目录

图目录

表目录

摘要

温度是我们日常生产和生活中实时在接触到的物理量，但是它是看不到的，仅凭感觉只能感觉到大概的温度值，传统的指针式的温度计虽然能指示温度，但是精度低，使用不够方便，显示不够直观，数字温度计的出现可以让人们直观的了解自己想知道的温度到底是多少度。

数字温度计是一种精度高、稳定性好、适用性极强的新型现场温度显示仪，是传统现场指针式金属温度计的理想替代产品，广泛应用于各类工矿企业，大专院校，科研院所。

数字温度计采用温度敏感元件也就是温度传感器（如铂电阻，热电偶，半导体，热敏电阻等），将温度的变化转换成电信号的变化，如电压和电流的变化，温度变化和电信号的变化有一定的关系，如线性关系，一定的曲线关系等，这个电信号可以使用模数转换的电路即 AD 转换电路将模拟信号转换为数字信号，数字信号再送给处理单元，如单片机或者 PC 机等，处理单元经过内部的软件计算将这个数字信号和温度联系起来，成为可以显示出来的温度数值，如 LED,LCD 或者电脑屏幕等显示出来给人观察。这样就完成了数字温度计的基本测温功能。

此次课程设计我们借助 Proteus 软件仿真实现 PT100 测量温度的功能。

关键字：电压采集、单片机、MSP430、A/D 转换、PT100

Abstract

Temperature is in our daily life and production in real-time access to physical quantities, but it is invisible, just by feeling can only feel about temperature, although the traditional pointer thermometer can indicate the temperature, but the precision is low, use convenient enough, show not intuitive, the emergence of digital thermometer can let people intuitively understand how much he'd like to know what the temperature is.

Digital thermometer is a new field temperature indicator with high precision, good stability and strong applicability. It is an ideal substitute for traditional field pointer metal thermometer and is widely used in various industrial and mining enterprises, colleges and universities, and scientific research institutes.

Digital thermometer is a temperature sensitive element temperature sensor (such as platinum resistance, thermocouple, semiconductor, thermistor, etc.), the temperature changes of converted into electrical signals, such as the change of the voltage and current, the temperature change and the change of the electrical signal has a certain relationship, such as linear relationship, the curve of a certain relationship, such as the electrical signal can be used in adc circuit that AD conversion circuit converts analog signals to digital signals, the digital signal to send to processing unit, such as SCM and PC, processing unit through internal software calculation will be linked to the digital signal and temperature,

It becomes a temperature value that can be displayed, such as LED,LCD or computer screen, for people to observe.This completes the basic temperature measurement function of the digital thermometer.

This course design we use Proteus software simulation to achieve PT100 temperature measurement function.

Key words: voltage acquisition, SCM, MSP430, A/D conversion, PT100

温度测量仪设计

1 设计题目

Proteus 仿真实现温度测量仪的功能。

2 设计目的

运用单片机原理及其应用等课程知识,根据题目要求进行软硬件系统的设计和调试，从而加深对传感器的理解，把学过的比较零碎的知识系统化，比较系统地学习开发单片机应用系统的基本步骤和基本方法，使应用知识能力、设计能力、调试能力以及报告撰写能力等有一定的提高。

3 设计内容及要求

设计并制作一个水温测量仪，温度范围为 0～100℃，测量精度±1%,显示分辨率 0.1℃。

4 系统总体结构

此次温度测量仪系统包括：恒压源模块，电桥模块，显示模块，运放模块，控制模块。

系统上电，由 TL431 以及运放 OP07 构成的恒压源模块产生恒压源供给电桥模块恒定的电压。电桥模块通过动态平衡输出变化的电压，将输出的差分电压交给由 LM324 配合电阻构成的运放模块进行一定倍数的电压放大，单端输出，交由控制模块，即单片机 MSP430G2553 开发板，进行 ADC 转化。转化完成后由显示模块 LCD1602A 液晶显示屏显示。

系统总设计框图如下：

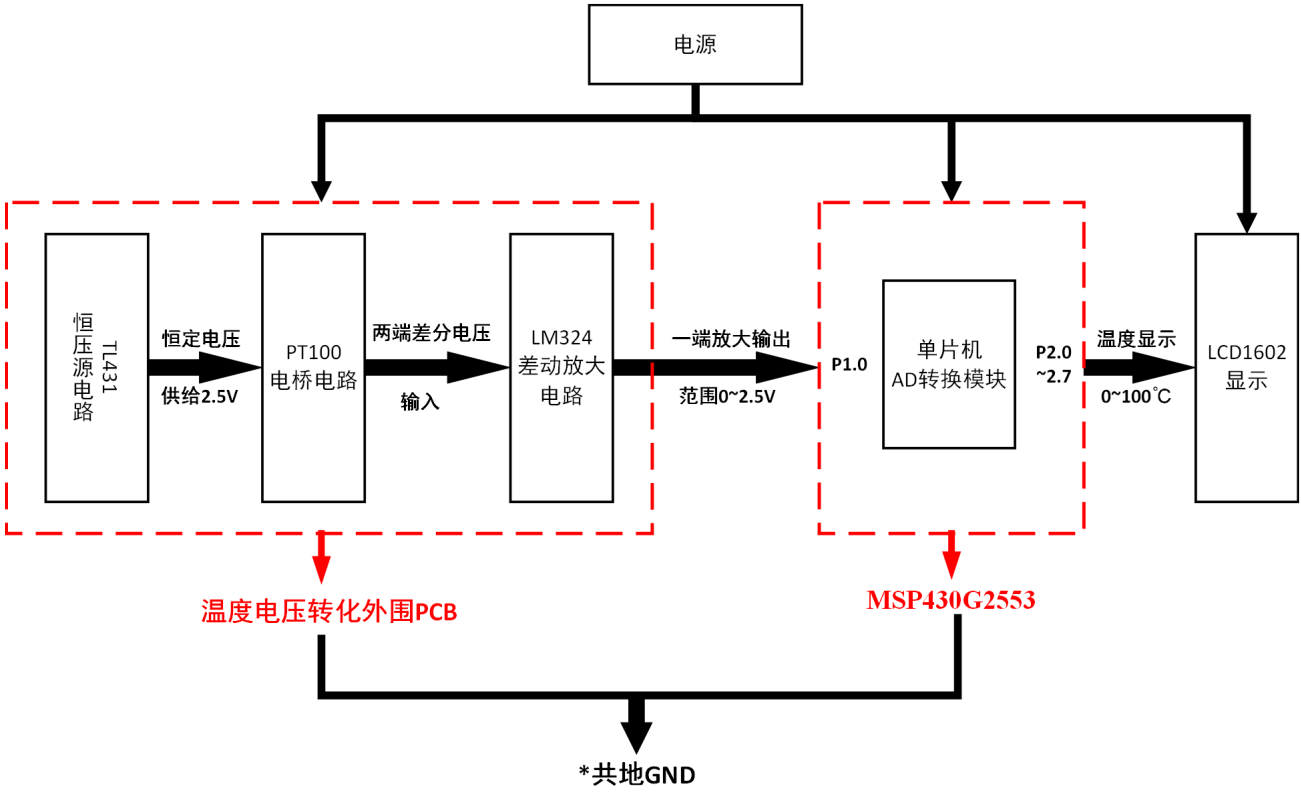


图 4-1 系统框图

5 硬件设计

5.1 元件选取

(一) MSP430G2553

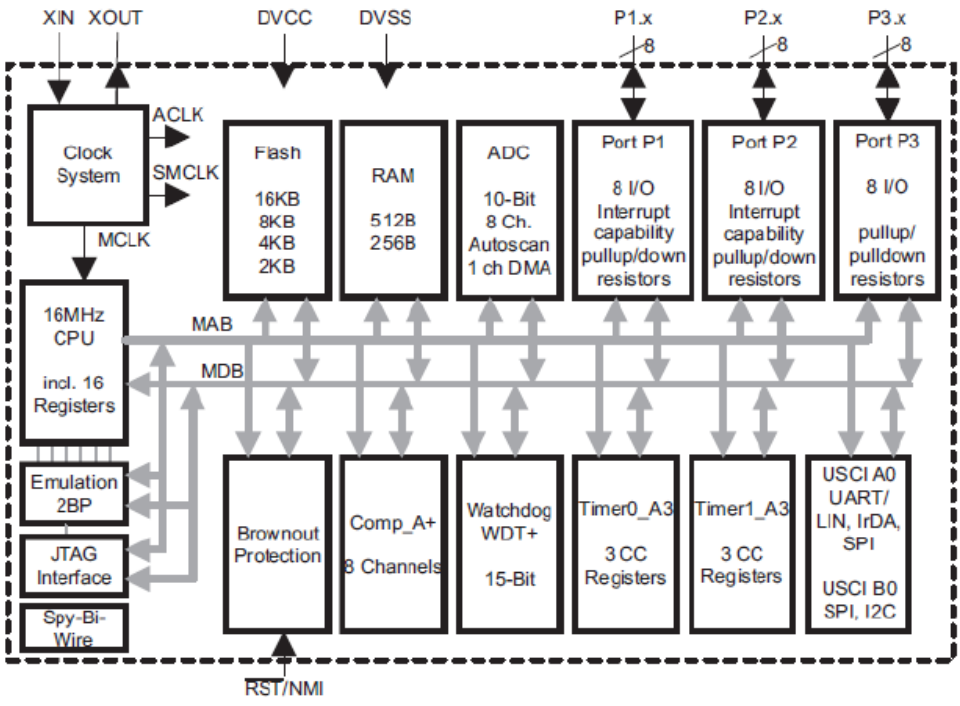


图 5-1 MSP430G2553 微控制器结构

如图 2 所示，此次课程设计主要用到了 MSP430G2553 内部的 ADC 功能。

ADC10 是 MSP430 单片机的片上模数转换器，其转换位数为 10bit，该模块内部是一个 SAR 型的 AD 内核，可以在片内产生参考电压，并且具有数据传输控制器。数据传输控制器能够在 CPU 不参与的情况下，完成 AD 数据向内存任意位置的传输。

ADC10 模块工作的核心是 ADC10 的核，即图中的 10-bitSAR。ADC10 的核将模拟量转换成 10 位数字量并储存在 ADC10MEM 寄存器里。这个核使用 VR+和 VR-来决定转换模拟值的门限。模拟电压 Vin 的输入范围：VR_-≤Vin≤VR_+。当输入 Vin≥VR+，NADC=1023。当输入 Vin≤VR-，NADC=0。采样值的计算公式为：

$$N_{ADC} = 1023 \times \frac{V_{in} - V_{R-}}{V_{R+} - V_{R-}} \tag{5-1}$$

反推模拟量的大小 Vin

$$V_{in} = \frac{VR_{+} \times N_{ADC}}{1023} \tag{5-2}$$

本次被测电阻两端的电压可由 Vin 得到，以下是 ADC10 总体结构框图。

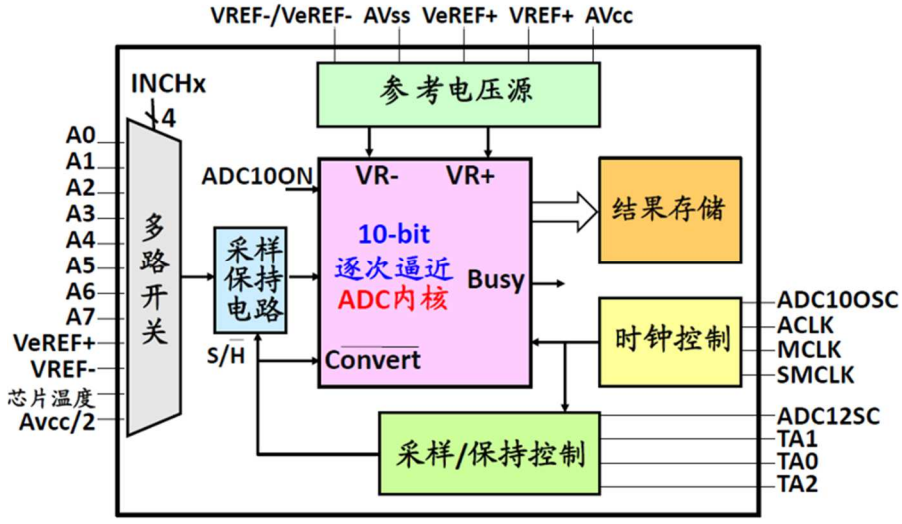


图 5-2 ADC10 总体结构框图

本次课程设计采样引脚为 P1.0，对应的通道为 A0，内部采取的参考电压为 2.5V，所以单片机实际采样得到的电压计算公式为

$$V_{in} = \frac{2.5 \times N_{ADC}}{1023} \tag{5-3}$$

(二) PT100

由于本次课程设计是测量水温 0～100℃，考虑到防水和温度区间，选择的温度传感器为 PT100，广泛用于测量-200～850℃范围内的温度。

PT100 铂热电阻与温度变化存在一定的线性关系，具体公式为:

$$R_t = 100 \times (1 + At) \tag{5-4}$$

其中 t 为温度，R_t为铂热电阻的阻值， A=3.851×10^{-3}/℃

(三) LCD1602

LCD1602 液晶显示器是广泛使用的一种字符型液晶显示模块。它是由字符型液晶显示屏（LCD）、控制驱动主电路 HD44780 及其扩展驱动电路 HD44100，以及少量电阻、电容元件和结构件等装配在 PCB 板上而组成。

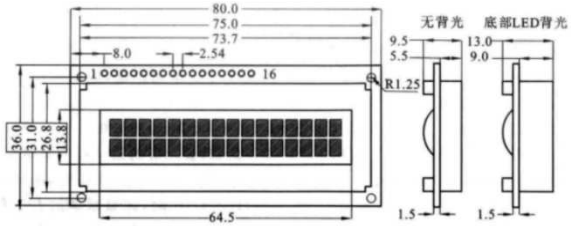


图 5-3 LCD 俯视图

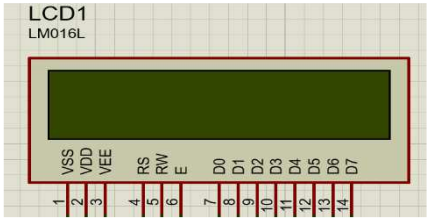


图 5-4 对应引脚

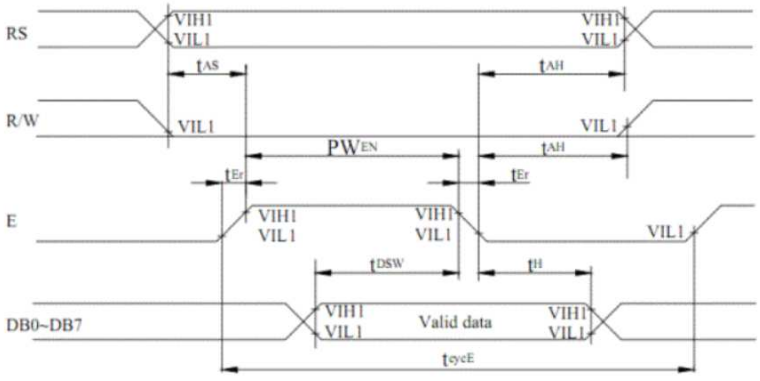


图 5-5 LCD 写操作时序图

引脚 1、2：分别为地和电源。

引脚 3：VL 为液晶显示器对比度调整端。

引脚 4：RS 为寄存器选择脚，高电平时选择数据寄存器、低电平时选择指令寄存器。

引脚 5：R/W 为读/写信号线，高电平时进行读操作，低电平时进行写操作。当 RS 和 R/W 共同为低电平时可以写入指令或显示地址；当 RS 为低电平，R/W 为高电平时，可以读忙信号；当 RS 为高电平，R/W 为低电平时，可以写入数据。

引脚 6：E 端为使能端，当 E 端由高电平跳变为低电平时，液晶模块执行命令。引脚 7~14：D0~D7 为 8 位双向数据线。引脚 15、16：分别为背光源正极和负极。

此次设计 MSP430G2553 的 P2.0~P2.7 与 LCD1602 的 D0~D7 相连。P1.1 与 E 相连，P1.2 控制 RW，P1.3 控制 RS。

(四) OP07

0P07 芯片是一种低噪声，非斩波稳零的双极性运算放大器集成电路。由于 0P07 具有非常低的输入失调电压（对于 0P07A 最大为 25HV），所以 OP07 在很多应用场合不需要额外的调零措施。OP07 同时具有输入偏置电流低（OP07A 为土 2ns）和开环增益高（对于 0P074 为 3007/mV）的特点，这种低失调、高开环增益的特性使得 0P07 特别适用于高增益的测量设备和放大传感器的微弱信号等方面。

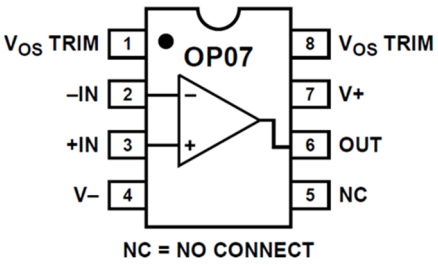


图 5-7 OP07 内部结构

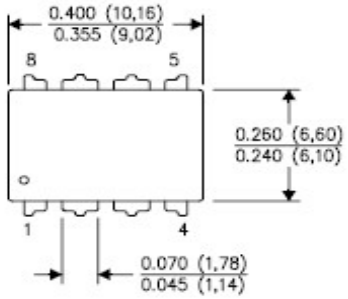


图 5-6 OP07 俯视图

OP07 芯片引脚功能说明： 1 和 8 为偏置平衡(调零端)，2 为反向输入端，3 为正向输入端，4 接负电源，5 空脚 6 为输出，7 接正电源。

本次设计 OP07 作为稳压和反馈的功能。

(五) LM324

LM324 系列是低成本的四路运算放大器，14 脚双列直插塑料封装，具有真正的差分输入外形如下图所示。单双电源均可工作。共模输入范围包括负电源

因此在众多应用中无需外部偏置元器件。输出电压范围也包括负电源电压。

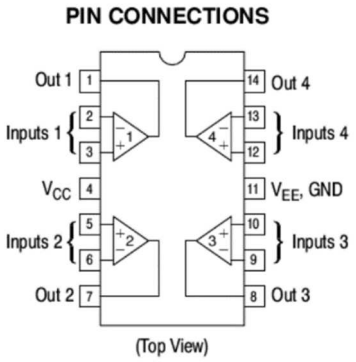


图 5-9 LM324 内部透视图

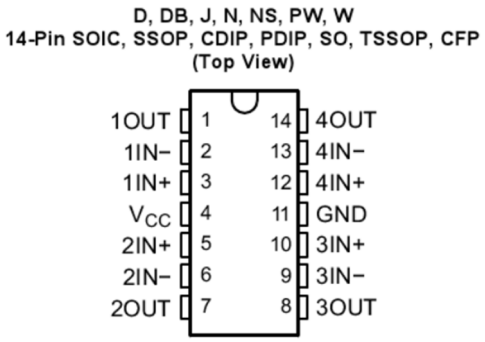


图 5-8 LM324 实物俯视图

(六) TL431

TL431 是可控精密稳压源。它的输出电压用两个电阻就可以任意的设置到从 Vref（2.5V）到 36V 范围内的任何值。该器件的典型动态阻抗为 0.2Ω，在很多应用中用它代替稳压二极管，例如，数字电压表，运放电路，可调压电源，开关电源等。

以下是分别是 TL431 的实物草图、软件中的元件图。

图 5-11 TL431 的实物

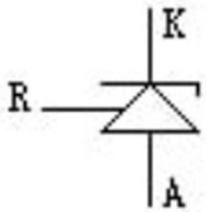


图 5-10 元件图

TL431 可等效为一只稳压二极管，其基本连接方法如下图所示。下图 a 可作 2.5V 基准源，下图 b 作可调基准源，电阻 R2 和 R3 与输出电压的关系为 U0=(1+R2/R3)2.5V

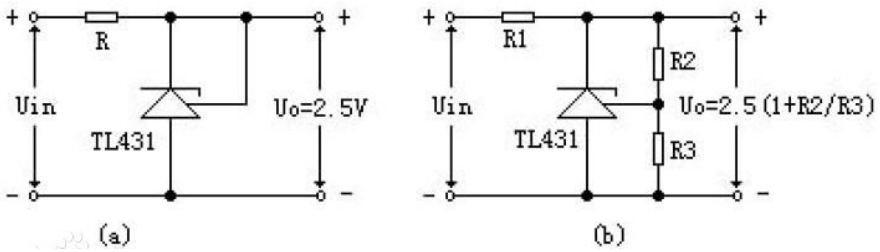


图 5-12 TL431 的使用方法

5.2 电路设计

(一) 恒压源、电桥模块

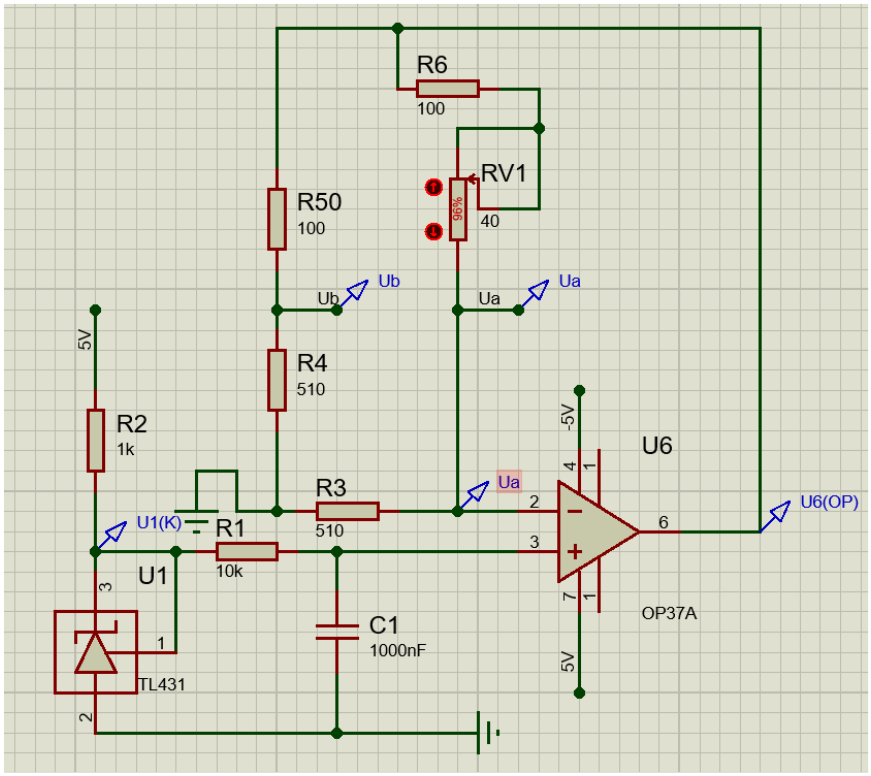


图 5-13 恒压源、电桥模块

电桥电路如图 5-13 所示，注意经过计算，仿真用 100Ω的电阻和 40Ω的滑动变阻器替代 PT100 温度传感器。TL431 和 R2 提供恒压源 Vref=2.5V, R1 和 C1 作为低通滤波。根据理想运算放大器“虚断”，没有电流流入同相输入端，所以 OP07 的 3 引脚的电压=Vref=2.5V。又根据理想运算放大器“虚短”, OP07 的 2 引脚电压等于 3 引脚的电压=Vref=2.5V，同时由于“虚断”，流过 R3 的电流只流经铂热电阻 R_t, 所以相当于有一个恒流源流经 R4 和 Rt，同时 R4, R3, R50, R_t 构成测量电桥, 通过检测 U_a 和 U_b 的电压差 U_{ab} 来反应温度的变化。当电桥达到动态平衡时,有:

$$R50 \cdot R3 = R4 \cdot Rt \tag{5-5}$$

通过计算可知:

$$U6 = \frac{Vref}{R3} \cdot Rt + Vref = Vref \cdot \left(\frac{Rt}{R3} + 1\right) \tag{5-6}$$

$$Ub = \frac{U6 \cdot R4}{R4 + R50} = \frac{Vref \cdot R4 \cdot \left(\frac{Rt}{R3} + 1\right)}{R4 + R50} \tag{5-7}$$

$$\mathrm{Uab} = \mathrm{Vref} - \mathrm{Ub} = \mathrm{Vref} - \frac{\mathrm{Vref} \cdot \mathrm{R4} \cdot \left(\frac{\mathrm{Rt}}{\mathrm{R3}} + 1\right)}{\mathrm{R4} + \mathrm{R50}} \tag{5-8}$$

$$\mathrm{Uab} = \mathrm{Vref} \cdot \left[1 - \frac{\mathrm{R4} \cdot \left(\frac{\mathrm{Rt}}{\mathrm{R3}} + 1\right)}{\mathrm{R4} + \mathrm{R50}}\right] \tag{5-9}$$

取 Ua=Vref=2.5V，R4=R3=510，R50=100，Rt=R6+ΔRt=100+ΔRt

则电桥电路的输出电压 Uab 两端的电压为:

$$\mathrm{Uab} = -Vref\frac{\Delta Rt}{R4 + R50} \tag{5-10}$$

从而求得铂热电阻阻值变化量:

$$\Delta\,\mathrm{Rt} = -\,\mathrm{Uab}\frac{R4 + R50}{Vref} \tag{5-11}$$

可见 $\Delta\,\mathrm{Rt}$ 与 Uab 成线性关系。

(二) 运放模块

图 5-13 电路中，取 R3=R4=510Ω，R50=R6=100Ω

又因为$Vref$=2.5V

$$\mathrm{Uab} = -Vref\frac{\Delta Rt}{R4 + R50} = -2.5\frac{\Delta Rt}{510 + 100} = -4.90836 \times 10^{-3} \times \Delta Rt \tag{5-12}$$

$$\Delta\,\mathrm{R}_t = \mathrm{R}_t - 100 = 100 \times (1 + At) - 100 = 100 \times At \tag{5-13}$$

其中 A 为常数，A=3.851×10^{-3}/℃

当温度

t=0℃时，ΔRt=0Ω，Uab=0V

t=100℃时，ΔR_t=38.51Ω，Uab=-0.15783V

因此，放大模块所需的放大倍数为：

$$A_u = \frac{2.5}{|Uab|} = \frac{2.5}{0.15783} \approx 16 \tag{5-14}$$

放大电路如下图 5-14 所示：

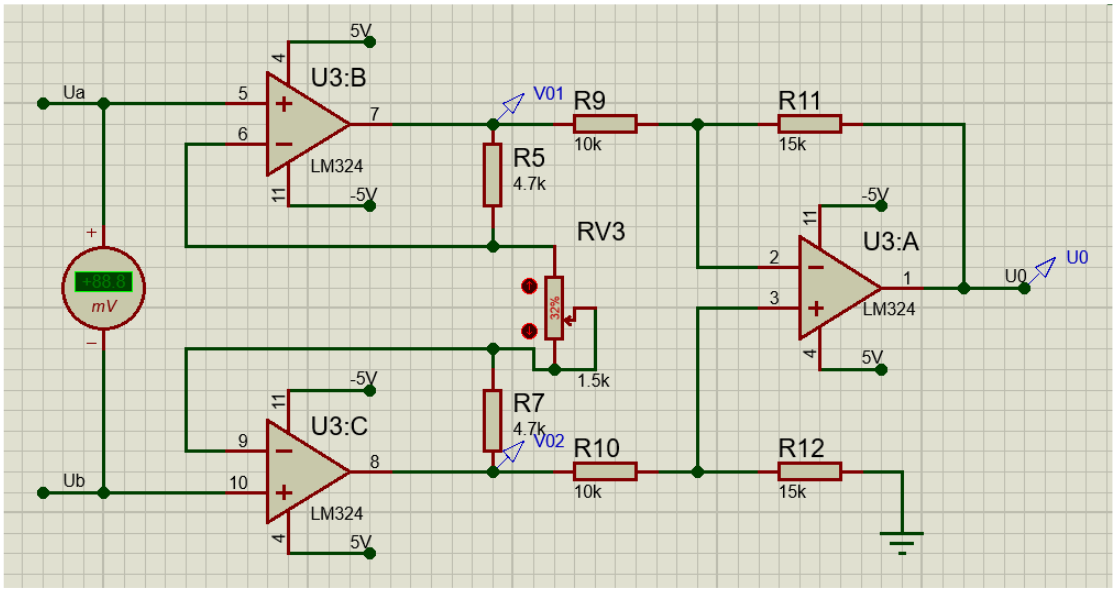


图 5-14 放大模块

由图 5-14 可知，RV3 上的电压是 Ub-Ua，流过 R5、R7 的电流等于流过滑动变阻器 RV3 的电流。

$$\mathrm{V02}-\mathrm{V01}=(2\mathrm{R5}+\mathrm{RV3})\frac{\mathrm{Ub}-\mathrm{Ua}}{RV3} \tag{5-15}$$

$$\mathrm{U0}=-\frac{R11}{R9}(\mathrm{V02}-\mathrm{V01}) \tag{5-16}$$

$$\text{总放大倍数}A_u=-\frac{R11}{R9}\times\left(1+\frac{2\mathrm{R5}}{\mathrm{RV3}}\right) \tag{5-17}$$

取 R5=R7=4.7KΩ，RV3=972Ω，则 A1=10.76

取 R11=R12=15KΩ，R9=R10=10KΩ，则 A2=1.5

A_u=A1×A2≈16

(三) 显示、控制模块

如下图 5-15 所示，MSP430G2553 控制 LCD1602，显示采样以后的电压。

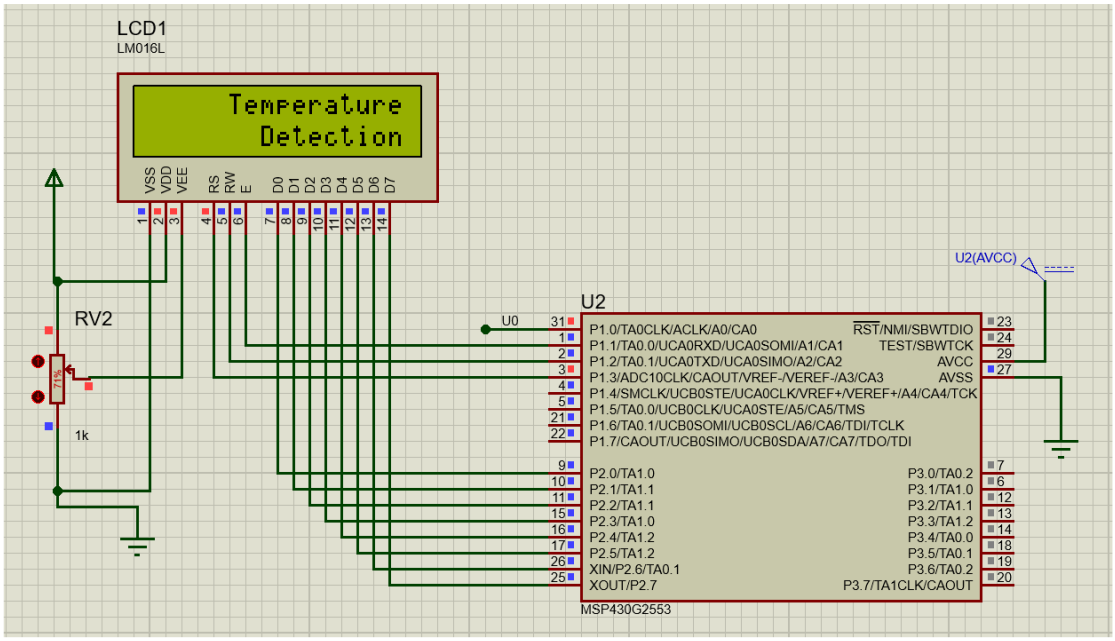


图 5-15 显示及控制模块

6 软件设计

6.1 中断流程图

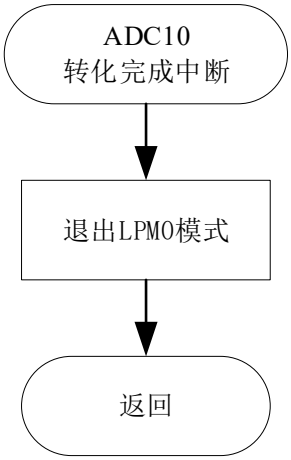


图 6-1 子程序流程图

中断子程序就是当 ADC 转换完成之后，使得 CPU 退出低功耗模式，继续执行主函数里低功耗语句后未完成的程序。

6.2 主函数流程图

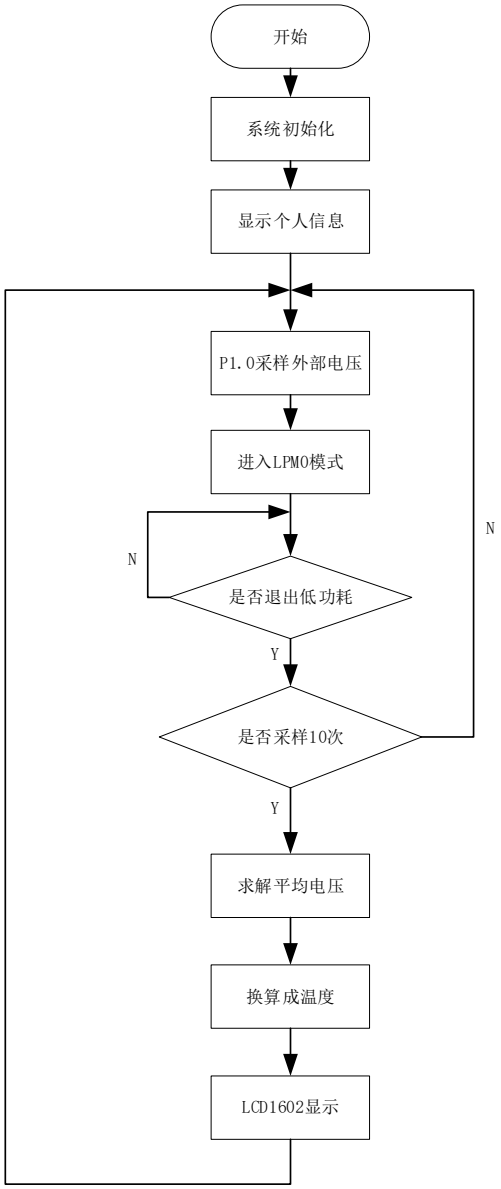


图 6-2 主函数流程图

如图 6-2 所示，为此次主程序的流程图，P1.0 复用作为外部模拟电压的输入端口。系统初始化完成后，首先显示个人的信息。之后开始使能 ADC 中断，开始 ADC 转化，CPU 进入 LPM0 模式。一旦 ADC 转化完成后，将转化后的值保存在 ADC10MEN 寄存器，同时 CPU 退出 LPM0 模式。接下来将 ADC10MEN 进行代数运算，最后将温度值显示在 LCD1602 上即可，考虑到电压采样的不稳定，采取采样满 10 次求平均才进行显示的算法。

ADC10 的初始化借助 Grace 来进行配置。

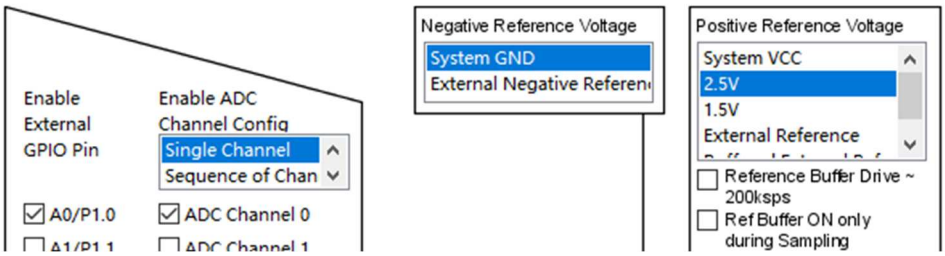


图 6-3 通过 Grace 配置 ADC10

ADC10 一共有 8 个外部输入，且全部与 IO 复用，所以首先需要使能 IO 的复用功能。如图 6-3 所示，我将 P1.0 复用作为外部模拟电压的输入，选择单通道采样，通道选择 ADC Channel 0。由于外部输入电压的最大值为 2.5V，在参考电压配置处选择内部基准电压 2.5V。

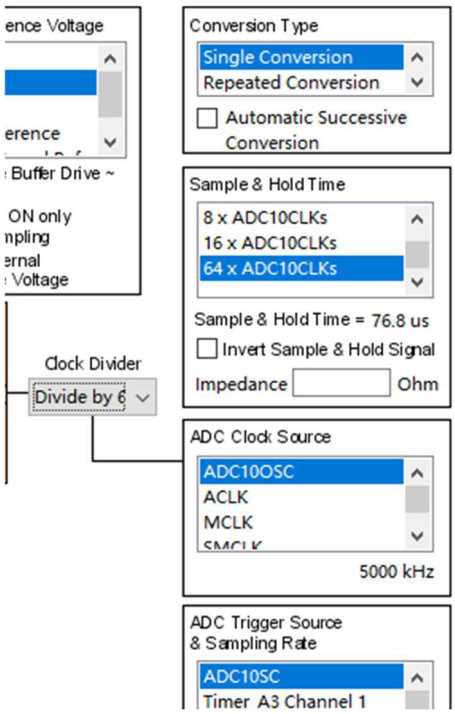


图 6-4 ADC10 相关时钟配置

选择转换模式为单次转换，“吱一声” 动一下。人工设定采样保持时间为 64 倍 ADC10CLK。ADC 的时钟选择 5MHz 专用振荡器，8 分频。采样起始信号选择软件写 ADC10SC 位，通过写 ADC10SC 位来开始一次 AD 转换。

至此，ADC10_init()配置到此结束，写程序时只需找到将 Grace 工程目录下的 ADC10_init.c，复制其中的语句即可。

6.3 显示函数流程图

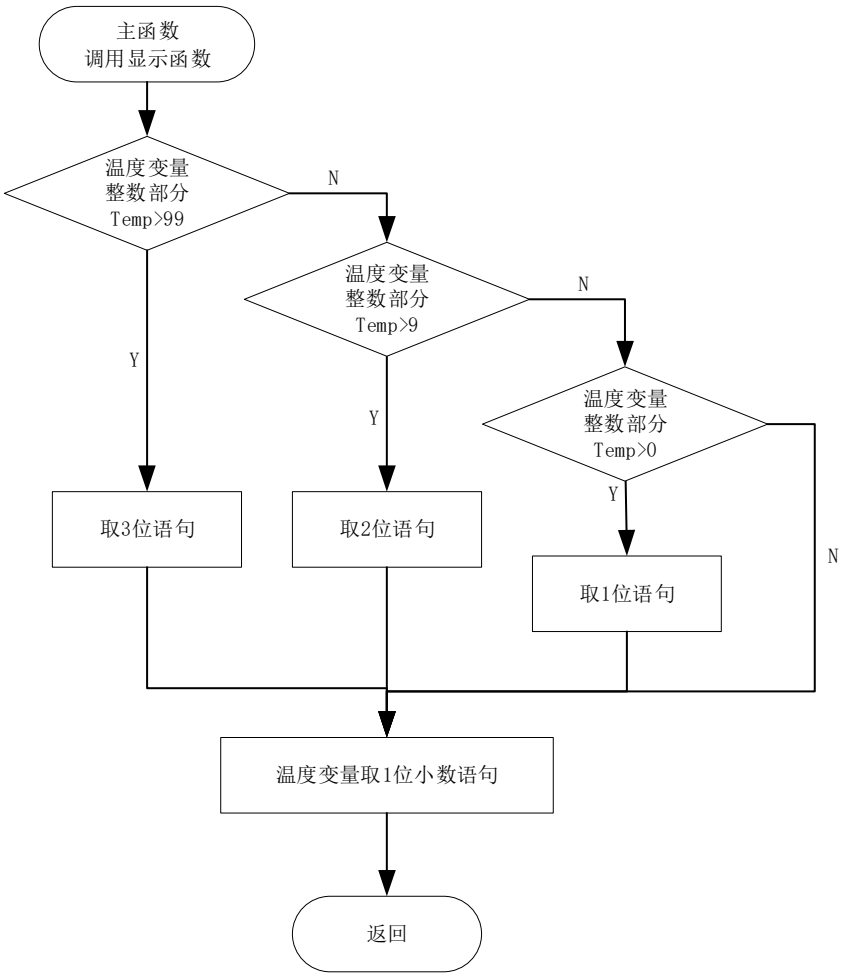


图 6-5 显示函数流程图

主函数调用显示函数之前，已经对温度的整数部分和小数部分完成处理，将整数部分保存在变量 temp 中，将小数部分保存在变量 xiaoshu 里。调用显示函数时，首先判断整数部分的温度区间，然后再执行不同的取位函数，最后将整数和小数一同显示在 LCD1602 上。

7 系统调试

7.1 生成 Hex 文件过程

首先在 Code Composer Studio 9 编写程序，进行编译，确保编译成功，没有错误，如下图 7-1 所示。

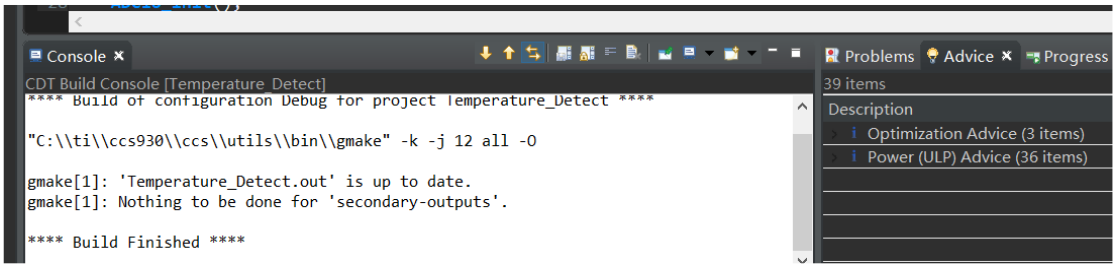


图 7-1-编译成功

之后勾选工程文件的 Hex 输出选项，将输出模式选择为 Output intel hex format，再进行一次编译，此时 Hex 文件已经输出到对应目录下。

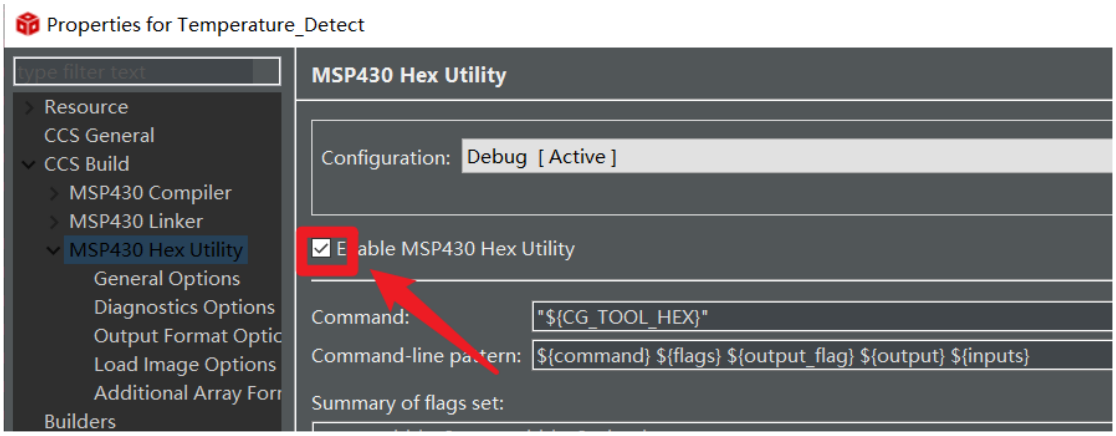


图 7-2 使能 Hex 文件输出

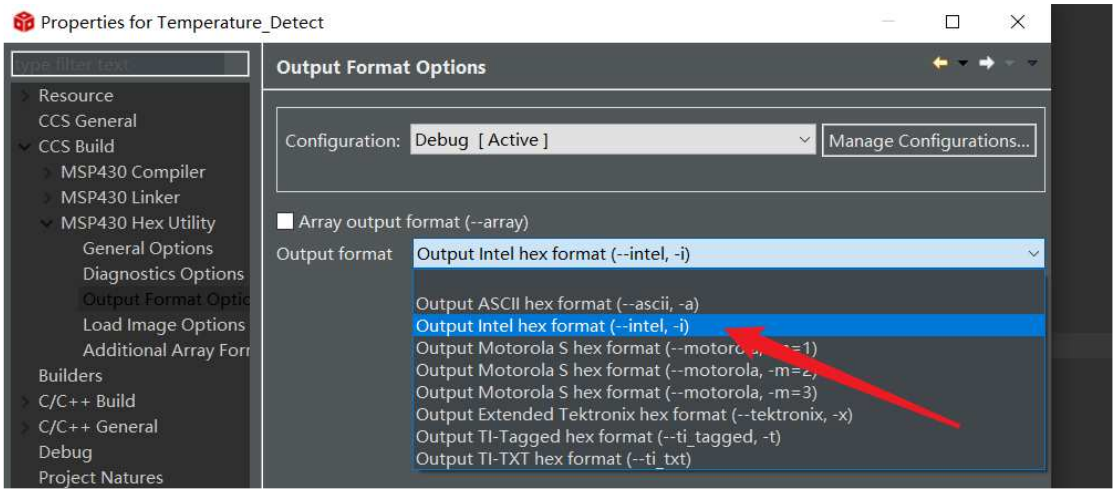


图 7-3 选择 Hex 输出模式

7.2 Proteus 装载 Hex 文件

接下来打开 Proteus 先前绘制好的仿真电路图，双击 MSP430G2553，进入芯片界面并且装入 Hex 文件，如下所示。

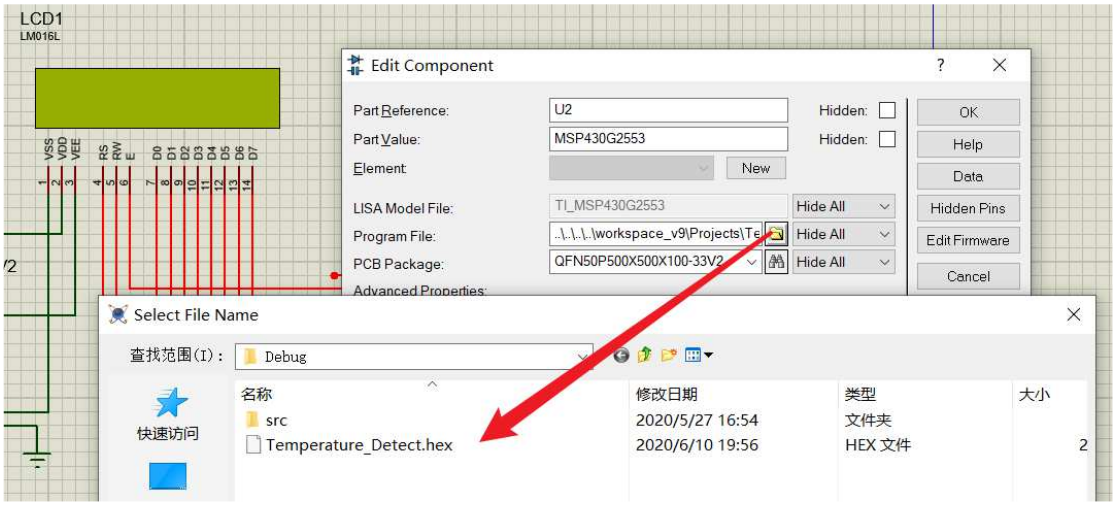
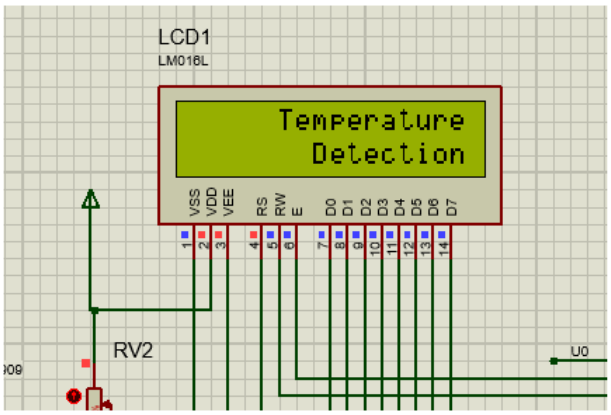
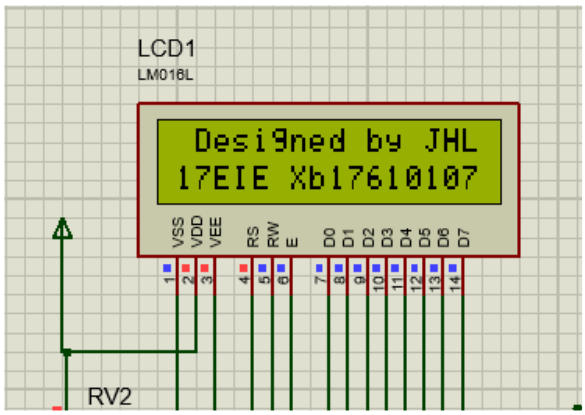


图 7-4 装载 Hex 文件

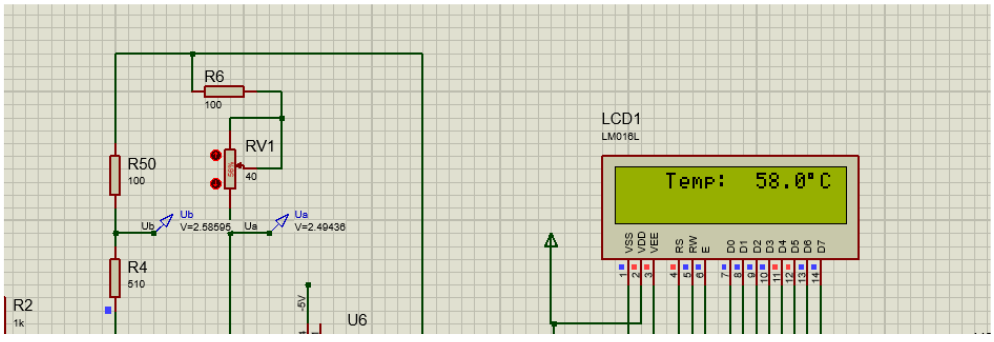
7.3 仿真过程

点击运行，系统开始运行。





系统一开始上电，显示系统信息和个人信息，功能正常，一段时间后，将会显示当前温度值，通过改变 PT100 的阻值，可以显示不同的温度。



7.4 数据记录

表格 7-1 调试数据记录

温度℃	铂热电阻变化量 单位Ω	仿真滑动变阻器比例	拟合前 LCD 显示 单位℃	拟合前误差	第一次拟合后 LCD 显示 单位℃	拟合后误差	第二次拟合后 LCD 显示 单位℃	拟合后误差
0.0	0.0	0%	0.4		0.2		0.1	
2.0	0.8	2%	2.1	5.0%	2.0	**0.0%**	1.8	-10.0%
4.0	1.5	4%	4.2	5.0%	4.1	**2.5%**	3.9	-2.5%
6.0	2.3	6%	6.2	3.3%	6.2	**3.3%**	6.1	1.7%
8.0	3.1	8%	8.2	2.5%	8.2	**2.5%**	8.1	1.3%
10.0	3.9	10%	10.2	2.0%	10.4	**4.0%**	10.3	3.0%
12.0	4.6	12%	12.2	1.7%	12.4	**3.3%**	12.4	3.3%
14.0	5.4	13%	13.1	-6.4%	13.4	**-4.3%**	13.5	-3.6%
16.0	6.2	15%	15.2	-5.0%	16.5	**3.1%**	15.7	-1.9%
18.0	6.9	17%	17.2	-4.4%	18.7	**3.9%**	17.8	-1.1%
20.0	7.7	19%	19.2	-4.0%	19.7	**-1.5%**	20.0	0.0%
22.0	8.5	21%	21.2	-3.6%	21.7	**-1.4%**	21.7	-1.4%
24.0	9.2	23%	23.2	-3.3%	23.8	-0.8%	23.8	-0.8%
26.0	10.0	25%	25.2	-3.1%	25.9	-0.4%	25.9	-0.4%
28.0	10.8	27%	27.2	-2.9%	28.0	0.0%	28.0	0.0%
30.0	11.6	29%	29.2	-2.7%	30.0	0.0%	30.0	0.0%
32.0	12.3	31%	31.2	-2.5%	32.1	0.3%	32.1	0.3%
34.0	13.1	33%	33.2	-2.4%	34.2	0.6%	34.2	0.6%
36.0	13.9	35%	35.2	-2.2%	36.3	0.8%	36.3	0.8%
38.0	14.6	37%	37.2	-2.1%	38.3	0.8%	38.3	0.8%
40.0	15.4	39%	39.2	-2.0%	40.4	1.0%	40.4	1.0%
42.0	16.2	40%	41.2	-1.9%	41.4	-1.4%	41.4	-1.4%
44.0	16.9	42%	42.3	-3.9%	43.6	-0.9%	43.6	-0.9%
46.0	17.7	44%	44.2	-3.9%	45.6	-0.9%	45.6	-0.9%
48.0	18.5	46%	46.3	-3.5%	47.7	-0.6%	47.7	-0.6%
50.0	19.3	48%	48.2	-3.6%	49.7	-0.6%	49.7	-0.6%
52.0	20.0	50%	50.3	-3.3%	51.9	-0.2%	51.9	-0.2%
54.0	20.8	52%	52.2	-3.3%	53.9	-0.2%	53.9	-0.2%
56.0	21.6	54%	54.3	-3.0%	56.0	0.0%	56.0	0.0%
58.0	22.3	56%	56.3	-2.9%	58.0	0.0%	58.0	0.0%
60.0	23.1	58%	58.3	-2.8%	60.1	0.2%	60.1	0.2%
62.0	23.9	60%	60.3	-2.7%	62.2	0.3%	62.2	0.3%

64.0	24.6	62%	62.3	-2.7%	64.3	0.5%	64.3	0.5%
66.0	25.4	64%	64.3	-2.6%	66.3	0.5%	66.3	0.5%
68.0	26.2	65%	66.3	-2.5%	67.4	-0.9%	67.4	-0.9%
70.0	27.0	67%	68.3	-2.4%	69.4	-0.9%	69.4	-0.9%
72.0	27.7	69%	69.4	-3.6%	71.6	-0.6%	71.6	-0.6%
74.0	28.5	71%	71.3	-3.6%	73.6	-0.5%	73.6	-0.5%
76.0	29.3	73%	73.4	-3.4%	75.7	-0.4%	75.7	-0.4%
78.0	30.0	75%	75.3	-3.5%	77.7	-0.4%	77.7	-0.4%
80.0	30.8	77%	77.4	-3.2%	79.8	-0.3%	79.8	-0.3%
82.0	31.6	79%	79.3	-3.3%	81.9	-0.1%	81.9	-0.1%
84.0	32.3	81%	81.4	-3.1%	84.0	0.0%	84.0	0.0%
86.0	33.1	83%	83.3	-3.1%	86.0	0.0%	86.0	0.0%
88.0	33.9	85%	85.4	-3.0%	88.1	0.1%	88.1	0.1%
90.0	34.7	87%	87.3	-3.0%	90.2	0.2%	90.2	0.2%
92.0	35.4	89%	89.4	-2.8%	92.3	0.3%	92.3	0.3%
94.0	36.2	90%	91.3	-2.9%	93.3	-0.7%	93.3	-0.7%
96.0	37.0	92%	93.4	-2.7%	95.3	-0.7%	95.3	-0.7%
98.0	37.7	94%	94.4	-3.7%	97.4	-0.6%	97.4	-0.6%
100.0	38.5	96%	96.3	-3.7%	99.4	-0.6%	99.4	-0.6%

实验测量数据如上两表所示，第一次测量温度误差在 4%上下波动。由于是仿真，判断误差来源为代替 PT100 的滑动变阻器比例取整造成，这是仿真器件本身问题。经过思考后，采取在软件中进行拟合方法弥补。第一次拟合以 LCD 显示的温度为横轴，标准温度为纵轴，绘制二阶多项式曲线，如图 7-5 所示。

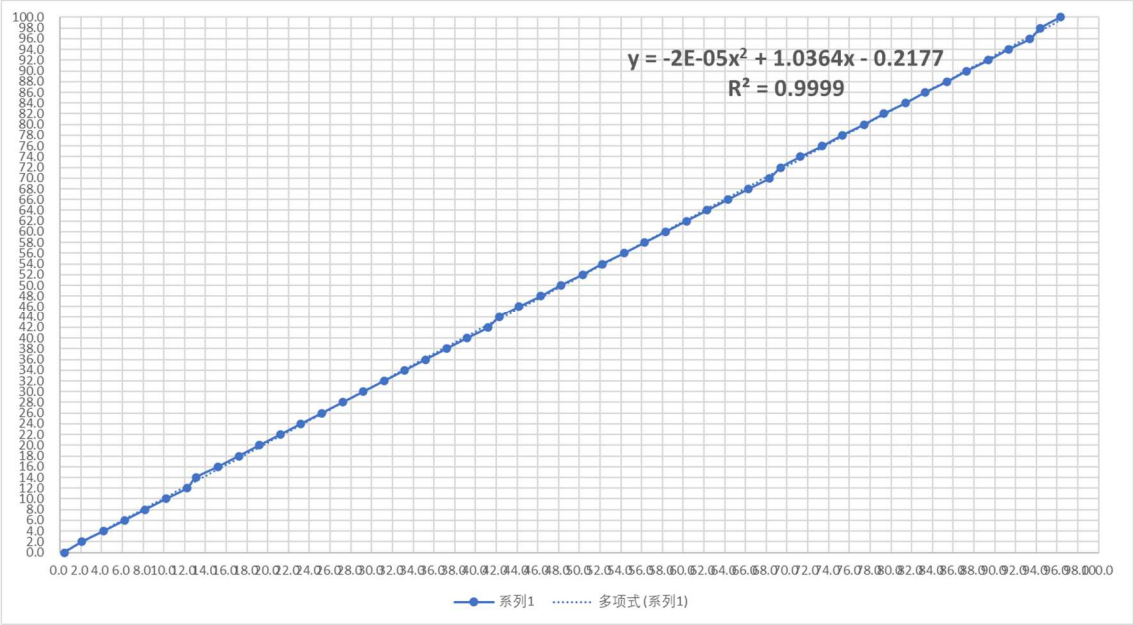


图 7-5 第一次拟合曲线

将第一次数据拟合的曲线公式：

$$y = -2 \times 10^{-5}x^2 + 1.0364x - 0.2177 \quad (7\text{-}1)$$

将公式 7-1 带入软件重新编译运行，记录第 2 组数据。由上表格 7-1 可以观察到，24℃~100℃的误差控制在 1%以下，满足要求，0℃~24℃误差还是偏大。所以取出该段数据单独进行拟合，得出第二次数据拟合的曲线公式，如下图 7-6 所示：

$$y = 0.0016x^2 + 1.0323x - 0.3848 \quad (7\text{-}2)$$

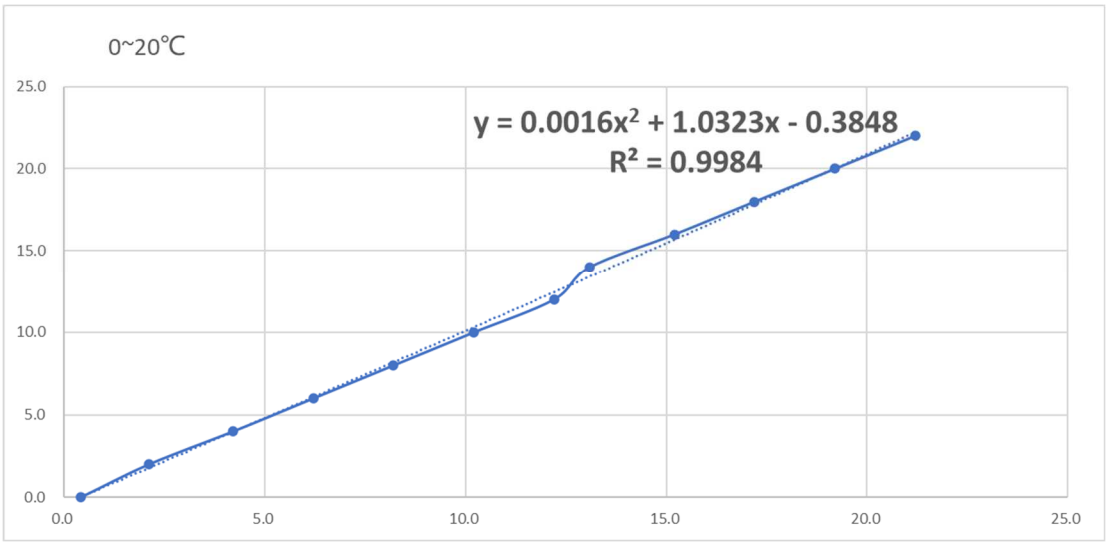


图 7-6 第二次拟合曲线

此时，观察表格 7-1，0℃~24℃误差相较之前的误差有一定程度的改善。

8 设计小结

此次的温度测量仪的设计让我对温度传感器有了更加深刻的认识。设计之初温度传感器的选型很重要，需要符合设计的要求，另外还要考虑预算。每种温度传感器的特性不一样，比如此次的 PT100 的特性是电阻会随着温度的变化而变化，我们需要设计电路将电阻的变化量转变成电压的变化量供单片机采样。

单片机采样显示的电压和实际电压之间一定会有误差，我们需要做的就是将显示值和实际电压记录下来进行拟合处理，将拟合后得到的曲线公式带入程序中进行再一次测试和数据记录。在第一次拟合效果不好的情况下，我学会进行分段拟合，即每个温度区间对应不同的拟合曲线公式，这样一来误差能够被减小。

另外，由于每次采样得到的电压值有细微的波动，我采取多次采样求平均值的方法来使电压稳定。仔细想了一下，电压光取平均值还不行，我的改进思路是将多次采样得到的电压进行排序，剔除最大和最小值后再取平均，这样能够更加稳定。

此次的传感器的课程设计，感谢李老师和季老师的指导与帮助，不管是在硬件电路设计上还是程序设计上他们都给了我特别大的启发。

附件 1 源程序代码

```
#include "msp430g2553.h"
#define SET_RS P1OUT |= BIT3
#define RST_RS P1OUT &= ~BIT3
#define SET_RW P1OUT |= BIT2
#define RST_RW P1OUT &= ~BIT2
#define SET_E P1OUT |= BIT1
#define RST_E P1OUT &= ~BIT1

unsigned char a[] = {"Temperature"};
unsigned char b[] = {"Detection"};
unsigned char c[] = {"Designed by JHL"};
unsigned char d[] = {"17EIE Xb17610107"};
unsigned char e[] = {"Temp:"};
int temp=0,xiaoshu=0;
void ADC10_init(void);
void display_normal();
void LCD_Write_Command(unsigned char com); //写命令函数
void LCD_Write_Data(unsigned char dat); //写数据函数
void lcd1602_init();
void lcd1602_display();

int main()
{
    float Vin=0,sum=0;
    char count=0;
    WDTCTL = WDTPW + WDTHOLD; //关看门狗
    lcd1602_init();
    ADC10_init();
    display_normal();
    while(1)
    {
      ADC10CTL0 |= ENC + ADC10SC; //在主循环中开启 ADC10 转换
      _bis_SR_register(CPUOFF + GIE); //休眠开总中断，等待 ADC10 转换完
成后，进入中断运行唤醒 CPU 代码

      Vin=(ADC10MEM*2.5*40)/1023;  //电压表达式，先扩大 40 倍
      sum=Vin+sum;
      count++;
      if(count==10)
      {
          Vin=sum/10.0;
```

```
            if(Vin>20)
            Vin=-0.00002*Vin*Vin+1.0364*Vin-0.2177;
            else
            Vin=0.0016*Vin*Vin+1.0323*Vin-0.3848;
            temp=(int)Vin;
            xiaoshu=(Vin-temp)*10;
            lcd1602_display();
            sum=0;
            count=0;
        }
    }
}

void display_normal()//显示学号
{
    unsigned char i=0;
    LCD_Write_Command(0x80);
    LCD_Write_Command(0x80 + 0x05); //数据指针设置，第一行显示
    __delay_cycles(1000);
    for (i = 0; i < sizeof(a); i++)
    {
        LCD_Write_Data(a[i]);
    }
    LCD_Write_Command(0x80 + 0x47); //数据指针设置，第二行显示
    __delay_cycles(1000);
    for (i = 0; i< sizeof(b); i++)
    {
        LCD_Write_Data(b[i]);
    }
    __delay_cycles(1000000);
    LCD_Write_Command(0x01);//清屏指令 1

    LCD_Write_Command(0x80);
    LCD_Write_Command(0x80 + 0x01); //数据指针设置，第一行显示
    __delay_cycles(1000);
    for (i = 0; i < sizeof(c); i++)
    {
        LCD_Write_Data(c[i]);
    }
    __delay_cycles(1000);
    LCD_Write_Command(0x40+0x80);//学号从第 2 行第 7 个开始显示,显示地址
+指针
    for (i = 0; i < sizeof(d); i++)
    {
```

```
        LCD_Write_Data(d[i]);
    }
    __delay_cycles(1000000);
    LCD_Write_Command(0x01);//清屏指令 1

    LCD_Write_Command(0x80);
    LCD_Write_Command(0x80+0x03);
    __delay_cycles(1000);
    for (i = 0; i < sizeof(e); i++)
    {
        LCD_Write_Data(e[i]);
    }

    LCD_Write_Command(0x80);
    LCD_Write_Command(0x80+0x0E);
    __delay_cycles(1000);
    LCD_Write_Data(0xdf);
    LCD_Write_Data(0x43);//显示最高位
}

void ADC10_init(void)
{
    ADC10CTL0 &= ~ENC;
    ADC10CTL0 = ADC10IE|ADC10ON | REFON| REF2_5V | ADC10SHT_0 |
SREF_1;
    //启用内部 2.5V 参考电压，采样保持时间为 4 x ADC10CLKs
    ADC10CTL1 = CONSEQ_0 | ADC10SSEL_0 | ADC10DIV_1 | SHS_0 | INCH_0;
    //ADC 时钟分频为不分频
    ADC10AE0 = 0x1;//MSP430 的 P1.0 为 ADC 输入端
}


void lcd1602_display()
{
    LCD_Write_Command(0x09+0x80);//送电压显示初地址
    if(temp>99)//100~999（3 位）
    {

        LCD_Write_Data(temp/100+0x30);//显示最高位
        LCD_Write_Data((temp/10)%10+0x30);//显示第 2 位数
        LCD_Write_Data(temp%10+0x30);//显示第 3 位数

        LCD_Write_Data(0x2E);//显示小数点
        LCD_Write_Data(xiaoshu+0x30);//显示小数
```

```
    }
    else
    {
        LCD_Write_Data(0x20);
        if(temp>9)//10~99
        {
            LCD_Write_Data(temp/10+0x30);//显示最高位
            LCD_Write_Data(temp%10+0x30);//显示第 2 位小数
            LCD_Write_Data(0x2E);//显示小数点
            LCD_Write_Data(xiaoshu+0x30);//显示小数
        }
        else if(temp>0)
        {
            LCD_Write_Data(0x20);
            LCD_Write_Data(temp+0x30);//显示最高位
            LCD_Write_Data(0x2E);//显示小数点
            LCD_Write_Data(xiaoshu+0x30);//显示小数
        }
        else
        {
            LCD_Write_Data(0x20);
            LCD_Write_Data(0+0x30);//显示最高位
            LCD_Write_Data(0x2E);//显示小数点
            LCD_Write_Data(xiaoshu+0x30);//显示小数
        }
    }
}

void lcd1602_init()
{
    P2DIR = 0xff;                   //端口初始化
    P1DIR |= BIT1 + BIT2 + BIT3;
    RST_E;
    LCD_Write_Command(0x38);//功能设置指令 6：8 位数据，双行显示，5*7 字
形
    __delay_cycles(1000);
    LCD_Write_Command(0x0c);//显示开关控制指令 4：开启显示屏，关光标，
光标不闪烁
    __delay_cycles(1000);
    LCD_Write_Command(0x06);//置输入模式指令 3：数据读写后光标右移，画
面不移动
    __delay_cycles(1000);
    LCD_Write_Command(0x01);//清屏指令 1
}
```

```
void LCD_Write_Command(unsigned char com)
{
	RST_RS;
	RST_RW;
	P2OUT = com;
	SET_E;
	__delay_cycles(800);
	RST_E;
}

void LCD_Write_Data(unsigned char dat)
{
	SET_RS;
	RST_RW;
	P2OUT = dat;
	SET_E;
	__delay_cycles(800);
	RST_E;
}

#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
	LPM0_EXIT;
}


/*#include "msp430g2553.h"
#include "LCD_128.h"
#include "HT1621.h"
#include "TCA6416A.h"

int temp=0,xiaoshu=0;
void ADC10_init(void);
void LCD_Init();
void LCD_Display();
int main()
{

	float Vin=0;
	WDTCTL = WDTPW + WDTHOLD; //关看门狗
	ADC10_init();
	LCD_Init();
```

```
    while(1)
    {
      ADC10CTL0 |= ENC + ADC10SC; //在主循环中开启 ADC10 转换
      _bis_SR_register(CPUOFF + GIE); //休眠开总中断，等待 ADC10 转换完成后，进入中断运行唤醒 CPU 代码
      //-----ADC 转换完成中断唤醒 CPU 后才执行以下代码-----
      Vin=(ADC10MEM*2.5*40)/1023;
      temp=(unsigned int)Vin; //转换为电阻值，并 100 倍处理，精确到小数点后两位
      Vin=Vin-temp;
      xiaoshu=10*Vin;
      LCD_Display();
    }
}


void LCD_Init()
{
    TCA6416A_Init();
    HT1621_init();
    //相关硬件的初始化，其中 I2C 模块的初始化由 TCA6416A 初始化函数在内部完成了， LCD_128 库函数由 HT1621 初始化函数在内部引用了
    //-----显示固定不变的 LCD 段-----
    LCD_DisplaySeg(_LCD_AUTO);
    LCD_DisplaySeg(_LCD_RUN);
    LCD_DisplaySeg(_LCD_TI_logo);
    LCD_DisplaySeg(_LCD_QDU_logo);
    LCD_DisplaySeg(_LCD_OHOM);
    LCD_DisplaySeg(_LCD_DOT4);
}


void LCD_Display()
{
        //-----清除 6 位显示数字-----
        LCD_DisplayDigit(LCD_DIGIT_CLEAR,1);
        LCD_DisplayDigit(LCD_DIGIT_CLEAR,2);
        LCD_DisplayDigit(LCD_DIGIT_CLEAR,3);
        LCD_DisplayDigit(LCD_DIGIT_CLEAR,4);
        LCD_DisplayDigit(LCD_DIGIT_CLEAR,5);
        LCD_DisplayDigit(LCD_DIGIT_CLEAR,6);
        //-----根据 temp 拆分并显示数字-----

        if(temp>99)//100~999（3 位）
```

```
            {
                LCD_DisplayDigit(temp/100,3);
                LCD_DisplayDigit((temp/10)%10,4);
                LCD_DisplayDigit(temp%10,5);
                LCD_DisplayDigit(xiaoshu,6);
            }
            else if(temp>9)
            {
                LCD_DisplayDigit(temp/10,4);
                LCD_DisplayDigit(temp%10,5);
                LCD_DisplayDigit(xiaoshu,6);
            }
            else
            {
                LCD_DisplayDigit(temp,5);
                LCD_DisplayDigit(xiaoshu,6);
            }
            HT1621_Reflash(LCD_Buffer);//-----更新缓存，真正显示-----
}



void ADC10_init(void)
{
    ADC10CTL0 &= ~ENC;
    ADC10CTL0 = ADC10IE|ADC10ON | REFON| REF2_5V | ADC10SHT_0 |
SREF_1;
    //启用内部 2.5V 参考电压，采样保持时间为 64 x ADC10CLKs
    ADC10CTL1 = CONSEQ_0 | ADC10SSEL_0 | ADC10DIV_7 | SHS_0 | INCH_0;
    //ADC 时钟分频为 7 分频
    ADC10AE0 = 0x1;//P1.0 为 ADC 输入端
    __delay_cycles(30000);
    //ADC10CTL0 |= ENC;
}



#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
    LPM0_EXIT;
}*/
```

附件 2 Proteus 仿真布局图