****

课程设计报告书

**题目：简易数字电压表**

**学 院 机械与汽车工程学院**

**专 业 机械电子工程**

**学生姓名 邓朝阳**

**学生学号 201930030529**

**指导教师 徐静**

**课程编号 030101921**

**课程学分 2.0**

**起始日期 2022.2.21**

目录

[一、选题背景 1](#_Toc97752874)

[1.1 概述 1](#_Toc97752875)

[1.2 课程目的 1](#_Toc97752876)

[1.3 选题确定及其功能需求 1](#_Toc97752877)

[二、方案论证(设计理念) 2](#_Toc97752878)

[2.1键盘扫描： 2](#_Toc97752879)

[2.2数码管驱动 2](#_Toc97752880)

[2.3电压信号的采集 3](#_Toc97752881)

[2.4功能按键的确定 3](#_Toc97752882)

[三、过程论述 3](#_Toc97752883)

[3.1原理图设计 3](#_Toc97752884)

[3.1.1主控 4](#_Toc97752885)

[3.1.2 USB接口 5](#_Toc97752886)

[3.1.3调试器接口 5](#_Toc97752887)

[3.1.4矩阵键盘接口 6](#_Toc97752888)

[3.1.5数码管电路 7](#_Toc97752889)

[3.1.6采样电路 7](#_Toc97752890)

[3.2 PCB设计 8](#_Toc97752891)

[3.3印制与焊接 9](#_Toc97752892)

[3.3.1转印 9](#_Toc97752893)

[3.3.2显影 9](#_Toc97752894)

[3.3.3腐蚀 10](#_Toc97752895)

[3.3.4打孔 10](#_Toc97752896)

[3.3.5焊接 10](#_Toc97752897)

[3.4编写流程框图 10](#_Toc97752898)

[3.5编写程序代码 11](#_Toc97752899)

[3.5.1 主函数 11](#_Toc97752900)

[3.5.2 键盘扫描函数 14](#_Toc97752901)

[3.5.3 数码管轮换显示与单通道显示 18](#_Toc97752902)

[3.5.4 数码管单通道滚动显示 20](#_Toc97752903)

[3.5.5 A/D 转换 23](#_Toc97752904)

[3.5.6 中断处理函数 25](#_Toc97752905)

[3.6 代码仿真 26](#_Toc97752906)

[四、结果分析 27](#_Toc97752907)

[4.1问题与调整 27](#_Toc97752908)

[4.1.1读取到的电压值不准确 27](#_Toc97752909)

[4.1.2滑动变阻器失效 28](#_Toc97752910)

[4.1.3悬空端电压不稳定 28](#_Toc97752911)

[4.2效果演示 28](#_Toc97752912)

[五、课程设计总结 29](#_Toc97752913)

[5.1 原理图设计总结 29](#_Toc97752914)

[5.2 PCB布线 29](#_Toc97752915)

[5.3 腐蚀与焊接 30](#_Toc97752916)

[5.4 程序设计 30](#_Toc97752917)

[参考文献 32](#_Toc97752918)

**简易数字电压表**

# 一、选题背景

## 1.1 概述

本次的课程设计是使用PIC16F877A芯片以及其他元器件进行电路硬件及软件的设计，最终制作出一个简易的电压表，能够通过A/D转换将选定的芯片引脚上的电压信号转换为数字信号并通过数码管进行显示。

## 1.2 课程目的

1、增进对单片机的感性认识，加深对单片机理论方面的理解；

2、掌握单片机的内部功能模块的应用，如定时器/计数器、中断、片内存贮器、I/O口、A/D转换等；

3、了解和掌握单片机应用系统的软硬件设计过程、方法及实现，为以后设计和实现单片机应用系统打下良好基础。

## 1.3 选题确定及其功能需求

本组获得的选题为：简易数字电压表的系统设计。

需要满足的系统功能为：

1、测0~5V的8路（用A口的8路模拟口）输入电压值；

2、在LED数码管上8路通道轮流显示；

3、单路选择显示。

学生需要通过利用PIC16F877A微机芯片设计一个简易的数字电压表系统。由于该系统具有数码管、矩阵键盘、A/D转换功能，并且需要满足延时和流动显示等功能，学生能够综合地训练硬件与软件设计方面的素质。

本设计选题需要考虑以下几个主要的问题：数码管的显像问题、矩阵键盘的扫描问题、电压模拟量的A/D转换问题，在总体设计中还需要考虑电子元器件的选用安排、电路板的布线合理性、调试过程中的各种产生的问题的修复等等。

# 二、方案论证(设计理念)

本次的简易数字电压表的系统的需求是需要将电压这一模拟量转换为数字量并进行输出，即需要使用ADC对电压进行模拟量采样转换为数字量。

关键的芯片选取PIC16F877A芯片，该芯片具有丰富的外设资源，且PIC16F877A具有8路的ADC通道，可以进行八路电压的采样，使用定时器规定了ADC的采样频率，并借助数码管对采样结果进行显示，也使用了定时器对数码管实现进行定时显示等功能。本次采用此芯片实现简易数字电压表的系统是较为理想的选择。

通过课程设计任务书的设计工作要求，本次设计的简易电压表在软件上设计应该重点关注于键盘扫描、数码管驱动、电压信号的采集以及键盘相关功能按键的确定。

## 2.1键盘扫描：

本次设计采用了矩阵键盘，而确定矩阵键盘上哪个键按下的方法有两种：

1. 逐行逐列扫描法

逐列将列线依次置低电平，读取行线，如果某一条行线为低电平，则说明该行线与当前置为低电平的列线交叉点处的按键被按下，从而可以获取按键的键值。

1. 线反法

键盘的行线被拉高，列线被拉低。如果有按键按下，则某一条行线将被拉低，从而确定按键所在的行。然后，键盘的列线被拉高，行线被拉低。如果有按键按下，则某一条列线将被拉低，从而确定按键所在的列。

## 2.2数码管驱动

数码管是一种发光器件，内部由七个条形发光二极管和一个小圆点发光二极管构成，根据各段的组合不同，便能够显示出不同的字符。按照八个数码管的公共接线端不同，数码管可分为共阴极数码管与共阳极数码管两种，本次设计采用共阳极数码管，即数码管的公共端接高电平，当段选口为低电平时，对应的二极管便会发光。本次设计采用动态显示方法，即每一时刻，数码管仅显示一位，位与位之间交替显示，由余晖效应可知，当数码管的显示周期足够小时，在人眼看来数码管便像是同时点亮了四位数字。

## 2.3电压信号的采集

电压信号的采集使用到单片机中的模/数转换器，简称为 ADC；ADC 种类较多，较为常

用的是逐次比较型 ADC，实现原理为通过比较模拟量的输出，确定原设定位的标定值。

PIC16F877A 单片机拥有 10 位的逐次比较型 ADC，通过对被测模拟电压的采样，并将

其与基准电压进行比较，将相关的寄存器置 0 或置 1，最终得到一个 10 位的二进制数来描述输入的模拟信号。由于仅有 10 位二进制数用于描述模拟量，故 1023 为最大电压对应的数字量，本次设计电压的测量范围为 0~5V，故当使用数字量计算电压值时，计算公式为𝑉 = 数字量 ∗ 5/1023。

## 2.4功能按键的确定

在本次设计中，简易电压表一共有四种功能，分别为各通道轮流显示，单通道显示，单通道向左滚动显示，显示熄灭；因此在键盘功能按键分配上，A 键为功能 1，各通道轮流显示；B 键为功能 2，单通道显示，按下 B 后再按下 0~7 的数字便可在数码管上显示对应的通道电压；C 键为功能 3，单通道向左滚动显示，按下 C 键后，选择通道号，再按下 C 键便可在数码管上滚动显示对应的通道电压；D 键为功能 4，按下后数码管熄灭。

# 三、过程论述

本次设计共分为五个部分，分别为原理图设计、PCB设计、印制与焊接、编写流程框图、编写程序代码。

## 3.1原理图设计

根据简易数字电压表的设计任务书，在硬件设计方面，需要考虑的方面有主控、键盘、数码管、系统的供电等。根据课程提供的硬件以及需要考虑的 ，使用Altium Designer软件进行硬件原理图设计如下：

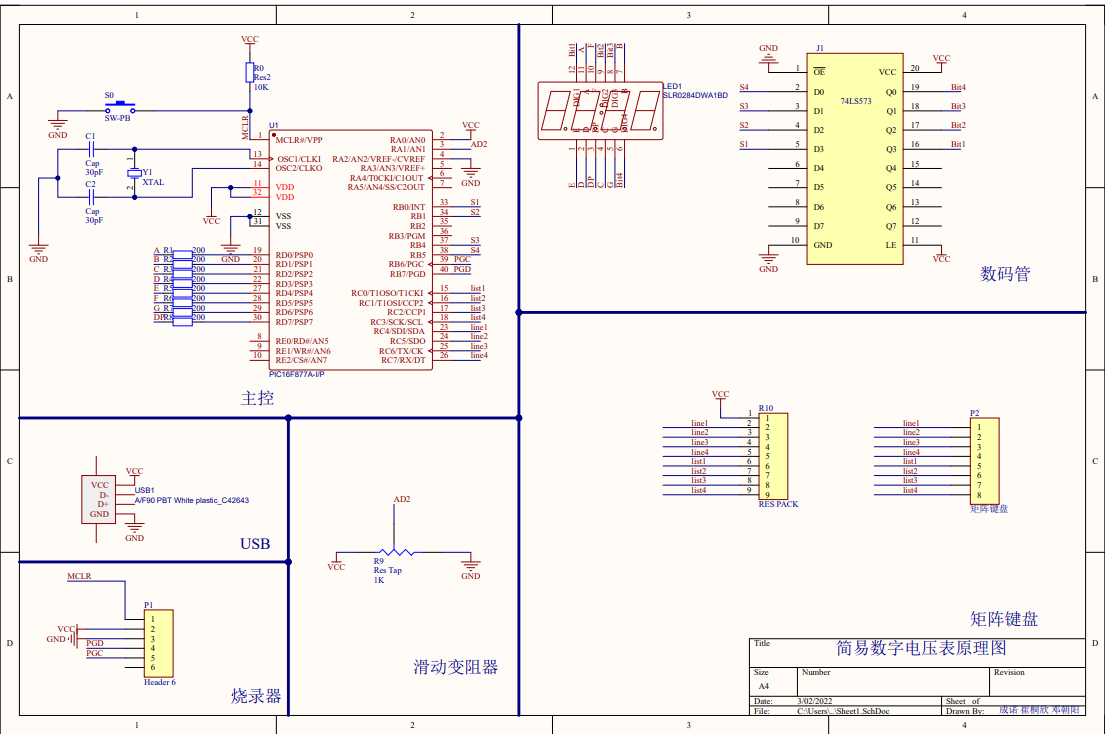


图3.1 原理图

### 3.1.1主控

芯片采用PIC16F877A，根据最小系统板的设计方法，首先为芯片添加晶振电路，并且在晶振与GND连线之间添加两个30pF的电容。其次对于复位电路的设计，根据资料可以确定当1脚为低电平时，单片机会复位，故1脚通过一个10K欧姆电阻接至VCC，通过按键接至GND，以实现通过按下按键将单片机复位。11脚与32脚接VCC，12脚与31脚接GND。对于AD采集的8个口，一个接至VCC，一个接至滑动变阻器的滑动端，一个接至GND。其余引脚的通过后续模块的设计再进行分配。

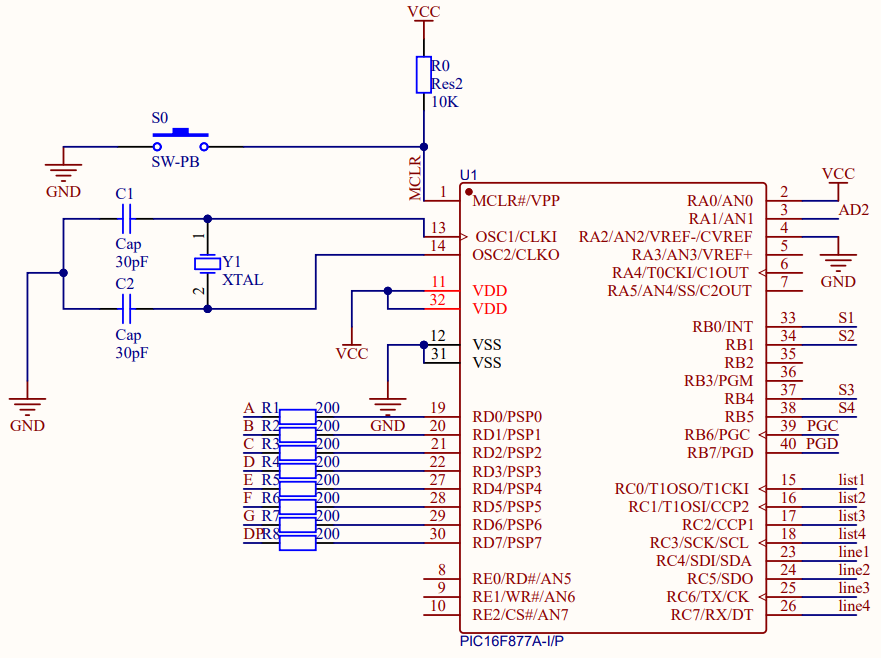


图3.1.1 主控

### 3.1.2 USB接口

在本课程设计中，USB模块仅当作供电输入口，故仅连接VCC与GND接口，其余引脚浮空。

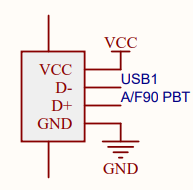


图3.1.2 USB接口

### 3.1.3调试器接口

烧录器的型号为PICkit2，共有六个插口，因此需要在硬件设计中为烧录器预留一个6脚的排针，根据烧录器的各插口的功能，将排针上对应的各引脚与PIC16F877A上对应的引脚相连接，其中烧录器上PGM插口没有使用到，因此排针的第6脚浮空，并没有与芯片相连接。

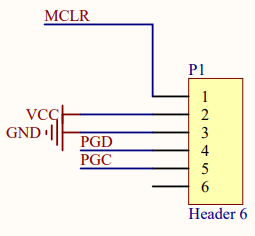


图3.1.3 调试器接口

### 3.1.4矩阵键盘接口

在此次设计中，键盘采用4X4行列式薄膜键盘，所以在硬件上预留一个8脚排针，当需要使用键盘时，仅需将键盘插入排针。为键盘分配引脚时，考虑到连续的I/O口对于后续程序的编写较为方便，因此将C口的8个引脚分配给键盘。

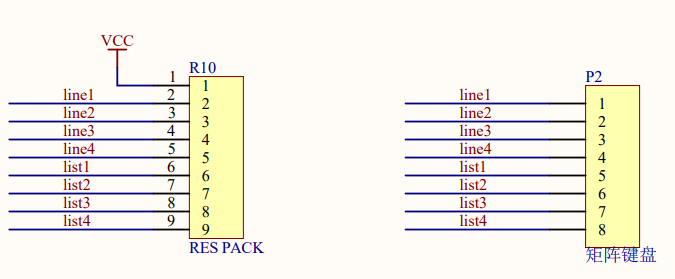


图3.1.4 矩阵键盘接口

### 3.1.5数码管电路

数码管需要考虑的资源分配有两个，分别是位选口与段选口。其中为段选口分配连续的I/O口有利于后续的程序编写，因此将仅剩的连续的D口分配给数码管的8个段选口，且每个段选口都经过一个200欧姆的限流电阻连接至数码管，防止电流过大损坏数码管。数码管的位选驱动选用74LS573芯片，芯片上用作位选的I/O口先连接至74LS573芯片的输入端，再将74LS573芯片的输出端与数码管的位选口相连。

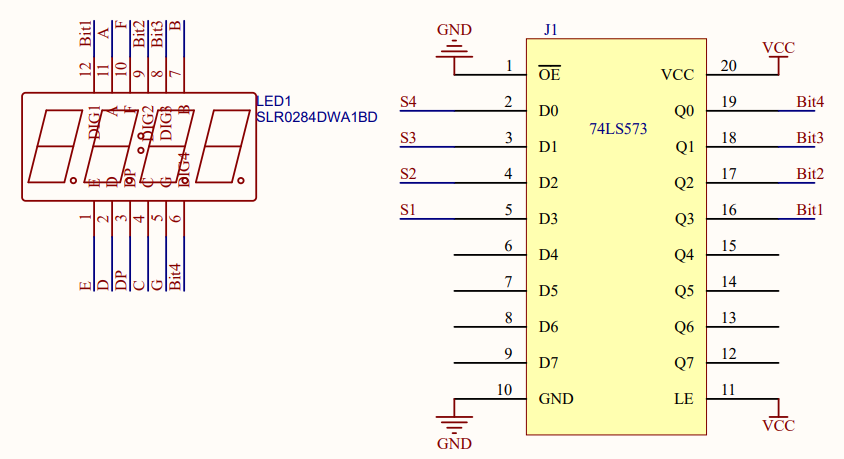


图3.1.5 数码管电路

### 3.1.6采样电路

由于本次课程设计的是一个简易电压表，故需要使用一个滑动变阻器，一端连接VCC，一端连接GND，一端连接其中一个AD采样输入口，通过调整滑动变阻器并读取该AD口采集到电压值判断程序是否正确。如图3.1.6

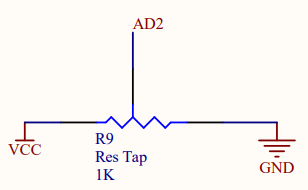


图3.1.6 采样电路

## 3.2 PCB设计

在进行pcb布板时，设计布板大小为100mm\*150mm，首先按照相关元器件的功能进行大致排布。USB作为供电模块，将其放置于板的边缘处；烧录器、键盘作为接入模块，因此将其对应的排针同样放置于板的边缘处；将芯片放置于中心偏旁边的位置，方便后续布线的同时为数码管腾出足够的空间；晶振模块放置于靠近芯片且远离其他元器件的位置，保证其运行不受到干扰；数码管模块则放置于右侧，预留出的空间有助于缓解后续的布线压力。

在进行布线时，首先将简单的信号线相连接，然后再连接复杂的信号线与电源线；但在布线过程中发现，有些引脚的布线无法实现，因此在已有的基础上对原理图部分引脚的连接顺序进行修改，保证其在PCB布线中能够顺利。

完成布线后，根据线路排版要求，电源线设置为60mil，信号线设置为25mil，管脚焊盘设置为80mil\*80mil，排针焊盘设置为80\*100mil，线与线、线与孔的间距设置为30mil，通孔大小设置为30mil。部分调整后进行规则检查，保证PCB的设置符合的设定好的规则。

最后将整块板子进行敷铜，方便后续缩短制版时间。

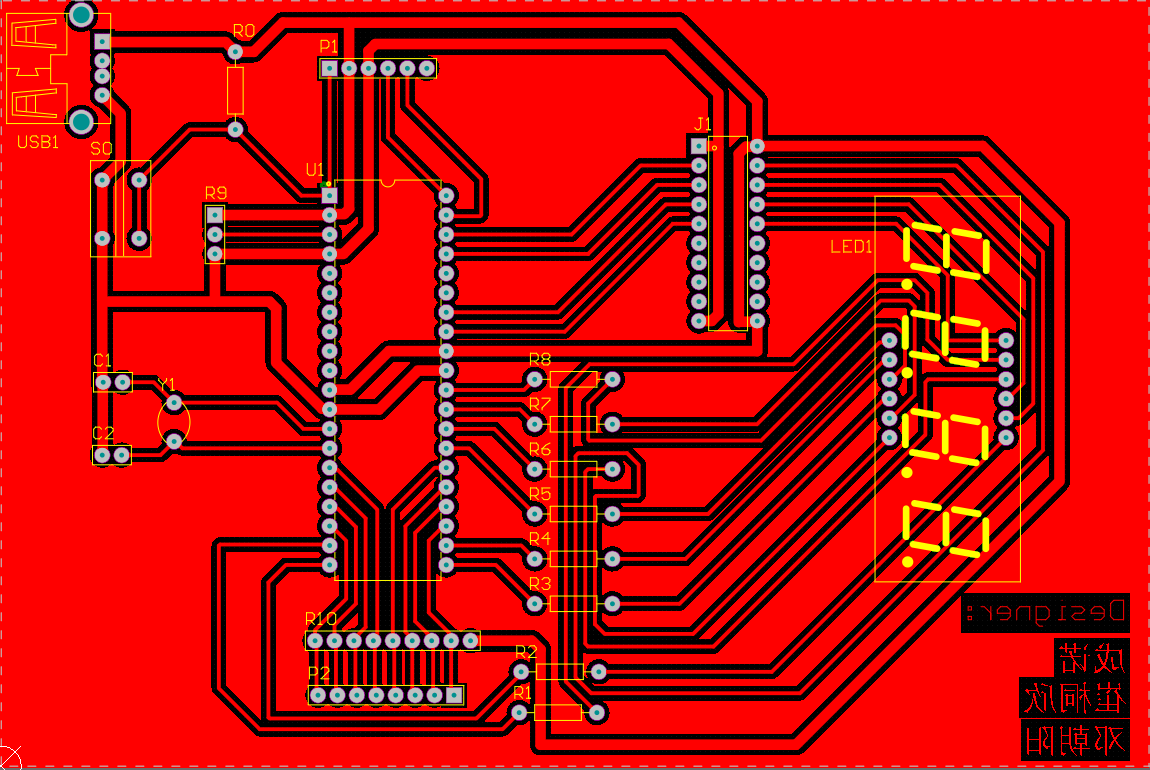


图3.2 PCB计

## 3.3印制与焊接

### 3.3.1转印

将设计好的PCB交给老师检查，修改错误后将正确的PCB布板图打印至A4纸上，领取电路板，将电路板对准并盖在PCB布板图上，放入真空转印机中，将PCB转印至电路板上。

### 3.3.2显影

将转印完成的电路板放入显影液中进行显影，并不断抖动电路板以加快进程，当电路清晰可见后取出放入腐蚀液中。

### 3.3.3腐蚀

将电路板放入腐蚀液中静置十分钟左右，期间每隔一段时间取出观察腐蚀情况，当从电路板背面能够清晰看到电路后，便可取出放入清水中清洗，并用吹风机将表面吹干。

### 3.3.4打孔

使用打孔机将电路板上对应的引脚进行打孔，其中USB固定引脚与排针需要使用较大的钻头进行打孔。

### 3.3.5焊接

根据PCB布板图将元器件焊接至电路板上，完成后使用万用表测量每条路是否连通，相邻引脚间是否短路。

## 3.4编写流程框图

对于软件的整体设计，首先设计一个大概的流程图，分析每个部分所需要实现的功能，然后逐步编写、调试、修改各模块的功能，最后在将各模块串起来进行调试，逐步实现最终的功能。

在本次程序设计中，根据使用到的外设已经最终实现的功能，程序流程图如下：

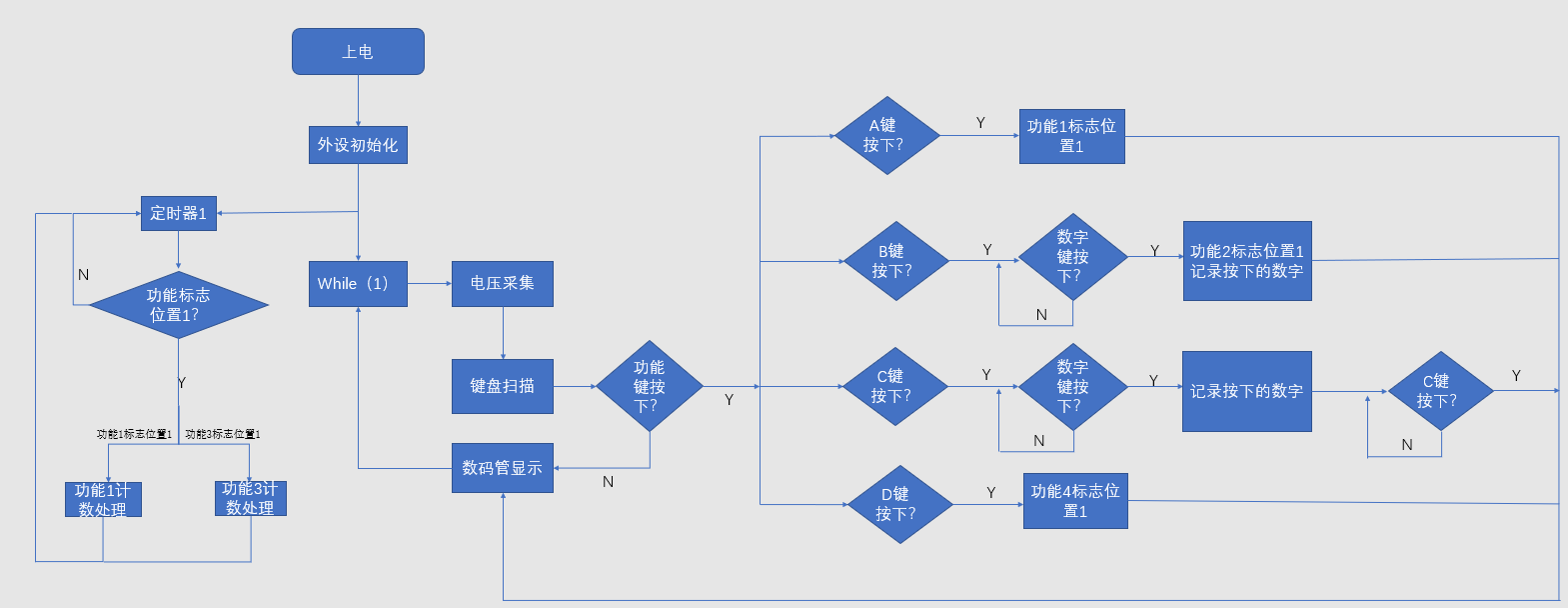


图3.4 程序流程图

根据程序流程图，需要设计的程序有如下几类：外设初始化、电压采集、键盘扫描、数码管显示。

## 3.5编写程序代码

下面根据程序的主要流程图,注意编写代码,并且在会附上每个环节流程图。

### 3.5.1 主函数

在主函数中，主要是初始化设定 I/O 的输入输出功能，预设置好与 ADC 相关的寄存器，并且开启定时器，随后程序便进入 while 循环里面，进行键盘扫描、AD 采集以及数码管驱动显示。思路如图3.5.1：

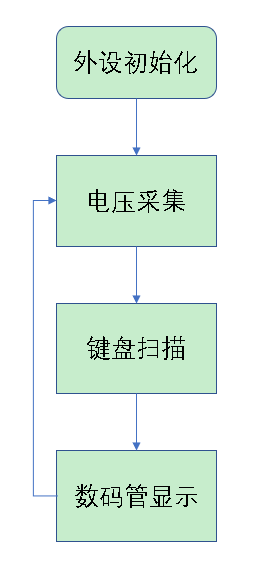


图 3.5.1

void main(void)

{

/\*\*\*\*\*\*\*端口输入输出初始化\*\*\*\*\*\*\*\*\*\*/

TRISA = 0X2F; //A 端口 0、1、2、3、5 设置为输入

TRISE = 0X07; //E 端口 0、1、2 设置为输入

TRISD = 0X00; //D 端口全部设为输出

TRISB = 0XCC; //B 端口 0、1、4、5 位设为输出

/\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*AD 寄存器初始化\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ADCON0 = 0X81; //三十二分频，选择 AN0 即 PA0 口作为输入检测，AD 在转换，使能 AD

ADCON1 = 0X80; //右对齐，参考英文手册 128 页，低四位取 0 为八口皆为模拟输入通道

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*定时器 1 初始化\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

T1CKPS1 = 1; //设定时钟分频

T1CKPS0 = 1; //f=foc/8 ，时钟频率为晶振频率/4=1MHz

TMR1CS = 0; //工作模式为定时器模式

TMR1IE = 1; //使能时钟 1 的中断

TMR1H= 0XCF; //设定初值，设置每次进入中断时间为 0.1s，0.1=（65536-

X）\*（1/(1MHz/8)）

TMR1L= 0X2C; // 得 X=53036，高四位为 X/256=207,第四位为 X%256=44

PEIE = 1; //允许外部中断源的中断请求

GIE = 1; //允许所有中断源的中断请求

TMR1ON = 1;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function4\_flag=1; //上电默认进入功能 4

delayms(5);

while(1)

{

AD\_handle1(ad\_num); //AD 采集

count++;

if(count==10) //当 count 到达 10，说明通道 ad\_num 已经采样十次，

换通道进行采集

{

count=0;

ad\_num++;

if(ad\_num==8)

{

ad\_num=0;

}

}

key\_scan(); //键盘扫描函数

if(function1\_flag==1) //轮流显示

{

display(function1\_display\_num);

}

else if(function2\_flag==1) //单口显示

{

display(function2\_display\_num);

}

else if(function3\_flag==1) //单口滚动显示

{

function3\_display();

}

else if(function4\_flag==1) //显示的内容熄灭

{

function4\_display();

}

else if((function1\_flag|function2\_flag|function3\_flag|function4\_flag)==0) //取消键按下后进入功能 1

{

function1\_flag=1;

TMR1ON = 1;

}

}

}

### 3.5.2 键盘扫描函数

键盘扫描函数采用较为典型的行列式扫描法，芯片 C 口的高四位与键盘行相连接，第四位与键盘列相连接。首先将 C 口高四位设置为输出，第四位设置为输入，令 PORTC=0X0F，当读取 C 口的值不为 0X0F 时，说明有按键被按下，通过 switch 语句判断出哪一列有按键被按下，并将该列第一行的值赋予一个变量，随后 C 口高四位设置为输入，低四位设置为输出，令 PORTC=0XF0，通过 switch 语句判断该按键位于哪一行，将最终的键值赋予一变量。待按键松开后，使用 switch 语句判断该按键是否属于功能键，若为功能键，则将所有的功能标志位清 0，对定时器进行处理，随后再判断是哪一个功能按键被按下，再执行相对应的处理。流程如图3.5.2：

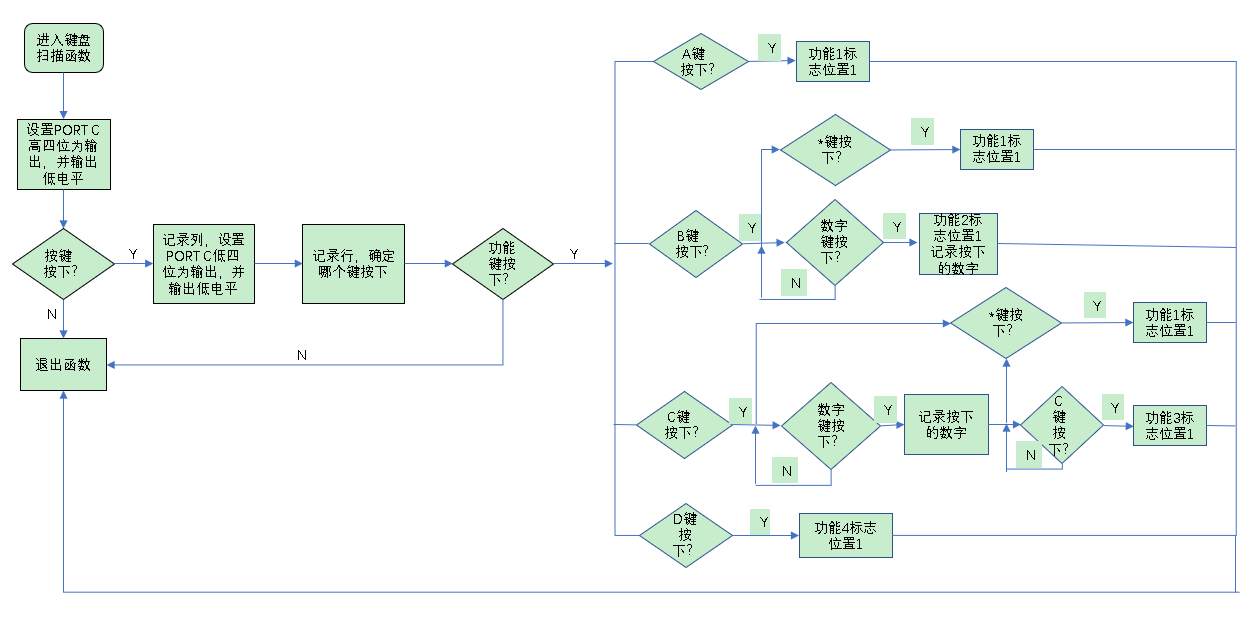


图 3.5.2

void key\_scan()//键盘扫描函数

{

uint temp; //变量用于记录 PC 口的值

int num=0; //若有按键按下，则记录按下的键值，无按下为 0

TRISC=0X0F; //高四位为输出，第四位为输入

PORTC=0X0F; //PC 口高四位输出 0

temp = PORTC;

if(temp!=0X0F) //若不为 0X0F，则说明有按键被按下，导致低四位有一位被置

零

{

delayms(5); //消抖

if(temp!=0X0F)

{

switch(temp)

{

case 0x0e: num=1; break; //第一列

case 0x0d: num=2; break; //第二列

case 0x0b: num=3; break; //第三列

case 0x07: num=10; break; //第四列

}

TRISC=0XF0; //低四位为输出，高四位为输入

PORTC=0XF0; //PC 口低四位输出 0

temp = PORTC;

switch(temp)

{

case 0xe0: //第一行

num=(num+0);

break;

case 0xd0: //第二行

if(num==10)

num+=1;

else

num+=3;

break;

case 0xb0: //第三行

if(num==10)

num+=2;

else

num+=6;

break;

case 0x70: //第四行

if(num==1)

num+=14;

else if(num==2)

num-=2;

else if(num==3)

num+=11;

else if(num==10)

num+=3;

break;

}

while(PORTC!=0XF0); //等待按键松开

}

}

//运行到此处，若功能键被按下，则进行处理（假设 num=A(10)、B(11)、C(12)、D(13)，\*（退出）（15）为功能按键）

if(num==10||num==11||num==12||num==13)

{

//将所有功能标志都置零

function1\_flag=0;

function2\_flag=0;

function3\_flag=0;

function4\_flag=0;

//定时器关闭,并且再次赋予初值，放止前面程序的影响

//2.26,定时器保持开启，保证 AD 的正常采集

//TMR1ON=0;

TMR1H= 0XCF;

TMR1L= 0X2C;

function1\_count=0;

function3\_count=0;

function1\_display\_num=0;

switch(num)

{

case 10:

function1\_flag=1;

TMR1ON=1;

//function1\_display\_num=0;

break;

case 11:

//function2\_flag=1;

while(fun2\_keyscan());

break;

case 12:

//function3\_flag=1;

while(fun3\_keyscan\_num());

//TMR1ON=1;

break;

case 13:

function4\_flag=1;break;

}

}

}

### 3.5.3 数码管轮换显示与单通道显示

轮换显示与单通道显示均使用同一个数码管驱动函数 void display(intdisplay\_num)，通过改变参数 display\_num 便可令数码管显示不同的内容。在调用过程中，轮换显示输入变量 function1\_display\_num，该变量会每隔 5s 增加 1，在 0~7 之间进行循环；而单通道显示输入变量 function2\_display\_num，该变量是在键盘扫描过程中确定的。流程如图3.5.3

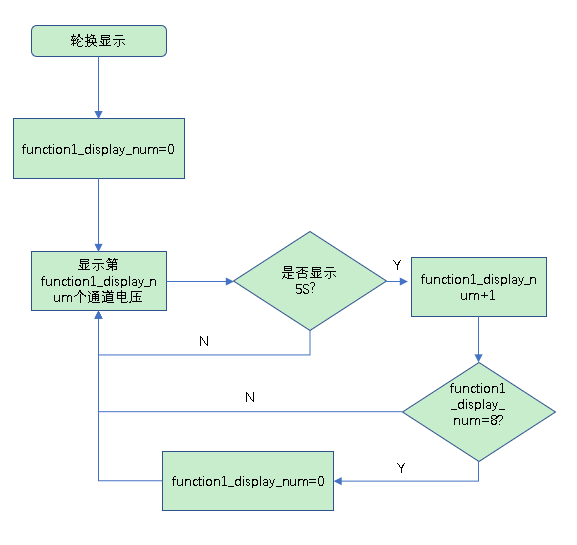


图 3.5.3

void display(int display\_num) //数码管显示第 display\_num 口的电压值

{

PORTD=0XFF; //熄灭所有段选，防止闪烁

PORTD=table[display\_num]; //第一位,AD 通道号

delayms(1);

PORTB=0X01;

delayms(1);

PORTB=0X00; //关闭位选

PORTD=0XFF;

PORTD=table[AD\_result[display\_num]/100]&0X7F; //第二位，个位（添加一个点的处理，与上 0X7F）

delayms(1);

PORTB=0X02;

delayms(1);

PORTB=0X00;

PORTD=0XFF;

PORTD=table[(AD\_result[display\_num]/10)%10]; //第三位,小数点后一位

delayms(1);

PORTB=0X10;

delayms(1);

PORTB=0X00;

PORTD=0XFF;

PORTD=table[AD\_result[display\_num]%10]; //第四位，小数点后两位

delayms(1);

PORTB=0X20;

delayms(1);

PORTB=0X00;

}

### 3.5.4 数码管单通道滚动显示

单通道滚动显示需要用到的变量有两个，一个是在键盘扫描中确定的通道号function3\_display\_num，另一个是记录显示时间的 function3\_count；function3\_count是在定时器中进行累加的变量，通过判断其值，便可确定数码管滚动显示进行到一个周期的哪个位置。流程如图 3.5.4

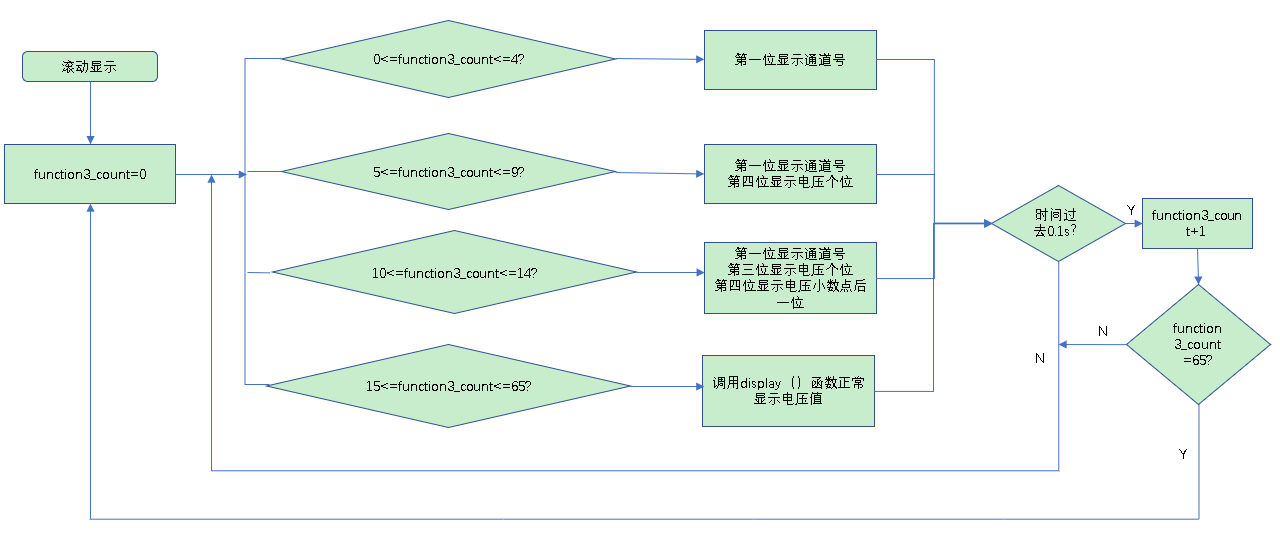


图 3.5.4

void function3\_display()

{

if(function3\_count<=4) //0~0.5s，只显示通道号

{

PORTD=0XFF;

PORTD=table[function3\_display\_num];

delayms(1);

PORTB = 0X01;

delayms(1);

PORTB=0X00;

}

if(function3\_count>=5&&function3\_count<=9) //0.5~1s，显示通道号和个位电压值

{

PORTD=0XFF;

PORTD=table[function3\_display\_num];

delayms(1);

PORTB = 0X01;

delayms(1);

PORTB=0X00;

PORTD=0XFF;

PORTD=table[AD\_result[function3\_display\_num]/100]&0X7F;

delayms(1);

PORTB = 0X20;

delayms(1);

PORTB=0X00;

}

if(function3\_count>=10&&function3\_count<=14) //1~1.5S，显示通道号，显示电压个位和小数点后一位

{

PORTD=0XFF;

PORTD=table[function3\_display\_num];

delayms(1);

PORTB = 0X01;

delayms(1);

PORTB=0X00;

PORTD=0XFF;

PORTD=table[AD\_result[function3\_display\_num]/100]&0X7F;

delayms(1);

PORTB = 0X10;

delayms(1);

PORTB=0X00;

PORTD=0XFF;

PORTD=table[(AD\_result[function3\_display\_num]/10)%10];

delayms(1);

PORTB=0X20;

delayms(1);

PORTB=0X00;

}

if(function3\_count>=15&&function3\_count<=65) //连续 5 秒全部显示

{

display(function3\_display\_num);

}

}

### 3.5.5 A/D 转换

A/D 转换功能使用到的寄存器有 ADCON0 与 ADCON1，其中通过改变 ADCON0 的值便可实现对不同通道的电压采集。在程序设计时设置两个数组：AD\_result[8]与 AD[8]，在进行A/D 转换时，将寄存器读取到的十位二进制数字量处理后存放于 AD[8]中，当同一通道累计采样十次后，将 AD[8]中的数据进行处理存放于 AD\_result[8]，AD\_result[8]作为最终电压值推送至数码管显示。主要流程如图3.5.5

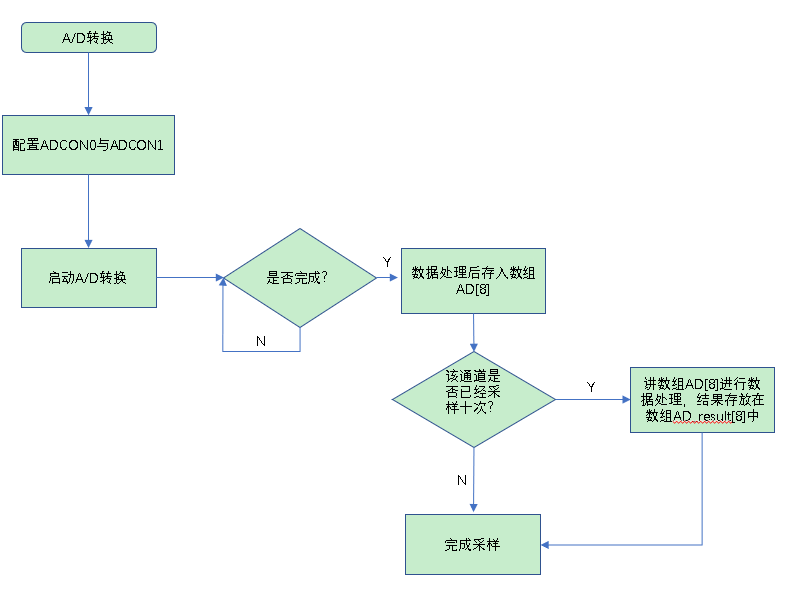


图 3.5.5

void AD\_handle1(int ad\_num)

{

uint adval=0; //用于数据处理

float advalf=0;

static int ad\_count; //记录累计次数

//采样计数+1

ad\_count++;

ADCON0 = 0X81|(ad\_num<<3); //选择 AD 采集口

ADCON1 = 0X80; //必须加上，不然出 bug

GO\_DONE = 1; //ADCON0 的 AD 启动位，置 1 为启动

while(GO\_DONE);//等待转换完成

adval=ADRESH; //高两位

adval=(adval<<8)|ADRESL; //高两位左移八位再或第八位，即得到十位数据

advalf=(float)adval\*5.0/1023;

adval=advalf\*100;

AD[ad\_num]+=advalf;

if(ad\_count==10) //当计数到第十次时，进行平均处理

{

AD\_result[ad\_num]=(uint)(AD[ad\_num]\*100/10);

AD[ad\_num]=0;

ad\_count=0;

}

}

### 3.5.6 中断处理函数

中断处理函数主要服务于功能 1 和功能 3 的计时，设置中断时间为 0.1s，进入中断函数后通过判断功能 1 标志位和功能 3 标志位进行相对于的处理。

void \_\_interrupt() Timer1(void)

{

if(TMR1IF==1) //溢出标志位，说明过了 0.1s

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 定 时 器 1 的 中 断

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

TMR1H= 0XCF; //重新赋予初值

TMR1L= 0X2C;

TMR1IF=0; //清除标志位

if(function1\_flag==1)

{

function1\_count++;

if(function1\_count==50) //达到 5s

{

function1\_count=0;

function1\_display\_num++;

if(function1\_display\_num==8)

function1\_display\_num=0;

}

}

if(function3\_flag==1)

{

function3\_count++; //假设每移动一位的时间为 0.5s，则一个

循环为 14 个 0.5s

if(function3\_count==65)

function3\_count=0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

}

## 3.6 代码仿真

在电路板完成制作后，根据电路板实际布板在 proteus 软件上搭建一个类似的电路，如图：

ADCON1=0X80 后便解决了电压不准确的问题。

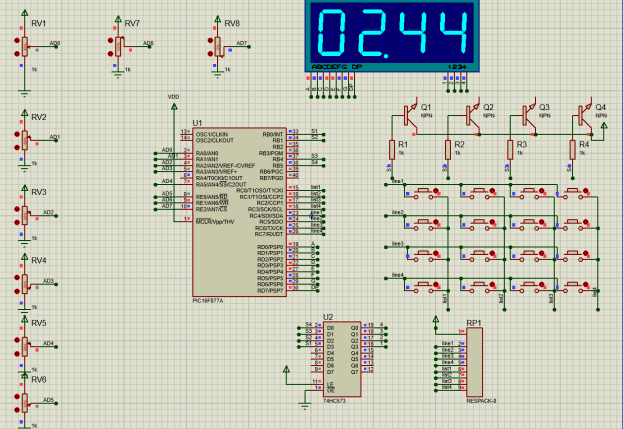


图 3.10 仿真环境

在 MAPLAB 上完成相关程序的编写后，便可使用 proteus 验证代码是否可行，在仿真平台上得到较为理想的结果后，再将程序通过 PICkit2 烧录器烧录至实物上，寻找存在的问题并进行修改，直至得到较为满意结果。

# 四、结果分析

最终得到的简易电压表能够按照预期设计实现其功能，但在设计过程中遇到了不少的问题，遇到的问题及其解决方法有以下：

## 4.1问题与调整

### 4.1.1读取到的电压值不准确

第一次实现多通道采集电压的时候发现，接至 VCC 的采集口电压显示出来仅有 2V 左右，在排除数据处理、数码管驱动显示等方面没有问题后，初步怀疑 A/D 采集过程中存放采样结果的寄存器可能因为采样速度太快而导致数据错误；使用 MAPLAB 的 debug 功能，一步一步运行程序发现能够采集到正常的 5V，所以确定很有可能是由于采样速度太快而导致寄存器数据处理错误，后经过调整采样周期、寄存器分频调整等多种方法后，最终确定为每次进行 A/D 转换时，不仅需要配置 ADCON0,还需要配置 ADCON1。但是程序上电的时候已经将 ADCON1 配置为 0X80 且后续的程序中均没有改动，按道理来说并不需要再次配置ADCON1 为 0X80,但在 A/D 转换程序加上ADCON1=0X80 后便解决了电压不准确的问题。

### 4.1.2滑动变阻器失效

按照原理图的设计，采集口 1 本该接至滑动变阻器的滑动端，通过调整其位置便可读取到不同的电压值，但在程序调试过程中发现采集口 1 显示出来的电压非常接近 0，且无法调整，由于之前考虑过 GND 接至滑动端可能出现的情况，故当遇到这个问题后我便查看PCB 图，确定是因为滑动变阻器封装的问题导致 GND 接至滑动端，最终通过飞线解决了这个问题。

### 4.1.3悬空端电压不稳定

在设计中有5个 A/D 采集口为悬空端，在进行采集时，数码管显示出来的电压非常不稳定，在老师的指点下，对 A/D 采集的方式进行修改，当某一采集口采集够 10 次后，对这10 次的值进行累加然后取平均值作为最终的数据，最终悬空端电压值得浮动值减小。

## 4.2效果演示

最终,在解决以上问题后,可以正常运行,演示效果如下图:



图 4.1 运行演示效果

# 五、课程设计总结

为期两周的实验实习课程，本人学会到了如何运用基本的材料来做一个比较简单实用的仪器。原理图设计、PCB元件封装和设计、腐蚀与焊接、软件设计，每个环节都或多或少地有或深或浅的体会。

## 5.1 原理图设计总结

通过对目标的功能分析后，画出相应的电路图是比较清晰容易的。然后，根据所需的功能元件，寻找相应的实体元件。体会比较深的是，可以好好利用“立创”平台寻找到自己所需的封装，方便设计。然而，由于不知道课程提供的元件是什么，尽管原理图的基本结构是对的，但具体的封装却与实际的不符，比如矩阵键盘的封装，找到的封装不是1\*8的整齐排口，还有上拉电阻也不符合1\*9的排口。这些细节问题需要等到老师检验，才能正确地修改。本人建议老师可以提供给学生元件的封装，或者给具体实物参考，或者提供学长运用的封装照片，强调一下容易选错的封装。这样可以减少老师检验的工作量。

## 5.2 PCB布线

在上学期的电工实习当中已经有一些PCB的布线基础，将原理图导入PCB的工程，然后布线。基本的思路是根据元件的连接关系，初步摆放元件的位置，然后自动布线。根据自动布线结果的提示，调整元件的相对位置。实际过程中，电脑全力运算都无法布置出理想的线。仔细分析，发现一些元件的连接关系是相互交叉的。在只能使用单层板的情况下，这给布线提供了相当大的难度。虽然也可以使用飞线，但是希望飞线的数量越少越好，能让硬件稳定点。于是，为解决元件连接关系的交叉，本人发现可以从原理图入手，修改管脚的对应关系。其中，数码管引脚和键盘引脚的排布比较乱，交叉的走线较多。本人重点调整这两个部分管脚的连接关系，使得元件之间的连线是尽量平行的。修改后，按照原来的迭代调整的方法，最终电脑可以完成自动布线。为了避免出错，本人参考着电脑的连接方式，重新人工布线。

布置好位置后进行焊盘大小调整和对布线规则进行设置，电源和地的走线宽度较信号线宽，为40mil，信号线走线25mil，最后采用自动布线和手动布线结合，将不合理的走线修改，最终的布线还是相对合理。

## 5.3 腐蚀与焊接

腐蚀基本是在老师的指导下完成的。钻孔与焊接靠的是个人的耐心。在焊接的过程中，发现可以和队友良好地合作。比如，排针的焊接，如果让队友帮助自己抵住元件，避免从板子上滑出来，就非常好焊接。

## 5.4 程序设计

在开学之前，本人特地阅读了一些关于嵌入式的编程技巧，《C语言模块化编程》一书，收益匪浅。在编程的开始，本人就着手思考如何让代码尽可能具有通用性，避免后续管脚修改的时候，大量修改名称；如何模块化编程，使得层次分明，降低代码之间耦合度，使得后续修改的时候，不同模块的影响最小。（本人没用直接参与电压表的设计，但是本人独立完成了2个8\*8led显示的程序设计，本人组的程序设计主要由例外一个同学完成，所以本人重点根据关于led显示设计来谈自己对程序设计的感受。）

在编程的初期，本人就定义特殊含义的宏名称和宏函数，使得后续代码与具体的管脚无关。如果管脚发生改变，非常容易修改。而且宏函数的定义，使得操作管脚的输入输出非常方便。比如设置管脚状态为高电平，为“set\_pin(B0,1)”,通过宏展开，会变成TRISB0=0；RB0=1；。这样，可以简化对代码的使用和思考。（一些相关的定义，可以见附录）

在程序框架的构建中，本人借鉴了游戏的设计的框架。一个while循环，首先是扫描按键的输入。将得到的信息存放到临时的空间当中。然后根据按键的信息，进行相应的逻辑处理，并且转化为预备显示的图像，最后根据预备显示的图像显示图像。由此完成一个循环。当然，稍稍不同的是中断的运用。本人是借鉴Windows操作系统的设计想法，将定时器中断当作一个时钟源，由此模拟出多个时钟信息，由此来构造出1s，3s等不同的时间间隔。剩下的就是具体的细节实现了。像显示，按键扫描，已经有非常成熟的过程，结合一点C语言的模板函数的设计，就可以写出非常良好的函数接口。

以上是本人在led显示中比较深刻的感受。程序设计的最终结果是可以良好地运行，而且非常容易扩展。可以显示任意数量的任意字形，如果时间富裕的话，本人考虑过可以做一些小动画，写一个led版的俄罗斯方块（基本的图形界面已经搭建了）。

此次的实验，本人不只是在本组负责任务，同时也帮助其他组编写代码，思考一些发生的问题。其中一些问题，让本人重新认真的思考了“什么是系统”。

整个课程是被分解成几个环节，但是问题的出现让本人意识到这是一个整体。比如在编写led显示的时候，发现一排led的显示不受软件控制，进而在硬件上进行测试，发现原来PIC16F877A的RA4口是需要加上上拉电阻才可以输出数字量。在帮助其他同学关于按键软件编写总不成功的问题中，通过软件调试的手段，加上电压表的测量，发一些电路板的一些地方存在虚焊，还用一些焊接导致电路短路。透过这些问题，本人意识到一个系统的模块不是独立的，它和其他模块会发生关联。软件出现的问题，可能是因为焊接引起的，也可能是原理图设计的不对。

在解决问题的过程中，本人体会到如何灵活利用工具取排除问题。比如最终，按键出现故障，可以通过软件方法输出按键的行和列，哪一些不能正常输出的说明有问题，然后再通过电路表测试相应的电路，最终找到问题。

总之，收益良多，非常感谢老师开设这门课程，让本人实践能力和代码编写能力得到提升，学会调试找bug，团队合作，感谢老师的耐心教导。

# 参考文献

[1] 洪晓斌，何振亚，徐静编. 教育部高等学校机械类专业教学指导委员会规划教材 微机原理及应用 PIC系列单片机. 北京：清华大学出版社, 2021.09.

[2] 刘向宇编著. PIC单片机C语言程序设计实例精粹. 北京：电子工业出版社, 2010.02.

[3] 王代华，宋林丽，王恩怀编著. PIC单片机及其嵌入式应用. 北京：国防工业出版社, 2013.01.

[4] 高显生编著. 我和PIC单片机 入门卷 基于PIC18. 北京：机械工业出版社, 2013.07.