目　录

自动平衡电桥原理研究与制作

廖阳平

西南大学工程技术学院，重庆 400716

摘　要: 本文论述了测量表征元件参数的物理量——电阻、电抗、电感、电容、损耗值和品质因数的几种方法，并从测量结果的精确度、测量项目及功能的多样性以及测量仪器本身在设计上和制造上的复杂度等项目加以比较，从中优选出基于“自动平衡电桥原理”的测量仪器解决方案。方案确定后，对具体的可以实际实施的硬件电路进行设计，其中包括元器件的选择，元器件参数值的计算，元器件类型的优选，误差对测量结果的影响,设计上的注意事项以及实际制作过程中的具体实施方法进行了论证，最后对主控单片机的程序设计给出总体的流程和规划并完成代码编写，在此基础上综合软硬件，制作出一台实用的样机。

关 键 词: 测量; 自动平衡电桥; 仪器;

Studying the Theory of Auto-balancing Bridge and DIY

Liao Yangping

College of Engineering and Technology, Southwest University, Chongqing 400716, China

Abstract: This paper discussed some kinds of methods for the measurement of passive component's physical value, including the resistance, the reactance, the capacitance, the inductance, the dissipation factor and the quality factor. Comparing their exactness of the result, the functions they build-in and the difficulty or the cost the design or manufacture themselves, I selected *the theory of auto-balancing bridge* as my solution. Then I'll show you how to design the electric circuit, including how to select the right type of component, how to calculate the value of them, how the tolerated error affecting the final result, what note should we pay when we designed it or put it into practice. In the end, I'll give you the flow chart of the micro controller's firmware. As a result, we'll build a sample at all.

Key Words: Measurement; Auto-balancing Bridge; Meter;

0 文献综述

0.1 测量方法综述

经查阅大量国内和国外的相关科技文献资料，特别是著名电子测量仪器解决方案公司的商业化产品的说明书后，总结出现今比较常用的无源元件参数测量方法包括：电桥法、谐振法、电压电流法、RF 电压电流法、自动平衡电桥法和网络分析法。综观各种测量测试方法，他们各有其优缺点，因此有其最佳的应用领域。其中有些是有着比较悠久的历史，几乎伴随着电子这门学科的发展而同时诞生的，比如电压电流法，其中一些是随着电子科学技术发展到特定的程度，在某个特定方向和领域有了一定的研究深度后才发展起来的，比如网络分析法，其中的一些则是专注于解决对自动化测试测量的需求，以及计算机技术软件和硬件发展到一定水平才渐渐发展起来的，因而其具有比较大的技术上的优势和使用便利上的优势，比如自动平衡电桥法。

0.2 研究现状

经查阅各方资料后发现，最早的与自动平衡电桥测量法相关的专利始见于美国，专利号：719810，标题：METHOD OF AND APPARATUS FOR AUTOMATIC MEASUREMENT OF IMPENDANCE OF OTHER PARAMETERS WITH MICROPROCESSOR CALCULATION TECHNIQUES[1]，发明者：Henry P. Hall，申请日期：1976 年 9 月 2 日。该专利讲述了一种基于数字相敏检波器，积分式 ADC 和微控制器的元件参数测量解决方案。1979 年 2 月，美国 HP 公司（现在其电子测量仪器部门已独立，更名为 Agilent 公司）在其 1979 年 2 月份的 HEWLETT-PACKARD JOURNAL[2]技术月刊的 24 到 31 页刊载了其最新的基于自动平衡电桥的测量仪器 4274A 和 4275A 的产品介绍，里面也详细介绍了其主要组成电路的原理。

国内大部分发表于各种期刊（包括学术性质的）的论文[3,4,5,6,7]都对一些基本原理有着大量重复性的论述，但缺乏对某种测试测量理论有更深入的理解和更具体的论述。早期的主要是介绍 HP 公司的仪器维修方法，其后陆陆续续有对测量电路具体形式的研究和高频化的设计措施描述，近期文章在创新性上已无进展，只在细枝末节上下功夫或是对使用该种仪器的注意事项、使用方法进行介绍。这从某种方面

也体现了我国在电子测量仪器方面与国外先进水平差距巨大。

测试与测量技术是与实际应用和需求密切相关的，比如各种电子元件的生产就迫切要求对大批量的产品快速进行分捡和定级、对于一些高精度等级的元件更是要求测试测量仪器具有更高的精度等级，因此造就了这门艺术与其他技术领域的不同，很多新兴技术是源于盈利性的科技公司。查看了比较多 HP 公司的相关开放性文档和资料[8]后对于元件参数测试测量方面有了更多的深入了解，可以说它在当前的电子元件测量仪器商业化产品领域取得了最高成就。所以，如果进行相关方面的研究，我建议从外国科技公司发布的公开资料攫取更富有用的信息。

0.3 商业化产品

这部分选取有代表性的手持式数字电桥产品进行比较以了解最新的该方面进展，其中包括了 Agilent 公司的 U1732A[9]，国内常州市同惠电子有限公司的 TH2822C[10]。

表 0.0　产品比较

Tab. 0.0　Compare of Products

型号	U1732A	TH2822C
功能	L/C/R/D/Q/θ	L/C/R/Z/D/Q/θ/ESR
测试频率	100Hz,120Hz,1kHz,10kHz	100Hz,120Hz,1kHz,10kHz,100kHz
测试电平	0.6V_{RMS}	0.6V_{RMS}
测试端	三端；五端	三端；五端
基本准确度	0.5%	0.25%
温度系数	0.15*(规定精度)/℃	/
测量速度	1SPS	4SPS
等效电路	串联；并联	串联；并联
量程方式	自动；手动	自动；手动
容差模式	1%；5%；10%；20%	1%；5%；10%；20%
清零功能	开路；短路	开路；短路
显示器	LC 主、副参数双显示带背光	LCD 主、副参数双显示带背光
接口	IR-USB	Mini-USB(虚拟串口)
读数	L/C/R：最大显示 19999 D/Q：最大显示 999	L/C/R：最大显示 40000 D/Q/θ：最大显示 9999
功耗	40mA	25mA
其他	自动关机	/

1 引言

无源元件是电子电路中应用最为广泛的元器件类型之一，是电子电路不可或缺的基石。因其所起的重要而基础的作用，其质量对整个电路的性能起着关键性的作用。综观各种测试与测量仪器，几乎都具备了一定的无源元件参数检测能力。万用表是平常使用得最多的手持式测量仪器，它具有很好的便携性，加上数字电子技术的引入，测量精度也达到了可以接受的范围，但对于无源元件的测量，大多数万用表仅能测量其电容值和电阻值，测量范围也十分有限，测量方法更是与其实际的工作情况和标准测量方法相去甚远。台式数字电桥克服了万用表的不足，更是具有极高的测量精度，成为元件生产厂家和实验室等专业部门对无源元件进行检测，分级和测量的首选，不过其极差的便携性和高昂的价格限制了其进一步地拓展应用。手持式数字电桥不论在功能上，便携性上，还是成本上都具有很大的竞争优势。本论文对数字电桥的原理就应用方面的可行性进行了深入研究，在此基础上制作出一台样机进行验证。全篇从对测量方案的比较开始，最终选定方案，阐述了方案实现中的关键而基本的几个原理，由原理形成整体的硬件电路设计思路，继而分模块逐一地选择元器件并实现具体电路，进而设计控制软件，最终进行整机的软硬件联合调试，成功制作出一台样机，通过总结调试过程和最终结果提出了改善性能的方法和建议。

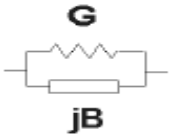
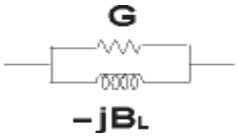
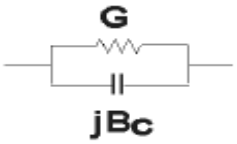
2 元件等效电路模型和测量方法

2.1 元件等效电路模型

无源元件参数测量一般包括其电阻、电感、电容以及与其相关的副参数，比如损耗角（θ角）、品质因数（Q值）和损耗值（D值）的测量。电阻表示电路中能量的损耗，电容和电感则表示电路中电场能量和磁场能量的存储。对于一个实际的无源元件，电阻、电感和电容都不是理想的，而是存在着寄生电容、寄生电阻和损耗等。表2.1给出了电阻、电感和电容在考虑了主要寄生参数后的等效电路模型[8]。

表 2.1　等效电路模型

Tab. 2.1　Equivalent of Circuit Model

参数	模型	参数	模型
$Z = R + jX$		$Y = G + jB$	
$Z = R + jX_L$		$Y = G - jB_L$	
$Z = R - jX_C$		$Y = G + jB_C$	

2.2 测量方法简述[8]

(1) 电桥法　如图 2.1，当电桥平衡时流过检流器 D 的电流为零，被测阻抗可以从与其他三个已知桥臂的关系中求出。桥臂使用不同类型的无源元件（电阻、电感或电容）组合，可以测量不同类型的无源元件参数。

(2) 谐振法　如图 2.2，调节电容 C 使得电路谐振，根据谐振频率（OSC 的振荡频率）、Q 值和电容 C 值可以算出待测阻抗的电阻和电抗并由此计算出电感值。由于测量电路损耗低，可以测试高达 1000 的 Q 值。除此之外，还可以交换电容和电感的位置或将将电路改成串联形式以适应不同参数的测量。

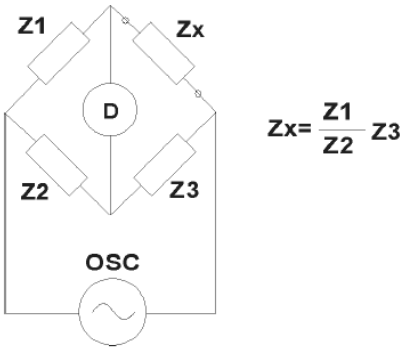


图 2.1　电桥法

Fig. 2.1　Bridge Method

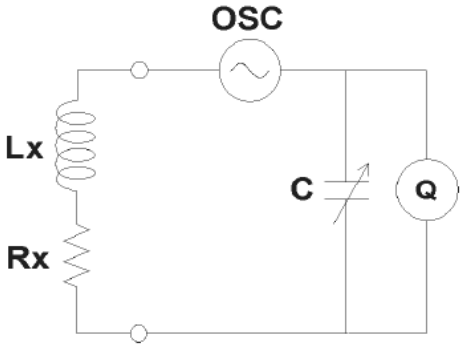


图 2.2　谐振法

Fig. 2.2　Resonant Method

(3) 电压电流法　如图 2.3，由电压 V1 和电流 I 计算被测阻抗，其中，电流由标准电阻 R 上的电压 V2 算出。

(4) RF 电压电流法　如图 2.4，与电压电流法的原理相同。由于使用了高的测试频率，要在阻抗匹配的条件下测量；有两种连接方法，以分别适应高阻抗和低阻抗的测量。

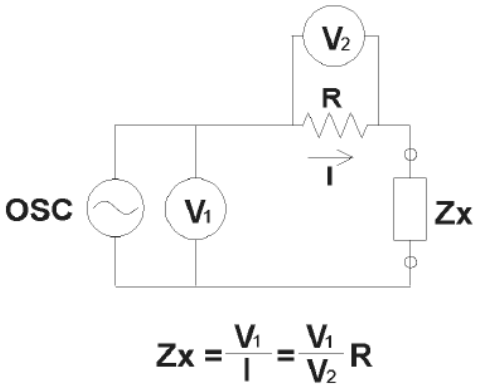


图 2.3　电压电流法

Fig. 2.3　I-V Method

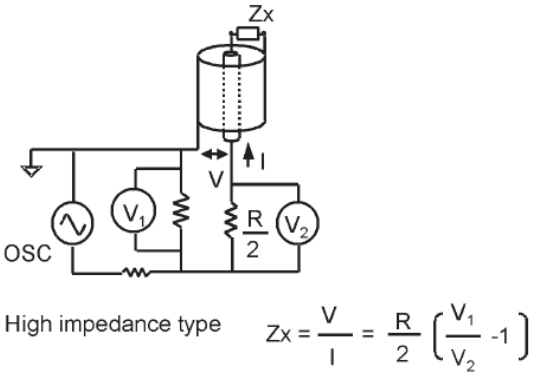


图 2.4　RF 电压电流法

Fig. 2.4　RF I-V Method

(5) 自动平衡电桥法　如图 2.5，通过被测无源元件 Zx 的电流 Ix 也流过电阻 Rr（Ir= Ix），Low 点电位由于运放的反馈作用保持为零(即虚地)，通过 Vx 和 Vr 可以计算出 Zx。

(6) 网络分析法　如图 2.6，通过测量输入信号与反射信号之比得到反射系数，由此计算出 Zx。由于利用了反射系数，因此该方法只适用于较高频率的无源元件参数测量。

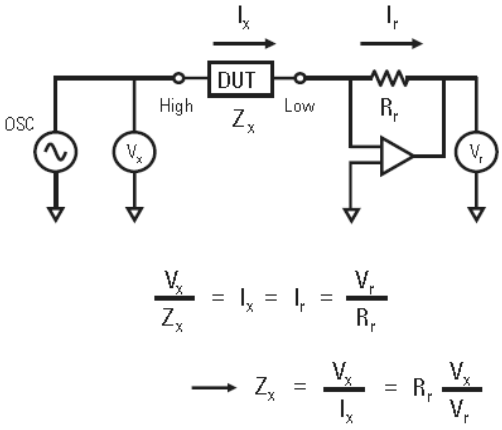


图 2.5　自动平衡电桥法

Fig. 2.5　Auto-balancing Bridge Method

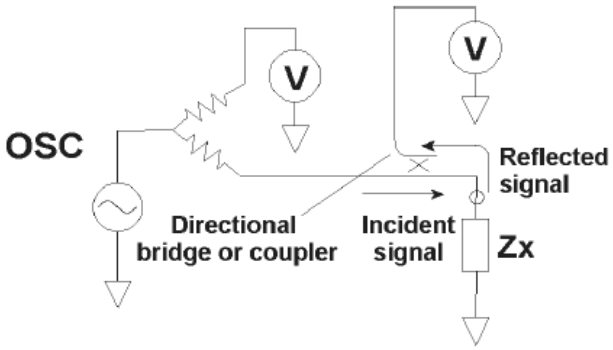


图 2.6　网络分析法

Fig. 2.6　Network Analysis Method

2.3 测量方法对比

各种测量方法的比较见表 2.2[1]。

表 2.2　各种测量方法比较

Tab. 2.2　Comparing of Measurement Methods

方法	优势	劣势	可用频率范围	一般应用
电桥法	精度高；频率宽；成本低	需人工调节；单一电桥频率窄	DC～300MHz	标准实验仪器
谐振法	可精确测量很大的 Q 值	需调至谐振；阻抗精度低	10KHz～70MHz	高 Q 值测量
电压电流法	可测量接地元件；适合探头测试需求	频率受限于探头传输性能	10KHz～100MHz	接地元件测量
RF 电压电流法	精度高；范围宽	频率受限于探头传输性能	1MHz～3GHz	射频元件测量
自动平衡电桥法	精度高；范围宽；可测量接地元件	高频不可用	20Hz～110MHz	通用元件测量；接地元件测量
网络分析法	宽频率，未知阻抗接近特性阻抗时精度高	变频时需校正，范围窄	300KHz～	射频元件测量

综合以上，选择基于“自动平衡电桥”原理的元件参数测量解决方案具有更大的比较优势。这是基于其更适宜于通用无源元件参数测量，具有比较好的测量精度，有很宽的测量范围，更适用于平常对于常见无源元件的参数进行甄别、筛选和测量。

3 自动平衡电桥原理分析

3.1 自动平衡电桥原理

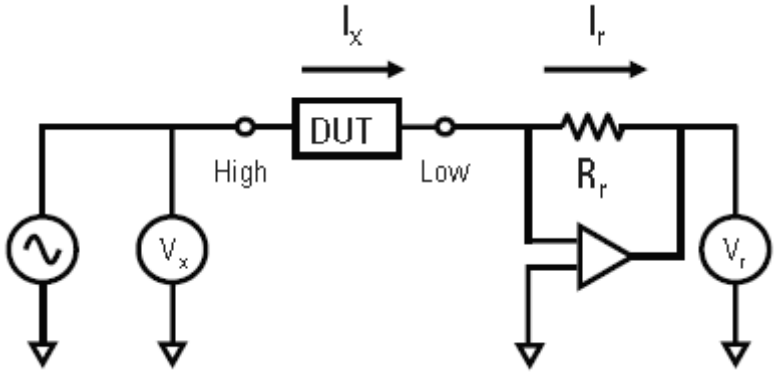


图 3.1　自动平衡电桥原理

Fig. 3.1　Theory of Auto-balancing Bridge

运算放大器是一个高增益的放大系统，理想情况下其开环增益 $K_0 \to \infty$，输入

阻抗 $Z_i \to \infty$，在同相输入端接地的情况下，其反相输入端也可以认为是零电位（即虚地点）。故

$$I_x \approx I_r,\ V_x = I_x \cdot Z_x,\ V_r = I_r \cdot R_r \tag{3-1}$$

$$\therefore Z_x = {V_x}/{I_x} = R_r \cdot \left({V_x}/{V_r}\right) \tag{3-2}$$

其中，Z_x 为待测阻抗，V_x, I_x, V_r, I_r, Z_x 都为实数，包含实部与虚部分量。

由（3-2）式可见，若能分别测量出这两个电压的实部和虚部分量的大小，就可以计算出被测阻抗的值并由此计算出需要的无源元件参数。

设 DUT 两端电压可以表示为 $V_x = a + jb$，流过 R_r 的电流 I_r 在其两端产生的压降可以表示为 $V_r = c + jd$。若被测阻抗 Z_x 表示为串联形式，即 $Z_x = R + jX$，将以上各式代入（3-2）式并加以整理和变换后得到[8]：

$$R = R_r \cdot \frac{ac + bd}{c^2 + d^2},\quad X = R_r \cdot \frac{bc - ad}{c^2 + d^2} \tag{3-3}$$

若被测阻抗表示为并联形式，同样可以推导出相关公式。

3.2 相敏检波器原理

相敏检波器是实现一个电压实部和虚部分量分离的关键部件。若被测阻抗 Z_x 两端电压可以表示为 $V_x = a + jb = |V_x| e^{j\varphi}$，则有 $a = |V_x| \cos\varphi$，$b = |V_x| \sin\varphi$，如图 3.2 所示。若控制相敏检波器的信号为 0^o 基准信号，则相敏检波器的输出为[8]：

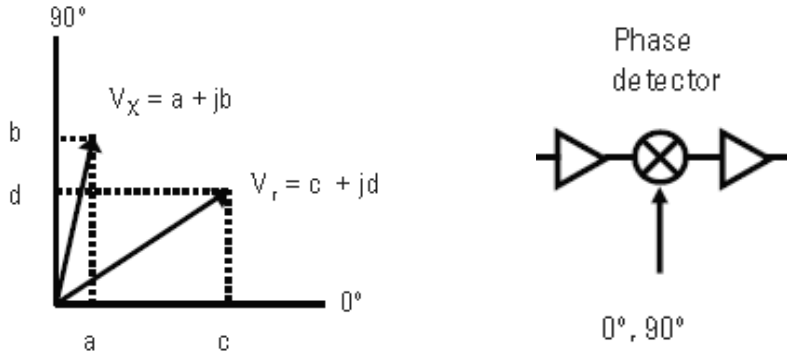


图 3.2　相敏检波器原理

Fig. 3.2　Theory of Phase Detector

$$V_0=\begin{cases}|V_x|\sin(\omega t+\varphi) & 0\le t\le T/2\\ 0 & T/2<t\le T\end{cases} \qquad (3\text{-}4)$$

$$\overline{V_0}=\frac{1}{T}\int_0^T V_0(t)dt=\frac{1}{T}\int_0^{\frac{T}{2}}|V_x|\sin(\omega t+\varphi)dt=-\frac{1}{\pi}|V_x|\cos\varphi=-\frac{1}{\pi}a \qquad (3\text{-}5)$$

可见，相敏检波器完成了对电压实部分量的分离。当控制相敏检波器的信号为 90^o 基准信号时，相敏检波器可以分离出电压的虚部分量。

4 硬件设计

本机由 6 大模块组成，分别是：电源模块、按键与显示模块、滤波器模块、前端信号处理模块、模数转换模块和 CPU 模块。整机合计使用 22 块集成电路外加一块液晶显示模组。

整机硬件方框图如图 4.1。

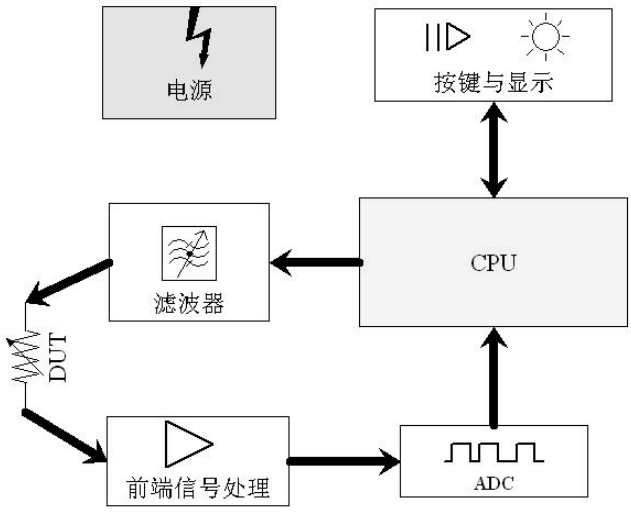


图 4.1　硬件方框图

Fig. 4.1　The Block Diagram of Hardware

4.1 电源模块

本机仅使用一个额定电压值为+8.5V（实际允许容限值为+6.5V～+10V）的外接直流电源供电，主要的模拟电路和数字电路均使用电压值为+5V 的稳压电源供电，仅部分电路（包括模拟开关、运算放大器和模数转换器）需要另一路电压为-5V 的电源供电，这是由机内的负电源电压电路从外接直流电源产生的。这个设计，大大简化了对外接直流电源的要求——无需双电源供电，提高了使用上的便利性。

图 4.2 中，低导通压降的肖特基二极管 D101（1N5817）提供了简单的输入电源反接保护，防止用户不慎接反外接直流电源损坏内部电路；U101（AMS1117-5.0）为最大输出电流达 1A 的低压差（Max. 1.3v）三端直流稳压集成电路，为整机的数字电路和模拟电路供电；U102（MC34063）为 DC-DC 变换器，由其将外接直流输入电源变换为-7.2V 的负电源电压，因为开关变换器固有的较大干扰，加入了由 C109、L103 和 C110 组成的Π型滤波器和 U103（LM7905）三端直流三端稳压集成电路进行二次稳压以进一步抑制开关电源噪声，获得比较纯净的直流电源，提高负电源质量，减少对负载电路的噪声注入。

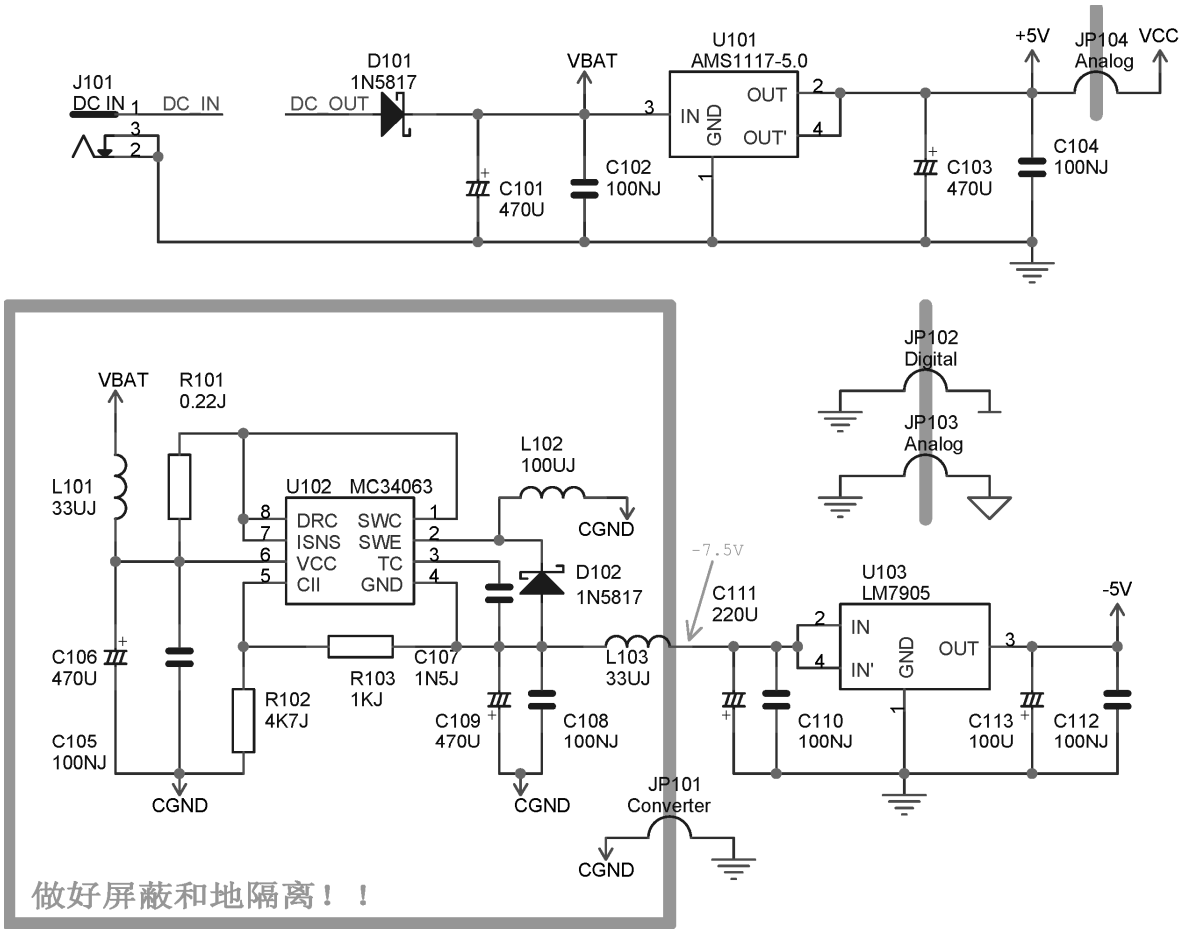


图 4.2　电源

Fig. 4.2　Power Supply

4.2 按键与显示模块

本机与用户的交互使用了 6 个输入按键，可供用户选择测量项目、选择量程、选择频率、选择等效电路模式、开关自动量程以及开关背光等功能；显示器使用了一块 1602（U402）的液晶模组，最多可以显示 16 字 X2 行的 ASCII 字符，直接使用

字符液晶相比于数码管的纯数字显示，改善了人机交互的界面，增加对用户的友好性。

图 4.3 中，U401（74HC164）为 8 位的串入并出移位寄存器，它的加入在不增加大量成本的情况下，简化了按键与显示组件和主电路板的连线数量，此处仅有 2 条，也大大减少了对单片机端口资源的占用，降低了对单片机的要求，提高了单片机端口的使用效率，使采用少引脚的单片机成为可能，从另一方面降低了总成本；Q401（8050）用于开关液晶模组的背光，在环境亮度充足的情况下降低整机功耗，在使用电池的情况下延长使用时间；RV401（10KJ）用于调节液晶模组的对比度；S101 为整机的电源开关；LD401 为整机电源指示灯；S401-S406 为 6 个输入按键；J401 为连接按键与显示模块和主板的排线座。

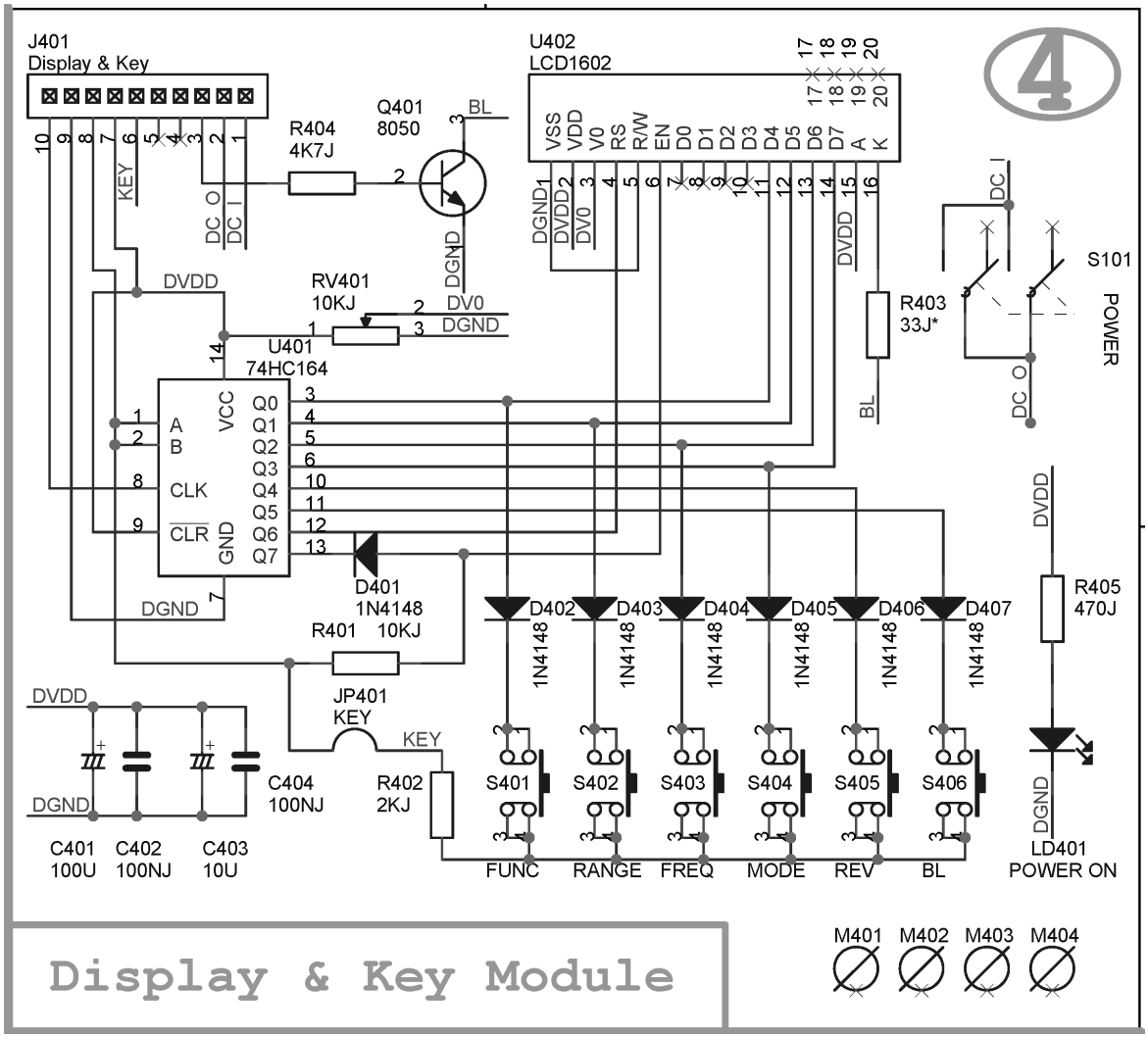


图 4.3　按键和显示模块

Fig. 4.3　Display and Keys Module

4.3 滤波模块

本机共有 4 个测式频率可供选择，这包括了 100Hz、120Hz（专供电解电容测量用）、1KHz 和 10KHz，相应地，本机使用了 4 通道的带通滤波器将由双 D 触发器产生的正交方波信号中的一路滤波成正弦波供测量电路作为激励信号激励待测件，相比于传统的使用模拟电路产生正交信号的方法，数字电路大大简化了对正交信号产生电路的要求并大幅度地提升了两个信号正交精度。

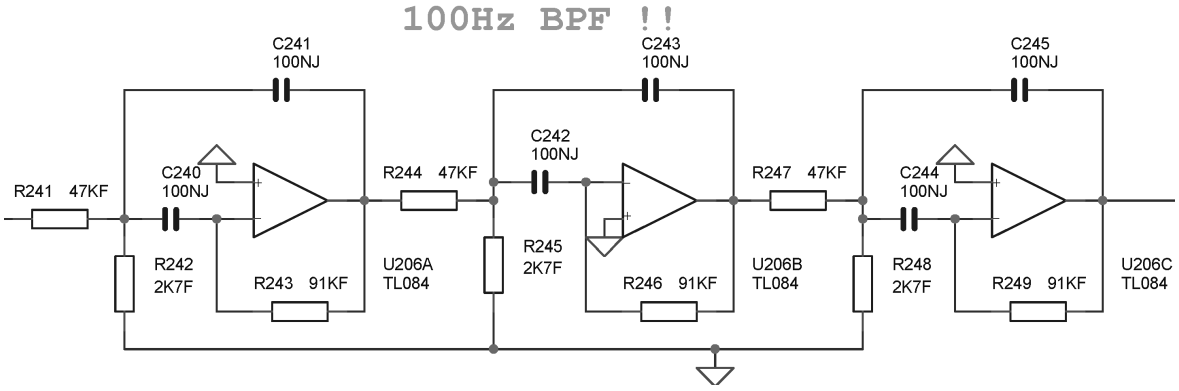


图 4.4　100Hz 带通滤波器

Fig. 4.4　100Hz BPF

图 4.4 为 4 通道中的一路，中心频率 100Hz，由 3 级的二阶无限增益多路反馈带通滤波器[13]级联而成。其幅频特性的仿真结果（使用 Linear Technology Corporation 的免费仿真软件 LTSpice IV）见图 4.5。

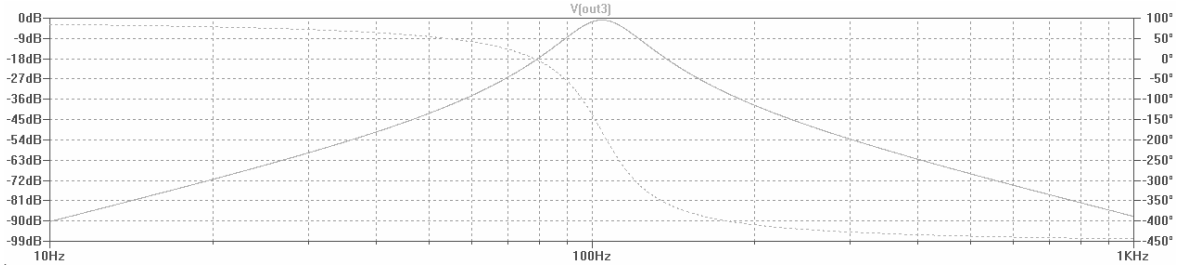


图 4.5　100Hz 带通滤波器特性

Fig. 4.5　Character of the 100Hz BPF

4.4 前端信号处理模块

前端信号处理模块主要包括量程电阻切换电路、自动平衡电桥核心电路和差分放大器电路。

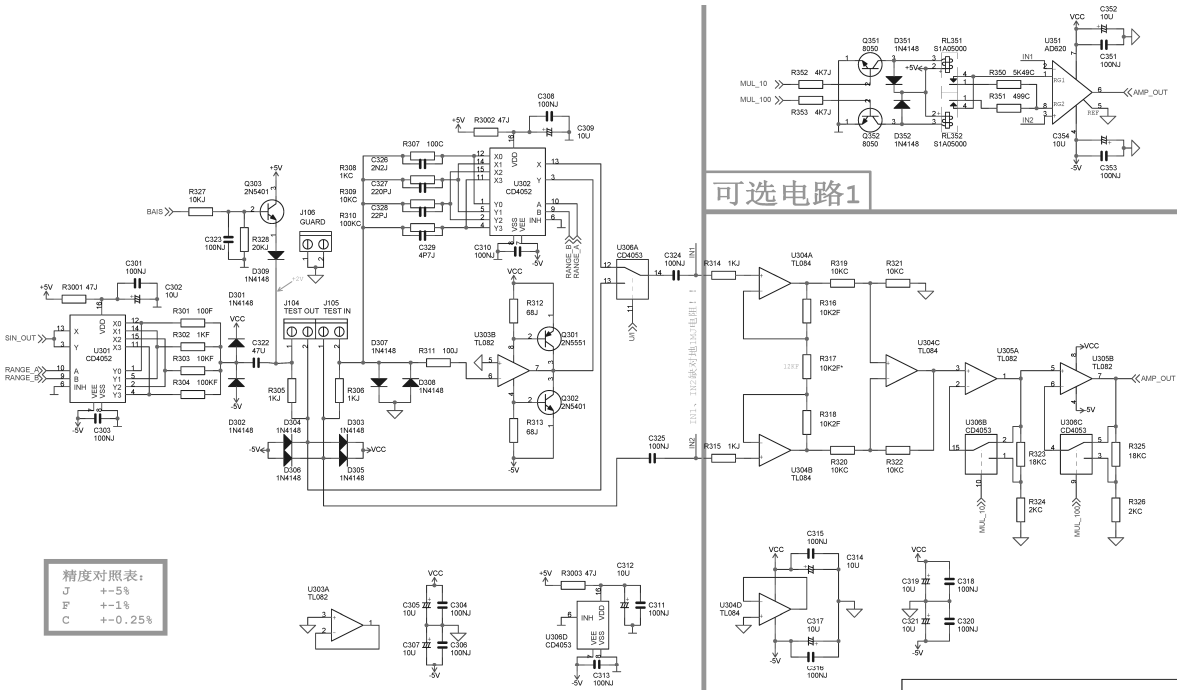


图 4.6　前端信号处理模块

Fig. 4.6　Signal Pre-processing Module

图 4.6 中，U301、U302（CD4052）为双刀四掷模拟开关，共同完成量程电阻的切换；U303B（TL072）为采用 JFET 输入级的低偏置电流、高压摆率和低失真率的双运放，组成自动平衡电桥的核心部分，自动完成电桥的平衡过程；U304（TL074）中的三路运放组成典型的三运放仪表放大器，具有较高的 CMRR 和输入阻抗，可以抑制输入共模电压对输出信号的影响；U305（TL072）和 U306（CD4053）组成 1、10 和 100 倍三档可选的程控放大器，将信号幅度调理到合适的值已匹配模数转换器的满量程电压，获得最高的转换精度；Q303（2N5401）在测量电解电容时为其提供 2V 的直流偏置电压以模拟电解电容实际工作状态，获得准确的测量参数；D301-D308（1N4148）为输入和输出端提供过电压保护，可以防止因为电容未放电测量时输入的过电压使 U301 和 U302 产生闩锁而损坏集成电路；U306A（CD4053）用于切换待测电压或电流；由于由分立元件组成的三仪表放大器对电阻的匹配要求极高，考虑到这些电阻的实际可获得性和成本，添加了由 U351（AD620）成品仪表放大器组成的备用电路以备对比测试和电路优化，另一方面它可以提供比自制三运放仪表放大器更优异的性能并降低功耗，不足之处是成本比较高。

4.5 模数转换模块

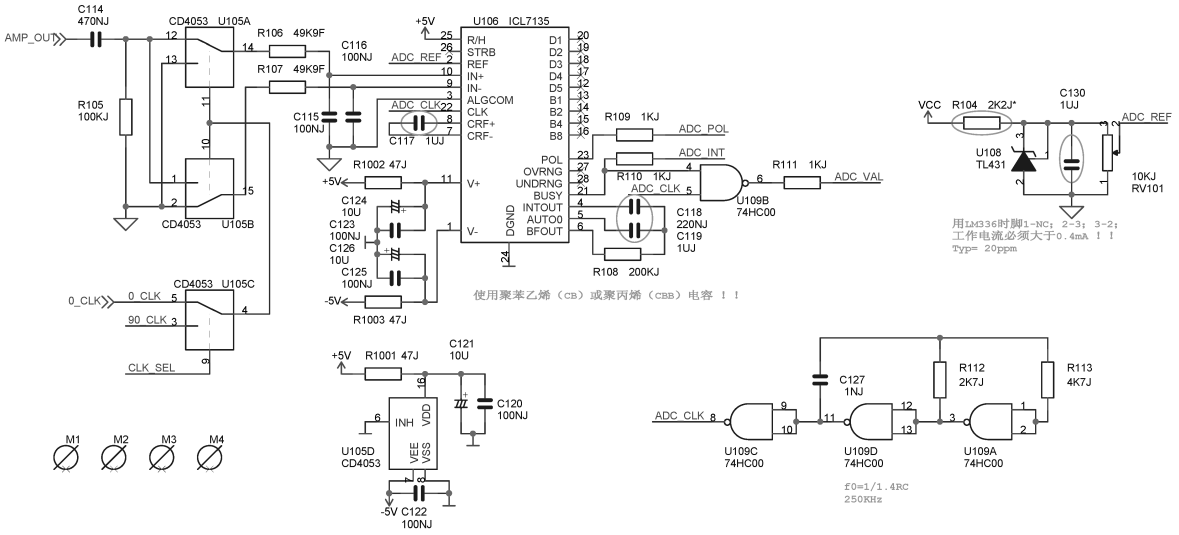


图 4.7　模数转换模块

Fig. 4.7　ADC Module

图 4.7 中，U105（CD4053）组成相敏检波器，由其完成对两个被测电压的实部和虚部分量的分离；U106（ICL7135[11]）为 $4\frac{1}{2}$ 位的模数转换器，输出格式为 BCD 码，本设计巧妙得利用了 BUSY 引脚，改用计数方式，仅用 3 线就完成了与单片机的接口，简化了与单片机的连线，减少了对单片机端口资源的占用，提高了单片机端口的使用效率；U108（LM336）为低温漂电压基准（Typ. 20ppm/℃）,提高了测量的精确度；U109（74HC00）为四与非门，用于产生时钟信号驱动模数转换器并将 U106 的 BUSY 引脚输出转变成一连串的脉冲输出到单片机的定时、计数器 T/C1 进行计数以获取转换结果；值得注意的是电容 C118（220NJ）的介质类型挑选很重要，必须使用极低介质吸收性能的电容类型，比如聚丙烯（CBB）电容，这对整个模数转换器的线性度会产生极大的影响。

4.6 CPU 模块

本机使用一块 Atmel 公司生产的 RISC 指令集单片机 ATmega8[12]（U104），当时钟频率为 8MHz 时，处理的峰值速度可达 8M MIPS，并且配备了丰富的外围设备控制部件。这里主要使用了其控制资源中的一个外部边沿中断 INT0，一个 8 位的定时器 T/C0，一个 16 位的计数器 T/C1 和一个运行于 PWM 模式的一个 8 位定时器 T/C2。相比于 51 单片机，AVR 单片机具有的硬件 PWM 模块可以产生精确频率的方波，简化硬件电路的同时提高性能，另一方面，本机也涉及了大量的数学运算，比如 arc tan，对单片机的运算性能要求较高，传统的 51 单片机可能无法满足高实时性的要求。

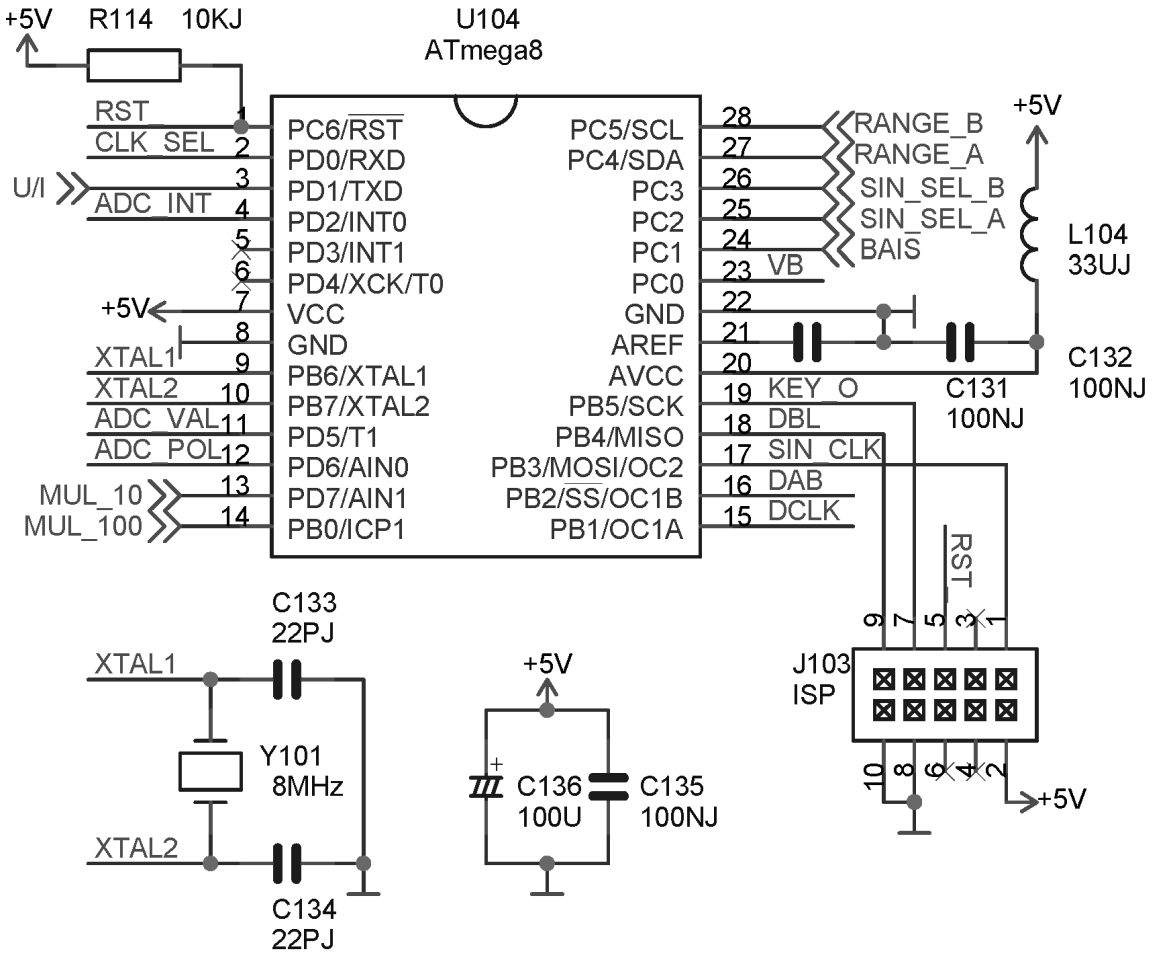


图 4.8　CPU 模块

Fig. 4.8　CPU Module

4.7 其他

本机除了以上几个大的功能模块外，还有一些其他的辅助电路。

最主要的是由 U107（74HC74）双上升沿触发的 D 触发器组成的正交信号发生电路，由单片机产生的方波信号进入双 D 触发器后进行了 4 分频，同时由于连接的方式，两个 D 触发器 Q 输出端的信号相位差 1/4 的周期，由此产生出正交的两路方波信号。

测试电解电容时，通常都要求为其添加一个直流偏置电压，以使其测量条件接近于实际的正常工作环境，提高测量结果的准确性。本机由单片机引脚直接产生高电平，经由电阻分压滤波后由三极管射极跟随器 Q303（2N5401）跟随后输出固定的 +2V 电压作为测试时的直流偏置电压。

本机涉及 DC-DC 电源变换器、数字电路与模拟电路，还要对比较小的信号进行放大与测量，因此，解决好电路之间的相互干扰问题至关重要。这里主要采用了以

下几种设计上和具体方案实施上的解决方法：①.对于 DC-DC 电源变换器、数字电路与模拟电路使用相互分离的地，仅在电源处一点相接，防止数字电路的地电流从模拟电路经过时在地电阻上产生的压降成为信号注入模拟电路而产生的干扰；②.模拟电路和数字电路有自己相对独立的电源供给线路，也仅在稳压器输出端处一点相接，DC-DC 变换器使用电感与整机主电源隔离开；③.PCB 布局上，加强了各个集成电路的电源滤波和退耦的措施，尽量减少来自电源的干扰和器件本身对外释放的干扰。对于模拟电路部分，仅在电路板背面使用大面积铺铜，正面元件密集区无铺铜，没有增加运放同相输入端的对地寄生电容，避免因其而产生不稳定甚至是自激振荡。对于数字电路部分，采用大面积铺铜，抑制并吸收干扰信号，对几个易产生干扰的信号线（比如：晶振引脚）提前进行优化布局布线并锁定，后期充分利用软件的自动功能简化多余的人工布线操作，提高效率。

5 软件设计

本机全部程序使用 C 语言编写，充分利用了标准库函数，简化了一些重复性的工作，将更多的精力用于界面的易用性，性能的稳定性，功能的多样性等方面的开发工作。程序编译器采用 IAR 公司的 IAR Embedded Workbench for Atmel AVR（版本号 V5.10A），这是一个专门用于开发基于 AVR 单片机的集成开发环境，可以使用汇编语言、C 语言和 C++语言进行源程序的编写和编译，最终生成单片机固件。此外，IAR 集成开发环境本身集成了一个强大且易用的软件或硬件调试工具，生成的代码可以按需定级优化以获得最优的性能。

在软件架构方面，充分利用了 C 语言结构化的组织优势，本机将各个功能独立编写成子程序，提高了源程序的可读性，可以便利的增加新功能，还降低了调试的复杂度，在不到半个月的时间内完成了源程序的编写和上机调试。整机共有 6 个源程序文件，包括 C 文件“main.c”、“hc164.c”、“my.c”及其对应的头文件“main.h”、“hc164.h”、“my.h”，总计近千行的源代码。

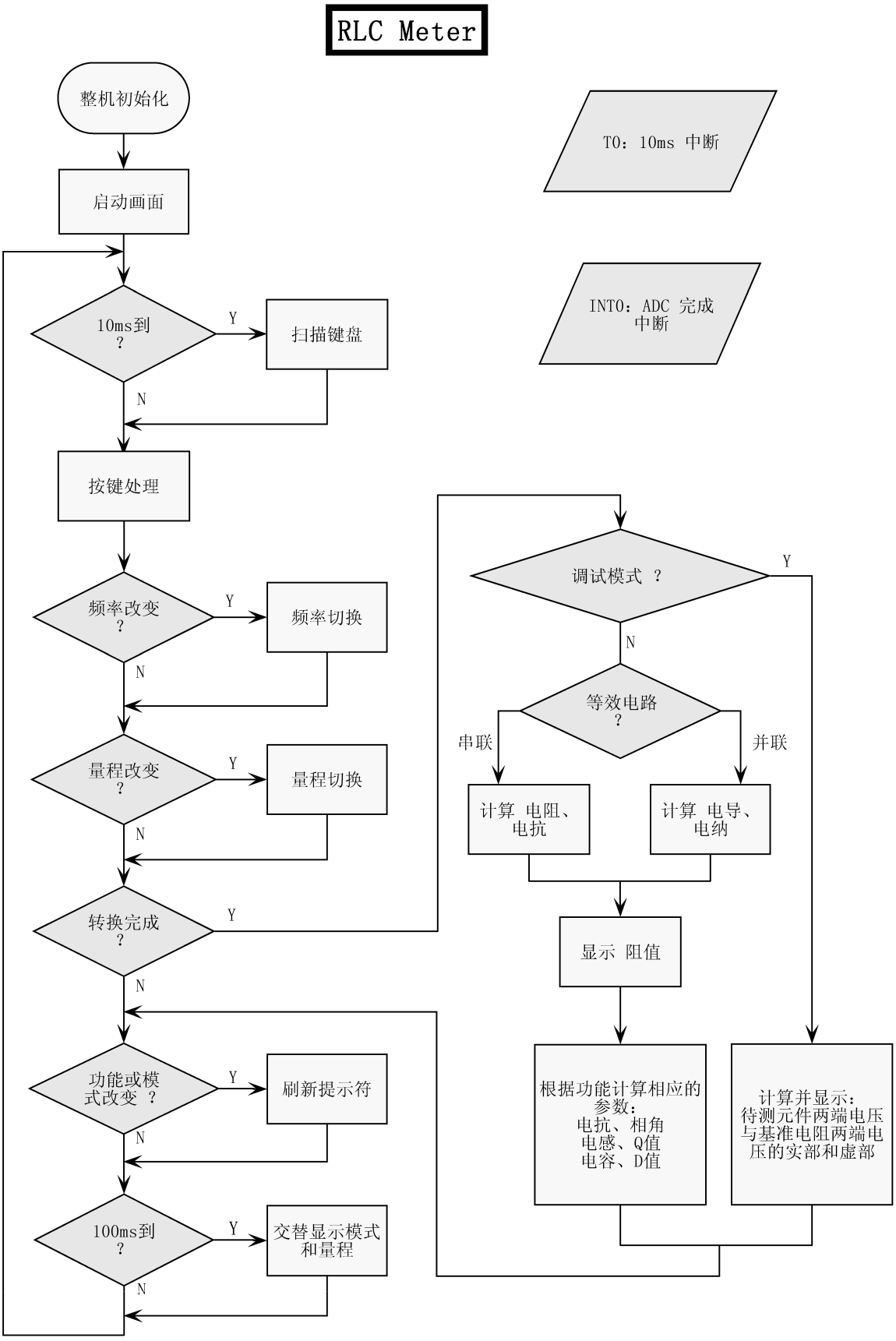


图 5.1　程序流程图

Fig. 5.1　The Flow-chart of Program

整机的主程序流程图如图 5.1。为了提高效率，在程序主循环中采取了一定的措施，仅对必要的操作进行响应，比如，如果频率未变时，就没有重复刷新提示符并对硬件的操作，这提升了整体的效率并且提高了吸引速度，也为后续加入新功能提供了运行时间上的富裕。

6 调试

开发一个新电路，基本上只有电路原理可供借鉴，而对于元器件参数的取值只靠设计上的保证通常是远远不够的，要经过一番调试和多次试验才能定下参数并取得最优的效果。

对于一个复杂的系统，硬件上分模块的调试是比较好的做法，这大大减少了其他电路模块的不确定性对这部分的影响，减少了调试操作的复杂性。在本样机的试制过程中，每焊好一个模块后马上进行该模块的调试，防止问题一步步累积以至最终在一个复杂系统中难于确定确切的故障点。

6.1 电源调试

电源是整个电路的能量之源，他与所有的电路组成部分都有紧密的联系，因此其质量的高低对于整个电路的功能实现和稳定性等起着至关重要的作用，对于高精度的模拟电路而言，做好电源部分至少就成功了 50%，使用一个失败的电源，其他的也就无从谈起。

本电路初步设计时使用的是由外部的 220V 市电变换过来的直流电源，为了简化整机对供电电源的要求，内部使用了 DC-DC 直流变换器来获得一路电压为-5V 的电源。调试时发现该直流变换器的干扰是比较严重的，用示波器观测经过低压差稳压器稳压后的+5V 主电源时，上面叠加了 $20\,mV_{P-P}$ 的高频尖峰脉冲；即使使用了一块负压输出的三端稳压器对直流变换器的输出进行了二次稳压后，$15\,mV_{P-P}$ 的高频尖峰脉冲还是出现在了-5V 电源输出中。

可以预期的解决方法有：①使用开关电容电源变换器，查阅了多家半导体公司的资料后发现大电流输出的型号售价颇高，如 Linear Technology 公司的 LTC660[13] 的最大输出电流可达 100mA，售价$3（约合￥20）；②使用两块锂电池来提供电源，这是一个比较好的解决方案，满足了便携性和高性能的要求，由于正负电源供电电流的不同，这增加了电池过放电检测与保护和充电电路的复杂性。

在此次制作过程中，很感谢 Linear Technology 公司，感谢其免费为我提供LTC660 及其他的半导体芯片样品。

6.2 模数转换器调试

本模数转换器最高分辨率可达 100 μV，其对稳定性的要求不言而喻。仔细研读 Intersil 公司的 datasheet 后，可以确定以下两个调试的关键点：基准电压；积分电容。

基准电压：本设计对于绝对精度的要求不是很高，这是由于由模数转换器直接得到的 4 个值进行了相对运算，相反地，对于转换器的线性度和稳定性要求很高。本设计使用美国国家半导体公司生产的 LM336 带隙基准电压源作为电压基准，温度漂移的典型值为 20ppm/℃，可以满足高的温度稳定性。由于缺乏相应的高精度仪器，其性能由其自身规格予以保证。

积分电容：ICL7135 对积分电容的极小介质吸收性能要求很高，推荐使用聚丙烯类型电容，相应的国内代号为 CBB。具体调试步骤如下：（注意断开外部输入）将负输入端（脚 2）接地，正输入端（脚 10）接至参考电压端（脚 2），同时按下“Mode”和“Auto-Range”两个按键以进入调试模式，之后液晶显示器的将显示比率值，要求其范围达到 9997.00～10000.00（可能开头有负号）。换用不同的积分电容（C118）可能有不同的效果，试过几种电容后发现，由两只 100nF 的安规电容（上标商标为 TENNA）并联成的积分电容，比率值可达 9999.00 以上，另外一个是不知名的红色外壳电容，也可达 9998.00 以上，其他的电容，不管是绿色外壳（以前通常是涤纶（CL）电容）的还是红色外壳的，最大只能达到 9985.00，差的甚至低至 9970.00，且与其上标的耐压值无关，应该避免使用此类电容，它们会极大影响转换的线性度和精确度。至此，模数转换模块调试结束。

6.3 降低功耗

经过考虑后，对原先的设计进行了改进，以降低功耗。①将用于切换带通滤波器的电子开关一分为二，一半用于切换输入和输出，一半用于切换该滤波器所用运算放大器的正电源和负电源，这样可以降低一半以上的功耗，实测发现电子开关和逻辑电路耗电很低，主要是运算放大器功耗很大（可达 6mA/块）。②使用开关电容电源变换器 LTC660（Linear Technology 公司，内阻典型值 6.5ohm,实测 5ohm）产生负电源，经过比较，MC34063 电源变换效率很低，而且干扰很大，使用 LTC660 的

主要缺点是比较贵，报价$3。

经以上改进后，负电源电流 20mA。用 7V 的电源供电时，实测整机电流小至 83mA，关闭液晶背光还可以减小 14mA，这个指标用两节锂电池串联供电，是一个不错的结果。

6.4 提高稳定性

实际发现，主参数（比如电阻，电容，电感）仅最后一位有±2 以内的跳动（显示都取 4 位有效数字），但副参数跳动比较大，而且在不同量程的值有无法容忍的差异，部分由于我用的基准电阻精度不够（1%），主要还是其值偏小（相对主参数而言），希望能进一步加以改进。

7 使用指南

7.1 显示器显示说明

LCD1602 液晶可以显示 2 行，每行 16 个字符，总计 32 个字符，对于显示内容的安排主要是充分利用其资源，在显示尽可能多的内容时，又兼顾条理性和美观性。主界面可以分为 4 部分，每一行由两部分组成，各个部分都有英文简写的引导符对显示内容进行提示，增加了人机界面的友好性。以图 7.1 显示时的一个情况为例，下面对各个显示的要素进行解析。

R	=		1	.	0	0	0	K		R	N	G	1	0	0
X	=		7	.	3	7	6	m		θ			0	.	5

图 7.1　显示示例

Fig. 7.1　Example of Display

第一部分：对于各种功能，该部分显示的参数相同——“电阻”大小，以“R=”引导。

第二部分：该部分分时显示两种参数，“量程”和“等效电路模式+测量频率”，交替时间为 1s。“量程”以“RNG”引导，自动量程时变为“ATO”，其后跟频率，分“100”，“120”，“1K”和“10K”四种。“等效电路模式+测量频率”以“Ser”（串联）或“Par”（并联）引导，其后跟量程，分“100”，“1K”，“10K”和“.1M”四种。

第三部分：依据功能的不同，显示内容也有所不同。以“X=”（电抗），“L=”

（电感），“C=”（电容）和“CP”（电解电容）引导，其后显示测量值。

第四部分：显示相对于功能的副参数。以“θ”（损耗角），“Q”（品质因数），“D”（损耗值）和“D”（损耗值）引导，一一与第三部分的功能相对应。

对于第一、三部分的示值，格式采用科学计数法的底数形式，保留 4 位有效数字，其后跟国际单位制的词头“p”（10^{-12}），“n”（10^{-9}），“μ”（10^{-6}），“m”（10^{-3}），“ ”（10^{0}），“K”（10^{3}），“M”（10^{6}）和“G”（10^{9}）共 8 种。第四部分，θ 和 Q 保留一位小数，D 保留四位小数。

7.2 测试连接说明

本机连接待测元件的方式比较全面，可以使用两线制，三线制和四线制三种方式。两线制方式适合中等阻值元件的连接；三线适宜在干扰比较大或小电容测量时减少杂散电容的影响；四线制方式适宜低值元件的测量，可以减小接线电阻的影响。

7.3 按键操作说明

本机含有 6 个按键，负责各种功能的切换，此外还用一个按钮开关用于开关电源。各个按键的具体功能分配见图 7.2。

键	Function	Range	Frequency	Mode	Auto-Range	Light
内	R=	100	100	Ser	Auto-On	On
	L=	1K	120	Par	Auto-Off	
容	C=	10K	1K	同时按下时进入调试模式		Off
	CP	100K	10K			

图 7.2　按键功能

Fig. 7.2　Function of the Keys

各个按键的功能采取轮流选中内容的方式进行循环，比如，每按一次 Function 键，则功能依次在电阻（“R=”）→电感（“L=”）→电容（“C=”）→电解电容（“CP”）间轮流切换。其他的按键操作方法与此类似。

为了便于制作与调试，本机内置了调试模式，可以通过特殊的按键组合进入该模式，即同时按下“Mode”和“Auto-Range”两个按键可启动该功能，此时显示四个测量值，即待测元件两端的电压和流过其中的电流在基准电阻上产生的电压的实部和虚部分量大小，此模式可以用于对模数转换部分及前端电路等进行调试。退出方法也是同时按下这两个按键，即返回正常模式。

8 样机测试

本部分对制作出来的样机进行各项测试，主要包括整机工作电流，各项功能的测试结果等。由于缺乏必备的高精度仪器，这里所进行的测试仅仅是粗略的。用于对比测试的仪器为深圳胜利 VC9208 3.5 位数字万用表。

8.1 工作电流

测试时所用的电源为一 SONY PlayStation2 交流电源适配器，标称输出电压为 8.5V，实测值为 8.88V。测试条件：量程 100，测试频率 100Hz，测试端开路。

表 8.1　工作电流

Fig. 8.1　Power Supply Current

开启液晶背光	关闭液晶背光
74.7mA	60.2mA

8.2 电阻测试

该项测试分别使用 100Hz 和 1KHz 的测试频率测量精度为 0.1%的标称值为 100Ω，1KΩ，10KΩ和 47KΩ的电阻，量程置于自动选择模式；为了比较，也列出了 VC9208 的对比测试结果，其中 VC9208 指标为±(0.8%+3)。

表 8.2　电阻测试

Fig. 8.2　Test of Resistors

	100Hz	1KHz	VC9208
100Ω	100.0Ω	102.0Ω	99.75Ω
1KΩ	1.000K	1.000K	999Ω
10KΩ	10.00K	10.01K	9.99K
47KΩ	47.30K	47.18K	47.0K

对于测量精度为 0.1%的电阻，无法将数字万用表的测试结果作为基准，因其精度要差于电阻的标称精度（如果电阻品质可靠的话），本数字电桥相比于万用表测量中等电阻（1K 和 10K）时有很好的精度。

8.3 电容测试

该项测试分别使用 100Hz 和 1KHz 的测试频率测量 1nF（±5%）有机薄膜电容、10nF（未标明精度）瓷介电容和 100nF（±10%）的有机薄膜电容，量程置于自动选择模式；为了比较，也列出了 VC9208 的对比测试结果，其中 VC9208 指标为±(2.5%+20)。

表 8.3　电容测试

Fig. 8.3　Test of Capacitors

	100Hz	1KHz	VC9208
1nF	904.7pF	874.2pF	0.90nF
10nF	12.47nF	12.08nF	13.90nF
100nF	99.42nF	99.28nF	0.101nF

测量电容时，暂且可以将数字万用表的结果作为基准，本数字电桥也有很好的精度。测 10nF 瓷介电容时偏差有点大，可能归因于瓷介电容的容值受所施加的电压影响较大，数字万用表未给出测试信号的具体信息，本数字电桥则使用 V_{RMS}= 0.6 V 的正弦信号。

该项测试使用 120Hz 的测试频率分别测量 22μF/25V（未标明精度）铝电解电容和 100μF/50V（未标明精度）铝电解电容，量程置于自动选择模式。为了比较，也列出了 VC9208 的对比测试结果，其中 VC9208 指标为±(2.5%+20)。

表 8.4　电解电容测试

Fig. 8.4　Test of Electrolytic Capacitors

	120Hz	VC9208
22μF/25V	20.25μF	22.6μF
100μF/50V	101.5μF	104.4μF

测电解电容时标准的做法是使用 120Hz 的正弦信号，同时施加直流偏置电压使正极电压大于负极，本数字电桥实现方法接近标准，数字万用表未给出具体情况。

8.4 电感测试

该项测试使用 1KHz 的测试频率测量 1mH（未标明精度）磁芯电感，量程置于自动选择模式；VC9208 无此测量功能，未能列出对比测试结果。

表 8.5　电感测试

Fig. 8.5　Test of Inductor

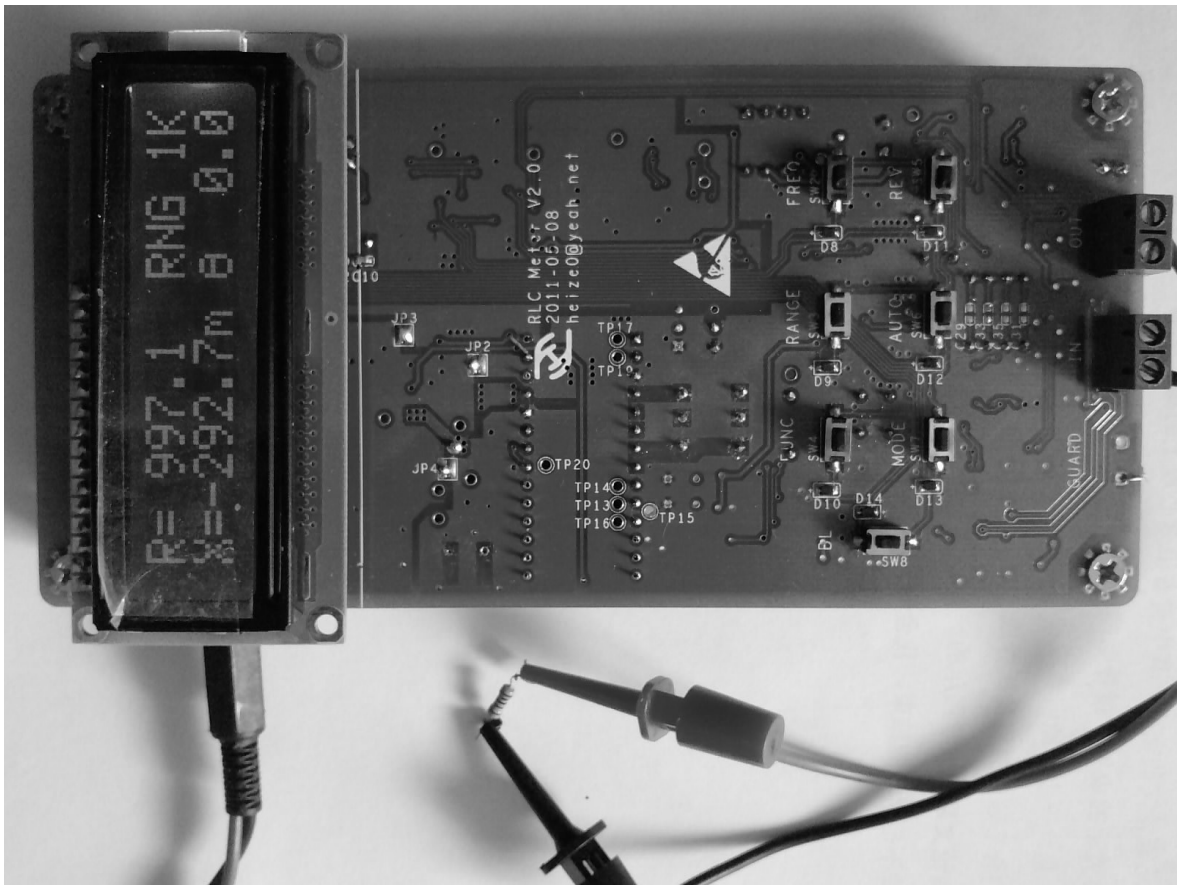
	1KHz
1mH	1.011mH

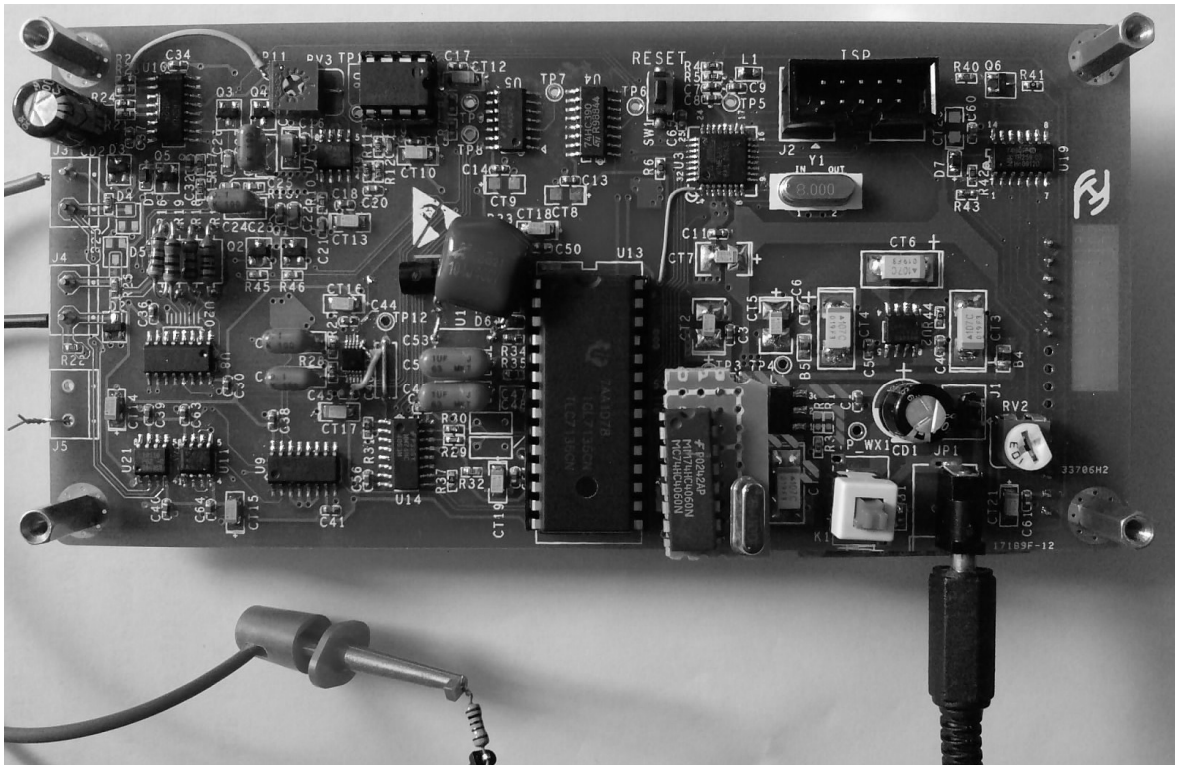
测电感时由于没有比较对象，暂且认为电感标称值正确，则本数字电桥结果接近标称值。

9 结论

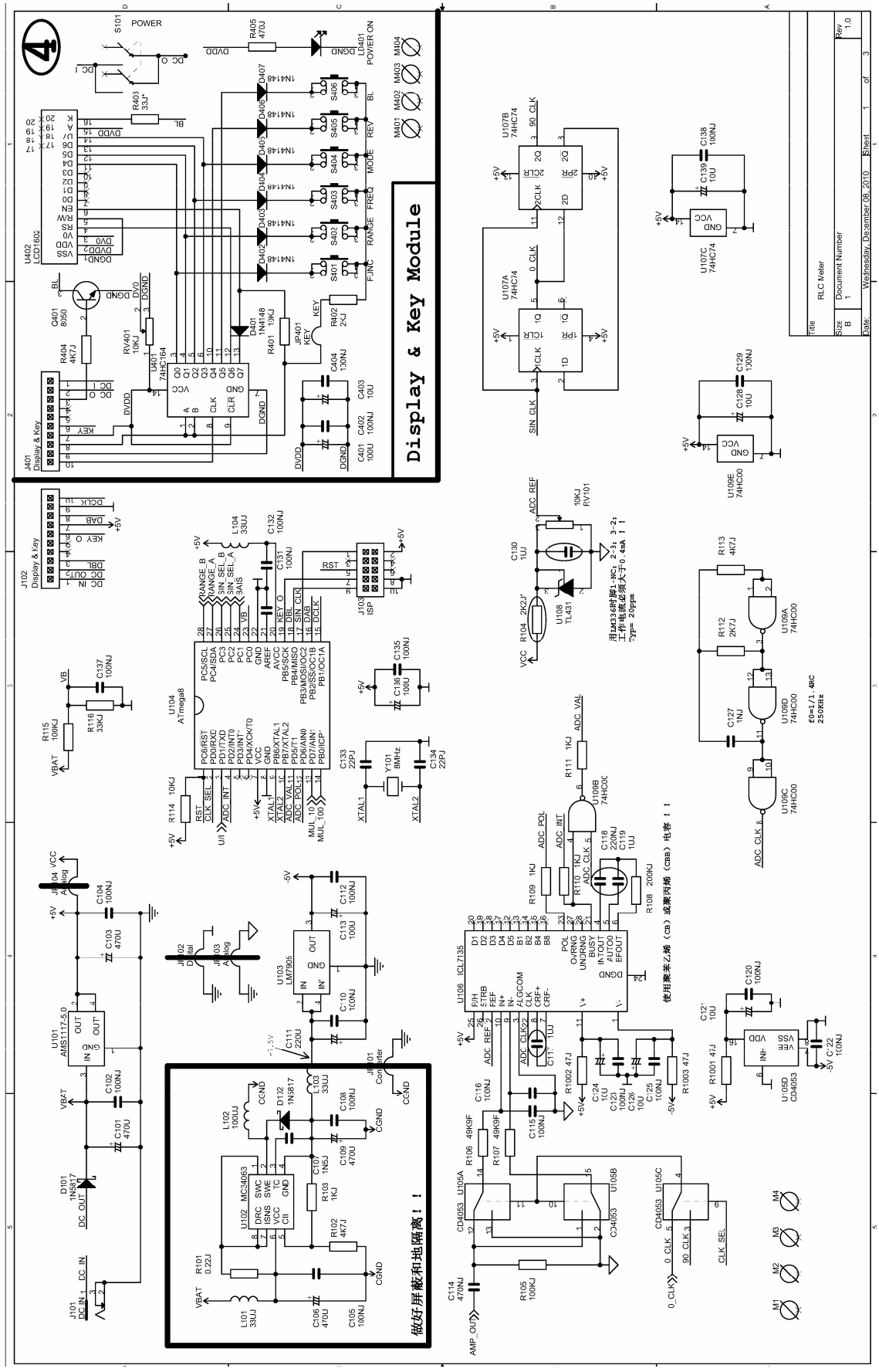
经过对最终制作出的一台样机的调试和测试，证明“自动平衡电桥”理论是实际可行并且富有成效的电子元件测量解决方案。由于是首次由现有的理论外加参考他人设计后独立自主确定电路形式和计算元件参数，设计的电路板又是模数混合的高精度电路，即使还有测量结果的稳定性没有预期的好等等缺陷，这对于我来说却是很有意义和成就感的。

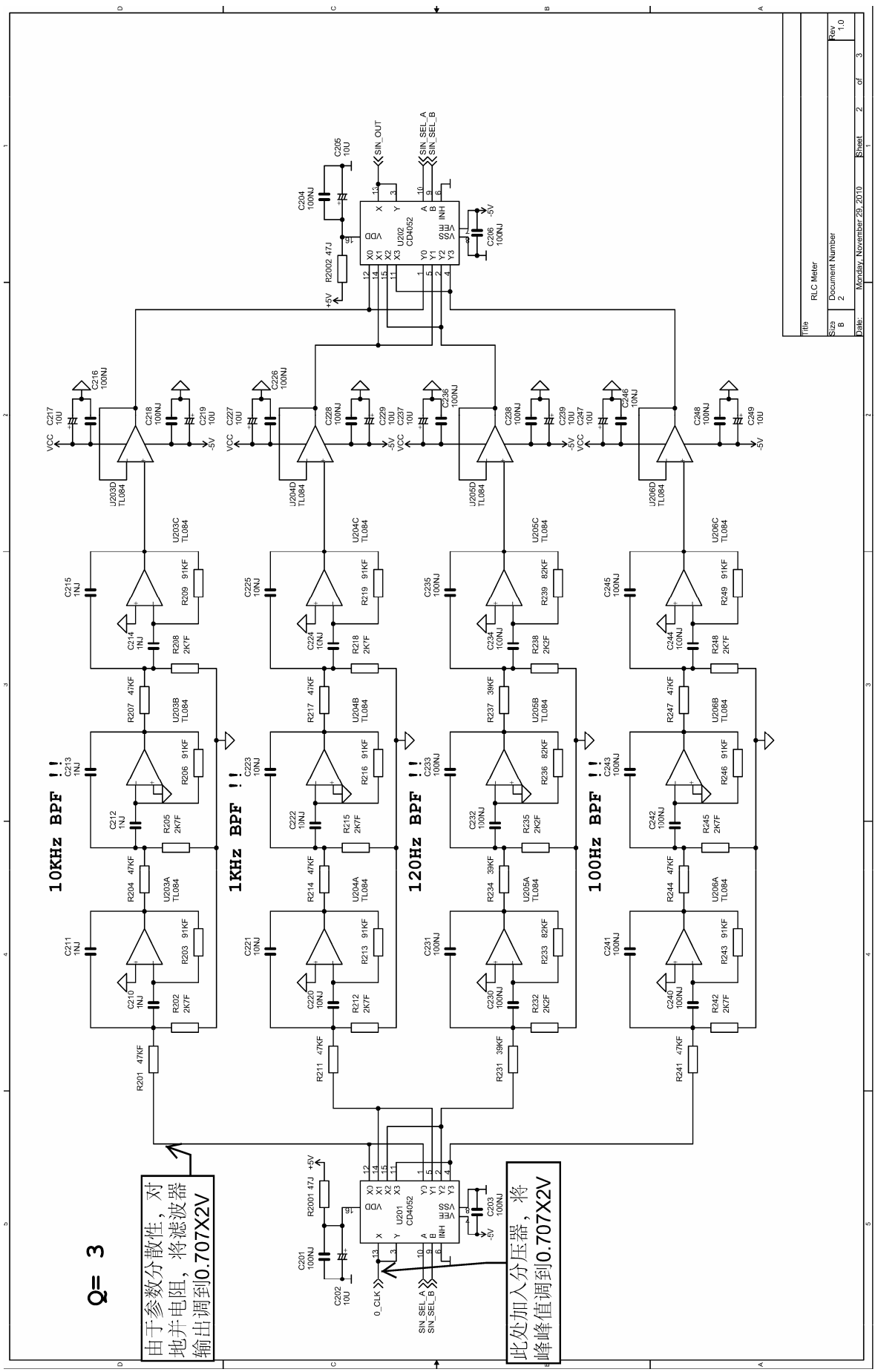
附录 A：实物照片

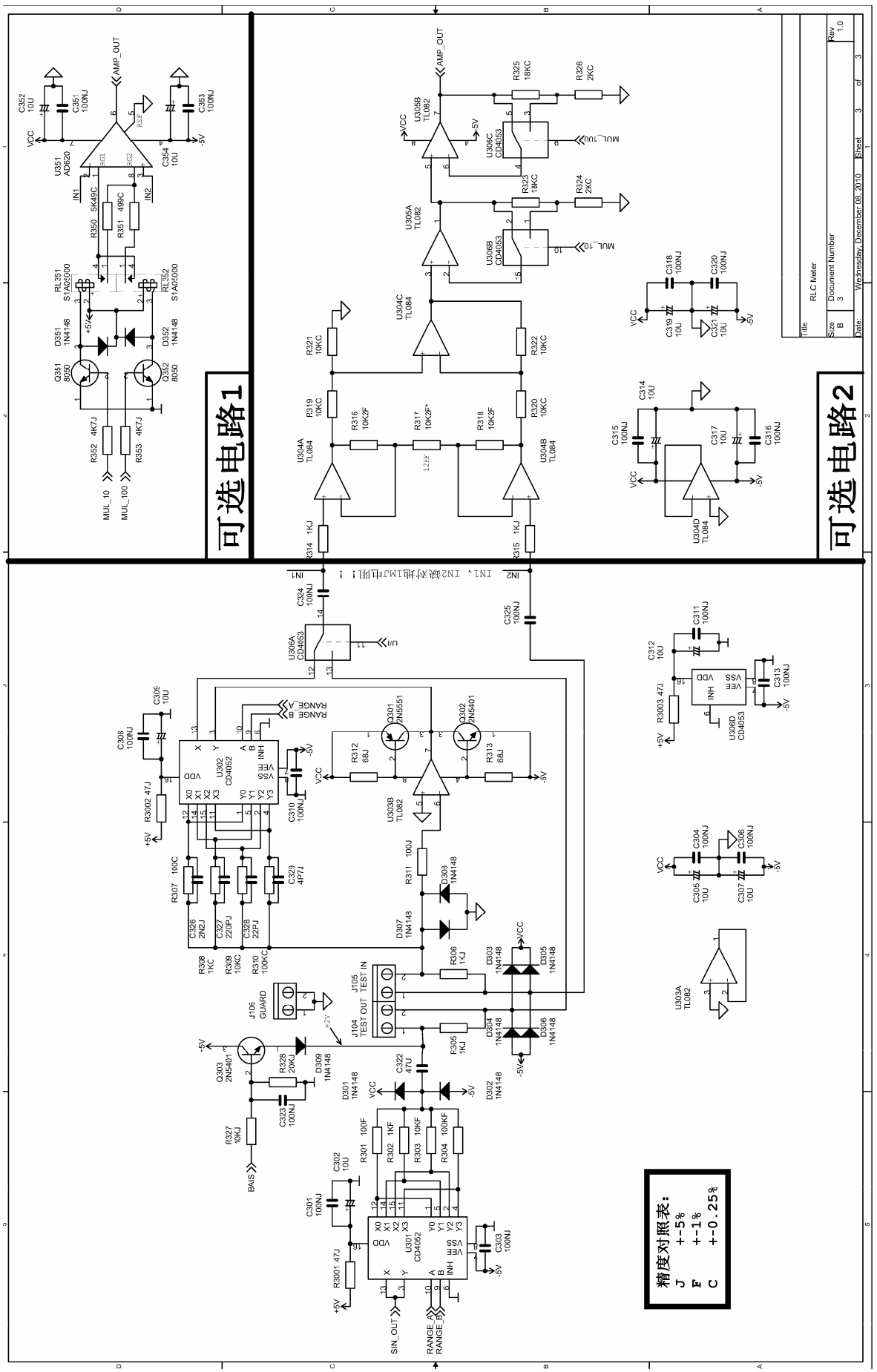

RLC Meter V2.00
2011-05-08
heize0@yeah.net
FREQ
REV
RANGE
AUTO
FUNC
MODE
GUARD
TP17
TP19
TP20
TP14
TP13
TP16
TP15
SW8


RESET
8.000

附录 B：电路图


Display & Key Module
POWER
LCD1602
74HC164
74HC74
ATmega8
ICL7135
CD4053
AMS1117-5.0
LM7905
MC34063
TL431
ISP
RST
ADC_INT
ADC_POL
ADC_VAL
ADC_CLK
DCLK
DAB
DBL
KEY_O
RANGE_A
RANGE_B
SIN_SEL_A
SIN_SEL_B
SIN_CLK
AMP_OUT
CLK_SEL
做好屏蔽和地隔离！！
使用聚苯乙烯（CB）或聚丙烯（CBB）电容！！
Title
RLC Meter
Document Number
Rev
Size
Date: Wednesday, December 08, 2010
Sheet


10KHz BPF !!
1KHz BPF !!
120Hz BPF !!
100Hz BPF !!
Q= 3
由于参数分散性，对地并电阻，将滤波器输出调到0.707X2V
此处加入分压器，将峰峰值调到0.707X2V
RLC Meter
Monday, November 29, 2010


可选电路1
可选电路2
RLC Meter
Document Number
Wednesday, December 08, 2010
精度对照表：
J +-5%
F +-1%
C +-0.25%

附录 C：源程序代码

1. 《main.c》

```
//*****************************************
//器件：        ATmega8(L)
//软件版本：    V1.0
//作者：        heize0
//邮箱：        heize0@yeah.net
//编译器：      IAR Embedded Workbench for Atmel AVR V5.10A
//时钟：        8.000MHz
//*****************************************

#include "..\inc\my.h"
#include "..\inc\hc164.h"
#include "..\inc\main.h"
#include "math.h"


U8          u8Tick= 0;              //时钟滴答
U8          u8ADCFlag= 0;
U8          u8ADC_OK= 0;            //模数转换完成标志


// Uv= a+jb;       Ui= c+jd;
S32         a= 0,b= 0,c= 0,d= 0;


//*****************************************
void main(void)
{
    U8 keyVal= KB_NONE;
    U8 t0Flag= 0;
    U8 dispFlag= 0;
    U8 blFlag= 1;                   //液晶背光标志
    U8 debugFlag= 0;
    U8 freqFlagOld= 1,freqFlagNew= 0;   // 测试频率标志
    U8 rangeFlagOld= 1,rangeFlagNew= 0; // 量程标志
    U8 funcFlagOld= 1,funcFlagNew= 0;   // 功能标志
    U8 modeFlagOld= 1,modeFlagNew= 0; // 模式标志
    U8 autoFlag= 0;         //自动量程标志
    U8 siFlag= 5;           // SI 单位制词头标志

    U16 F0= 0;              //测试频率
    U32 R0= 0;              //量程电阻
    double RxOld,XxOld,RxNew,XxNew,tt;

    //*************************************
    //整机初始化
    M8Init();
    PortInit();
    HC164Init();

    TCCR0= 0X05;  // 1024 分频
    TCNT0= 0XB2; //定时 10ms
    TIMSK |= 0X01;//允许 T0 中断

    MCUCR |= 0X02;      // INT0 下降沿触发
    GICR |= 0X40;       //允许 INT0 中断

    TCCR1B= 0X07;       // T1 外部时钟,下降沿驱动

    OCR2= OR2_100;
    TCCR2= TR2_100;     // T2 产生测试频率

    //ADMUX= 0XE0;      //内部 2.56V 基准,左对齐,通道 ADC0
    //ADCSRA= 0XCE;     // 64 分频,125KHz,中断使能,单次转换

    IE_ON;              //开中断

    //*************************************
    //启动画面
    LCDXYStr(0,0,welcome);
    LCDXYStr(0,1,name);
    DelayMs(2000);
    LCDXYStr(0,0,author);
    LCDXYStr(0,1,email);
    DelayMs(2000);
    LCDXYStr(0,0,blank);
    LCDXYStr(0,1,blank);

    IE_OFF;
    WDTCR |= 0X18;
    WDTCR |= 0X1E;      //开启看门狗,定时 1S
    IE_ON;

    //*************************************
    //主程序开始
    while(1)
    {
        if(u8Tick == 1)
        {
            u8Tick= 0;
            t0Flag= !t0Flag;
            if(++dispFlag >= 200)
                dispFlag= 0;

            keyVal= KeyBoard();
        }

        if(t0Flag)
            T10MS_H;
        else
            T10MS_L;


    //*************************************
        switch(keyVal)          //按键处理
        {
        case KB_FUNC:
            if(++funcFlagNew >= 4)
                funcFlagNew= 0;

            if(funcFlagNew == 3)  //测电解电容时自动开启直流偏置
                BAIS_ON;
```

```
                else
                        BAIS_OFF;
                break;
        case KB_RANGE:
                if(++rangeFlagNew >= 4)
                        rangeFlagNew= 0;
                break;
        case KB_FREQ:
                if(++freqFlagNew >= 4)
                        freqFlagNew= 0;
                break;
        case KB_MODE:
                if(++modeFlagNew >= 2)
                        modeFlagNew= 0;
                break;
        case KB_:                       //保留按键
                if(++autoFlag >= 2)
                        autoFlag= 0;
                break;
        case KB_DEBUG:
                if(++debugFlag >= 2)
                        debugFlag= 0;
                break;
        case KB_BL:
                if(blFlag == 1)
                {
                        BL_OFF;
                        blFlag= 0;
                }
                else
                {
                        BL_ON;
                        blFlag= 1;
                }
                break;
        default:
                keyVal= KB_NONE;
                break;
        }

        keyVal= KB_NONE;  //清除每隔 10ms 扫描键盘一次的延迟


    //***********************************
        if(freqFlagOld != freqFlagNew)   // 频 率切换
        {
                freqFlagOld= freqFlagNew;

    //LCDXYStr(10,0,mode[modeFlagOld]);
    //显示 Ser 或 Par

    //LCDXYStr(13,0,freq[freqFlagOld]);
                switch(freqFlagOld)
                {
                case 0:
                        OCR2= OR2_100;
                        TCCR2= TR2_100;
                        SIN_100;
                        F0= 100;
                        break;
                case 1:
                        OCR2= OR2_120;
                        TCCR2= TR2_120;
                        SIN_120;
                        F0= 120;
                        break;
                case 2:
                        OCR2= OR2_1K;
                        TCCR2= TR2_1K;
                        SIN_1K;
                        F0= 1000;
                        break;
                case 3:
                        OCR2= OR2_10K;
                        TCCR2= TR2_10K;
                        SIN_10K;
                        F0= 10000;
                        break;
                default:
                        //freqFlagOld=0;
                        break;
                }

                //u8ADCFlag= 0;
        }


    //+++++++++++++++++++++++++++++++++
        if(rangeFlagOld != rangeFlagNew)        //量程切换
        {
                rangeFlagOld= rangeFlagNew;
                //LCDXYStr(10,0,rng[autoFlag]);
                //显示 RNG

    //LCDXYStr(13,0,range[rangeFlagOld]);
                switch(rangeFlagOld)
                {
                case 0:
                        RANGE_100;
                        R0= 100;
                        break;
                case 1:
                        RANGE_1K;
                        R0= 900;
                        break;
                case 2:
                        RANGE_10K;
                        R0= 9000;
                        break;
                case 3:
                        RANGE_100K;
                        R0= 90000;
                        break;
                default:
                        //rangeFlagOld= 0;
                        break;
                }

                //u8ADCFlag= 0;
        }
        //---------------------------------------

    //+++++++++++++++++++++++++++++++++
        if(u8ADC_OK == 1)          //处理结果
        {
                u8ADC_OK= 0;

                if(debugFlag == 0)
                {
                        if(modeFlagNew == 0)// 串 联
```

```
等效电路
                    {
                        RxNew=
(double)R0*((double)a*c+(double)b*d)/
    ((double)c*c+(double)d*d);
                        XxNew=
(double)R0*((double)b*c-(double)a*d)/
    ((double)c*c+(double)d*d);

                    }
                    else
                    {
                        RxNew=
(double)R0*((double)a*a+(double)b*b)/
    ((double)a*c+(double)b*d);
                        XxNew=
(double)R0*((double)a*a+(double)b*b)/
    ((double)b*c-(double)a*d);
                    }

                    RxNew= fabs(RxNew);
                    RxOld= (RxOld+RxNew)/2;
                    XxOld= (XxOld+XxNew)/2;

                    LCDCmd(0x04);           //
以下反序显示

                    //Rx= 345678000.0;
                    //Xx= 0.000000345678;
                    tt= RxOld+0.5;
                    Converter(&tt,&siFlag);

    LCDXY(8,0),LCDDat(si[siFlag]);
                    Show((S32)tt);          //
显示阻值

                    switch(funcFlagNew)
                    {
                    case 0:
                        if((autoFlag == 1) &&
(u8ADCFlag == 6))
                            rangeFlagNew=
AutoRange((S32)RxOld);
                        tt= XxOld;
                        Converter(&tt,&siFlag);
                        if(tt > 0)
    //四舍五入
                            tt += 0.5;
                        else
                            tt -= 0.5;

    LCDXY(8,1),LCDDat(si[siFlag]);
                        Show((S32)(tt));

    //LCDXY(15,1),ShowVal(0,1,7);
                        tt= XxOld/RxOld;
                        if(modeFlagNew == 1)//
并联等效电路
                            tt= 1/tt;
                        tt= 1800/PI*atan(tt);
                        if(tt > 0)
                            tt += 0.5;
                        else
                            tt -= 0.5;

    LCDXY(15,1),ShowVal((S32)tt,1,5);
                        break;
                    case 1:
                        if((autoFlag == 1) &&
(u8ADCFlag == 6))
                            rangeFlagNew=
AutoRange((S32)XxOld);
                        tt= XxOld/(2*PI*F0);
                        Converter(&tt,&siFlag);

    LCDXY(8,1),LCDDat(si[siFlag]);
                        //tt= fabs(tt);
                        if(tt > 0)
                            tt += 0.5;
                        else
                            tt -= 0.5;
                        Show((S32)tt);
                        tt= XxOld/RxOld;
                        if(modeFlagNew == 1)//
并联等效电路
                            tt= 1/tt;
                        tt= fabs(tt);

    LCDXY(15,1),ShowVal((S32)(10*tt+0.5),1,5);
                        break;
                    case 2:          //电解电容和
无极性电容的计算方法相同
                    case 3:
                        if((autoFlag == 1) &&
(u8ADCFlag == 6))
                            rangeFlagNew=
AutoRange((S32)XxOld);
                        tt= -1/(2*PI*F0*XxOld);
                        Converter(&tt,&siFlag);

    LCDXY(8,1),LCDDat(si[siFlag]);
                        if(tt > 0)
                            tt += 0.5;
                        else
                            tt -= 0.5;
                        Show((S32)tt);
                        tt= RxOld/XxOld;
                        if(modeFlagNew == 1)//
并联等效电路
                            tt= 1/tt;
                        tt= fabs(tt);

    LCDXY(15,1),ShowVal((S32)(10000*tt+0.5),4,5
);
                        break;
                    default:
                        //funcFlagNew= 0;
                        break;
                    }

                    LCDCmd(0x06);       // 恢 复
正序显示

                    RxOld= RxNew;
                    XxOld= XxNew;
                }//end if(debugFlag == 0)
                else
                {
                    LCDCmd(0x04);       // 以 下
```

```
反序显示
                    LCDXY(5,0);
                    ShowVal(a,0,6);
                    LCDXY(11,0);
                    ShowVal(b,0,6);
                    LCDXY(5,1);
                    ShowVal(c,0,6);
                    LCDXY(11,1);
                    ShowVal(d,0,6);
                    LCDCmd(0x06);       // 恢 复
正序显示
                }

                if(++u8ADCFlag >= 8)
                    u8ADCFlag= 0;

            }
            //---------------------------------------


    //*************************************
            //模式或功能更改时刷新提示符
            if((funcFlagOld != funcFlagNew) ||
(modeFlagOld != modeFlagNew))
            {
                modeFlagOld=
modeFlagNew,funcFlagOld= funcFlagNew;

    LCDXYStr(0,0,disp[modeFlagOld][8]);       //
显示 R 或 R

    //LCDXYStr(10,0,mode[modeFlagOld]);
    //显示 Ser 或 Par

    LCDXYStr(0,1,disp[modeFlagOld][funcFlagOld
*2]);

    LCDXYStr(9,1,disp[modeFlagOld][funcFlagOld
*2+1]);
            }

            if(dispFlag <= 100)
            {
                if(debugFlag == 0)

    LCDXYStr(9,0,mode[modeFlagOld]);          //
显示 Ser 或 Par
                else
                {

    LCDXYStr(12,1,mode[modeFlagOld]);
                    LCDXY(12,0),LCDDat(' ');
                }


    LCDXYStr(13,0,freq[freqFlagOld]);
            }
            else
            {
                if(debugFlag == 0)
                    LCDXYStr(9,0,rng[autoFlag]);
                //显示 RNG
                else

    LCDXYStr(12,1,rng[autoFlag]);

    LCDXYStr(13,0,range[rangeFlagOld]);
            }
            //---------------------------------------


            WDT_R;

        } //end while(1)
} //end main()

//*****************************************
void PortInit(void)
{
    CLK_SEL_OUT,CLK_SEL_0;    //
    UI_OUT,UI_U;              //
    ADC_INT_IN,ADC_INT_L;
    T10MS_OUT,T10MS_L;
    INT1_OUT,INT1_L;
    ADC_VAL_IN,ADC_VAL_L;
    ADC_POL_IN,ADC_POL_L;
    MUL_10_OUT,MUL_100_OUT;
    MUL_1;                    //放大 1 倍
    SIN_CLK_OUT;              //
    BL_OUT,BL_ON;             //开启背光
    VB_IN,VB_L;               //
    BAIS_OUT,BAIS_OFF;        //无直流偏置
    SIN_SEL_A_OUT,SIN_SEL_B_OUT;
    SIN_100;                  //测试频率 100Hz
    RANGE_A_OUT,RANGE_B_OUT;
    RANGE_100;                //量程 100 档
}

//*****************************************
void Converter(double *dat,U8 *siFlag)
//数据格式处理
{
    (*siFlag)= 6;
    if(fabs(*dat) > 1000)
        for(;fabs(*dat) > 1000000;(*siFlag)++)
        {
            (*dat) /= 1000;
            if(*siFlag == 0)  //防止死循环
                break;
        }
    else
        for(;fabs(*dat) < 1000;(*siFlag)--)
        {
            (*dat) *= 1000;
            if(*siFlag == 9)
                break;
        }
}

//*****************************************
void ShowVal(S32 val,U8 pt,U8 cnt)
//显示函数,将整数 val 以 pt 位小数形式显示,占 cnt
位
{
    U8 i,nFlag= 0;//,num;

    if(val < 0)
    {
        nFlag= 1;
        val= -val;
    }
```

```
    //LCDCmd(0x04);      //反序显示

    for(i=1;i <= cnt;i++)
    {
        //num='0' + (S8)(val%10);
        LCDDat('0' + (U8)(val%10));
        val /= 10;
        if(i == pt)
        {
        LCDDat('.');
            i++;
        }
        if((val == 0) && (i>(pt+1)))
        {
            i++;
            break;      //剔除前导零
        }
    }

    if((nFlag==1) && (i<=cnt))
    {
        LCDDat('-');
        i++;
    }
    for(;i <= cnt;i++)
        LCDDat(' ');

    //LCDCmd(0x06);      //恢复正序显示
}

//*****************************************
void Show(S32 val)
//取 4 位有效数字
{
    if(FABS(val)<10000)
        ShowVal(val,3,6);
    else
        if(FABS(val)<100000)
        {
            if(val >= 0)
                val += 5;
            else
                val -= 5;
            ShowVal(val/10,2,6);
        }
        else
        {
            if(val >= 0)
                val += 50;
            else
                val -= 50;
            ShowVal(val/100,1,6);
        }
}

//*****************************************
U8 AutoRange(S32 val)
{
    U8 temp;
    val= FABS(val);
    if(val <= 500)
        temp= 0;
    else
    {
        if(val <= 5000)
            temp= 1;
        else
        {
            if(val <= 50000)
                temp= 2;
            else
                temp= 3;
        }
    }

    return(temp);
}
//*****************************************


//*****************************************
#pragma vector= TIMER0_OVF_vect
__interrupt void T0Isr(void)
{
    TCNT0= 0XB2; //定时 10ms
    u8Tick= 1;
}

//*****************************************
#pragma vector= INT0_vect
__interrupt void INT0Isr(void)
{
    U8 temp;
    S16 adcVal;
    temp= TCNT1L,adcVal= TCNT1H;
    TCNT1H= 0,TCNT1L= 0;
    adcVal= (adcVal<<8)-10001+temp;
    if(adcVal < 0)
        adcVal= 0;

    if( !ADC_POL_VAL ) //结果为负
        adcVal= -adcVal;

    //++++++++++++++++++++++++++++++++++
    // Uv= a+jb;      Ui= c+jd;
    switch(u8ADCFlag)   //读取转换结果
    {
    case 0:
        a= -adcVal;
        CLK_SEL_90;   //下次转换电压虚部
        //u8ADCFlag++;
        u8ADC_OK= 1;
        break;
    case 1:
        u8ADCFlag++;  //延迟等待稳定
        break;
    case 2:
        b= adcVal;
        UI_I,CLK_SEL_0;//下次转换电流实部
        //u8ADCFlag++;
        u8ADC_OK= 1;
        break;
    case 3:
        u8ADCFlag++;  //延迟等待稳定
        break;
    case 4:
        c= adcVal;
        CLK_SEL_90;   //下次转换电流虚部
        //u8ADCFlag++;
        u8ADC_OK= 1;
        break;
    case 5:
        u8ADCFlag++;  //延迟等待稳定
```

```
            break;
        case 6:
            d= -adcVal;
            UI_U,CLK_SEL_0;
            //u8ADCFlag++;
            u8ADC_OK= 1; //一次测量结束
            break;
        case 7:
            u8ADCFlag++;  //延迟等待稳定
            break;
        default:
            UI_U,CLK_SEL_0;
            u8ADCFlag= 0;
        }//end switch
        //-----------------------------------

}

//****************************************
```

2. 《main.h》

```
//****************************************
//器件：        ATmega8(L)
//软件版本：    V1.0
//作者：        heize0
//邮箱：        heize0@yeah.net
//编译器：      IAR Embedded Workbench for Atmel
AVR V5.10A
//时钟：        8.000MHz
//****************************************


#ifndef __MAIN_H__
#define __MAIN_H__


#define CLK_SEL         D,0
#define UI              D,1
#define ADC_INT         D,2
#define INT1            D,3
#define T10MS           D,4
#define ADC_VAL         D,5
#define ADC_POL         D,6
#define MUL_10P         D,7

//****************************************
#define MUL_100P        B,0
#define SIN_CLK         B,3
#define DBL             B,4
#define KEY             B,5

//****************************************
#define VB              C,0
#define BAIS            C,1
#define SIN_SEL_A       C,2
#define SIN_SEL_B       C,3
#define RANGE_A         C,4
#define RANGE_B         C,5

//****************************************

#define CLK_SEL_OUT     DDR_OUT(CLK_SEL)
#define CLK_SEL_0   PORT_L(CLK_SEL)
#define CLK_SEL_90 PORT_H(CLK_SEL)

#define UI_OUT      DDR_OUT(UI)
#define UI_I        PORT_L(UI)
#define UI_U        PORT_H(UI)

#define ADC_INT_IN DDR_IN(ADC_INT)
#define ADC_INT_L  PORT_L(ADC_INT)

#define INT1_OUT    DDR_OUT(INT1)
#define INT1_H      PORT_L(INT1)
#define INT1_L      PORT_H(INT1)

#define T10MS_OUT  DDR_OUT(T10MS)
#define T10MS_H         PORT_L(T10MS)
#define T10MS_L         PORT_H(T10MS)

#define ADC_VAL_INDDR_IN(ADC_VAL)
#define ADC_VAL_L  PORT_L(ADC_VAL)

#define ADC_POL_INDDR_IN(ADC_POL)
#define ADC_POL_L  PORT_L(ADC_POL)
#define ADC_POL_VAL     PIN_VAL(ADC_POL)

//****************************************
#define MUL_10_OUT      DDR_OUT(MUL_10P)
#define MUL_100_OUT     DDR_OUT(MUL_100P)

#define MUL_1       do{\

    PORT_L(MUL_10P);\
    PORT_L(MUL_100P);\
                    }while(0)
#define MUL_10      do{\

    PORT_H(MUL_10P);\
    PORT_L(MUL_100P);\
                    }while(0)
#define MUL_100         do{\

    PORT_H(MUL_10P);\
    PORT_H(MUL_100P);\
                    }while(0)

//****************************************
#define SIN_CLK_OUT     DDR_OUT(SIN_CLK)
#define SIN_CLK_H  PORT_H(SIN_CLK)

#define BL_OUT          DDR_OUT(DBL)
#define BL_ON       PORT_H(DBL)
#define BL_OFF      PORT_L(DBL)

#define KEY_IN      DDR_IN(KEY)
#define KEY_H       PORT_H(KEY)
#define KEY_VAL         PIN_VAL(KEY)

#define VB_IN       DDR_IN(VB)
#define VB_L        PORT_L(VB)

#define BAIS_OUT    DDR_OUT(BAIS)
#define BAIS_ON         PORT_H(BAIS)
#define BAIS_OFF    PORT_L(BAIS)

//****************************************
#define SIN_SEL_A_OUT  DDR_OUT(SIN_SEL_A)
#define SIN_SEL_B_OUT  DDR_OUT(SIN_SEL_B)

#define SIN_10K         do{\
```

```
    PORT_L(SIN_SEL_A);\
    PORT_L(SIN_SEL_B);\
                    }while(0)
#define SIN_1K      do{\

    PORT_H(SIN_SEL_A);\
    PORT_L(SIN_SEL_B);\
                    }while(0)
#define SIN_120     do{\

    PORT_L(SIN_SEL_A);\
    PORT_H(SIN_SEL_B);\
                    }while(0)
#define SIN_100     do{\

    PORT_H(SIN_SEL_A);\
    PORT_H(SIN_SEL_B);\
                    }while(0)

//*****************************************
#define RANGE_A_OUT
    DDR_OUT(RANGE_A)
#define RANGE_B_OUT
    DDR_OUT(RANGE_B)

#define RANGE_100  do{\

    PORT_L(RANGE_A);\
    PORT_L(RANGE_B);\
                    }while(0)
#define RANGE_1K        do{\

    PORT_H(RANGE_A);\
    PORT_L(RANGE_B);\
                    }while(0)
#define RANGE_10K       do{\

    PORT_L(RANGE_A);\
    PORT_H(RANGE_B);\
                    }while(0)
#define RANGE_100K          do{\

    PORT_H(RANGE_A);\
    PORT_H(RANGE_B);\
                    }while(0)

//*****************************************

#define TR2_100     0X1E//实际频率 100.16Hz
#define OR2_100     0X26
#define TR2_120     0X1C//实际频率 120.19Hz
#define OR2_120     0X80
#define TR2_1K      0X1A//实际频率 1KHz
#define OR2_1K      0X7C
#define TR2_10K     0X19//实际频率 10KHz
#define OR2_10K     0X63

#define PI          (double)__PI
//#define CHECK_0(x)    (((x)<=1e-15) ? 1e-15:x)
#define FABS(X)         (((X)>=0) ? X:(-(X)))


U8 flash welcome[]= "Welcome !!!";
U8 flash name[]= "RLC Meter";
U8 flash author[]= "Liao YangPing";
U8 flash email[]= "heize0@yeah.net";
U8 flash blank[]= "                ";


U8 flash si[]= {'f','p','n',0xe4,'m',' ','K','M','G','T'};
U8 flash freq[][4]= {"100","120"," 1K","10K"};
U8 flash disp[][9][3]= {"X=",{' ',0xf2,0x00},"L="," Q","C="," D","CP"," D","R=","X=",{'
',0xf2,0x00},"L="," Q","C="," D","CP"," D","R="};
U8 flash mode[][5]= {" Ser"," Par"};
U8 flash range[][4]= {"100"," 1K","10K",".1M"};
U8 flash rng[][5]= {" RNG"," ATO"};

//*****************************************
void main(void);

//*****************************************
void PortInit(void);

//*****************************************
void Converter(double *dat,U8 *siFlag);
//数据格式处理

//*****************************************
void ShowVal(S32 val,U8 pt,U8 cnt);
//显示函数,将整数 val 以 pt 位小数形式显示,占 cnt
位

//*****************************************
void Show(S32 val);

//*****************************************
U8 AutoRange(S32 val);

//*****************************************
****
#pragma vector= TIMER0_OVF_vect
__interrupt void T0Isr(void);

//*****************************************
#pragma vector= INT0_vect
__interrupt void INT0Isr(void);

//*****************************************

#endif
```

3.《hc164.c》

```
//*****************************************
//器件：        ATmega8(L)
//软件版本：    V1.0
//作者：        heize0
//邮箱：        heize0@yeah.net
//编译器：      IAR Embedded Workbench for Atmel
AVR V5.10A
//时钟：        8.000MHz
//*****************************************


#include "..\inc\my.h"
#include "..\inc\hc164.h"


// DAB out,L;    DCLK out,L;
void HC164Clr(void);
```

```
void HC164Dat(U8 dat);
U8 HC164Key(void);
void LCDCheckBusy(void);

//*****************************************
void HC164Clr(void)
//清零
{
	U8 cnt;
	//DAB_OUT,DAB_L;
	for(cnt=0;cnt<=7;cnt++)
	{
		DCLK_H;
		//NOP;
		DCLK_L;
	}
}

//*****************************************
void HC164Dat(U8 dat)
//输出数据
{
	U8 cnt;
	HC164Clr();
	dat= (dat&0x4f)|0x80;
	for(cnt=0;cnt<=7;cnt++)
	{
		if(dat&0x80)
			DAB_H;
		else
			DAB_L;
		DCLK_H;
		//NOP;
		DCLK_L;
		dat <<= 1;
	}

	DAB_H;
	NOP;
	DAB_L;		// EN 产生下降沿
}

//*****************************************
U8 HC164Key(void)
//读入数据
{
	U8 cnt,dat=0;
	HC164Clr();
	DAB_H;		//第一个为高电平
	DCLK_H;
	//NOP;
	DCLK_L;		// HC164 左移一位
	DAB_L;

	for(cnt=0;cnt<=5;cnt++)
	{
		DAB_IN;//,DAB_H;
		NOP;//VERY IMPORTANT!!!
		NOP;//VERY IMPORTANT!!!
		if(DAB_VAL)
			dat |= 0x40;
		dat >>= 1;
		/*DAB_L,*/DAB_OUT;
		DCLK_H;
		//NOP;
		DCLK_L;
	}

	return dat;
}

//*****************************************
void HC164Init(void)
// HC164  初始化
{
	DAB_OUT,DAB_L;
	DCLK_OUT,DCLK_L;

	DelayMs(15);	//15
	HC164Dat(0x30>>4);
	DelayMs(5);
	HC164Dat(0x30>>4);
	DelayUs(100);
	HC164Dat(0x30>>4);
	HC164Dat(0x20>>4);

	LCDCmd(0x28);
	LCDCmd(0x08);
	 LCDCmd(0x01);
	LCDCmd(0x06);


	LCDCmd(0x0c);
}

//*****************************************
U8 KeyBoard(void)
//键盘扫描
{
	static U8 step= 0,dat= 0;
	U8 temp;
	switch(step)
	{
	case 0:
		dat= HC164Key();
		step++;
		break;
	case 1:
		temp= HC164Key();
		if(dat == temp)
		{
			step++;
			return dat;
		}
		else
			step--;
		break;
	case 2:
		temp= HC164Key();
		if(dat != temp)
			step= 0;
		break;
	default:
		step= 0;
		break;
	}

	return KB_NONE;
}

//*****************************************v
oid LCDCheckBusy(void)
{
```

```
    DelayUs(1200);  //750
}

//*****************************************
void LCDCmd(U8 cmd)
//写指令
{
    LCDCheckBusy();
    HC164Dat(cmd>>4);
    HC164Dat(cmd&0x0f);
}

//*****************************************
void LCDDat(U8 dat)
//写数据
{
    LCDCheckBusy();
   HC164Dat((dat>>4)|0x40);
    HC164Dat((dat&0x0f)|0x40);
}

//*****************************************
void LCDXYStr(U8 x,U8 y,U8 flash *str)
//在  x(0~15)列,y(0~1)行处写一字符串
{
   if(y == 1)
        x += 0x40;
   LCDCmd(x+0x80);
   while(*str != '\0')
        LCDDat(*(str++));
}

//*****************************************
void LCDXY(U8 x,U8 y)
//光标定位在  x(0~15)列,y(0~1)行
{
   if(y == 1)
        x += 0x40;
   LCDCmd(x+0x80);
}

//*****************************************
```

4. 《hc164.h》

```
//*****************************************
//器件：        ATmega8(L)
//软件版本：    V1.0
//作者：        heize0
//邮箱：        heize0@yeah.net
//编译器：      IAR Embedded Workbench for Atmel
AVR V5.10A
//时钟：        8.000MHz
//*****************************************


#ifndef __HC164_H__
#define __HC164_H__


#define DAB             B,2

#define DCLK            B,1

//*****************************************

#define KB_NONE         0X00
#define KB_FUNC         0X01
#define KB_RANGE        0X02
#define KB_FREQ         0X04
#define KB_MODE         0X08
#define KB_             0X10
#define KB_DEBUG        0X18//同时按下  MODE
和保留键
#define KB_BL           0X20

#define DAB_OUT     DDR_OUT(DAB)
#define DAB_H       PORT_H(DAB)
#define DAB_L       PORT_L(DAB)
#define DAB_IN      DDR_IN(DAB)
#define DAB_VAL     PIN_VAL(DAB)

#define DCLK_OUT    DDR_OUT(DCLK)
#define DCLK_H      PORT_H(DCLK)
#define DCLK_L      PORT_L(DCLK)

//*****************************************
void HC164Init(void);
// HC164  初始化

//*****************************************
U8 KeyBoard(void);
//键盘扫描

//*****************************************
void LCDCmd(U8 cmd);
//写指令

//*****************************************
void LCDDat(U8 dat);
//写数据

//*****************************************
void LCDXYStr(U8 x,U8 y,U8 flash *str);
//在  x(0~15)列,y(0~1)行处写一字符串

//*****************************************
void LCDXY(U8 x,U8 y);
//光标定位在  x(0~15)列,y(0~1)行

//*****************************************


#endif
```

5. 《my.c》

```
//*****************************************
//器件：        ATmega8(L)
//软件版本：    V1.0
//作者：        heize0
//邮箱：        heize0@yeah.net
//编译器：      IAR Embedded Workbench for Atmel
AVR V5.10A
//时钟：        8.000MHz
//*****************************************


#include "..\inc\my.h"
```

```
//*****************************************
// 七段数码管的字型码,从 0 到 9 和 无显示
U8 flash u8LED7SEG[]=
{0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f
,0x00};

// 七段数码管的段选码,从 低位 到 高位（右-->左）
U8 flash u8LED7BIT[]=
{0x7f,0xbf,0xdf,0xef,0xf7,0xfb,0xfd,0xfe};

//*****************************************
void DelayUs(U16 us)
{
    U16 i;
    us= us*5/4;              // 5/4 是在 8MHz 晶振下
    for( i=0;i<us;i++)
        NOP;
}

//*****************************************
void DelayMs(U16 ms)
{
    U16 i,j;
    for( i=0;i<ms;i++)
        for(j=0;j<1141;j++)
            NOP;//1141 是在 8MHz 晶振下
}

//*****************************************
void M8Init(void)      //ATMEGA16 初始化
{
    #ifndef __DEBUG__
   //SETB(MCUCSR,7);
   //SETB(MCUCSR,7);  // JTAG 引脚作 I/O 口
用
    #endif

   IE_OFF;                  //关中断
}
```

6. 《my.h》

```
//*****************************************
//器件：        ATmega16(L)
//软件版本：    V1.0
//作者：        heize0
//邮箱：        heize0@yeah.net
//编译器：      IAR Embedded Workbench for Atmel
AVR V5.10A
//时钟：        8.000MHz
//*****************************************


//条件编译的预处理指令,用于防止本头部文件被其
他文件重复引用
#ifndef __MYSELF_H__
#define __MYSELF_H__


#include <iom8.h>
#include <intrinsics.h>
#include <comp_a90.h>

typedef unsigned char  U8;
typedef unsigned int   U16;
typedef unsigned long  U32;

typedef signed char    S8;
typedef signed int     S16;
typedef signed long    S32;


#define SETB(y,x)     ((y) |= 1<<(x))        // y 第
x 位置位
#define CLRB(y,x)     ((y) &= ~(1<<(x)))     // y 第
x 位清零
#define XORB(y,x)     ((y) ^= 1<<(x))        // y 第
x 位取反
#define XNORB(y,x)    ((y) ^= ~(1<<(x)))     // y 除
了第 x 位外都取反
#define TESTB(y,x)    ((y) &= 1<<(x))        // y 第
x 位取出

#define _DDR_OUT(x,y)    (DDR##x |= (1<<y))   //
#define DDR_OUT(Z)       _DDR_OUT(Z)
      // Z 格式为(端口号,序号)
#define _DDR_IN(x,y)(DDR##x &= ~(1<<y))
#define DDR_IN(Z)        _DDR_IN(Z)
#define _PORT_H(x,y)     (PORT##x |= (1<<y))
#define PORT_H(Z)        _PORT_H(Z)
#define _PORT_L(x,y)     (PORT##x &= ~(1<<y))
#define PORT_L(Z)        _PORT_L(Z)
#define _PIN_VAL(x,y)    (PIN##x & (1<<y))
#define PIN_VAL(Z)       _PIN_VAL(Z)


#define CPU_F  8            // CPU 实际频率,以
MHz 为单位
#define DELAY_US(x) __delay_cycles(x*CPU_F)
          // 延时 x 微秒
#define                          DELAY_MS(x)
__delay_cycles(x*CPU_F*1000) //   延  时     x
（ x<=4 ）毫秒

#define NOP            _NOP()
#define IE_ON     _SEI()
#define IE_OFF    _CLI()
#define WDT_R _WDR()

#define ABS(X)  ((X)>=0 ? (X):(-(X)))

extern U8 flash u8LED7SEG[];
extern U8 flash u8LED7BIT[];


//*****************************************
void DelayUs(U16 us);

//*****************************************
void DelayMs(U16 ms);

//*****************************************
void M8Init(void);
//ATMEGA8 初始化,关中断

//*****************************************
#endif
```

参考文献

[1] Henry P.Hall. METHOD OF AND APPARATUS FOR AUTOMATIC MEASUREMENT OF IMPENDANCE OF OTHER PARAMETERS WITH MICROPROCESSOR CALCULATION TECHNIQUES[**P**]. United States 4196475，1980-04-01.
[2] Hewlett-Packard Co.. Multi-Frequency LCR Meters Test Components under Realistic Conditions[**J**]. HEWLETT-PACKARD Journal，1979，30（2）：24-32.
[3] 丁金林，王　峰. 智能RLC 测量仪的设计[**J**]. 现代电子技术，2009，32（15）：112-114
[4] 汪源浚. LCR测量仪的工作原理和使用技巧[**J**]. 教学与科技，2008，(1)：14-18
[5] 谢　敏. 基于MSP430的低功耗仪表设计[**J**]. 微计算机信息，2007，8（1）：142-144
[6] 袁益祥. 单片机在LCR测量中的应用[**J**]. 仪表技术，1992，(1)：3-9
[7] 周生景. 高精度LCR 测量系统的设计研究[**J**]. 电子测量于仪器学报，2003，17(3)：1-5
[8] Agilent Technologies，Inc.. Impendence Measurement Handbook[**M**]. USA：Agilent Technologies，Inc.，2000. 1-960
[9] Agilent Technologies，Inc.. Agilent U1731A/U1732A双显示手持式LCR测量仪用户及维修指南[**Z**]. USA：Agilent Technologies，Inc.，2004. 1-284
[10] 常州市同惠电子有限公司. TH2822系列手持式LCR数字电桥[**Z**]. 常州：常州市同惠电子有限公司，2004. 1-284
[11] Intersil Americas Inc.. ICL7135 Data Sheet[**Z**]. USA：Intersil Americas Inc.，2007. 1-15.
[12] Atmel Corporation. ATmega8(L) Data Sheet[**Z**]. USA：Atmel Corporation，2004. 1-284
[13] Linear Technology Corporation. LTC660 Data Sheet[**Z**]. USA：Linear Technology Corporation，1995. 1-12
[14] 李朝青. 单片机原理及接口设计[**M**]. 北京：北京航空航天大学出版社，2006：1-320.
[15] 马 潮. AVR 单片机嵌入式系统原理与应用实践[**M**]. 北京：北京航空航天大学出版社，2007. 1-536.
[16] 刘 辉. 电子仪器与测量技术[**M**]. 合肥：中国科学技术大学出版社，1992. 334-353
[17] Ron Mancini. Op Amp for Everyone[**M**]. USA：Texas Instruments, Inc.，2002. 1-450
[18] 谭浩强. C程序设计[**M**]. 北京：清华大学出版社，2005. 1-378
[19] 童诗白，华成英. 模拟电子技术基础[**M**]. 北京：高等教育出版社，2001. 1-632
[20] Walter G. Jung. Op Amp Applications[**M**]. USA：Analog Devices,Inc.,2006. 1-590
[21] 王 昊，李 昕. 集成运放应用电路设360例[**M**]. 北京：电子工业出版社，2007. 150-182
[22] 阎 石. 数字电子技术基础[**M**]. 北京：高等教育出版社，2006. 1-590

致　谢

这篇论文的完成，其中有着很多心理上的艰辛体验。成功完成不仅仅是我个人努力劳动的成果，指导老师给予了我极大的帮助和鼓励克服完成过程中的许多疑问，最终一步步地向着成功迈进，终于顺利完成了论文。在此，我感谢我的父母，他们含辛茹苦的将我抚养成人；感谢老师和学校，他们无私的教会了我很多东西。这篇论文是我大学四年的一个小小总结，伴随着它，我即将大学毕业，不过，我想这并不是学习的结束，我会一直坚持学习，向着理想不断迈进！

廖阳平

2011 年 5 月于西南大学