浙江大学

本科实验报告

数控恒流源设计

课程名称：	电子电路设计实验
姓　　名：	周灿松、王瑜昕、朱夏瑜
学　　院：	信息与电子工程学院
系：	
专　　业：	信息工程
指导老师：	王子立、金向东、龚淑君

2021 年 7 月 22 日

浙江大学实验报告

专业：信息工程
姓名：周灿松、王瑜昕、朱夏瑜
日期：2021 年 7 月 22 日
地点：东 4-227

课程名称：电子电路设计实验　　指导老师：王子立、金向东、龚淑君　　成绩：
实验名称：数控恒流源设计　　实验类型：设计实验　　同组学生姓名：

一、 系统方案

1. 系统总体框图

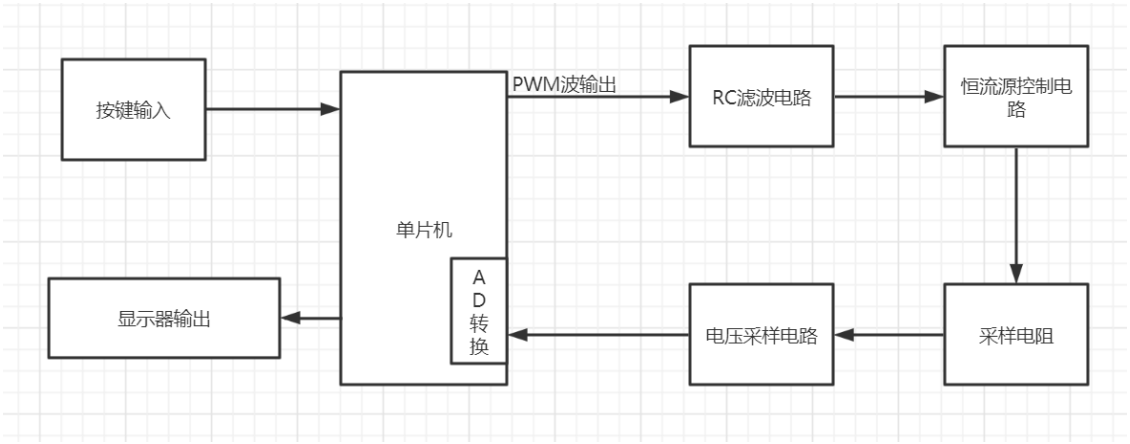

图 1: 系统总体框图

2. 方案比较与选择

2.1 采用开关电源的恒流源

采用开关电源的恒流源电路如图 1 所示。当电源电压降低或负载电阻 R1 降低时，采样电阻 RS 上的电压也将减少，则 SG3524 的 12、13 管脚输出方波的占空比增大，从而 BG1 导通时间变长，使电压 U0 回升到原来的稳定值。BG1 关断后，储能元件 L1、E2、E3、E4 保证负载上的电压不变。当输入电源电压增大或负载电阻值增大引起 U0 增大时，原理与前类似，电路通过反馈系统使 U0 下降到原来的稳定值，从而达到稳定负载电流 I1 的目的。

优点：开关电源的功率器件工作在开关状态，功率损耗小，效率高。与之相配套的散热器体积大大减小，同时脉冲变压器体积比工频变压器小了很多。因此采用开关电源的恒流源具有效率高、体积小、重量轻等优点。

缺点：开关电源的控制电路结构复杂，输出纹波较大，在有限的时间内实现比较困难。

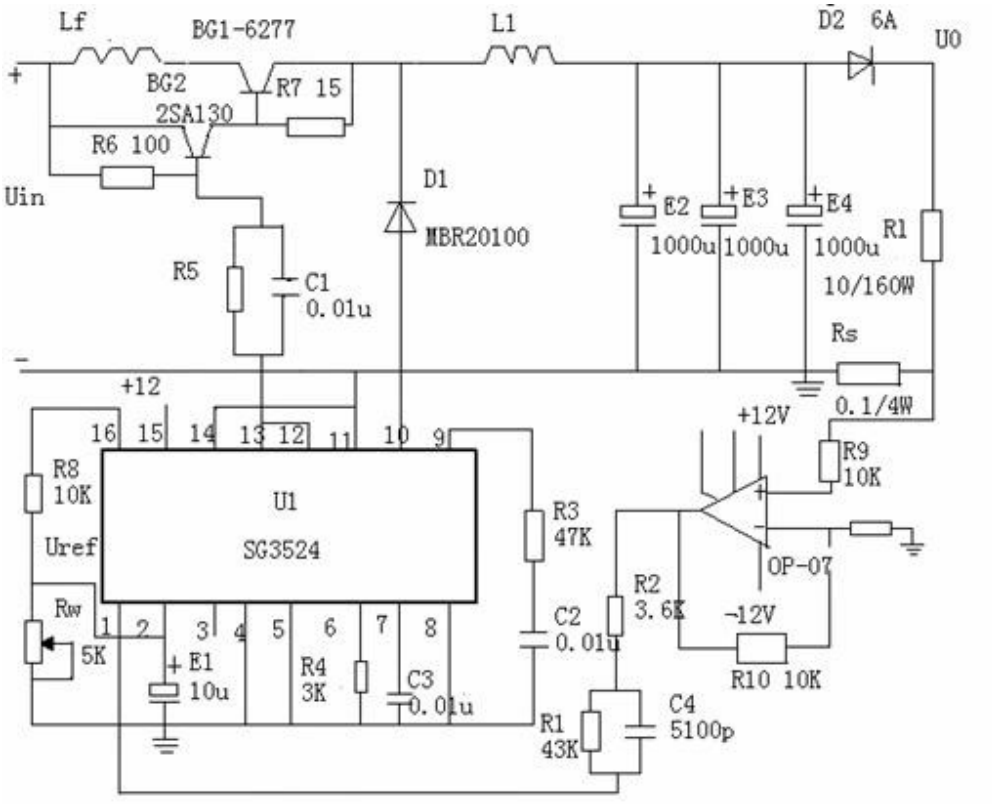

图 2: 采用开关电源的恒流源

2.2　采用集成稳压器构成的开关恒流源

系统电路构成如图 2 所示。MC7805 为三端固定式集成稳压器，调节 RW，可以改变电流的大小，其输出电流为：IL=(UOUT/RW)+I4，式中 I4 为 MC7805 的静态电流，小于 10mA。当 RW 较小即输出电流较大时，可以忽略，当负载电阻 RL 变化时，MC7805 改变自身压差来维持通过负载的电流不变。

优点：该方案结构简单，可靠性高　　缺点：无法实现数控。

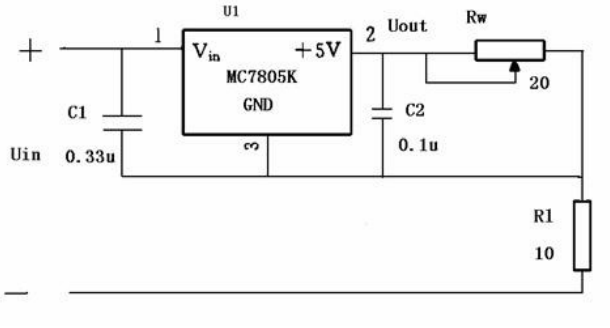

图 3: 采用集成稳压器构成的开关恒流源

3. 采用单片机控制的恒流源

该方案恒流源电路由 N 沟道的 MOSFET、高精度运算放大器、采样电阻等组成，其电路原理图如图 3 所示。利用功率 MOSFET 的恒流特性，再加上电流反馈电路，使得该电路的精度很高。

该电流源电路可以结合单片机构成数控电流源。通过键盘预置电流值，单片机输出相应的数字信号给 D/A 转换器，D/A 转换器输出的模拟信号送到运算放大器，控制主电路电流大小。实际输出的电流再通过采样电阻采样变成电压信号，A/D 转换后将信号反馈到单片机中。单片机将反馈信号与预置值比较，根据两者间的差值调整输出信号大小。这样就形成了反馈调节，提高输出电流的精度。本方案可实现题目要求，当负载在一定范围内变化时具有良好的稳定性，而且精度较高。

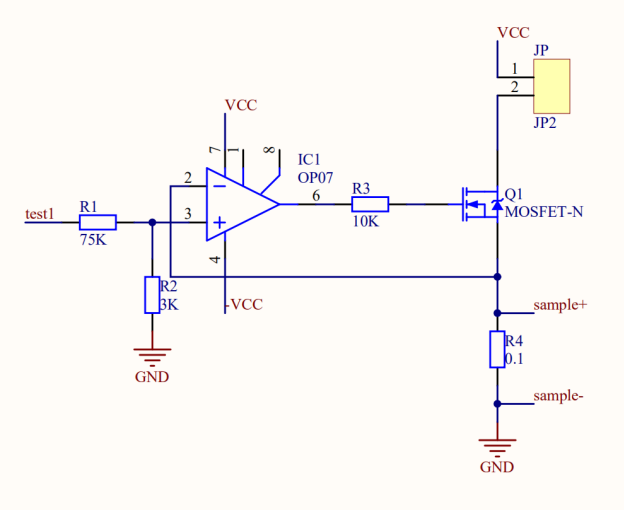

图 4: 采用单片机控制的恒流源

4. 方案描述

该数控恒流源电路主要包括按键输入、LCD 显示屏输出、单片机、RC 滤波电路、压控恒流源电路、电压采样电路几个模块。

系统工作原理为：通过按键对电流值 S_CURRENT 进行预置，并显示在 LCD 显示屏上，单片机根据预置的电流值输出一定占空比的 PWM 波，通过 RC 滤波电路后输出直流信号。直流信号输送至压控恒流源模块，通过电阻分压后输送至大功率场效应管 IRF640，产生相应的电流值。场效应管的漏极电流即为恒流源的实际输出电流。场效应管的漏极电流近似于源极电流，在电压采样模块，场效应管的源极电流经过采样电阻后转化为电压信号，经过差分放大电路后输送至单片机，单片机采集此信号，作出相应的调整处理后输出显示在 LCD 显示屏上，作为电流源的自测表的输出值 N_CURRENT，实际的输出电流通过电流表测量得到。

二、 理论分析与计算

1. 理论分析

我们组的电路主要分为 3 个部分，恒流控制电路，采样放大电路，单片机电路，这里主要对恒流控制电路和采样放大电路进行介绍分析。

恒流控制电路主要用于控制一个恒定的电流进行输出，首先根据你设定的值控制单片机输出一个占空比确定的 pwm 波，pwm 波经过 RC 滤波变为一个恒定的电压，也就是经过 pwm 输出电路，此时这个确定的电压值在经过分压进入恒流控制电路，这里进行分压的原因主要由于我们所取的采样电阻比较小，因此必须控制输入的电压值，经过确定分压比之后的电压再进入运算放大器，根据理想运放的特性我们可以指导，2、3 两脚的电压值近似相等，而 2 脚又与采样电阻相连，因此此时完整的电压进入采样电阻，采样电阻另一端又接地，因此此时采样电阻上的电流即为分压后的电压除以采样电阻的阻值，而根据 mos 管的特性，运算放大器和 mosfet 相连出来的电流可以近似忽略，因此采样电阻上的电流值与我们的输出接口 JP JP2 上的电流值相等，而据此可知，我们可以控制 pwm 波的占空比来线性地控制电流。

而电压采样电路主要起到一个放大反馈的作用，采样电阻上的电压进入我们的电压采样电路，而根据电压采样电路上的电阻阻值，我们可以确定一个恒定的放大比，在这个电路中，我们把 R7、R8 的值设定为相等，R9、R10 的值也设定为相等，最后得到的放大比例即为两者之比，放大后的电压后输出到 AD 输入电路，此时对放大后的电压进行 RC 滤波，之后将这个滤波后的电压输入到单片机的 Adin, 单片机根据输入进来的 Adin 可以进行数模转化，而数模转化比较后则可以对单片机的 pwm 占空比进行一个调节，以此来达到一个反馈调节的功能。

至于电源模块，LCD 电路我们则是根据老师所给的单片机学习板而设计，两者相吻合以此来保证正确性，我们也适当删减了一些单片机学习板上一些没有必要的电路。

恒流控制电路和电压采样电路我们主要参考了老师放在群里的资料。

1.1　理论计算

电路中主要需要计算的是恒流控制电路和电压采样电路的一些电阻值。

这里我先对恒流控制电路上的分压比进行计算，一开始在不知道采样电阻只能是 0.1 欧姆之前，我们是按照 0.15 欧姆来进行计算的，不过此处我就根据 0.15 欧姆的思路给出 0.1 欧姆的计算思路，首先是分压比调控，因为 pwm 波的占空比只能在 256 个值之内进行调节，因此我们取了一个比较好的占空比也即为 200，因此我们设定它的满量程也即是实验要求 20mA 到 1000mA，满量程对应此，1000mA 在采样电阻对应 0.15V，而 pwm 最大的电压量为 5V，因此 5V*k*200/256=0.15，据此我们可以得到 k=0.0384, 而根据标称电阻值，我们得到了两个电阻值是比较符合的，3K 欧姆和 75K 欧姆，3/75+3 我们可以得到也为 0.384，因此这就是我们在横流控制电路上的计算。

然后我们对于电压采样电路上的电阻值进行计算，此处我们采用了相同的电阻值进行控制，也就是 R7=R8=3K, 而 R9=R10=75K，而根据运放的特性我们可以得到放大的比例为 R9/R7=25，因此我们对于采样电阻上的电压放大了 25 倍。之后的一些电容值我们根据老师所给的资料以及老师之后的建议我们进行了调节，最后所得到的电容值如之前的电路图所示。

此时我们已经可以完整的量程范围 (20, 1000)mA, 而采样电阻上的电压值的范围为 (3,150)mV, 再到分压前的电压为 (78,3900)mV, 最后根据此得到占空比 (,0.0156,0.78), 对应有占空比的分子 (4，200)。

而在最后电路板的焊接过程中，我们又进行了调节，采样电阻改为了 0.1 欧姆，而在实验室的柜子中 75K 欧姆的电阻是不存在的，因此我们采取了 100K 欧姆的电阻进行了代替，当然根据我们电路板的测试中，在 Adin 电路和 pwm 输入电路中，本身就存在一定的压降，因此我们最后还是根据实际测出来 pwm 值，AD 值，以及实际电流值，来对我们程序内部的参数来进行设计。

三、 电路与程序设计

1. 硬件电路设计

1.1 各模块电路

(1) 总体电路

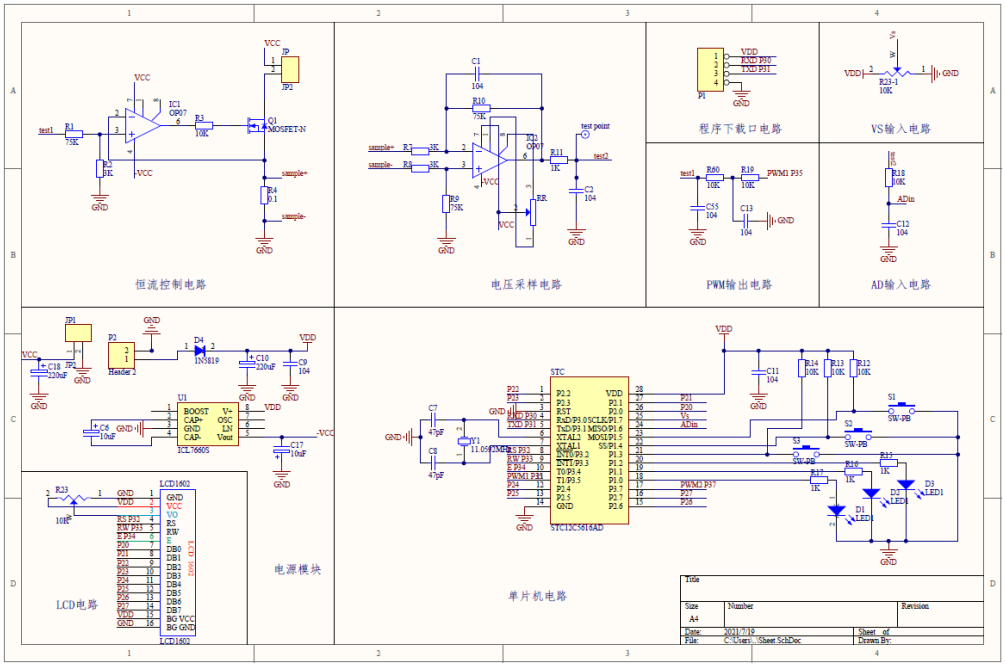

图 5: 总体电路

(2) 压控恒流源模块

压控恒流源是该系统的主要部分，通过电压来控制输出电流的变化。该压控恒流源电路由分压电阻 R1、R2，运算放大器 IC1，大功率场效应管 Q1，采样电阻 R4，电流输出模块 JP2 等组成。

压控恒流源电路的工作原理为：输入 RC 滤波电路得到的直流信号，经过分压后得到 U3 输入运算放大器，控制输出对应的电流 Io。运放采用 OP-07 作为电压跟随器，由于 U2=U3，场效应管 Id=Is，可以得到 Io=Is=U2/R53= U3/R53。从 Io=U3/R53 可以看出，电路输入电压 U3 控制输出电流 Io，即 Io 不随 RL 的变化而变化，从而实现压控恒流。

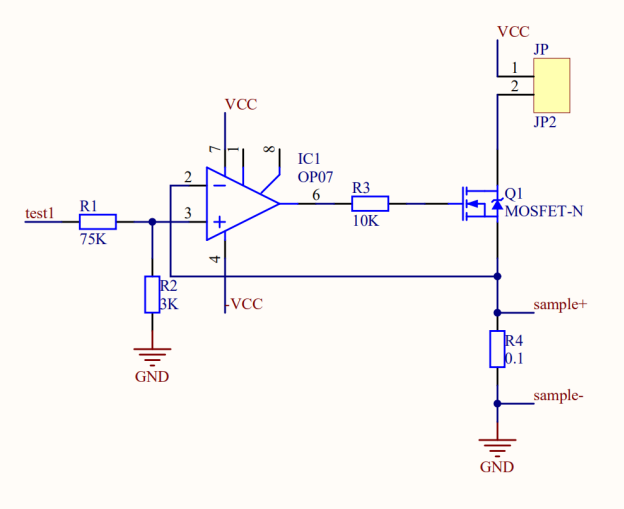

图 6: 压控恒流源模块

(3) 电压采样模块

电压采样模块是反馈部分。该电压采样模块采用差分放大电路，由运算放大器 IC2、若干电阻电容组成。电路原理图如图 7 所示。

其工作原理为：对采样电阻 R4 上的电压进行采样，经放大后输入单片机，由单片机内部的 ADC 模块转换为数字信号作为反馈，PID 调整输出的 PWM 波，实现负反馈。

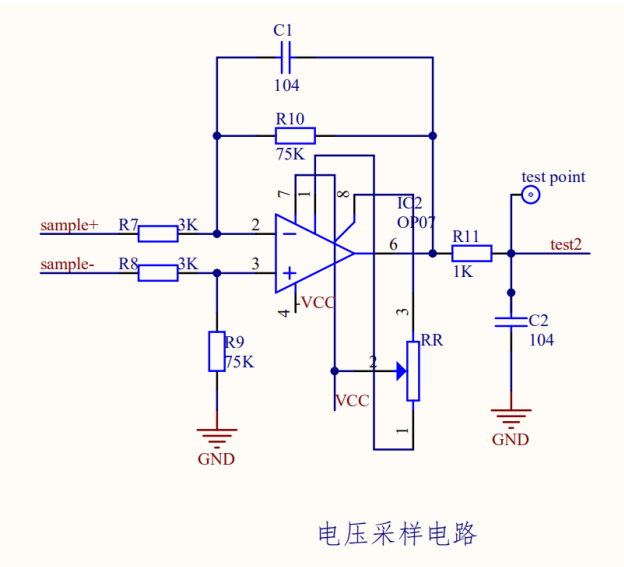

图 7: 电压采样模块

(4) 按键输入模块

该模块采用三个按键输入，按键 1 控制改变数字的位，按键 2 控制数字 +1，按键 3 为确认输入，将设定的数字输入单片机。按键采用防抖动设计。

(5) 单片机以及 A/D 转换模块

单片机为本系统的主控制器，本次设计采用型号为 STC1C5410AD 的单片机，指令周期短，工作速率快，功耗低，具有丰富的片上资源，可以在线下载，易于调试。

单片机自带的 A/D 转换模块为 10 位，取高八位为有效位，转换模块将输入的采样电压转换成数字信号反馈给单片机，单片机将此反馈信号与预置值比较，根据两者间的差值调整输出信号大小。这样就形成了反馈调节，提高输出电流的精度。同时，将采样回来的电压经过单片机处理传送到 LCD，可以显示当前的实际电流值。

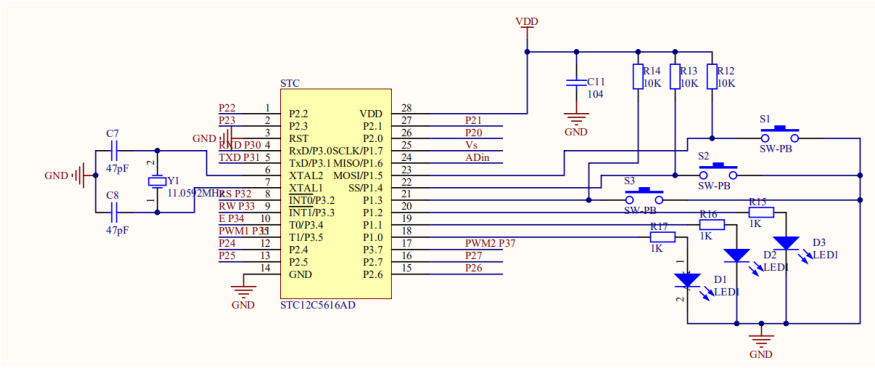

图 8: 单片机电路

(6) LCD 显示模块

LCD 显示屏型号为 LCD1602，第一行显示设定电流值 S_CURRENT，第二行显示单片机测量值 N_CURRENT。

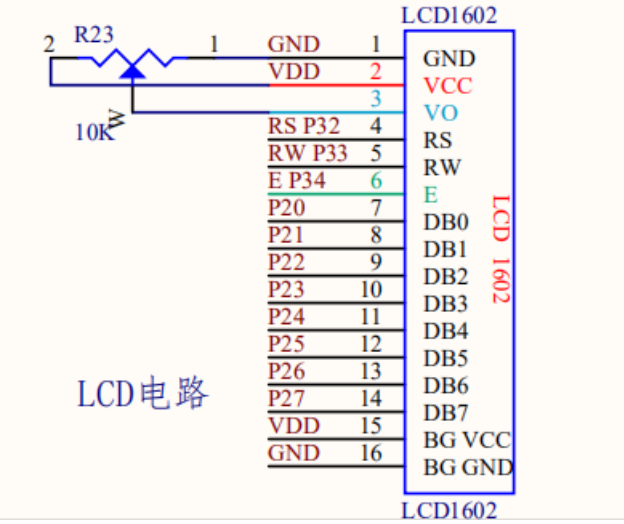

图 9: LCD 电路

(7) 电源模块

电源模块由学生电源提供 12V 直流，由 ICL7660S 生成-5V 的电压。

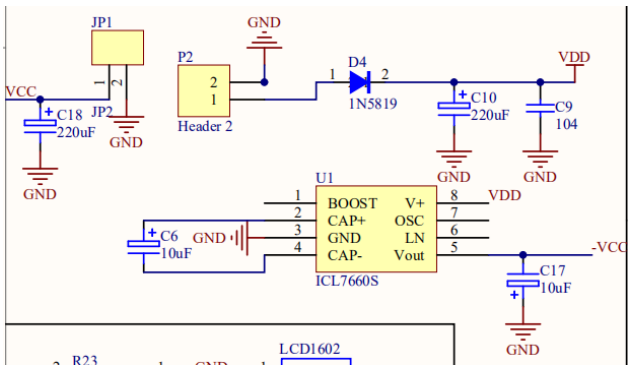

图 10: 电源模块

1.2　PCB 设计

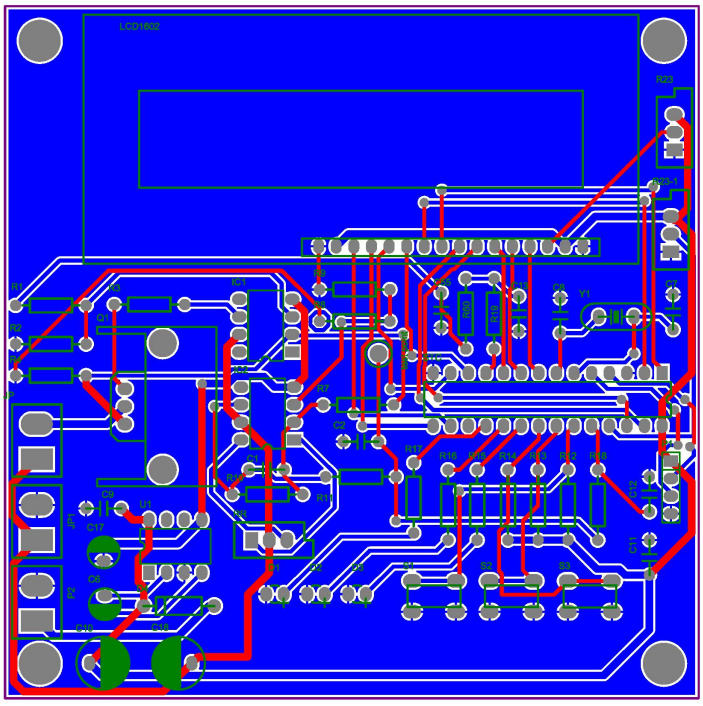

图 11: PCB 设计

2. 软件设计

2.1 流程图

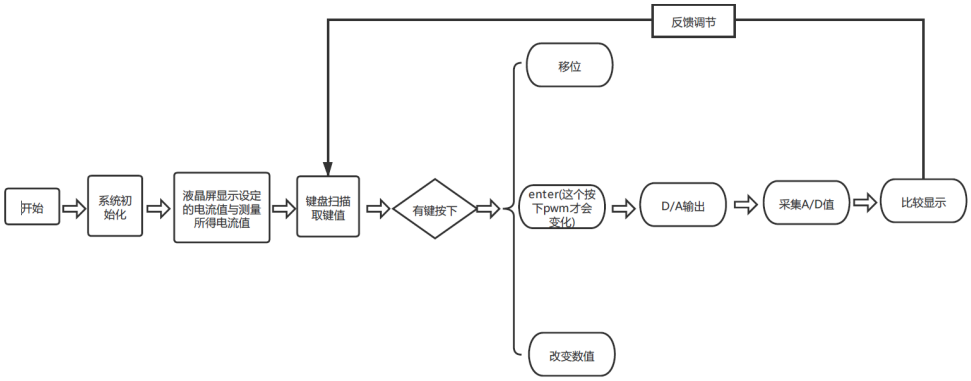

图 12: 流程图

2.2 模块核心代码

Listing 1: ADC 输入及转换为实际电流

```
        void ADC_Scan(){
        //开启ADC，通道六扫描
        ADC_CONTR = ADC_POWER | ADC_SPEEDLL | 6 | ADC_START;
        _nop_();                        //Must wait before inquiry
        _nop_();
        _nop_();
        _nop_();
        while (!(ADC_CONTR & ADC_FLAG));//Wait complete flag
        ADC_CONTR &= ~ADC_FLAG;         //Close ADC
        //将10位转换结果存入Adc_resuit变量
        Now_Current = ADC_DATA*4+(ADC_LOW2 && 0x03);

        //将电流转换为实际电流值
        Now_Current = Now_Current/0.65;

            }
```

Listing 2: 按键操作及 PWM 控制

```
void    Key_treat()
{
    if (Key3_press_flag)
    {
        //将预设值传入PWM
        result = Set[3]*1000 + Set[2]*100 + Set[1]*10 + Set
            [0];
        //将按键输入结果转化为PWM控制值
        result = 255 - result/gain;
        Key3_press_flag = 0;
    }

    if (Key2_press_flag)
    {
        switch (target)
        {
        case 0: Set[3-target] = (Set[3-target] + 1)%2;
            break;

        default: Set[3-target] = (Set[3-target] + 1)%10;
            break;
        }
        Key2_press_flag = 0;
    }

    if (Key1_press_flag)
    {
        target = (target+1)%4;
        Key1_press_flag = 0;
    }

}
```

四、 测试方案与测试结果

1. 测试步骤

(1) 单片机 PWM 波输出占空比设置为 50%，改变负载，测试恒流控制模块是否能达到恒流控制效果

(2) 断开 PWM 波反馈回路，测量单片机 ADC 输入，得到输出电流与 ADC 转换结果之间的关系，校准参数

(3) 连接 PWM 反馈回路，设定不同的电流值，计算得到 PWM 占空比与电流之间的关系，调整代码

2. 测试结果

2.1　电流范围与步进调节

设定值 (mA)	20	30	40	50	500	1000
测定值 (mA)	19.7	29.4	39.1	48.4	498.5	1003.1
显示值 (mA)	20	26	38	44	493	998

2.2　负载电阻改变电流变化测量

设定 500mA		
接入电阻	第一次测量	第二次测量
0 欧姆	498.5mA	500.1mA
7.5 欧姆	493.1mA	495.5mA

2.3　纹波电流测量

接入 7.5Ω 电阻，将示波器接在电阻两端，得到文波电流为 2mA

3. 数据处理

设定值 (mA)	20	30	40	50	500	1000
测定值 (mA)	19.7	29.4	39.1	48.4	498.5	1003.1
显示值 (mA)	20	26	38	44	493	998
设定与测量差值	0.3	0.6	0.9	1.6	1.5	-3.1
测量与显示差值	-0.3	3.4	1.1	4.4	5.5	5.1

设定 500mA		
接入电阻	第一次测量	第二次测量
0 欧姆	498.5	500.1
7.5 欧姆	493.1	495.5
变化值	5.4	4.6

4. 数据分析

4.1 电流范围与步进

由上述表格可知，我们能够通过设定不同的值使输出电流在 20mA 1000mA 的范围内变化，步进至少能达到 10mA 的精度，达到了基础部分对步进的要求，同时电流可调范围达到了提升要求部分。

4.2 输出电流与给定值偏差

由上述表格可知，设定不同的输出电流时，测量电流与设定值之间的最大差值出现在设定 1000mA 时，为 3.1mA，远远小于要求中的设定值的 1%+10mA

4.3 显示输出与电流差值

由上述表格可知，显示值与测量差值之间的误差最大发生在设定电流为 500mA 时，差值为 5.5mA，要求允许的最大差值为 $(1000 \times 1\% + 3)mA = 13mA$, 满足要求

4.4 改变负载电阻

我们将电流输出设定为 500mA，由表三可知，空载时输出电流与接入负载后输出电流变化值在 2mA 左右，而要求中的变化值为 3mA，符合系统设计要求

4.5 文波系统

测量出的文波电流为 2mA，达到了基础要求中的 5mA，但是没有达到提升部分中的 1mA 要求

五、 实验的心得体会

1. 王瑜昕

这次的小学期实验令我收获很大，学期内的电子电路设计实验因为当时选了一个比较容易的题目，，用 arduino 做一个不同量程的电容测量仪，如果不算 arduino 和 lcd1602, 整个电路图只有一些电阻电容，非常的简单，因此也可以料想到之后的设计电路图和 pcb 版图绘制比较简单，同时当时的实验老师对于 pcb 版图的绘制线的绘制覆铜等的要求都是比较低的，因此，这次的小学期实验反而是更令我完整的走了一遍从设计到制作的流程。

在这次的小学期中遇到了许许多多的问题，但也慢慢的将他们都解决了，首先我们的电路设计总体来说还是比较顺利的，因为老师已经给出了参考电路图，我们根据老师给出的参考电路图将其进行整合优化设计，但由于我们审题的不仔细，未注意题目要求是只有一个 12V 电压的供电，因此我们在设计时，默认了我们可以使用负电源，导致终稿提交时出现失误，同时在准备负电源的时候我们也找到了对应的芯片专门用于产生负电压，也知道了它也是有范围限制，同时在电路设计的时候也明白了滤波电容的重要性，一般来说，例如你的 pwm 波或者说之后进去的 ad_in 等，都是需要添加滤波电容来保证电源的稳定性，在设计过程中，对于电路分模块的理解进一步加深了。

这次的 PCB 版图设计与之前有所不同，这次的 pcb 版图对于覆铜和导线宽度是有要求的，而且 PCB 板子大小也进行了严格的规定，因此在手工布线的时候比较麻烦，同时由于电路图的修改，导致了 pcb 版图也进行了许多次的绘制，由于当时制作板子时，把 lcd1602 整一个放在了 pcb 版图中，导致了其余的元器件比较密集，因此在布线的时候相当麻烦，最终还是按照红竖蓝横这样来进行排线，同时添加了比较多的过孔，来保证线路的简洁性。

最令我感触深刻应该就是单片机编程和调试这一部分，首先一开始我们对于老师给的 demo 不是特别理解，最终是根据电路的测量结果和 stc 的产品说明书来进行一个理解，例如我们做的题目数控恒流源最主要的两个部分理解就是 pwm 占空比怎么控制，以及 ad_in 怎么进行模数转化，pwm 的控制我们通过调整程序内部的数值，然后测量单片机输出引脚上的电压值，以此来判断如何控制，然后便是模数转化，通过阅读 stc 的产品说明书，以及老师的 demo，我们对于模数转化也有了一个了解，之后便是编程，我们设置了三个按键，一个是位移键，另一个是数据调解键，最后一个是 enter 键，我们主要是根据 main5 这份程序进行改编编写我们的程序，首先我们通过 key_treat 来处理我们的三个按键，然后我们将输入进去的值，对应于 pwm 波占空比的变化，以此来改变占空比，之后我们通过高精度的电流计来测量我们的电流值的变化，我们发现确实是基本符合线性变化，算出其中的斜率值，截距，我们又对其进行了一个反馈调节。唯一可惜的是步进 1mA, 当时我们并没有想到可以对 pwm 再进行一次 pwm 调节，因此没能更高精度做到 1mA 步进有点可惜。随后便是纹波电流的调节，我们接了一个大功率的 7.5 欧姆的电阻，通过示波器测量它上面的电压值变化，再计算它上面的电流值变化，一次来进行纹波电流测量，在程序中，我们主要通过 pid 调节来控制它的反馈，根据测量，我们发现我们的纹波电流能满足 5mA 的基本要求，但要求更高的 1mA 还是可惜未能达到。

值得一提的是，我们在单片机调试过程中，发现它的 ad_in 输出一直是 0000，根据电压测量原来我们接入的是负电压，然后我们发现只要是负电压，它的模数转化就必然是 0，其实我们在一开始电路原理图设计的时候就已经发现输出是负电压，但当时就根据资料里的来，而未改变结果，所幸最后我们的采样电阻两端的电阻接的比较近，因此我们通过改变连接方式来使其输出正电压，我们先将之前焊接的电阻取下来，然后将 R7 的一端接到 R8，R8 的一端接到 R7，最终解决了这个问题。

这次的小学期实验令我特别有收获，对于 AD 软件的使用变得更加熟练，对于 PCB 的绘制变的更加得心应手，对于单片机有了一个初步的了解，同时上个学期未使用的吸焊枪现在也能够熟练使用，这是一个非常有收获的小学期。

2. 周灿松

在本次电子电路综合设计课程中，我们自主地设计了一个数控的恒流源。老实说，在最开始选定题目之后，我心中是非常忐忑的，在学期中我们电子电路设计实验二虽然也设计制作了一个倒计时计时器，但是当时老师给出了大致的电路图、以及近乎完整的代码，我们做的更多的是在老师给的参考文件的基础上把电路实现出来。但本次课程却并不是如此，从电路到代码都需要我们自主设计，仅有部分参考电路，同时老师对 PCB 板的要求也比学期中高了很多，难免会有一些彷徨。

在前期的电路设计过程中，我们遇到了一系列的问题。其中遇到的最大的问题便是我们没有注意到题目要求中的只能 12V 供电，我们将其理解成了可以利用正负 12V 的电源进行供电，所以在电路设计时我们选用了正负 12V 的学生电源。而且我们有一点心急，很快的把 PCB 板的连线完成了，等到老师指出问题时，我们只能在修改电路后重新进行布线。由于 OP07 运算放大器需要加上一个负电压的偏置，所以在查阅了治疗后我们选用了一块 ICL7660S 芯片来产生一个伏电压，本来我们是准备利用输入的正 12V 电压产生一个负 12V 的电压，但是在给老师检查了设计后，老师指出了这块芯片无法产生 12V 这么高的负电压。在进行了权衡之后，我们认为偏置电压不一定需要达到 12V 这么大，随机利用驱动单片机的 5V 电源产生了-5V，最终问题得到了解决。

此次的 PCB 绘制过程也与电子电路设计实验二有了很大的区别，因为当时的电路只需要跑小电流，所以对 PCB 板的线宽是没有要求的，布通就行。但这次不一样，因为电路是作为一个数控恒流源使用的，所以有一部分线需要加宽以使其能够承受更大的电路，这一点也给布线带来了一定的难度，不能像以前一样十分紧凑地布局，需要给走大电流的线留出足够的空间走线。除了遇到的困难，这次的 PCB 绘制过程也让我学到了很多的关于布线的知识，利于分区域进行布线、将覆铜层作为 GND 减少布线难度……同时也填补了理论电路设计的空缺，以前一直没意识到滤波电容不能离得太远，这一点在理论知识学习时是没有了解到的。

软件设计难度较低，主要难度来自对单片机的不熟悉，刚上手时不知道从何开始，这种深入底层的编程体验也是前所未有的，让人感受到了直接操作寄存器的痛快感。但是因为单片机性能有限，我以前奔放的编程方式也受到了挑战，因为定义了过多的变量，最后导致用尽了单片机可直接寻址的存储单元，在进行精简后才成功解决这一问题。

调试时在差分放大电路遇到了一个重大事故：单片机的 ADC 输入管脚无法识别我们输入的电压。这一问题成因是在电路设计时我们差分放大的输入反向了，最后放大的电压是一个负电压，因为单片机只能识别 0-5V 的电压，所以就导致无法识别了。因为设计电路时脱离了单片机的性能，我们忽视掉了这一问题，也酿成了这一悲剧结果。不幸中的万幸是 PCB 设计时采用了分模块布线，我们可以直接修改电阻的排布方式改变连线，如若不然，可能还需要割线，又会麻烦很多。

此次课程从电路图设计到 PCB 设计，再到软件编程、直至最后的调试测试。虽然问题很多，但是在解决问题的过程中也的确学到了很多的知识，也吸取到了不少的经验教训。首先、在拿到一个题目时，应当认真地理解要求所表达的意思，避免因为会错提议导致重新设计；其次、电路设计时不能忽视任何一个可能出现问题的点，往往一个很小的错误都会导致最后功能无法正常实现。此次差分放大电路的负电压通过调整电路接线解决了一定程度上是运气比较好，下一次再因为忽视细节导致同样问题发生便不一定能够这么幸运了；然后、在设计时应该充分阅读器件手册，了解元器件性能后再动手设计，不能想当然地认为他能做到；最后，在单片机编程时，慎重使用全局变量，因为单片机可直接寻址的内存单元有限，肆意挥霍极易导致存储单元不够用。

3. 朱夏瑜

在本次的小学期课程中，我们完整地接触到了从电路设计到 pcb 制作，再到调试电路、实现功能的一个过程，其中遇到了很多的困难，也有很多的收获。

我们组的选题是“数控恒流源”，在确定选题后，就开始了电路设计。在仔细阅读了老师给出的参考资料后，我基本了解了恒流源电路的工作原理，根据设计所需精确度的要求，我们选择了用单片机控制的数控直流恒流源电路，其中恒流源模块采用压控恒流源，通过电压来控制电流的变化。在设计电路时，要明确题目所给的限制条件，我们在设计时没有注意到只能采用 12V 直流电压的条件，而采用了 ±12V 电压串联，最后是由 5V 电压通过 ICL7660S 产生-5V 的电压加在运算放大器两端来解决。在用 AD 绘制电路图的时候，学习到了一个小技巧，就是按模块绘制电路，还有就是通过给相连的电线分别做上相同的标记，可以代替中间的线，减少了连线过程中的弯绕，使得电路图更加简洁明了，也便于后续 PCB 版图绘制时的器件摆放和连线。此外，在电路设计时，一定要弄明白每一个器件的作用，不然就会造成器件冗余的情况。在后续的 PCB 版图绘制中，也学习到了器件分模块放置的重要性，一般要将电源模块放置在板子的左右两侧；还要根据流经电流的大小来调整走线的粗细；在我们的实验中，用到了带有散热片的 MOS 管，但是因为一开始没有考虑到散热，器件布置的过于紧密了，对后续的测量产生了一定的影响；器件在放置时也要考虑后续测量的便捷性。在 AD 绘图中，也学习到了很多快捷键的使用方法，使得绘图更加的快速便捷。为了 PCB 版图更加清晰明了，一般采用横着的线用一种颜色，竖着的线用一种颜色，遇到走不通的地方，可以用过孔连接。

拿到板子后，就是编程和调试过程了。我们用的单片机型号是 STC1C5410AD，显示屏的型号是 LCD1602。老师给出了很多的 demo 供我们参考，每一个 demo 的开头也列出了具体的功能，方便阅读和理解代码。根据对电路的理解，我们设置显示屏显示设定恒流电流值和输出电流值，并通过三个按键来实现电流值的预置功能，一个是选择位，一个是步进 1，一个是确认输入。显示输出电流值较为麻烦，需要通过 PWM 调制和 AD 转换来实现。首先通过测量来确定实际电路板的反馈电压 AD_IN、PWM 占空比和设定电流值的关系，我们发现三者基本是线性的，可以根据范围乘上一个系数来匹配，并根据电压采样得到的反馈电压做一个反馈调节。在调试的过程中，我们遇到了由于器件布置的过于密集导致散热较慢，数据不稳定的现象，根据调节系数来降低发热带来的影响。此外，我们还遇到了一个问题，是由于线连反导致反馈模块的输出电压是负的，从而输入单片机的电压也是负的，单片机自动将其置为 0，从而显示的输出电流一直为 0000 不发生改变，在发现这个问题后，我们使用交换两个电阻的位置的方法代替割线完美的解决了这个危机，这也很大程度上得益于器件的模块化放置，使得两个电阻距离很近，而这两个电阻恰好是模块的输入端，调节起来非常地方便。但是，也提醒了我们在电路设计时也要注意连线的正负这样的小细节，从而避免不必要的麻烦。

此次的小学期学习，完成了一次完整的课题，从方案设计到答辩，到电路板制作、编程调试，我从中学习到了软件的使用、参考资料查询学习的方法、还有调试的方法等等，收获了很多。

六、 完整代码

最后附上完整代码

Listing 3: 完整代码

```
#include <reg52.h>
#include <STC12C.h>
#include <intrins.h>
#include <stdio.h>
#include <math.h>


//变量类型标识的宏定义
#define Uchar unsigned char
#define Uint unsigned int
#define uchar unsigned char
#define uint unsigned int

#define gain 4.57718
#define DataPort P2     // 数据端口
#define Busy    0x80

//接口定义
sbit    Led1 = P1^0;
sbit    Led2 = P1^1;
sbit    Led3 = P1^2;

sbit    Key1 = P1^5;
sbit    Key2 = P1^4;
sbit    Key3 = P1^3;

// 控制引脚定义，不同的连接必须修改的部分
sbit RS=P3^2;
sbit RW=P3^3;
sbit EN=P3^4;

#define rs RS
#define rw RW
#define en EN

//ADC相关变量
#define ADC_POWER   0x80            //ADC power control bit
#define ADC_FLAG    0x10            //ADC complete flag
```

```
#define ADC_START   0x08            //ADC start control bit
#define ADC_SPEEDLL 0x00            //420 clocks

/*按键相关变量*/
bit Key1_press_flag;    //Key1按下标志
bit Key2_press_flag;    //Key2按下标志
bit Key3_press_flag;    //Key3按下标志

bit Key1_back;      //按键的上一状态
bit Key2_back;
bit Key3_back;

//电流预设值修改显示相关
int result;
int Set[4] = {0,0,0,0};//电流预设值
int target; //选择修改位

//ADC输出控制PWM变量
Uint PWM_Ctrl = 0;
Uint Now_Current = 0;

//LCD显存
uchar   Dsp_buf[16];
uchar   Dsp_buf2[16];

//定时计数器
uint Timer_Count;

void InitADC();
void ADC_Scan();
void Key_Scan();
void Key_treat();
void LcdWriteData( char dataW );
void LcdWriteCommand( Uchar CMD );
void Lcd_init( void );
void Disp_XY( char posx,char posy);
void DispOneChar(Uchar x,Uchar y,Uchar Wdata);
void Display (void);
void Pwm_init(void);
void Pwm_treat();
void system_init(void);
void timer0_int (void);
void Delay(uint y);
```

```
83  /*********************延时函数*************************/
    //延时y ms
85  void    Delay(uint y)
    {
87      uint x;
        while (y--)
89      {
            x = 1000;
91          while (x--);
        }
93  }

95  /******************系统初始化***************************/
    void system_init(void)
97  {
        //定义端口输入输出,p1^0-P1^2为推挽输出, p1^3-P1^7为输入, 设置PxM0,
           PxM1,
99      P1M0 = 0xf8;         //#11111000b
        P1M1 = 0x07;         //#00000111b
101
        //T0,模式设定
103     TR0 = 0;          //停止计数
        ET0 = 1;          //允许中断
105     PT0 = 1;          //高优先级中断
        TMOD = 0x01;      //#00000,0001,16位定时模式
107
        TH0 = 0;
109     TL0 = 0;
        TR0 = 1;          //开始运行
111
        P1 &= ~0x07;      //清除LED
113 }

115 void timer0_int (void) interrupt 1
    {
117     TH0 = 0xfc;
        TL0 = 0x67;          //1ms定时中断
119
        Key_Scan();
121
        ++Timer_Count;
123
```

```
    if ((Timer_Count & 0x3f) == 1){
        ADC_Scan();
    }

    if (Timer_Count > 500)
    {
        Timer_Count = 0;
        Pwm_treat();
    }
}


//按键相关函数

/********************* 键盘扫描函数************************/
void    Key_Scan()
{
    if (!Key1)          //开机键扫描，检测下降沿
    {
        if(Key1_back)
        {
            Key1_press_flag = 1;
        }
    }

    if (!Key2)          //开机键扫描
    {
        if(Key2_back)
        {
            Key2_press_flag = 1;
        }
    }

    if (!Key3)          //开机键扫描
    {
        if(Key3_back)
        {
            Key3_press_flag = 1;
        }
    }

    Key1_back = Key1;
    Key2_back = Key2;
```

```
167     Key3_back = Key2;

169 }

171 /******************** 按键处理 **************************/
    void    Key_treat()
173 {
    if (Key3_press_flag)
175 {
        //将预设值传入PWM
177     result = Set[3]*1000 + Set[2]*100 + Set[1]*10 + Set[0];
        //将按键输入结果转化为PWM控制值
179     result = 255 - result/gain;
        Key3_press_flag = 0;
181 }

183 if (Key2_press_flag)
    {
185     switch (target)
        {
187     case 0: Set[3-target] = (Set[3-target] + 1)%2;
            break;
189
        default: Set[3-target] = (Set[3-target] + 1)%10;
191         break;
        }
193     Key2_press_flag = 0;
    }
195
    if (Key1_press_flag)
197 {
        target = (target+1)%4;
199     Key1_press_flag = 0;
    }
201

    }
203
    //初始化ADC寄存器
205 void InitADC(){
    ADC_DATA = 0; //将ADC结果寄存器清零
207 ADC_CONTR = ADC_POWER | ADC_SPEEDLL;
    Delay(2);
209 }
```

```
211 void ADC_Scan(){
    //开启ADC，通道六扫描
213     ADC_CONTR = ADC_POWER | ADC_SPEEDLL | 6 | ADC_START;
    _nop_();                          //Must wait before inquiry
215     _nop_();
    _nop_();
217     _nop_();
    while (!(ADC_CONTR & ADC_FLAG));//Wait complete flag
219     ADC_CONTR &= ~ADC_FLAG;          //Close ADC
    //将10位转换结果存入Adc_resuit变量
221     Now_Current = ADC_DATA*4+(ADC_LOW2 && 0x03);

223     //将电流转换为实际电流值
    Now_Current = Now_Current/0.65;
225
}
227
/*=========================================================
229 显示光标定位
=========================================================*/
231 void Disp_XY( char posx,char posy)
{
233 uchar temp;

235     temp = posx %0x40;// & 0x07;
    posy &= 0x01;
237     if ( posy==1)  {temp |= 0x40;}
    temp |= 0x80;
239     LcdWriteCommand(temp);
}
241
/*=========================================================
243 按指定位置显示数出一个字符
=========================================================*/
245 void DispOneChar(Uchar x,Uchar y,Uchar Wdata) {

247     Disp_XY( x, y );                //  定位显示地址
    LcdWriteData( Wdata );            // 写字符
249 }

251 /*=========================================================
初始化程序，必须按照产品资料介绍的初始化过程进行
```

```
253 ============================================================*/
    void Lcd_init( void )
255 {
        LcdWriteCommand(0x38);              // 显示模式设置(以后均检测忙信号)
257     Delay(2);
        LcdWriteCommand(0x0f);              // 开显示屏，管光标，光标不闪烁
259     Delay(2);
        LcdWriteCommand(0x06);              // 显示地址递增
261     Delay(2);
        LcdWriteCommand(0x01);              // 显示清屏
263     Delay(2);
        LcdWriteCommand(0x02);  //光标
265 }

267 /*============================================================
    写控制字符子程序: E=1 RS=0 RW=0
269 ============================================================*/
    void LcdWriteCommand(Uchar CMD)
271 {
        rs=0;           //选命令寄存器
273     rw=0;           //写
        DataPort=CMD;   //写命令
275     Delay(1);       //延时，让1602准备数据
        en=1;           //使能信号变化，写入数据
277     en=0;
    }
279

    /*============================================================
281 当前位置写字符子程序: E =1 RS=1 RW=0
    ============================================================*/
283 void LcdWriteData( Uchar Data )
    {
285     rs=1;           //选数据寄存器
        rw=0;           //写
287     DataPort=Data;  //写命令
        Delay(1);       //延时，让1602准备数据
289     en=1;           //使能信号变化，写入数据
        en=0;
291 }

293 /*********************显示函数*************************/
    void    Display (void)
295 {
```

```
    int n,m =0;
297 //将设定值输入显存
    sprintf(Dsp_buf,"S_Currute:%1d%1d%1d%1dmA" , Set[3] , Set[2] , Set[1]
       ,Set[0]);
299 //将AD采样值输入显存
    sprintf(Dsp_buf2,"N_Currute:%04dmA" ,Now_Current);
301 LcdWriteCommand(0x80);
    for(m=0;m<16;m++)
303 {
        LcdWriteData(Dsp_buf[m]);
305 }
    LcdWriteCommand(0x80+0x40);
307 for(n=0;n<16;n++)
    {
309     LcdWriteData(Dsp_buf2[n]);
    }
311 }

313 void Pwm_init(void)
    {
315 CCON = 0;                       //Initial PCA control register
                                    //PCA timer stop running
317                                 //Clear CF flag
                                    //Clear all module interrupt flag
319 CL = 0;                         //Reset PCA base timer
    CH = 0;
321 CMOD = 0x02;    //#00000010B    //Set PCA timer clock source as Fosc/2
                                    //Disable PCA timer overflow interrupt
323
    CCAP1H = CCAP1L = result;         //PWM1 port output 0% duty cycle
       square wave
325 CCAPM1 = 0x42;                   //PCA module-1 work in 8-bit PWM mode
       and no PCA interrupt

327 CR = 1;                         //PCA timer start run
    }
329
    void Pwm_treat()
331 {
    CCAP1H = CCAP1L = result+2;
333
    }
335
```

```
/*********************主 函 数**************************/
void main(void)
{
    system_init();
    Delay(100);
    Lcd_init();
    InitADC();
    Pwm_init();

    EA = 1;

    while (1)
    {
        Key_treat();
        Delay(50);
        Display();
        Disp_XY(target+10,0);
    }
}
```