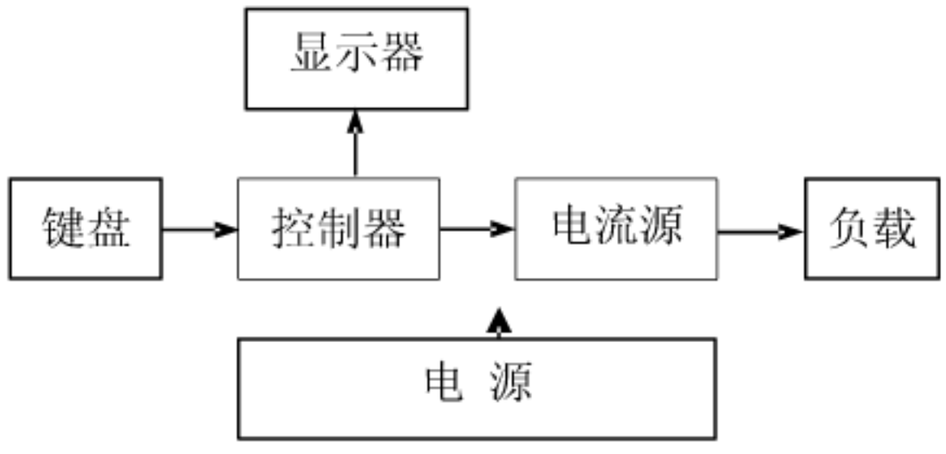
数控直流电流源（F 题）

一、任务

设计并制作数控直流电流源。输入交流 200～240V，50Hz；输出直流电压≤10V。其原理示意图如下所示。



二、要求

1、基本要求

（1）输出电流范围：200mA～2000mA；

（2）可设置并显示输出电流给定值，要求输出电流与给定值偏差的绝对值≤给定值的 1%+10 mA；

（3）具有“+”、“-”步进调整功能，步进≤10mA；

（4）改变负载电阻，输出电压在 10V 以内变化时，要求输出电流变化的绝对值≤输出电流值的 1%+10 mA；

（5）纹波电流≤2mA；

（6）自制电源。

2、发挥部分

（1）输出电流范围为 20mA～2000mA，步进 1mA；

（2）设计、制作测量并显示输出电流的装置 (可同时或交替显示电流的给定值和实测值)，测量误差的绝对值≤测量值的 0.1%+3 个字；

（3）改变负载电阻，输出电压在 10V 以内变化时，要求输出电流变化的绝对值≤输出电流值的 0.1%+1 mA；

（4）纹波电流≤0.2mA；

（5）其他。

三、评分标准

	项目	满分
基本要求	设计与总结报告：方案比较、设计与论证，理论分析与计算，电路图及有关设计文件，测试方法与仪器，测试数据及测试结果分析。	50
	实际完成情况	50
发挥部分	完成第(1)项	4
	完成第(2)项	20
	完成第(3)项	16
	完成第(4)项	5
	其他	5

四、说明

1、需留出输出电流和电压测量端子；

2、输出电流可用高精度电流表测量；如果没有高精度电流表，可在采样电阻上测量电压换算成电流；

3、纹波电流的测量可用低频毫伏表测量输出纹波电压，换算成纹波电流。

数控直流恒流源的设计与制作

发表日期：2006 年 5 月 1 日　出处：本站原创　　【编辑录入：zouwenkun】

指导老师:王贵恩博士　　制作人:彭浦能、梁星燎、林小涛

《数控直流恒流源 》　　《数控恒流源获奖证书》

摘要：本系统以直流电流源为核心，AT89S52 单片机为主控制器，通过键盘来设置直流电源的输出电流，设置步进等级可达 1mA，并可由数码管显示电流设定值和实际输出电流值。本系统由单片机程控设定数字信号，经过D/A转换器（AD7543）输出模拟量，再经过运算放大器隔离放大，控制输出功率管的基极，随着功率管基极电压的变化而输出不同的电流。单片机系统还兼顾对恒流源进行实时监控，输出电流经过电流/电压转换后，通过A/D转换芯片，实时把模拟量转化为数据量，再经单片机分析处理，通过数字量形式的反馈环节，使电流更加稳定，这样构成稳定的压控电流源。实际测试结果表明，本系统能有效应用于需要高稳定度的小功率恒流源的领域。

关键词：压控恒流源　智能化电源 闭环控制

The Digital Controlled Direct Current Source

Abstract: In this system the DC source is center and 89S52 version single chip microcomputer (SCM) is main controller, output current of DC power can be set by a keyboard which step level reaches 1mA, while the set value and the real output current can be displayed by LED. In the system, the digitally programmable signal from SCM is converted to analog value by DAC (AD7543), then the analog value which is isolated and amplified by operational amplifiers, is sent to the base electrode of power transistor, so an adjustable output current can be available with the base electrode voltage of power transistor. On the other hand, The constant current source can be monitored by the SCM system real-timely, its work process is that output current is converted voltage, then its analog value is converted to digital value by ADC, finally the digital value as a feedback loop is processed by SCM so that output current is more stable, so a stable voltage-controlled constant current power is designed. The test results have showed that it can be applied in need areas of constant current source with high stability and low power.

Keywords: voltage-controlled constant current source, intelligent power,closed loop control

前言

随着电子技术的发展、数字电路应用领域的扩展，现今社会，产品智能化、数字化已成为人们追求的一种趋势，设备的性能、价格、发展空间等备受人们的关注，尤其对电子设备的精密度和稳定度最为关注。性能好的电子设备，首先离不开稳定的电源，电源稳定度越高，设备和外围条件越优越，那么设备的寿命更长。基于此，人们对数控恒定电流器件的需求越来越迫切. 当今社会，数控恒压技术已经很成熟，但是恒流方面特别是数控恒流的技术才刚刚起步且有待发展，高性能的数控恒流器件的开发和应用存在巨大的发展空间。本文正是应社会发展的需求，研制出一种基于单片机的高性能的数控直流恒流源。本数控直流恒流源系统输出电流稳定，输出电流可在 20mA~2000mA 范围内任意设定，不随负载和环境温度变化，并具有很高的精度，输出电流误差范围±4mA，因而可实际应用于需要高稳定度小功率直流恒流源的领域。

1 系统原理及理论分析

1.1 单片机最小系统组成

单片机系统是整个数控系统的核心部分，它主要用于键盘按键管理、数据处理、实时采样分析系统参数及对各部分反馈环节进行

整体调整。主要包括 AT89S52 单片机、模数转换芯片 ADC0809、12 位数模转换芯片 AD7543、数码管显示译码芯片 74LS47 与 74LS138 等器件。

1.2 系统性能

本系统的性能指标主要由两大关系所决定，设定值与A/D采样显示值（系统内部测量值）的关系。内部测量值与实际测量值的关系，而后者是所有仪表所存在的误差。

在没有采用数字闭环之前，设定值与内部测量值的关系只能通过反复测量来得出它们的关系（要送多大的数才能使D/A输出与设定电流值相对应的电压值），再通过单片机乘除法再实现这个关系，基本实现设定值与内部测量值相一致。但由于周围环境等因素的影响，使设定值与内部测量值的关系改变，使得设定值与内部测量值不一致，有时会相差上百毫安，只能重新测量设定值与A/D采样显示值的关系改变D/A入口数值的大小才能重新达到设定值与内部测量值相一致，也就是说还不稳定。

在采用数字闭环后。通过比较设定值与A/D采样显示值，得出它们的差值，再调整D/A的入口数值，从而使A/D采样显示值逐步逼近设定值最终达到一致。而我们无须关心D/A入口数值的大小，从而省去了原程序中双字节乘除的部分，使程序简单而不受周围环境等因素的影响。

内部测量值与实际测量值的误差是由于取样电阻与负载电阻和晶体管的放大倍数受温度的影响和测量仪表的误差所造成的，为了减少这种误差，一定要选用温度系数低的电阻来作采样电阻，因此本系统选用锰铜电阻丝来做采样电阻。

1.3 恒流原理

数模转换芯片 AD7543 是 12 位电流输出型，其中 OUT1 和 OUT2 是电流的输出端。电流的输出级别可这样计算

$$DX=2^{n}$$

式中：DX 是控制级数

电压 u_i 由集成运算放大器 U8A 的 1 脚输出，根据 T 型电阻网络型的 D/A 转换关系可知，u_i 存在如下通式：

$$u_i=-(b_{n-1}*2^{n-1}+b_{n-2}*2^{n-2}+\cdots\cdots+b_1*2+b_0*2^0)*\frac{V_{REF}}{2^n*R}*R_f=-B\frac{V_{REF}}{2^n} \quad (1)$$

式中：u_i——输出电压(V)；V_{REF}——参考电压(V)；R ——T 网络电阻(Ω)；R_f——外接反馈电阻(Ω)。

电流放大电路存在如下关系：

$$I_b=\frac{-u_i}{R_4}\cdot\frac{(R_3+R_{RP})}{(\beta+1)R_3} \quad (2)$$

$$I_L=\beta I_b \quad (3)$$

式中： Ib——基极电流（mA）； Ui——输入电压（V）； IL——负载电流（mA）。

由式（1）、（2）可得到：

$$I_L=-\frac{u_i}{R_4}\cdot\frac{(R_3+R_{RP})}{(\beta+1)R_3}\cdot\beta \quad (4)$$

由于电路中的放大系数 β 值远大于 1，而 R_3 与 R_4 保持恒定，所以可推出负载电流与输入电压存在如下关系：

$$I_L=-k\frac{u_i}{R_4} \quad (5)$$

由式(5) 、（1）可得到:

$$I_L = kB\frac{V_{REF}}{2^n R_f} \quad (6)$$

其中，k 为比例系数。

式（6）可知，负载电流 I_L 不随外部负载 R_L 的变化而改变。当 B 保持不变时(即 AD7543 的输入数字量保持不变)，输出电流 I_L 维持不变，能够达到恒流的目的。为了实现数控的目的，可以通过微处理器控制 AD7543 的模拟量输出，从而间接改变电流源的输出电流。从理论上来说，通过控制 AD7543 的输出等级，可以达到 1mA 的输出精度。但是本系统恒流源要求输出电流范围是 20mA~2000mA，而当器件处于 2000mA 的工作电流时，属于工作在大电流状态，晶体管长时间工作在这种状态，集电结发热严重，导致晶体管 β 值下降，从而导致电流不能维持恒定。为了克服大电流工作时电流的波动，在输出部分增加了一个反馈环节来控制电流稳定，减小电流的波动，此反馈回路采用数字形式反馈，通过微处理器的实时采样分析后，根据实际输出对电流源进行实时调节。经测试表明，采用常用的大功率电阻作为采样电阻 R0，输出电流波动比较大，而选用锰铜电阻丝制作采样电阻，电流稳定性得到了改善。电路反馈原理如图 1 所示。

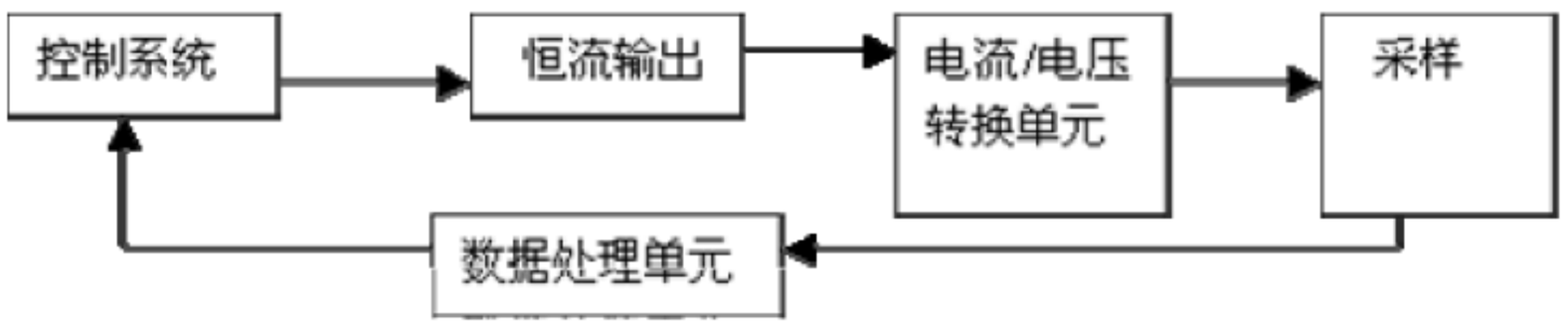


图 1　电流输出反馈电路原理

2 总体方案论证与比较

方案一：采用各类数字电路来组成键盘控制系统，进行信号处理，如选用 CPLD 等可编程逻辑器件。本方案电路复杂，灵活性不高，效率低，不利于系统的扩展，对信号处理比较困难。

方案二：采用 AT89S52 单片机作为整机的控制单元，通过改变 AD7543 的输入数字量来改变输出电压值，从而使输出功率管的基极电压发生变化，间接地改变输出电流的大小。为了能够使系统具备检测实际输出电流值的大小，可以将电流转换成电压，并经过 ADC0809 进行模数转换，间接用单片机实时对电压进行采样，然后进行数据处理及显示。此系统比较灵活，采用软件方法来解决数据的预置以及电流的步进控制，使系统硬件更加简洁，各类功能易于实现，能很好地满足题目的要求。本方案的基本原理如图 2 所示。

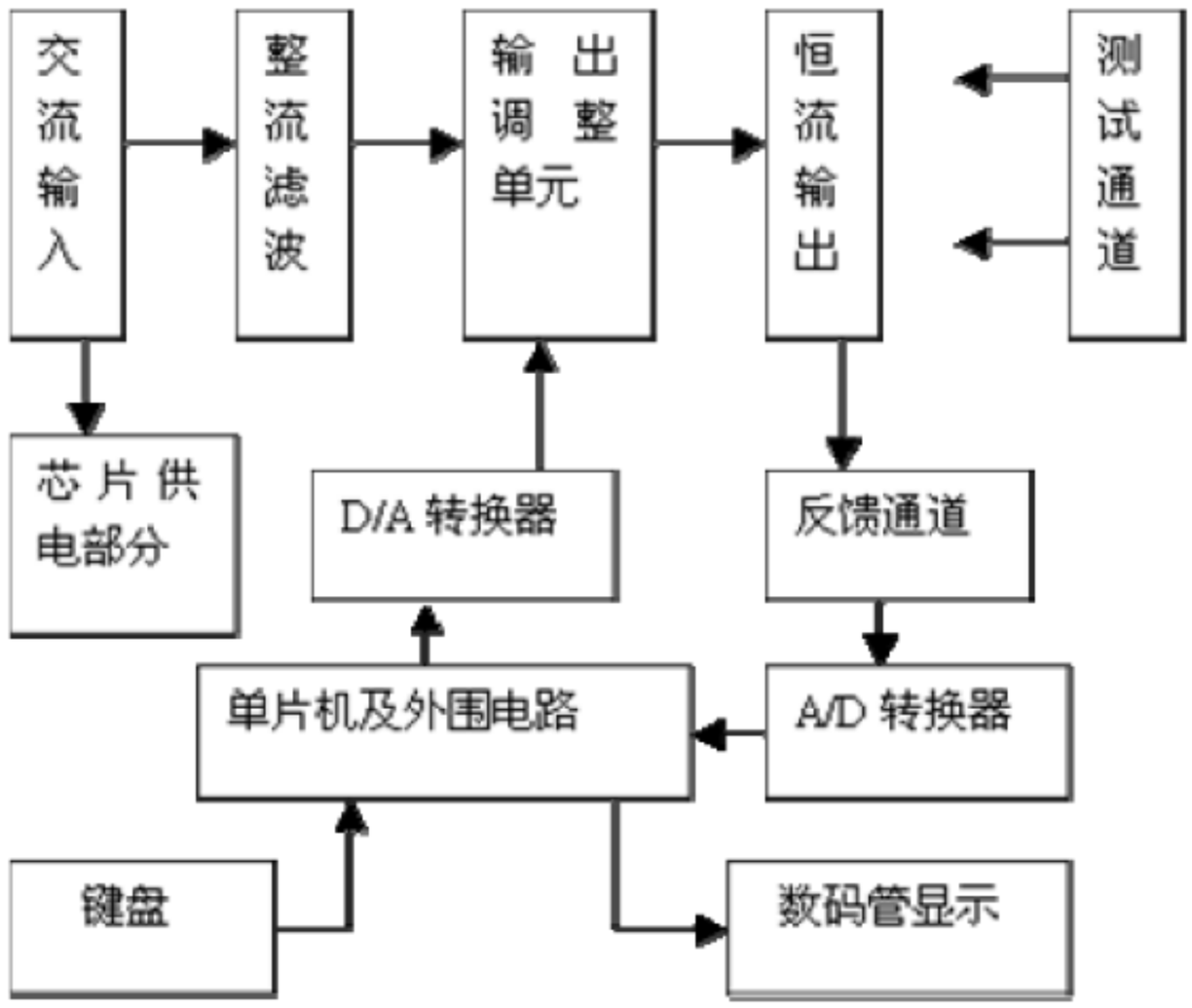


图 2　系统原理框图

比较以上两种方案的优缺点，方案二简洁、灵活、可扩展性好，能达到题目的设计要求，因此采用方案二来实现。

3 模块电路设计与比较

3.1 恒流源方案选择

方案一：采用恒流二极管或者恒流三极管，精度比较高，但这种电路能实现的恒流范围很小，只能达到十几毫安，不能达到题目的要求。

方案二：采用四端可调恒流源，这种器件靠改变外围电阻元件参数，从而使电流达到可调的目的，这种器件能够达到 1~2000 毫安的输出电流。改变输出电流，通常有两种方法：一是通过手动调节来改变输出电流，这种方法不能满足题目的数控调节要求；二是通过数字电位器来改变需要的电阻参数，虽然可以达到数控的目的，但数字电位器的每一级步进电阻比较大，所以很难调节输出电流。

方案三：压控恒流源，通过改变恒流源的外围电压，利用电压的大小来控制输出电流的大小。电压控制电路采用数控的方式，利用单片机送出数字量，经过 D/A 转换转变成模拟信号，再送到大功率三极管进行放大。单片机系统实时对输出电流进行监控，采用数字方式作为反馈调整环节，由程序控制调节功率管的输出电流恒定。当改变负载大小时，基本上不影响电流的输出，采用这样一个闭路环节使得系统一直在设定值维持电流恒定。该方案通过软件方法实现输出电流稳定，易于功能的实现，便于操作，故选择此方案。电路原理图如图 3 所示。

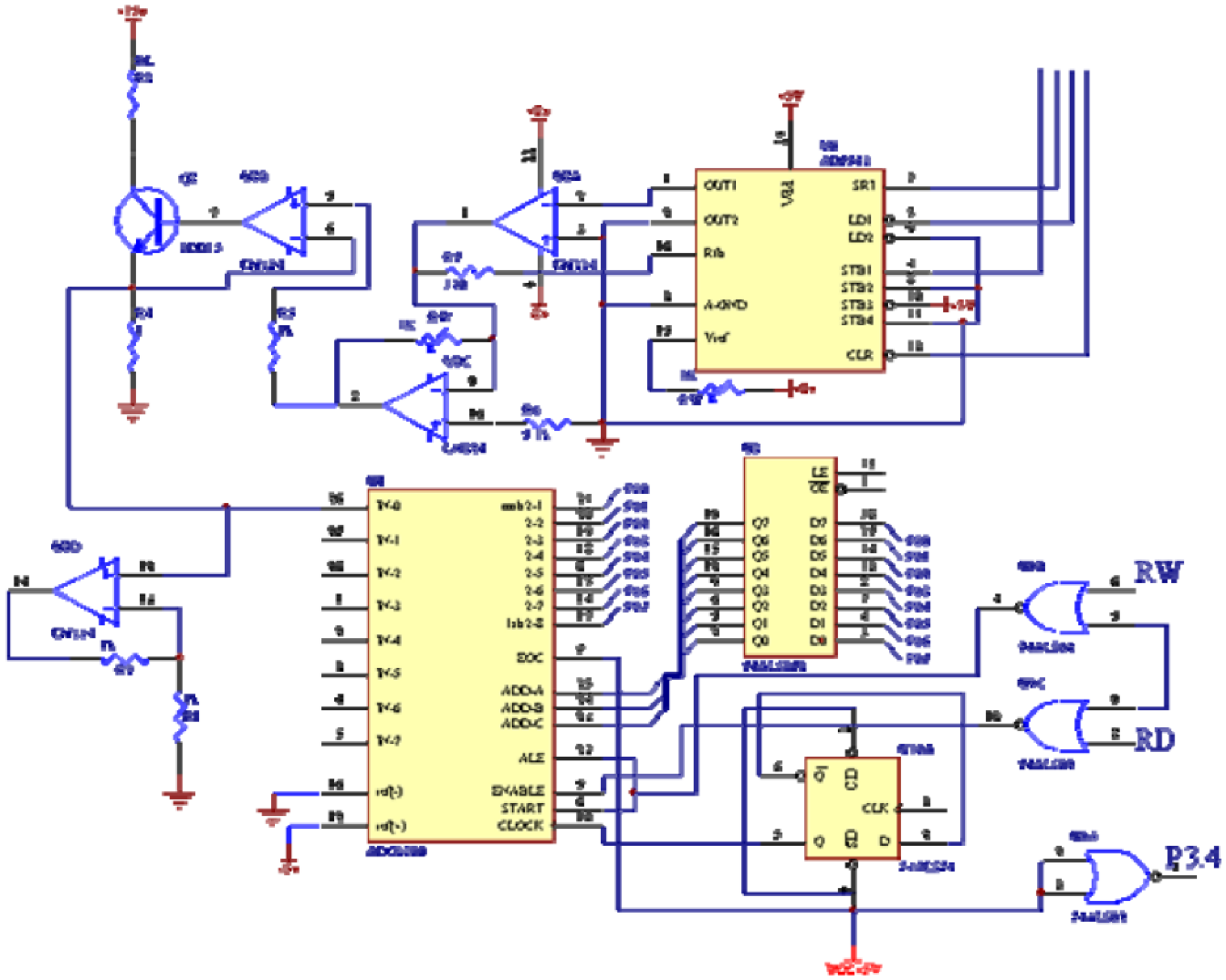


图 3　压控恒流源电路原理

3.2 反馈闭环方案选择

方案一：采样电阻 R_0 上的电压 $U_0 \approx I_L \cdot R_0$，可知输出电流与采样电阻存在近似线性关系，因此可以从检测电阻 R_0 上电压的大小来直接增减反馈深度。

方案二：从采样电阻 R_0 上得到一个反馈电压，由于采样电阻阻值比较小，在该电阻上的压降相应也小，为了提高系统控制的灵敏度，采用一级运算放大器对采样电压进行放大，再送到 ADC0809 进行 A/D 转换。数据由单片机系统进行相应处理，为了达到 1mA 步进，选用 12 位串行 D/A 转换器件 AD7543 可以满足题目要求，而且该芯片是采用串行数据传送方式，硬件电路简单。同时反馈系统控制灵活，易于达到 1mA 的步进要求。

3.3 控制单元方案选择

由于要实现人机对话，至少要有 10 个数字按键和两个步进按键，考虑到还要实现其它的功能键，选用 16 按键的键盘来完成整个系统控制。显示部分采用 8 位 LED 数码管，而且价格便宜，易于实现。考虑到单片机的 I/O 端口有限，为了充分优化系统，采用外部扩展一片 8155 来实现键盘接口与显示功能。电路原理如图 4 所示。

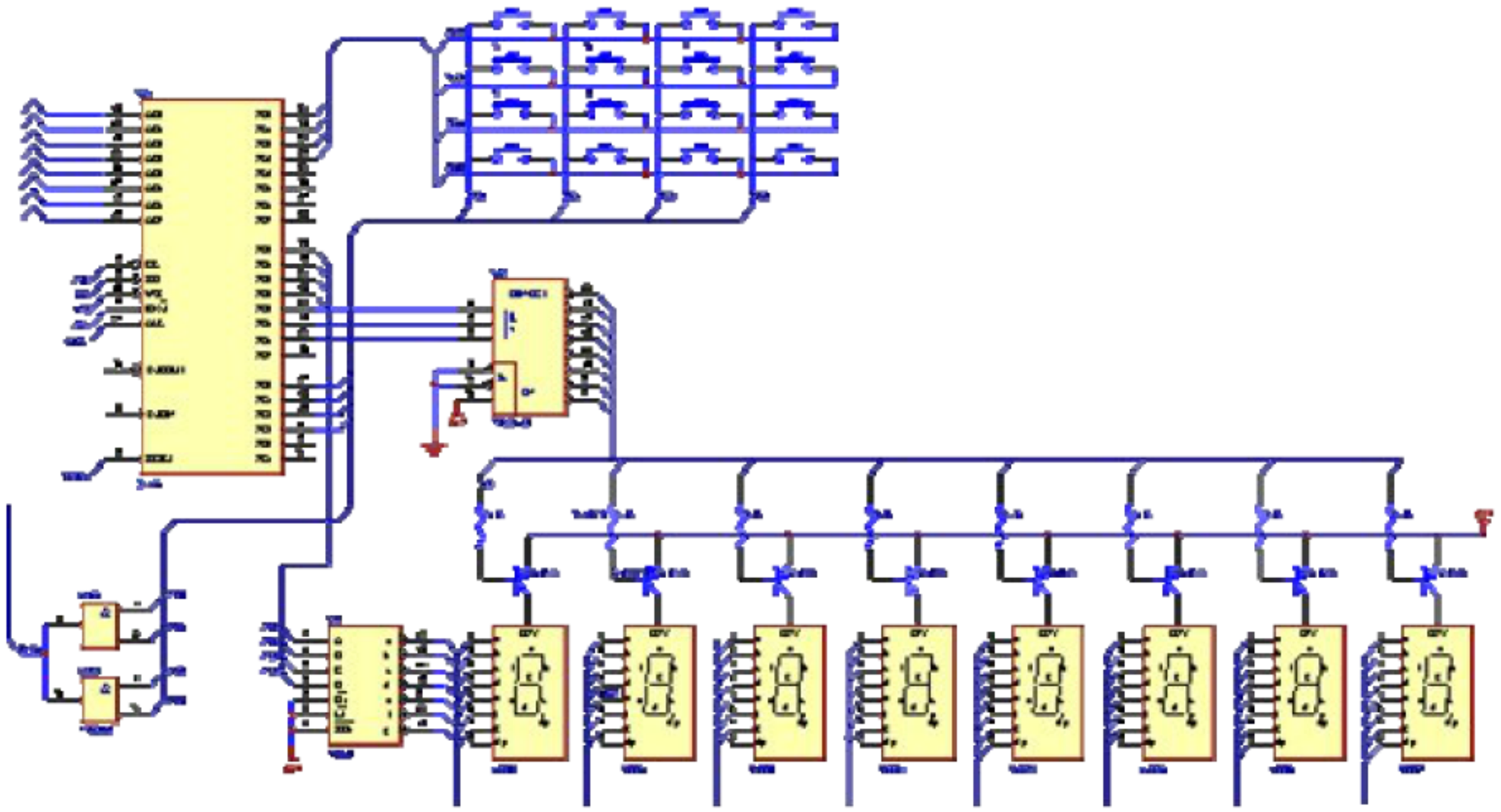


图 4 键盘及显示电路

3.4 电源方案选择

方案一：用开关稳压电源给整机供电，此方案能够完成本作品电流源的供电，但开关电源比较复杂，而且体积也比较大，制作不便，因而此方案难以实现。

方案二：单片机控制系统以及外围芯片供电采用 78 系列三端稳压器件，通过全波整流，然后进行滤波稳压。电流源部分由于要给外围测试电路提供比较大的功率，因此必须采用大功率器件。考虑到该电流源输出电压在 10V 以内，最大输出电流不大于 2000mA，由公式 P=U*I 可以粗略估算电流源的功耗为 20W。同时考虑到恒流源功率管部分的功耗，需要预留功率余量，因此供电电源要求能输出 30W 以上。为了尽量减少输出电流的纹波，要求供电源要稳定，因此采用隔离电源，选用由 LM338 构成的高精度大电流稳压电源。此方案输出电流精度高，能满足题目要求，而且简单实用，易于自制，故选用方案二。稳压电源原理如图 5 所示。

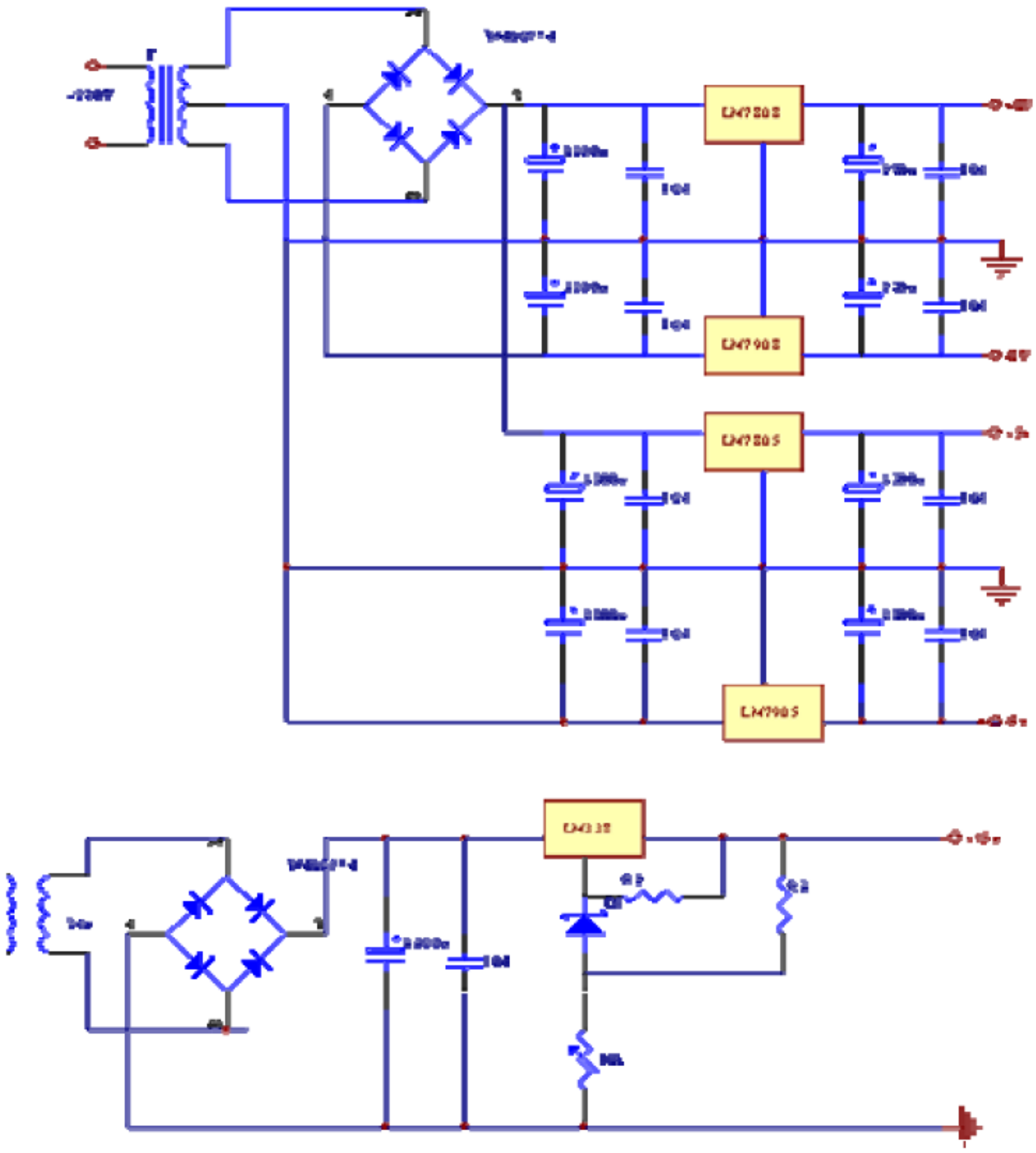


图 5　稳压电源原理

3.5 过压报警功能设计

为了使本数控直流电流源进一步智能化，考虑到要求输出电压不大于 10V，因此系统测试部分设计了一个过压报警电路，用于对电压的实时监测，一旦有过压现象，控制器响应后会发出报警控制信号。电路原理参见图 3。

4 软件设计

根据实际的硬件电路，为了有效地减小纹波电流，用软件方法实现去峰值数值滤波，以减小环境参数对输出控制量的影响。软件设计主程序流程图和闭环比较子程序流程图；电流设置子程序流程图；键盘中断子程序流程图；显示中断子程序流程图。分别如下图所示。

根据本系统的实际要求软件设计可分为以下几个功能模块：

4.1 主程序模块 MAIN：流程图如图 6 所示。

主程序负责与各子程序模块的接口和检查键盘功能号。

4.2 闭环比较子程序模块 BIHUAN：流程图如图 7 所示。

通过调用闭环比较子程序得出实际值与设定值的差值，如果是实际值大于设定值则将原来的 D／A 的入口数值减去这个差值再送去 D/A 转换，如果是实际值小于设定值则把原来的 D／A 的入口数值加上这个差值再送去转换。如果输出值与设定值仍然不一致，再

将差值和设定值相加送 D/A 转换，以逐步逼近的形式使实际值和设定值相一致后通过 LED 把稳定的实际值显示出来。而逐步逼近过程中的实际值不送显示因此减少了实际显示值的不稳定。这也是结构化程序的要点（合理设置程序的顺序结构）。

4.3 电流设置子程序模块 SETUP：流程图如图 8 所示。

通过键盘设置电流的大小，因为本系统最大输出电流是 2000mA，所以该子程序兼有电流设置合法性，也就是说设置电流不能大于 2000mA。

4.4 键盘中断子程序模块 KEYSCAN：流程图如图 9 所示。

本系统采用外部中断 1 来实现实时扫描，使程序及时响应按键请求而无需顾虑其它程序模块运行情况。

4.5 显示中断子程序模块 LED：流程图如图 10 所示。

本系统采用定时中断 0 来实现逐位动态显示，每位显示间隔固定为 2ms，使 LED 输示非常稳定，无法考虑定时刷新显示，使得该显示子程序简单灵活，适用性广。

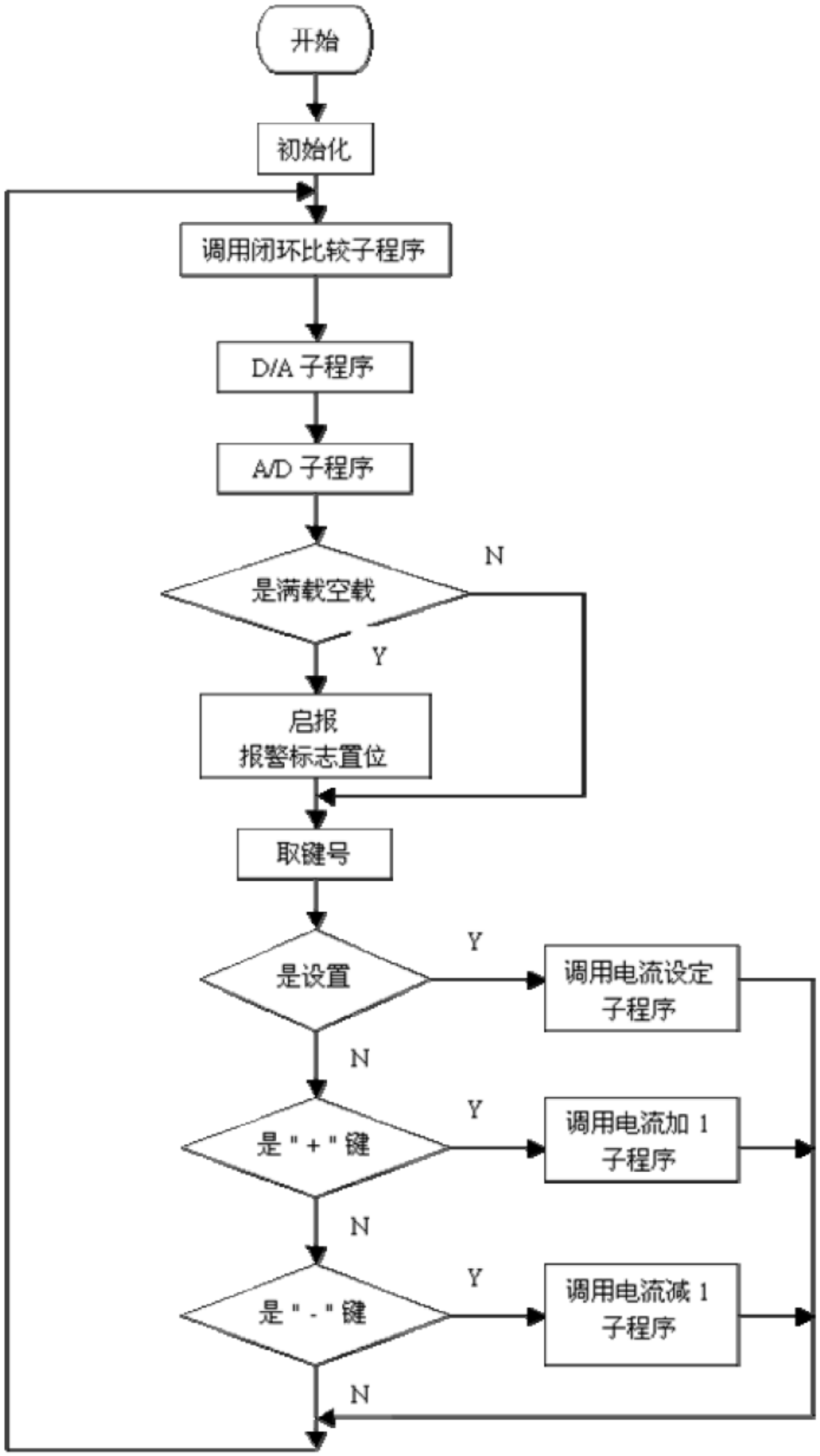

开始
初始化
调用闭环比较子程序
D/A 子程序
A/D 子程序
是满载空载
N
Y
启报
报警标志置位
取键号
是设置
Y
调用电流设定
子程序
N
是"+"键
Y
调用电流加 1
子程序
N
是"-"键
Y
调用电流减 1
子程序
N

图 6 主程序流程图

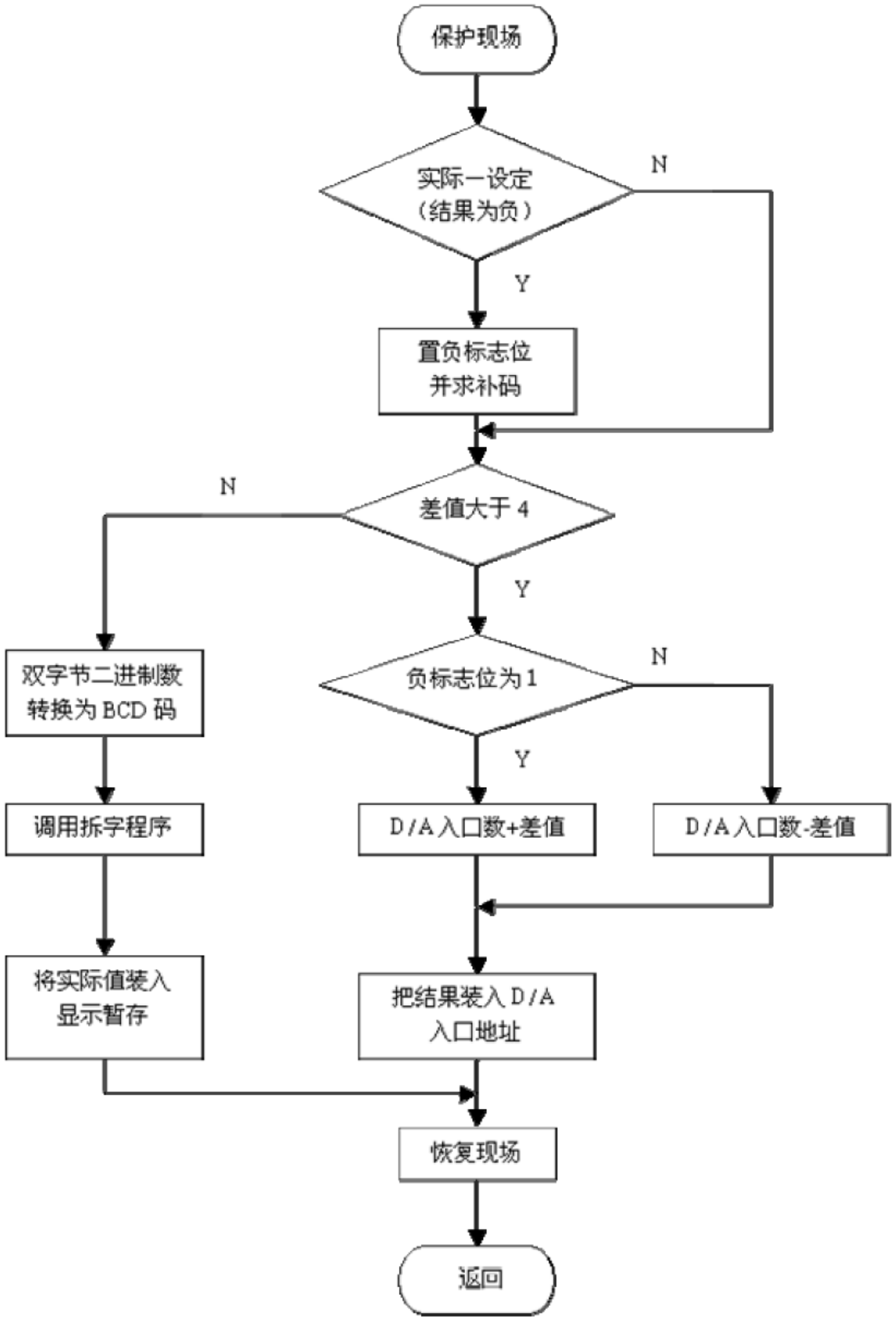


图 7 闭环比较子程序流程图

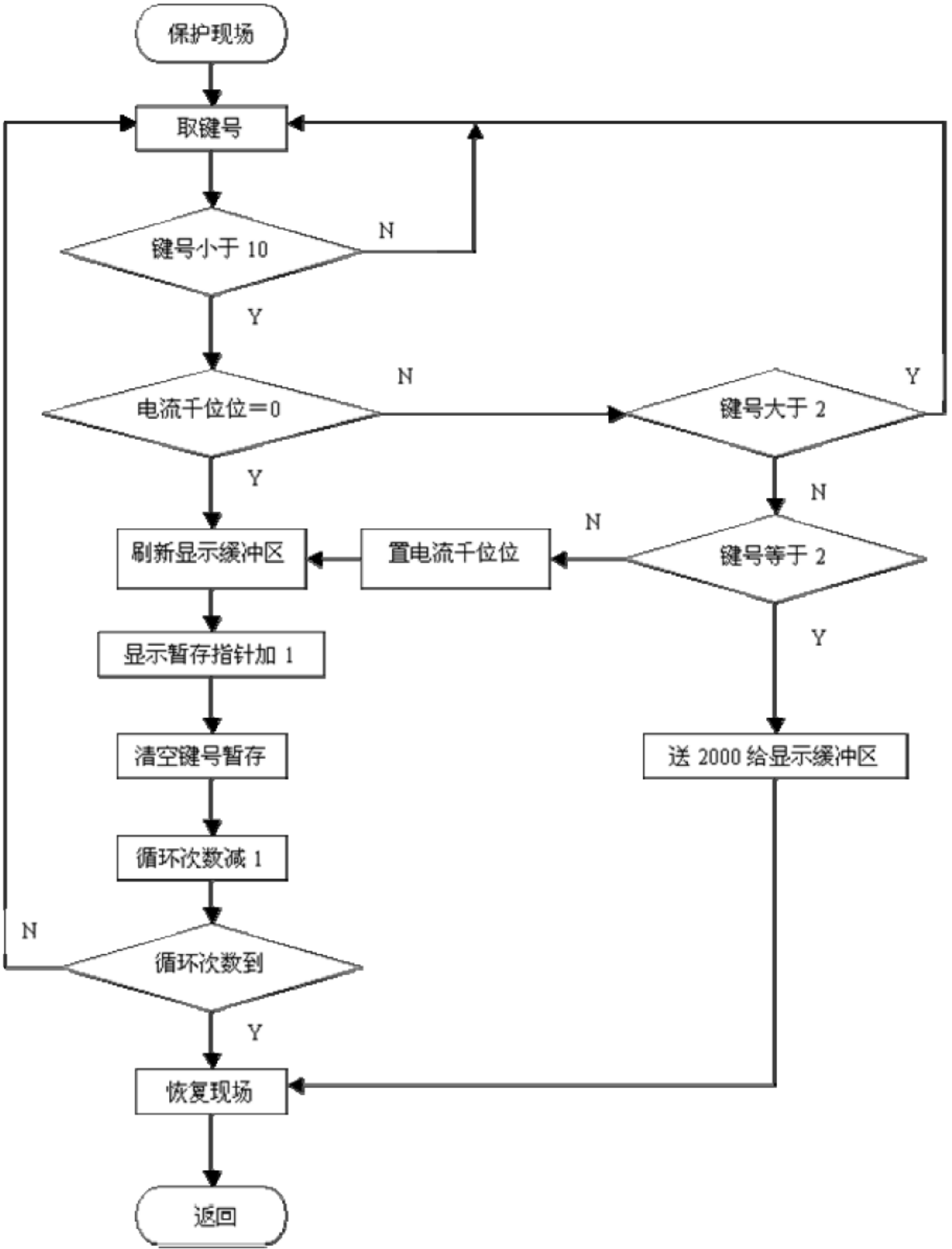


图 8　电流设置子程序模块

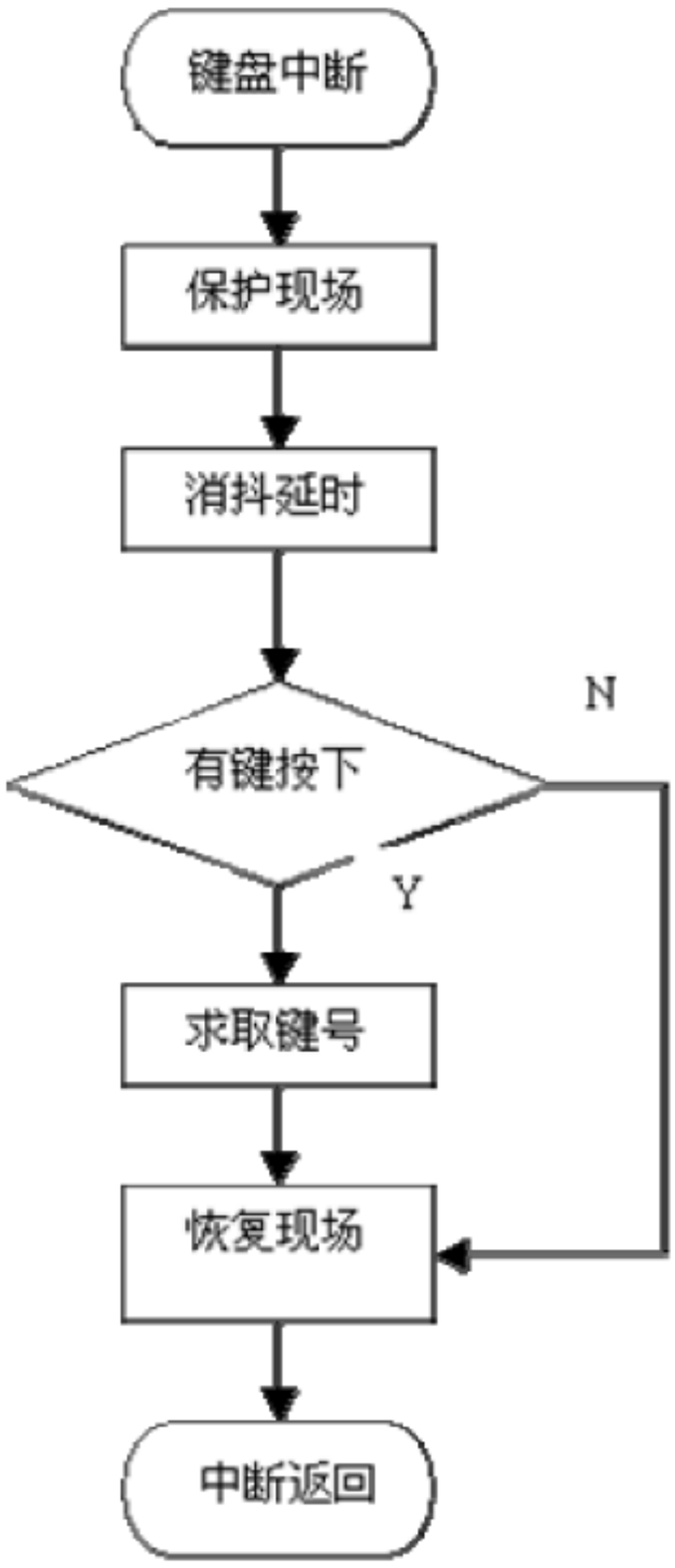


图 9 键盘中断子程序流程图

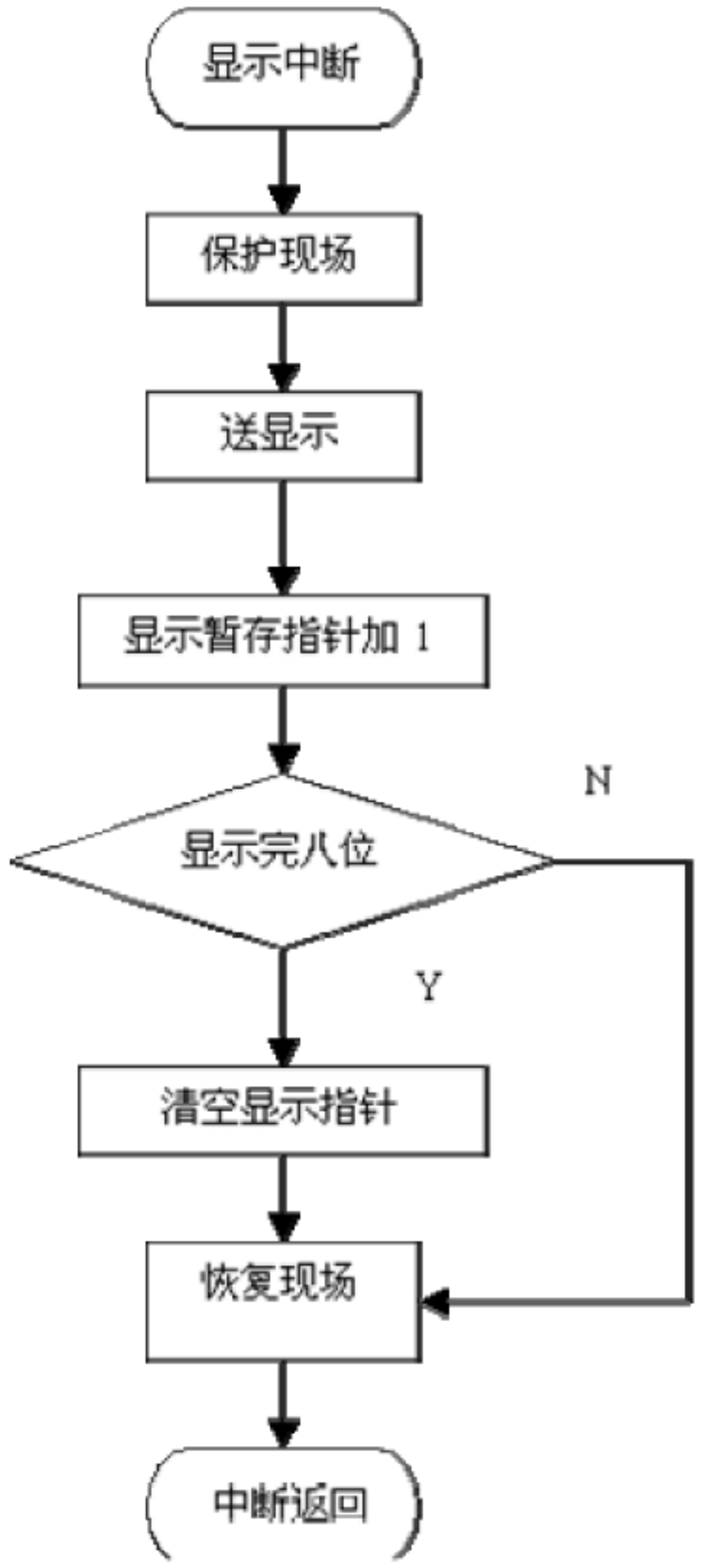


图 10 显示中断子程序流程图

5 数据测试及分析

数据测试是反映系统性能的重要指标。因此对本系统进行了全面的测试，分别为输出电流测试、步进电流测试、工作时间测试、负载阻值变化测试、纹波电流测试。本系统测试采用的仪表如下：当测试系统电流分别 0～200ｍＡ和 200ｍＡ~2000ｍＡ时，分别采用数字表ＤＴ9801 的 200ｍＡ档和 10Ａ档。测试电压采用数字表ＸＢ－9208Ｂ的 2Ｖ档和 20Ｖ档。测试纹波电流采用低频毫伏表 DA—16D 来测试纹波电压，但当测量值与对应量程相差较大时，会有一定的误差。

5．1 输出电流测试

给电流源上电，通过按键设定输出电流值，对应 D/A 转换输出电压、晶体管基极电压，电流源自身检测到实际输出电流值以及通过外部电流表测量的电流值，相关数据如表 1 所示。由表可知设定值的线性增大，相关数据也相应增大，但由于取样电阻负载电阻和晶体管的放大倍数受温度的影响和测量仪表的误差而造成大电流时实际值比设定值略小，小电流时实际值比设定值略大。所以实际调试时只能拿中间值 1000 毫安来作基准。

表 1　输出电流测试表

键盘设定值/mA	D/A 转换输出电压 /V	基极电压/V	显示输出值/mA	外部测量值/A
20	0.595	0.555	16	0.03
50	0.644	0.614	48	0.06
100	0.705	0.691	96	0.10
200	0.824	0.805	200	0.21
300	0.925	0.921	296	0.30
400	1.032	1.026	400	0.40
500	1.131	1.137	504	0.50
600	1.234	1.233	600	0.60
700	1.322	1.331	696	0.70
800	1.431	1.427	800	0.80
900	1.529	1.532	904	0.90
1000	1.640	1.638	1000	1.00
1100	1.800	1.798	1104	1.10
1200	1.902	1.900	1200	1.20
1300	2.00	2.00	1296	1.30
1400	2.10	2.10	1400	1.40
1500	2.20	2.21	1496	1.50
1600	2.37	2.37	1600	1.60
1700	2.46	2.47	1696	1.70
1800	2.57	2.58	1800	1.80
1900	2.72	2.72	1904	1.90
2000	2.82	2.83	2000	2.00

5.2 步进电流测试

由上面系统方案分析可知本系统由于现有器件限制只能采用 8 位的A/D作闭环反馈。则要A/D转换回来的数乘以 8 才能达到2000mA，即显示输出值是每隔 8 mA跳变的，而外部测量值也是 8 mA跳变的，所以理论上设定值与实际值的最大误差为 4mA。

表 2　步进电流测试表

键盘设定值/mA	显示输出值/mA	外部测量值/mA
45	48	61.4
46	48	61.4
47	48	61.4
48	48	61.4
49	48	61.4
50	48	61.4
51	48	61.4
52	48	61.4
53	56	70.9
54	56	70.9
145	144	156.1
146	144	156.1
147	144	156.1

148	144	156.1
149	152	163.3
150	152	163.3
151	152	163.3
152	152	163.3
153	152	163.3
154	152	163.3

5.3 工作时间测试

工作时间测试表见表 3。由表 3 可知，当系统工作在大电流时，电流外部测量值随着系统工作时间延长略有减小，而显示输出值不变。造成这种误差主要是因为随着系统工作时间延长，系统器件温度不断升高，采样电阻与负载电阻有所增大，且晶体管的放大倍数有所减小，因而造成输出电流减少而采样电阻两端电压不变。

表 3　工作时间测试表

设定值/mA	时间/Min	显示输出值/mA	外部测量值
150	0	152	163.0mA
150	1	152	162.7mA
150	2	152	162.5mA
150	3	152	162.6mA
150	4	152	163.2mA
150	5	152	162.5mA
500	0	504	0.49A
500	1	496	0.49A
500	2	496	0.49A
500	3	504	0.49A
500	4	496	0.48A
500	5	504	0.49A
1000	0	1000	1.00A
1000	1	1000	1.00A
1000	2	1000	1.00A
1000	3	1000	1.00A
1000	4	1000	0.99A
1000	5	1000	0.99A
2000	0	2000	2.00A
2000	1	2000	1.99A
2000	2	2000	1.99A
2000	3	2000	1.98A
2000	4	2000	1.98A
2000	5	2000	1.97A

5.4 负载阻值变化测试

测试结果表明，无论是大电流还是小电流，负载阻值的改变对系统的影响都是比较小，说明系统达到恒流这一基本要求。

表 4　负载阻值变化测试表

键盘设定值 /mA	负载R_L阻值 /Ω	显示输出值 /mA	外部测量值
200	1	200	199.0mA
200	5	200	199.3mA
200	10	200	199.5mA
200	20	200	200.0mA
200	50	200	200.0mA
500	1	504	0.49 A
500	5	504	0.49 A
500	10	496	0.50 A
500	15	496	0.50 A
500	20	504	0.50 A
1000	1	1000	1.00 A
1000	2	1000	1.00 A
1000	5	1000	1.00 A
1000	8	1000	1.00 A
1000	10	1000	1.00 A
2000	1	2000	2.00 A
2000	2	2000	2.00 A
2000	3	2000	1.99 A
2000	4	2000	2.00 A
2000	5	2000	1.99 A

5.5 纹波电流测试

测试及运算结果表明，输出纹波电流较小，维持在 0.1~0.2mA 之间，能够满足小于 0.2mA 的要求。同时表明，本系统输出电流稳定，可以满足直流恒流源的应用要求。

表 5　纹波电流测试

设定输出电流/mA	负载R_L阻值/Ω	纹波电压实测值/mV	转换成纹波电流/mA
50	10	1.21	0.121
90	10	1.23	0.123
335	10	1.45	0.145
756	5	0.925	0.185
1450	5	1.00	0.200
1700	5	0.970	0.194
1980	5	0.945	0.189

6 结束语

在设计制作数控直流恒流源的过程中，我们深切体会到，实践是理论运用的最好检验。本次设计是对我们三年所学知识的一次综合性检测和考验，无论是动手能力还是理论知识运用能力都得到了提高，同时加深了我们对网络资源认识，大大提高了查阅资料的能力和效率，使我们有充足的时间投入到电路设计当中。本系统的研制主要应用到了模拟电子技术、数字电子技术、单片机控制技术、大功率电源设计、电子工艺等多方面的知识，所设计的基于单片机程序控制的压控式恒流源，达到了应用要求。在数据测试和调试方面，由于仪表存在误差和电路器件因工作时间过长温度升高而产生的误差，使得测量数据不是很精确，本系统就此通过软件设计，减

少误差的存在，使输出电流的误差范围减小到±4mA，大大提高了系统的精度，与理论计算吻合。

程序部分

数控恒流源程序

```
#include <reg52.h>
#include <absacc.h>
#include<string.h>
#include<intrins.h>
#define unit unsigned int
#define uchar unsigned char
#define DELAY_TIME 60
#define TRUE 1
#define FALSE 0

uchar keyup;
uchar keydown;
uchar keyupstate;
uchar keydownstate;

static unsigned int s=0;
static unsigned int b=1;
static unsigned int q=0;
static unsigned int c=0;
static unsigned int a;

code unsigned char table[19]=
{11,17,23,28,34,39,45,51,56,62,68,73,79,84,90,96,101,107,113};
code unsigned char Seg7Code[11]=                    //用十六进数作为数组下
标，可直接取得对应的七段编码字节
{0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0xBF};

sbit SCL=P1^4;
sbit SDA=P1^5;

void DELAY(unsigned int t) /*延时函数*/
    {
        while(t!=0)
            t--;
    }

void I2C_Start(void)
    {
        /*启动 I2C 总线的函数，当 SCL 为高电平时使 SDA 产生一个负跳变*/
        SDA=1;
```

```
        SCL=1;
        DELAY(DELAY_TIME);
        SDA=0;
        DELAY(DELAY_TIME);
        SCL=0;
        DELAY(DELAY_TIME);
    }

void I2C_Stop(void)
    {
        /*终止 I2C 总线，当 SCL 为高电平时使 SDA 产生一个正跳变*/
        SDA=0;
        SCL=1;
        DELAY(DELAY_TIME);
        SDA=1;
        DELAY(DELAY_TIME);
        SCL=0;
        DELAY(DELAY_TIME);
    }

void SEND_0(void)    /* SEND ACK */
    {
        /*发送 0，在 SCL 为高电平时使 SDA 信号为低*/
        SDA=0;
        SCL=1;
        DELAY(DELAY_TIME);
        SCL=0;
        DELAY(DELAY_TIME);
    }

void SEND_1(void)
    {
        /*发送 1，在 SCL 为高电平时使 SDA 信号为高*/
        SDA=1;
        SCL=1;
        DELAY(DELAY_TIME);
        SCL=0;
        DELAY(DELAY_TIME);
    }

bit Check_Acknowledge(void)
    {
        /*发送完一个字节后检验设备的应答信号*/
        SDA=1;
        SCL=1;
```

```
    DELAY(DELAY_TIME/2);
    F0=SDA;
    DELAY(DELAY_TIME/2);
    SCL=0;
    DELAY(DELAY_TIME);
    if(F0==1)
        return FALSE;
    return TRUE;
}

void WriteI2CByte(char b)reentrant
    {
        /*向 I2C 总线写一个字节*/
        char i;
        for(i=0;i<8;i++)
            if((b<<i)&0x80)
                SEND_1();
            else
                SEND_0();
    }

char ReadI2CByte(void)reentrant
    {
        /*从 I2C 总线读一个字节*/
        char b=0,i;
        for(i=0;i<8;i++)
        {
            SDA=1;      /*释放总线*/
            SCL=1;      /*接受数据*/
            DELAY(10);
            F0=SDA;
            DELAY(10);
            SCL=0;
            if(F0==1)
                {
                    b=b<<1;
                    b=b|0x01;
                }
            else
                b=b<<1;
        }
        return b;
}
```

```
/**********以下为读写 24c02 的函数**********/
void Write_One_Byte(char addr,char thedata)
{
          bit acktemp=1;
          /*write a byte to mem*/
          I2C_Start();
          WriteI2CByte(0xa0);
          acktemp=Check_Acknowledge();
          WriteI2CByte(addr);/*address*/
          acktemp=Check_Acknowledge();
          WriteI2CByte(thedata);/*thedata*/
          acktemp=Check_Acknowledge();
          I2C_Stop();

}

char Read_One_Byte(char addr)
{         bit acktemp=1;
          char mydata;
           /*read a byte from mem*/
          I2C_Start();
          WriteI2CByte(0xa0);
          acktemp=Check_Acknowledge();
          WriteI2CByte(addr);/*address*/
          acktemp=Check_Acknowledge();
          I2C_Start();
          WriteI2CByte(0xa1);
          acktemp=Check_Acknowledge();

          mydata=ReadI2CByte();
          acktemp=Check_Acknowledge();

          return mydata;
          I2C_Stop();
}

void DisplayBrush( void )          //显示输出函数
{
unit m;
 P0=0xff;
 P0 = Seg7Code[ 10 ];
 P1=0xfe;
    for(m=0;m<1000;m++);
```

```
P0 = Seg7Code[ s ];
P1=0xfd;
for(m=0;m<1000;m++);

P0 = Seg7Code[ b ];
P1=0xfb;
for(m=0;m<1000;m++);

P0 = Seg7Code[ q ];
 P1=0xf7;
 for(m=0;m<1000;m++);
}

void jisuan (void)
{ s=a/100;
 b=(a/10)%10;
 q=a%10;
 }

void delays(void)                              //按键去斗延时函数
{  uchar i;
   for(i=300;i>0;i--);
}

void main( void )
{ c=Read_One_Byte(0x10);
 a=Read_One_Byte(0x20);
 if (c>18||a>100)
 {c=0;a=10;}
 P2=table[c];
  jisuan();
 DisplayBrush();
 EA=1;IT0=1;IT1=1;EX0=1;EX1=1;

while (1)
 {
if (keyupstate!=keyup)
 { delays();
 if (c<=18)
  {if (a==100)
   goto L1;
      else a+=5;
    c++;}
L1: keyupstate=keyup;
 P2=table[c];
```

```
    jisuan();
   DisplayBrush();
   Write_One_Byte(0x10,c);
   Write_One_Byte(0x20,a);
   }
////////////////////////////////////////////////////////
  if (keydownstate!=keydown)
   { delays();
   if (a==10)
      goto L2;
      else a-=5;
  c--;
L2: keydownstate=keydown;
  P2=table[c];
    jisuan();
  DisplayBrush();
  Write_One_Byte(0x10,c);
  Write_One_Byte(0x20,a);
  }
  DisplayBrush();

}
}


void intsvr0(void)  interrupt 0 using 1
{
 keyup=!keyup;
}


void intsur0(void)   interrupt 2 using 1
{
 keydown=!keydown;
}
```