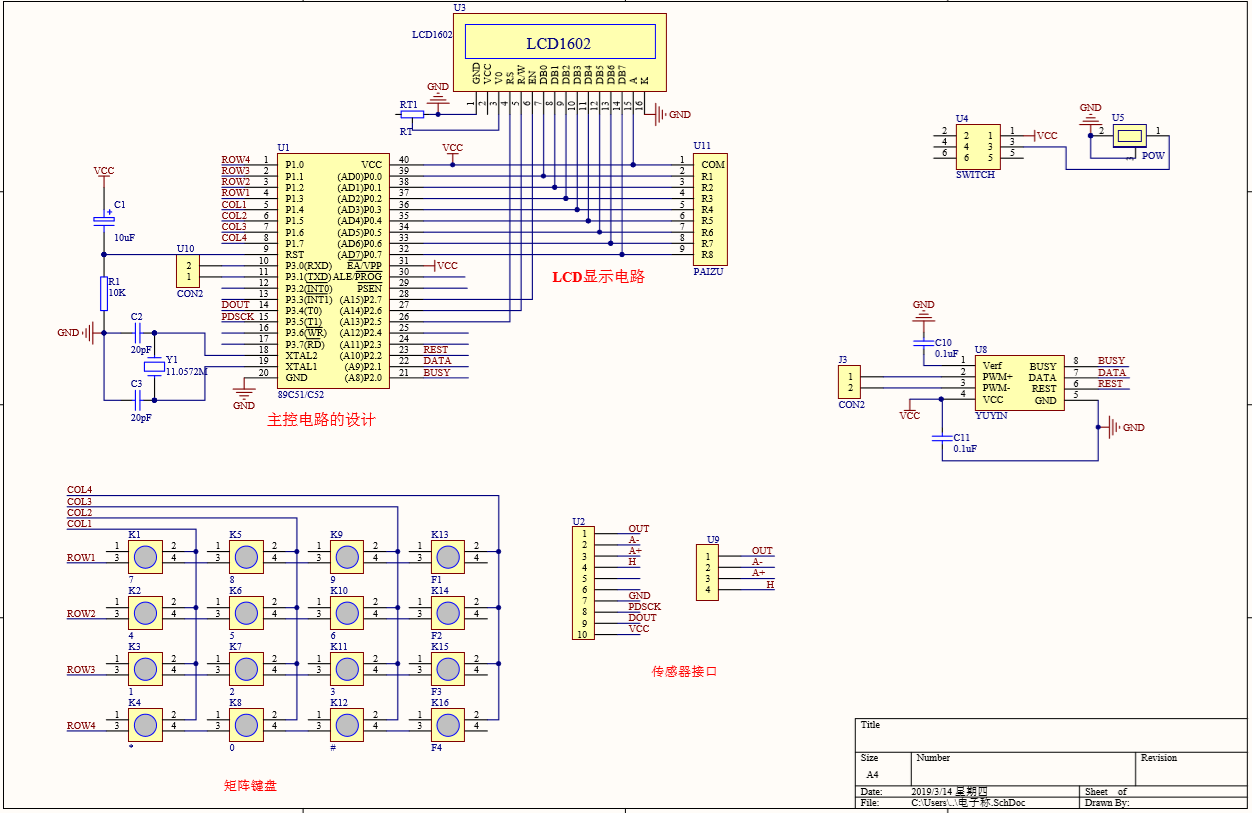
[18] Yang Y. Intelligent transmission system based on TC35i[J]. Electronic Test, 2016.

[19] Gan Y, Wang H, Shan Z, et al. USER DATA PROCESSING METHOD AND DEVICE:, WO/2016/000163[P]. 2016.

[20] 远生廖. Analysis of Cerebrovascular Function in Patients with Chronic Cerebral Circulation Insufficiency[J]. 2016, 05(1):9-13.

附 录

硬件电路图：



程序代码：

#include "hx711.h"

unsigned long ReadCount(void)

{

unsigned long Count;

unsigned char i;

ADSK=0;

Count=0;

while(ADDO);

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

{

ADSK=1;

Count=Count<<1;

ADSK=0;

if(ADDO) Count++;

}

ADSK=1;

Count=Count^0x800000;

ADSK=0;

return(Count);

}#include "keyboard.h"

//按键扫描函数 :

//功能；只扫描一个按键，按键为号小优先

//返回键值：

// 7 8 9 10

// 4 5 6 11

// 1 2 3 12

// 14 0 15 13

unsigned char Getkeyboard(void)

{

unsigned char number = 0;

unsigned char i;

//行输入低电平

ROW1=ROW2=ROW3=ROW4=0;

//如果列输出不全为高，则说明有键按下

if ((COL1!=1)||(COL2!=1)||(COL3!=1)||(COL4!=1))

{

//输入0111

ROW1=0;

ROW2=ROW3=ROW4=1;

for (i=0;i<20;i++);

if (COL1==0) return 1; // 7

else if (COL2==0) return 2; // 8

else if (COL3==0) return 3; // 9

else if (COL4==0) return 10; // 10

//输入1011

ROW2=0;

ROW1=ROW3=ROW4=1;

for (i=0;i<20;i++);

if (COL1==0) return 4; //4

else if (COL2==0) return 5; // 5

else if (COL3==0) return 6; // 6

else if (COL4==0) return 11; // 11

//输入1101

ROW3=0;

ROW1=ROW2=ROW4=1;

for (i=0;i<20;i++);

if (COL1==0) return 7; //1

else if (COL2==0) return 8; // 2

else if (COL3==0) return 9; // 3

else if (COL4==0) return 12; // 12

//输入0111

ROW4=0;

ROW1=ROW2=ROW3=1;

for (i=0;i<20;i++);

if (COL1==0) return 14;

else if (COL2==0) return 0;

else if (COL3==0) return 15;

else if (COL4==0) return 13;

//没有检测到列低电平

return 99;

}

return 99;

}#include "LCD1602.h"

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

//MS延时函数(12M晶振下测试)

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

void LCD1602\_delay\_ms(unsigned int n)

{

unsigned int i,j;

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

for(j=0;j<123;j++);

}

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

//写指令

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

void LCD1602\_write\_com(unsigned char com)

{

LCD1602\_RS = 0;

LCD1602\_delay\_ms(1);

LCD1602\_EN = 1;

LCD1602\_PORT = com;

LCD1602\_delay\_ms(1);

LCD1602\_EN = 0;

}

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

//写数据

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

void LCD1602\_write\_data(unsigned char dat)

{

LCD1602\_RS = 1;

LCD1602\_delay\_ms(1);

LCD1602\_PORT = dat;

LCD1602\_EN = 1;

LCD1602\_delay\_ms(1);

LCD1602\_EN = 0;

}

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

//连续写字符

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

void LCD1602\_write\_word(unsigned char \*s)

{

while(\*s>0)

{

LCD1602\_write\_data(\*s);

s++;

}

}

void Init\_LCD1602()

{

LCD1602\_EN = 0;

LCD1602\_RW = 0; //设置为写状态

LCD1602\_write\_com(0x38); //显示模式设定

LCD1602\_write\_com(0x0c); //开关显示、光标有无设置、光标闪烁设置

LCD1602\_write\_com(0x06); //写一个字符后指针加一

LCD1602\_write\_com(0x01); //清屏指令

}

#include <reg52.h>

#include <intrins.h>

#include <string.h>

#include "LCD1602.h"

#include "hx711.h"

#include "keyboard.h"

#include "eeprom52.h"

#define uchar unsigned char

#define uint unsigned int

//定义量程系数

#define RATIO 4000/1000 //1574/1000 //2114/1623

//定义标识

volatile bit FlagTest = 0; //定时测试标志，每0.5秒置位，测完清0

volatile bit FlagKeyPress = 0; //有键按下标志，处理完毕清0

volatile bit FlagSetPrice = 0; //价格设置状态标志，设置好为1。

//管脚定义

//sbit LedA = P2^2;

sbit beep = P3^6;

sbit alert = P3^7;

//显示用变量

int Counter;

int i, iTemp;

//称重用变量

unsigned long idata FullScale; //满量程AD值/1000

unsigned long AdVal; //AD采样值

unsigned long weight; //重量值，单位g

unsigned long idata price; //单价，长整型值，单位为分

unsigned long idata money; //总价，长整型值，单位为分

//键盘处理变量

unsigned char keycode;

unsigned char DotPos; //小数点标志及位置

static unsigned int RATIO\_deviation=492;

unsigned char a\_a=0;

void Data\_Init();

void Port\_Init();

void Timer0\_Init();

void Timer0\_ISR () ;

void INT1\_Init();

void KeyPress(uchar);

void To\_Zero();

void Display\_Price();

void Display\_Weight();

void Display\_Money();

void PlayNowInfo(void);

extern char PlaySoundCheckBusy(void);

extern void PlaySoundTick(int a);

/\*延时\*/

void key\_delay(int i)

{

int j,k;

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

for(k=0; k<500; k++);

}

//重新找回零点，每次测量前调用

void To\_Zero()

{

FullScale=ReadCount();

price=0;

}

//显示单价，单位为元，四位整数，两位小数

void Display\_Price()

{

LCD1602\_write\_com(0x8c);

LCD1602\_write\_data(price/100 + 0x30);

LCD1602\_write\_data(price%100/10 + 0x30);

LCD1602\_write\_data('.');

LCD1602\_write\_data(price%10 + 0x30);

}

//显示重量，单位kg，两位整数，三位小数

void Display\_Weight()

{

LCD1602\_write\_com(0x83);

LCD1602\_write\_data(weight/1000 + 0x30);

LCD1602\_write\_data('.');

LCD1602\_write\_data(weight%1000/100 + 0x30);

LCD1602\_write\_data(weight%100/10 + 0x30);

LCD1602\_write\_data(weight%10 + 0x30);

}

//显示总价，单位为元，四位整数，两位小数

void Display\_Money()

{

//unsigned int i,j;

if (money>9999) //超出显示量程

{

LCD1602\_write\_com(0x80+0x40+4);

LCD1602\_write\_word("---.-");

return;

}

if (money>=1000)

{

LCD1602\_write\_com(0x80+0x40+4);

LCD1602\_write\_data(money/1000 + 0x30);

LCD1602\_write\_data(money%1000/100 + 0x30);

LCD1602\_write\_data(money%100/10 + 0x30);

LCD1602\_write\_data('.');

LCD1602\_write\_data(money%10 + 0x30);

}

else if (money>=100)

{

LCD1602\_write\_com(0x80+0x40+4);

LCD1602\_write\_data(0x20);

LCD1602\_write\_data(money%1000/100 + 0x30);

LCD1602\_write\_data(money%100/10 + 0x30);

LCD1602\_write\_data('.');

LCD1602\_write\_data(money%10 + 0x30);

}

else if(money>=10)

{

LCD1602\_write\_com(0x80+0x40+4);

LCD1602\_write\_data(0x20);

LCD1602\_write\_com(0x80+0x40+5);

LCD1602\_write\_data(0x20);

LCD1602\_write\_data(money%100/10 + 0x30);

LCD1602\_write\_data('.');

LCD1602\_write\_data(money%10+ 0x30);

}

else

{

LCD1602\_write\_com(0x80+0x40+4);

LCD1602\_write\_data(0x20);

LCD1602\_write\_com(0x80+0x40+5);

LCD1602\_write\_data(0x20);

LCD1602\_write\_com(0x80+0x40+6);

LCD1602\_write\_data(0 + 0x30);

LCD1602\_write\_data('.');

LCD1602\_write\_data(money%10 + 0x30);

}

}

void PlayNowInfo(void)

{

uint tmp;

tmp = weight/1000;

if (tmp >= 10)

{

return;

}

while(PlaySoundCheckBusy() == 0);

///////重量

PlaySoundTick(17);//重

while(PlaySoundCheckBusy() == 0);

tmp = weight/1000;

PlaySoundTick(tmp);

while(PlaySoundCheckBusy() == 0);

PlaySoundTick(28);//点

while(PlaySoundCheckBusy() == 0);

tmp = weight%1000/100;

PlaySoundTick(tmp);

while(PlaySoundCheckBusy() == 0);

tmp = weight%100/10;

PlaySoundTick(tmp);

while(PlaySoundCheckBusy() == 0);

tmp = weight%10;

PlaySoundTick(tmp);

while(PlaySoundCheckBusy() == 0);

PlaySoundTick(16);//千克

///////单价

key\_delay(200);

while(PlaySoundCheckBusy() == 0);

PlaySoundTick(18);//单价

tmp = price/100;

if (tmp > 0)

{

while(PlaySoundCheckBusy() == 0);

PlaySoundTick(tmp);

while(PlaySoundCheckBusy() == 0);

PlaySoundTick(11);//十

}

while(PlaySoundCheckBusy() == 0);

tmp = price%100/10;

if (!(price/100 > 0 && tmp == 0))

{

PlaySoundTick(tmp);

}

while(PlaySoundCheckBusy() == 0);

PlaySoundTick(28);//点

while(PlaySoundCheckBusy() == 0);

tmp = price%10;

PlaySoundTick(tmp);

while(PlaySoundCheckBusy() == 0);

PlaySoundTick(55);//元

///////金额

key\_delay(200);

while(PlaySoundCheckBusy() == 0);

PlaySoundTick(19);//金额

tmp = money/1000;

if (tmp > 0)

{

while(PlaySoundCheckBusy() == 0);

PlaySoundTick(tmp);

}

tmp = money%1000/100;

if (tmp > 0)

{

while(PlaySoundCheckBusy() == 0);

PlaySoundTick(tmp);

}

tmp = money%100/10;

while(PlaySoundCheckBusy() == 0);

if (tmp > 0)

{

PlaySoundTick(tmp);

}

else

{

PlaySoundTick(tmp);

}

while(PlaySoundCheckBusy() == 0);

PlaySoundTick(28);//点

tmp = money%10;

while(PlaySoundCheckBusy() == 0);

PlaySoundTick(tmp);

while(PlaySoundCheckBusy() == 0);

PlaySoundTick(55);//元

}

//数据初始化

void Data\_Init()

{

price = 0;

DotPos = 0;

beep = 1;

alert = 1;

}

void write\_eeprom(void)

{

SectorErase(0x2000); //清空

byte\_write(0x2000,RATIO\_deviation >> 8 );

byte\_write(0x2001,RATIO\_deviation&0xFF);

byte\_write(0x2002,0x01);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*把数据从单片机内部eeprom中读出来\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void read\_eeprom(void)

{

RATIO\_deviation= byte\_read(0x2000);

RATIO\_deviation= (RATIO\_deviation << 8) + byte\_read(0x2001);

a\_a = byte\_read(0x2002);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*开机自检eeprom初始化\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void init\_eeprom(void)

{

a\_a = byte\_read(0x2002);

if(a\_a != 1) //新的单片机初始单片机内问eeprom

{

a\_a = 1;

write\_eeprom(); //保存数据

}

}

//按键响应程序，参数是键值

//返回键值：

// 7 8 9 10(清0)

// 4 5 6 11(删除)

// 1 2 3 12(未定义)

// 14(未定义) 0 15(.) 13(确定价格)

void KeyPress(uchar keycode)

{

switch (keycode)

{

case 0:

case 1:

case 2:

case 3:

case 4:

case 5:

case 6:

case 7:

case 8:

case 9:

//目前在设置整数位，要注意price是整型，存储单位为分

if (DotPos == 0)

{

//最多只能设置到千位

if (price<100)

{

price=price\*10+keycode\*10;

}

}

//目前在设置小数位

else if (DotPos==1) //小数点后第一位

{

price=price+keycode;

DotPos=2;

}

Display\_Price();

break;

case 10: //清零键

To\_Zero();

Display\_Price();

FlagSetPrice = 0;

DotPos = 0;

break;

case 11: //校零

RATIO\_deviation=RATIO\_deviation+1;

write\_eeprom();

break;

case 12:

if (RATIO\_deviation > 0)

RATIO\_deviation=RATIO\_deviation-1;

write\_eeprom();

break;

case 13: //确认键

FlagSetPrice = 1;

DotPos=0;

money = weight\*price/1000; //money单位为分

//显示总金额

Display\_Money();

PlayNowInfo();

break;

case 14: //删除键，按一次删除最右一个数字

if (DotPos==2)

{

price=price/10;

price=price\*10;

DotPos--;

}

else if (DotPos==1)

{

price=price/100;

price=price\*10;

DotPos--;

}

else if (DotPos==0)

{

price=price/100;

price=price\*100;

}

Display\_Price();

break;

case 15: //小数点按下

DotPos = 1; //小数点后第一位

break;

}

}

short int weight\_value\_filter(short wvalue)

{

/\*----------------------------------------------------------------\*/

/\* Local Variables \*/

/\*----------------------------------------------------------------\*/

unsigned char i,j;

static short lastest\_5times\_value[5] = {0};

static unsigned char filter\_count = 0;

static short calibration\_voltage = 0;

unsigned char max\_equal\_index = 0;

unsigned char value\_equal\_count[5] = {0};

/\*----------------------------------------------------------------\*/

/\* Code Body \*/

/\*----------------------------------------------------------------\*/

if(filter\_count < 5)

{

filter\_count ++;

lastest\_5times\_value[filter\_count -1] = wvalue;

return lastest\_5times\_value[filter\_count -1];

}

else

{

i = 0;

while(i < 4)

{

lastest\_5times\_value[i] = lastest\_5times\_value[i + 1];

i ++;

}

lastest\_5times\_value[4] = wvalue;

}

/\*Find Most Times Digit\*/

for(i = 0; i < 5; i ++)

{

for(j = 0; j < 5; j ++)

{

if(lastest\_5times\_value[i] == lastest\_5times\_value[j])

{

value\_equal\_count[i] ++;

}

}

}

for(i = 0; i < 4; i ++)

{

if(value\_equal\_count[max\_equal\_index] < value\_equal\_count[i + 1])

{

max\_equal\_index = i + 1;

}

}

/\*If "calibration\_voltage" exist in the "lastest\_5times\_value";Return the "calibration\_voltage"\*/

for(i = 0; i < 5; i ++)

{

if(calibration\_voltage == lastest\_5times\_value[i])

{

return calibration\_voltage;

}

}

calibration\_voltage = lastest\_5times\_value[max\_equal\_index];

return calibration\_voltage;

}

//===============main program===================//

void main(void)

{

unsigned long AdVal;

uchar test\_interval;

Init\_LCD1602();

EA = 0;

Data\_Init();

//初始化完成，开中断

EA = 1;

//背光

//LedA = 1;

LCD1602\_write\_com(0x80); //指针设置

LCD1602\_write\_word("Welcome to use! ");

LCD1602\_write\_com(0x80); //指针设置

LCD1602\_write\_word("WE: . PR: . ");

LCD1602\_write\_com(0x80+0x40); //指针设置

LCD1602\_write\_word("MON: . ");

Display\_Price();

init\_eeprom();

read\_eeprom();

To\_Zero();

while(1)

{

//每0.5秒称重一次

if (test\_interval++ > 50)

{

test\_interval = 0;

//称重，得到重量值weight，单位为g

AdVal = ReadCount();

if (AdVal <= FullScale)

{

weight = 0;

}

else

{

AdVal = AdVal - FullScale;

weight = AdVal \* RATIO\_deviation / 100000;

}

weight = weight\_value\_filter(weight);

//如果超量程，则报警

if (weight >= 10000)

{

beep = 0;

alert = 0;

LCD1602\_write\_com(0x83);

LCD1602\_write\_word("-.---");

}

//如果不超量程

else

{

beep = 1;

alert = 1;

//显示重量值

Display\_Weight();

//如果单价设定好了，则计算价格

if (FlagSetPrice == 1)

{

money = weight\*price/1000; //money单位为分

//显示总金额

Display\_Money();

}

else

{

LCD1602\_write\_com(0x80+0x40+4);

LCD1602\_write\_word(" . ");

}

//清测试标志

FlagTest = 0;

}

}

//获取按键

keycode = Getkeyboard();

//有效键值0-15

if ((keycode<16)&&(FlagKeyPress==0))

{

FlagKeyPress = 1;

KeyPress(keycode);

FlagKeyPress = 0;

}

while(COL1==0 || COL2==0|| COL3==0|| COL4==0);

LCD1602\_delay\_ms(1);

}

}

//以下为语音信息对应播放起始地址定义,A为开始，B为结束

#define sound\_0A 0x0010

#define sound\_0B 0x0012

#define sound\_1A 0x0014

#define sound\_1B 0x0016

#define sound\_2A 0x0018

#define sound\_2B 0x001A

#define sound\_3A 0x001B

#define sound\_3B 0x0020

#define sound\_4A 0x0021

#define sound\_4B 0x0024

#define sound\_5A 0x0026

#define sound\_5B 0x0028

#define sound\_6A 0x002A

#define sound\_6B 0x002C

#define sound\_7A 0x002D

#define sound\_7B 0x0030

#define sound\_8A 0x0032

#define sound\_8B 0x0034

#define sound\_9A 0x0036

#define sound\_9B 0x0038

#define sound\_10A 0x003A //单价

#define sound\_10B 0x003D

#define sound\_11A 0x003F //点

#define sound\_11B 0x0042

#define sound\_12A 0x0043 //金额

#define sound\_12B 0x0048

#define sound\_13A 0x0049 //千克

#define sound\_13B 0x0050

#define sound\_14A 0x0050 //元

#define sound\_14B 0x0053

#define sound\_15A 0x0054 //重

#define sound\_15B 0x0057

#define sound\_16A 0x0058 //十

#define sound\_16B 0x005D

#define sound\_17A 0x005E //百

#define sound\_17B 0x0061

#define sound\_18A 0x0062 //千

#define sound\_18B 0x0065