User’s Guide

NTC 热敏电阻至 TMP6 线性热敏电阻更换指南

摘要

线性热敏电阻使用的硬件和软件设计方法与线性正温度系数或负温度系数热敏电阻的常用方法相同。本文档的目的是介绍将 NTC 热敏电阻系统转换为线性热敏电阻的硬件和软件设计方法。

内容

商标

1 引言

热敏电阻可用于温度检测应用，而不是数字温度传感器，因为热敏电阻成本更低、占地面积更小而且响应速度更快。有关热敏电阻优于数字温度传感器的更多信息，可参阅*使用热敏电阻的检测温度*。

1.1 NTC 热敏电阻与 TMP6 线性热敏电阻系列

市场上的两种主要热敏电阻是 NTC 和线性热敏电阻。NTC 热敏电阻的工作原理是，随着温度的升高或下降而改变电阻。线性热敏电阻在有电流经过时，其有效电阻根据温度发生变化。两种热敏电阻最大的区别是，NTC 热敏电阻的阻值随温度的升高而呈对数下降，而线性热敏电阻的有效电阻随温度升高呈线性增加。下图显示了典型的 10kΩ NTC 热敏电阻与 TI TMP6 线性热敏电阻系列（特别是 TMP6131 DEC 封装）在电阻温度特性上的差异。

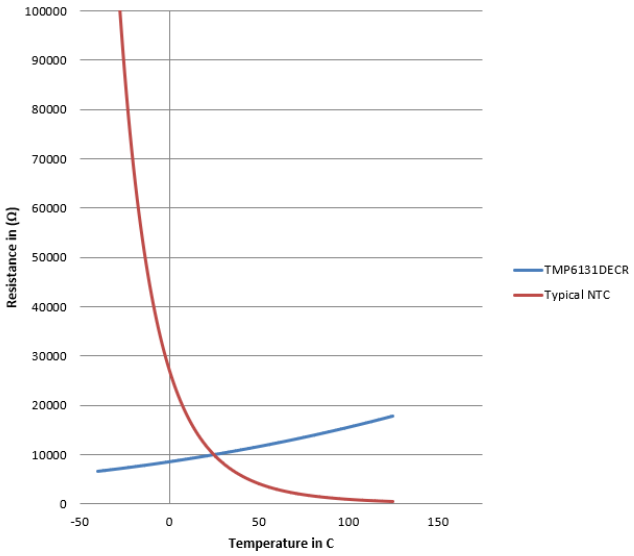

图 1-1. 典型 NTC 热敏电阻与 TMP6131DEC 热敏电阻的 RT 曲线

1.2 NTC/线性热敏电阻 TCR

温度系数电阻 (TCR) 可定义为器件的电阻随温度变化而发生的变化。使用方程式 1 计算测量的 TCR（单位为 ppm/°C）。

温度特性电阻 = ((R2 – R1)/R1×(T2 – T1))×10^6 (1)

由于 TMP61 热敏电阻具有线性，因此该器件在宽工作温度范围内具有一致的 TCR。与纯阻性器件 NTC 热敏电阻不同，TMP61 热敏电阻的有效电阻受器件中电流的影响，并且有效电阻会随温度变化而变化。TMP61 热敏电阻的 TCR (25°C) 为 6400ppm/°C，在整个温度范围内的典型 TCR 容差是 0.2%。但是，根据 TMP61 热敏电阻的偏置方式，该值略有变化。

TMP6 热敏电阻有很多优势，而 NTC 热敏电阻乍一看来具有在室温下电阻发生变化的优势。利用 TI 提供的简单增强功能，TMP6 可实现相同甚至更好的精度。

1.3 NTC 与硅基线性热敏电阻的利弊对比

设计中温度检测电路的配置将取决于许多因素。虽然电压偏置热敏电阻的结构更简单，但电流偏置热敏电阻的动态电压范围更宽，稳定性更强，而且输出电压 V_{TEMP} 在温度范围内的精度更高。

典型的 NTC 热敏电阻在极端温度之间的容差范围为（1% 至 5%），尽管这对于某些 NTC 热敏电阻来说比较高，但 TMP61 热敏电阻在极端温度 [-40°C，+150°C] 之间的容差范围为（0.5% 至 1.5%），可参见下面的图 1-2。

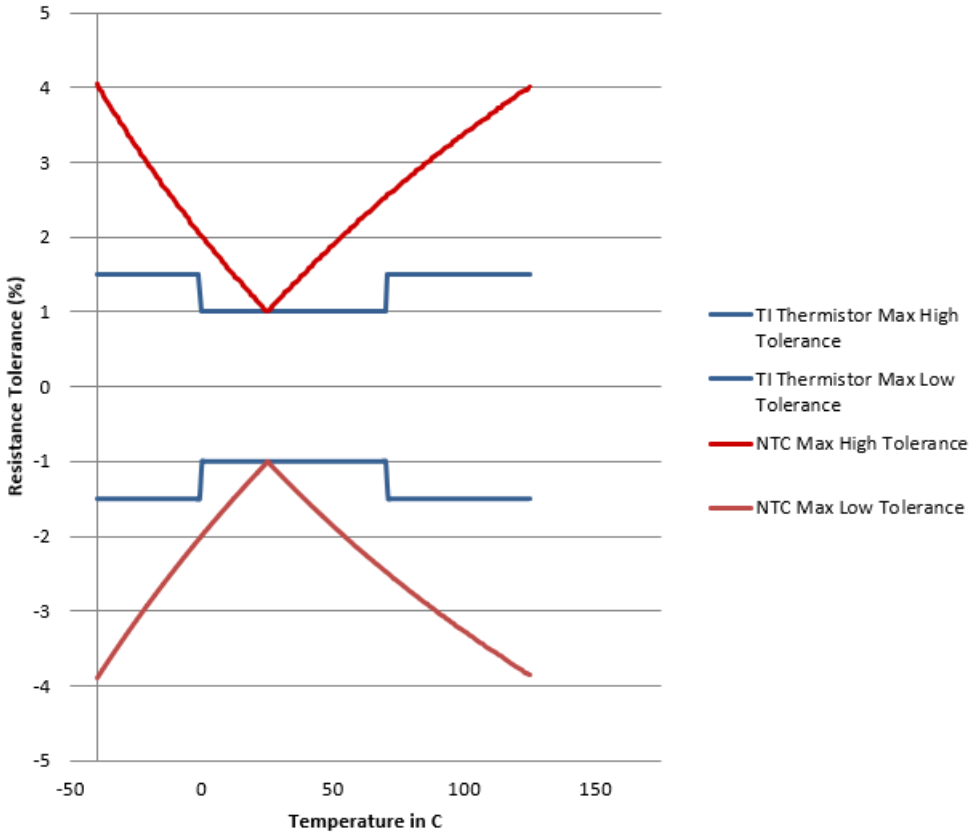

图 1-2. 典型 NTC 热敏电阻与 TMP6 线性热敏电阻的电阻容差

热敏电阻的最大优势是设计简单。在电压偏置或电流偏置网络中，可通过测量热敏电阻上的压降或经过热敏电阻的电流来进行检测。热敏电阻电路的主要配置是电压偏置（如图 1-3 中的分压器配置所示）或电流偏置（如图 1-4 所示）。输出电压 V_{TEMP} 可送入 ADC，以便在 MCU 中对温度数据进行数字化处理。

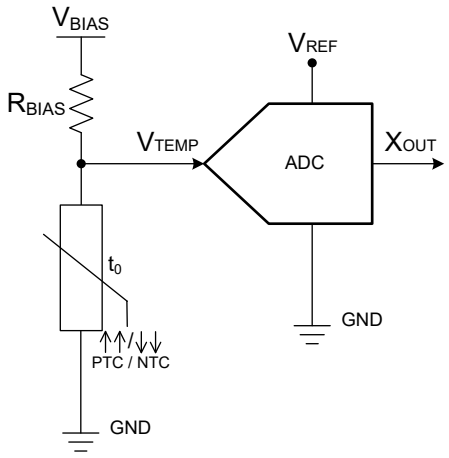

图 1-3. 热敏电阻分压器电路

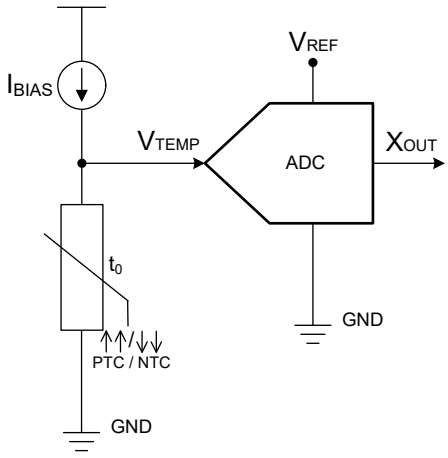

图 1-4. 热敏电阻电流源电路

观察 NTC 热敏电阻的特性时，应该注意，当环境温度很高时，很难通过 NTC 热敏电阻得知温度，因为它们在较高的温度下灵敏度很低。为了对传入的温度数据进行更简单的软件处理，可能需要对 NTC 热敏电阻的 R-T 表进行

线性化。对于 NTC 热敏电阻，这通常需要一个与热敏电阻并联的定值电阻。图 1-5 显示了典型 NTC 热敏电阻分压器电路，其并联电阻为 R_P 电阻，偏置电阻为 R_{BIAS} 。典型 NTC 热敏电阻分压器、典型线性热敏电阻分压器和具有并联电阻的 NTC 热敏电阻分压器的电压响应比较如 图 1-6 所示。

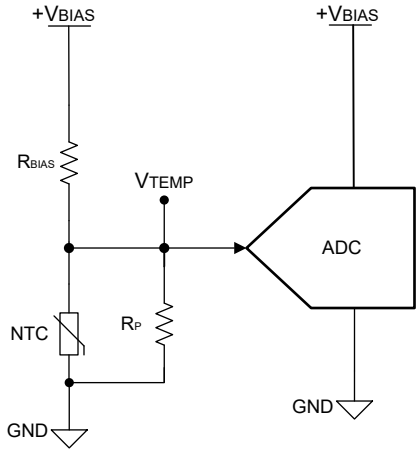

图 1-5. 具有并联电阻的 NTC 热敏电阻

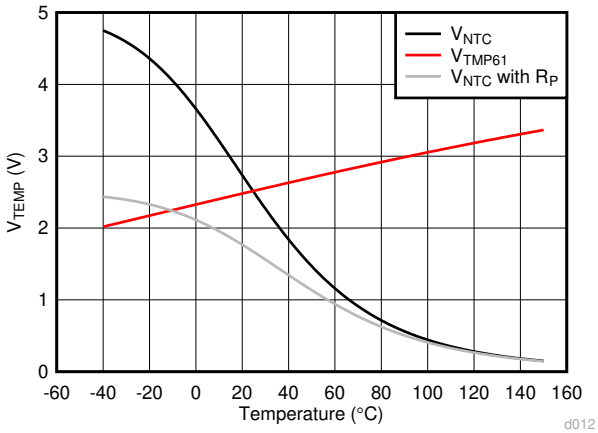

图 1-6. 带和不带线性电阻的 NTC 热敏电阻与 TMP61 热敏电阻温度电压

在有限的温度范围内，这样做是有效的，但使 NTC 热敏电阻在整个温度范围内实现线性化是比较困难的，而且仅靠硬件是无法实现的。反之，线性热敏电阻是使用线性 R-T 特性曲线制造的，因此不需要与热敏电阻并联的定值电阻。

系统设计人员可能需要对热敏电阻进行校准，以确保在器件工作范围内的准确性。为了在此范围内实现更高的精度， NTC 热敏电阻需要在不同的温度值（例如 -40°C、25°C 和 125°C ）进行多点校准，因为 NTC 热敏电阻是非线性的。同样的理论也适用于线性热敏电阻，但由于其是线性的，所以只需要单点校准（例如在 25°C 下 ）。)因此，使用线性热敏电阻将节省制造时间，并减少对 MCU 进行温度处理所需的内存。有关校准 TMP6 线性热敏电阻器件系列的更多信息，请参阅 Thermistor Design Tool。

1.4 TMP6 精度

按照本文档中包含的设计步骤，同时考虑包括 Rbias 容差/PPM、测试的温度范围、多项式/LUT、过采样、滤波和校准在内的各种设计，下表汇总了 TMP6 线性热敏电阻可实现的精度（单位为 °C）。

Bias Resistor	Temperature Range	With room temperature calibration	
		Polynomial 12bit	Polynomial + (Oversampling or Filtering)
± 0.1%, 10 PPM	0 to 70°C	+/- 0.74°C	+/- 0.54°C
± 0.1%, 10 PPM	Full Range	+/- 0.92°C	+/- 0.72°C
± 0.5%, 25 PPM	0 to 70°C	+/- 1.30°C	+/- 1.10°C
± 0.5%, 25 PPM	Full Range	+/- 1.60°C	+/- 1.40°C
± 1%, 100 PPM	0 to 70°C	+/- 1.94°C	+/- 1.74°C
± 1%, 100 PPM	Full Range	+/- 2.50°C	+/- 2.30°C

图 1-7. TMP6 热敏电阻精度（单位为 °C）

表中的精度值是在理想 Vbias 5V 和理想 Vref 且无 ADC 误差的假设下得到的。

2 典型 NTC 热敏电阻的设计注意事项

请以此设计过程为示例：

表 2-1. 电压偏置温度检测电路示例的系统要求

温度范围		偏置电压
T_{MIN}	T_{MAX}	V_{BIAS}
-40°C	125°C	5V

2.1 电压偏置的 NTC 热敏电阻网络

基于 NTC 热敏电阻的温度检测网络结构的最简单结构如 图 2-1 所示。对于 ADC，将使用 12 位的分辨率和 5 VDC 的参考电压。12 位分辨率是可接受的测量分辨率，而 5VDC 基准电压简化了电源轨要求。这些设计决策产生了如 图 2-2 所示的电压响应。

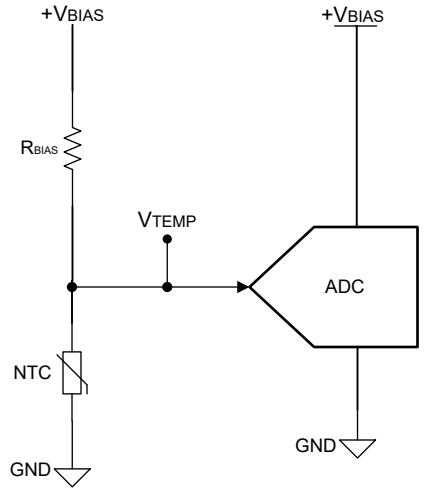

图 2-1. 简单的电压偏置 NTC 热敏电阻原理图

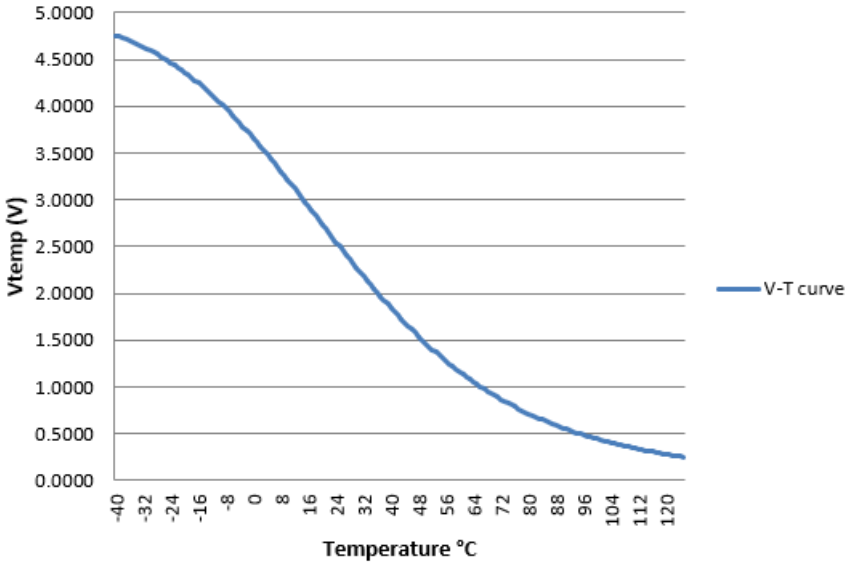

图 2-2. NTC 热敏电阻电压响应 (5V)

虽然这种电压偏置网络非常简单，但降低了所需的物料清单成本。温度输出 V_{TEMP} 显示了极端温度下的非线性行为。

2.2 引脚排列/极性

TMP6 线性热敏电阻是 2 个引脚的器件，因此是 NTC 热敏电阻的引脚对引脚替代品。由于 TMP6 热敏电阻采用 0402 (1005mm) 和 0603 (1608mm) 封装，因此这些器件的封装也是兼容的。应注意的是，0603 器件也可以用在 0805 封装上。有关更多详细信息，请参阅节 6.2。

需要注意的是，TMP6 热敏电阻有极性（参阅图 2-3）。

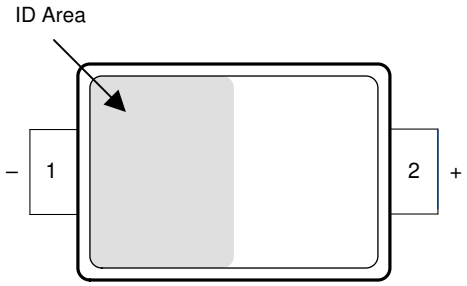

图 2-3. TMP6 热敏电阻 DYA 封装 2 引脚 SOT-5X3 底视图（斜置）

TI 使用特殊的硅工艺，其中掺杂水平和有源区器件控制关键特性（TCR 和标称电阻）。由于极化端子，该器件具有有源区和衬底。将正极端子连接到最高电压电位，将负极端子连接到最低电压电位，以确保正常工作。如果焊盘上的电压相反，热敏电阻将正常显现，直到 p-n 结开始传导（大概 0.6V），然后热敏电阻 I-V 特性将崩解。如果配置相反，这会导致器件端子上的测量值为 0.6V。

2.3 将 NTC 热敏电阻硬件设计转换为 TMP6 线性热敏电阻设计

对于简单的 NTC 热敏电阻设计，从 NTC 热敏电阻转换为 TMP6 线性热敏电阻很简单。唯一的硬件改动是交换 NTC 热敏电阻和 TMP61 热敏电阻，而 R_{BIAS} 可保持原样。最终的转变如下方图 2-4 所示。

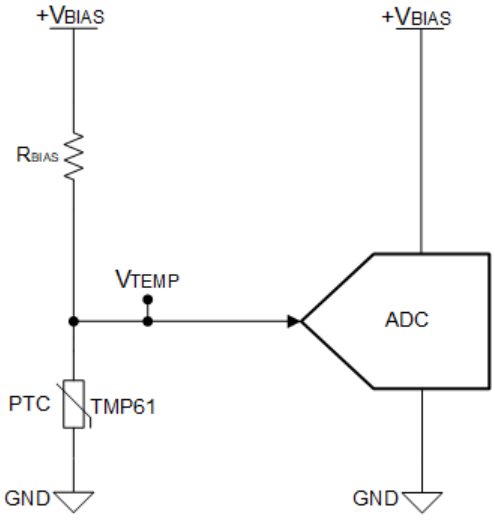

图 2-4. TMP6 线性热敏电阻分压器电路

上述电路产生以下电压响应。请注意，电压响应是正的。

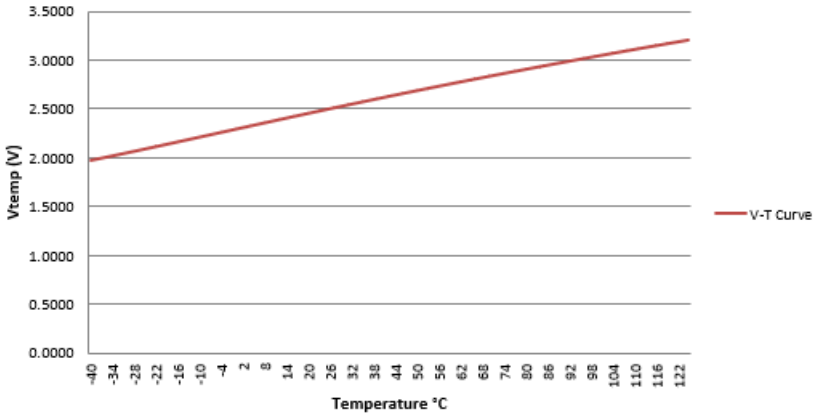

图 2-5. TMP6 线性热敏电阻正电压响应

正电压响应可以用于新的设计，但匹配原始 NTC 热敏电阻电路生成的负电压响应可能对旧的设计有用。为了使 TMP6 线性热敏电阻产生负电压响应，偏置电阻和 TMP6 线性热敏电阻必须交换位置。重新设计后，您可以看到下图中产生的负电压响应。

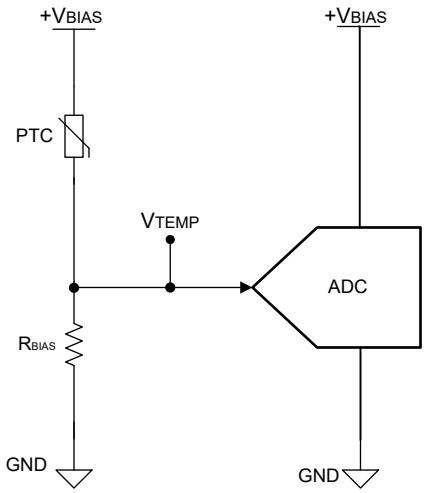

图 2-6. 用于负电压响应的 TMP6 线性热敏电阻电路

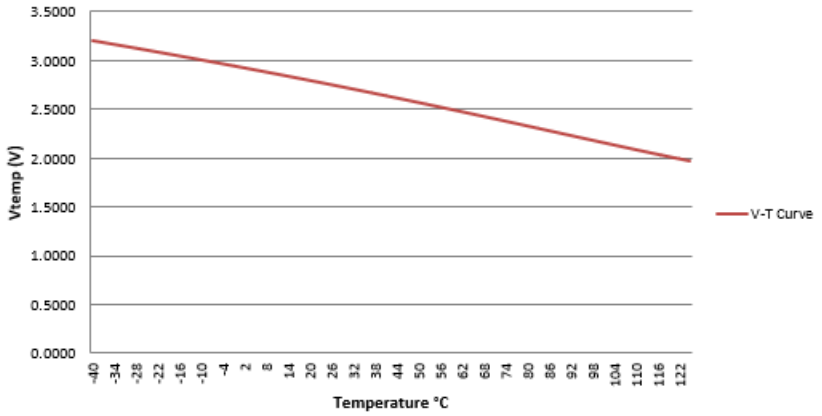

图 2-7. TMP6 线性热敏电阻负电压响应

2.4 简单的查找表

硬件设计完成后，可以使用 TI 的 Thermistor Design Tool 生成用于在 MCU 内部进行电阻-温度转换的软件。了解如何从 Thermistor Design Tool 生成代码片段。

我们将使用以下设计参数：

5V、10kΩ 偏置电阻，12 位 ADC，TMP6131DYA。

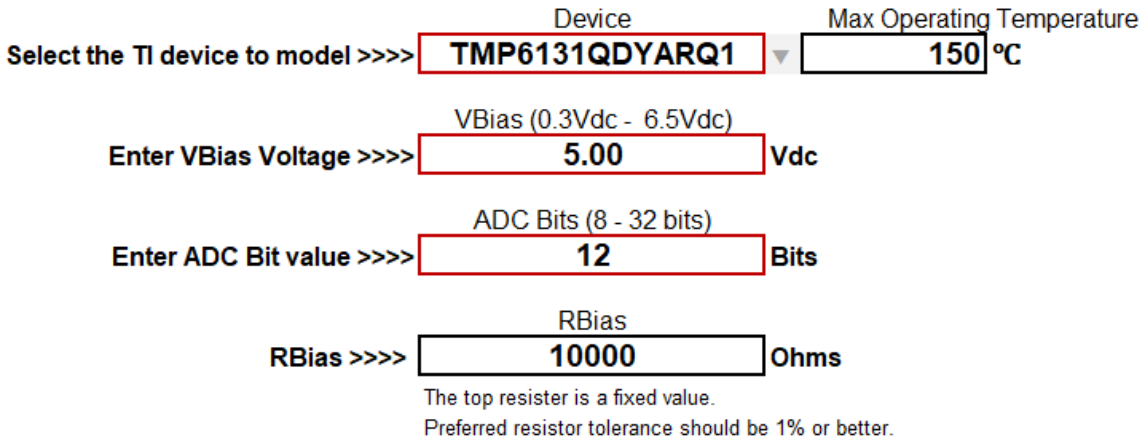

图 2-8. Thermistor Design Tool 参数

接下来，可进入 *Device Resistance Tables* 选项卡。在这里，我们可以找到 1°C 和 5°C 阶跃查找表。此页面根据最初设置的设计参数动态填充电阻表。 5°C 查找表如下所示。

5 °C Steps Example Code

Line #	Temperature (°C)	Min Resistance (Ω)	Typical Resistance (Ω)	Max Resistance (Ω)
1	-40	6501	6600	6699
2	-35	6710	6812	6914
3	-30	6927	7032	7138
4	-25	7151	7260	7369
5	-20	7384	7496	7609
6	-15	7624	7740	7856
7	-10	7871	7991	8111
8	-5	8126	8250	8374
9	0	8431	8517	8602
10	5	8703	8791	8878
11	10	8981	9072	9163
12	15	9267	9361	9454
13	20	9560	9657	9754
14	25	9861	9961	10060
15	30	10169	10272	10375
16	35	10484	10590	10696
17	40	10807	10916	11025
18	45	11137	11250	11362
19	50	11475	11591	11707
20	55	11820	11940	12059
21	60	12173	12296	12419
22	65	12534	12661	12787
23	70	12902	13033	13163
24	75	13212	13413	13614
25	80	13595	13802	14009
26	85	13985	14198	14411
27	90	14385	14604	14823
28	95	14792	15018	15243
29	100	15209	15440	15672
30	105	15634	15872	16110
31	110	16068	16313	16558
32	115	16512	16764	17015
33	120	16965	17224	17482
34	125	17428	17694	17959
35	130	17901	18174	18446
36	135	18384	18664	18944
37	140	18878	19165	19453
38	145	19382	19677	19972
39	150	19897	20200	20503
40	155	#N/A	#N/A	#N/A
41	160	#N/A	#N/A	#N/A
42	165	#N/A	#N/A	#N/A
43	170	#N/A	#N/A	#N/A

图 2-9. 5°C 查找表

在此页面上，我们还可以找到查找表的 C 代码。

Example of 5°C step C code **Copy the content of the box below, then paste into your C code**

```
int THRM_Res_S_Table[43][2] = {                                    // THRM short table
                                {   -40 ,   6600 }  ,
                                {   -35 ,   6812 }  ,
                                {   -30 ,   7032 }  ,
                                {   -25 ,   7260 }  ,
                                {   -20 ,   7496 }  ,
                                {   -15 ,   7740 }  ,
                                {   -10 ,   7991 }  ,
                                {    -5 ,   8250 }  ,
                                {     0 ,   8517 }  ,
                                {     5 ,   8791 }  ,
                                {    10 ,   9072 }  ,
                                {    15 ,   9361 }  ,
                                {    20 ,   9657 }  ,
                                {    25 ,   9961 }  ,
                                {    30 ,  10272 }  ,
                                {    35 ,  10590 }  ,
                                {    40 ,  10916 }  ,
                                {    45 ,  11250 }  ,
                                {    50 ,  11591 }  ,
                                {    55 ,  11940 }  ,
                                {    60 ,  12296 }  ,
                                {    65 ,  12661 }  ,
                                {    70 ,  13033 }  ,
                                {    75 ,  13413 }  ,
                                {    80 ,  13802 }  ,
                                {    85 ,  14198 }  ,
                                {    90 ,  14604 }  ,
                                {    95 ,  15018 }  ,
                                {   100 ,  15440 }  ,
                                {   105 ,  15872 }  ,
                                {   110 ,  16313 }  ,
                                {   115 ,  16764 }  ,
                                {   120 ,  17224 }  ,
                                {   125 ,  17694 }  ,
                                {   130 ,  18174 }  ,
                                {   135 ,  18664 }  ,
                                {   140 ,  19165 }  ,
                                {   145 ,  19677 }  ,
                                {   150 ,  20200 }  ,
                                {   155 ,  #N/A  }  ,
                                {   160 ,  #N/A  }  ,
                                {   165 ,  #N/A  }  ,
                                {   170 ,  #N/A  }  ,
};  // Remove all lines associated with unused temperatures and update the table length
```

图 2-10. 5°C 查找表 C 代码

通过简单地实施查找表，TMP6 热敏电阻的精度和典型 NTC 热敏电阻的精度在整个工作温度范围内的比较如下。

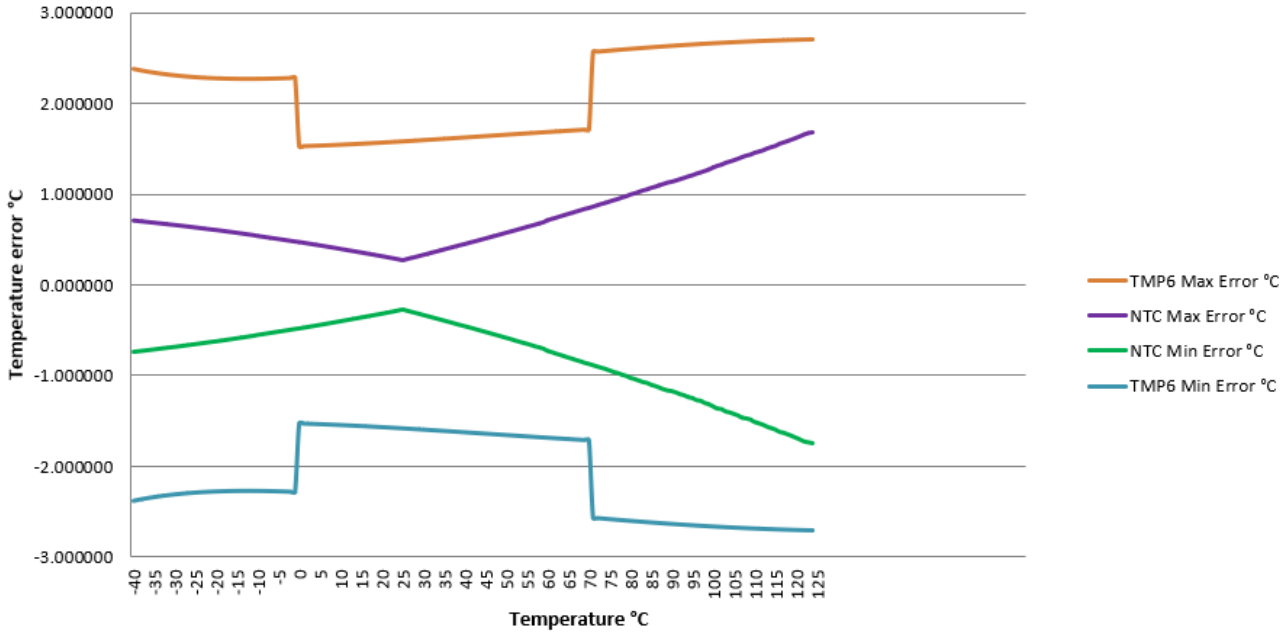

图 2-11. 未校正的热敏电阻精度比较

3 软件变化

当热敏电阻电路的输出由 ADC 检测并转换为数字信息以供 MCU 处理时，输出必须转换为温度值。常用的软件 R-T 转换方法之一是使用查找表。这需要预先在一张表中填充电阻以及这些电阻的相关温度值。代码将通过在点之间插值，确定哪个电阻值更贴近期望的温度值。这种方法使 R-T 表的设置非常简单，但对 MCU 的闪存要求很高，并且需要冗长的数组解析程序。由于容差变化和温度系数等系统误差可能导致偏离理想的 R-T 表，因此该方法也可能是不准确的。

第二种可节省内存的温度转换方法是下述 Steinhart-Hart 方程。该方程可实现为温度检测代码，以映射到热敏电阻的 R-T 曲线：

$$1/T = A + B \times \ln(R) + C \times \ln^3(R) \quad (2)$$

其中 T 是温度（开尔文度）；R 是测量的电阻值；A、B、C 是计算得出的系数。

然而，使用 TMP6 线性热敏电阻时，下面的 4 阶多项式回归模型是一种更好的转换算法：

$$T = A_4 \times R^4 + A_3 \times R^3 + A_2 \times R^2 + A_1 \times R + A_0 \quad (3)$$

其中 T 是温度（摄氏度）；R 是测量的电阻值；$A_{0\text{-}4}$ 是计算得出的多项式系数。

由于该器件具有线性，因此这种近似法效果很好，但对于非线性的 NTC 热敏电阻不起作用。在 Thermistor Design Tool 中可以生成多项式回归模型的多项式系数。

3.1 固件设计注意事项

计算 TI TMP6 线性热敏电阻产品系列温度值的推荐方法是 4 阶多项式回归法。这是最准确、最快速的温度计算方法，而且不需要查找表。我们进入 *4th Order Polynomial TMP vs. Res* 选项卡，发现“Quartic Function”和“Regression”两个模型，它们提供计算器件温度/电阻的 4 阶多项式。

4th order polynomial

Quartic Function

$R(\Omega) = A4*(T\text{^}4) + A3*(T\text{^}3) + A2*(T\text{^}2) + A1*T + A0$ — *Resistance as a function of temperature*

	Coefficients	
	8.516625E+03	A0
	5.404381E+01	A1
	1.497209E-01	A2
	-5.699002E-05	A3
	7.918690E-07	A4

Temperature	25.0	°C	Enter temperature here to get the resistance
	77.0	°F	
	298.2	°K	
Resistance	9961	Ohms	Calculated resistance from the 4th order polynomial above

Regression

$T°C = A4*(R\text{^}4) + A3*(R\text{^}3) + A2*(R\text{^}2) + A1*R + A0$ — *Temperature as a function of resistance*

	Coefficients	
	-2.694795E+02	A0
	5.094962E-02	A1
	-3.143945E-06	A2
	1.182160E-10	A3
	-1.810821E-15	A4

Resistance	*10052*	Ohms	Enter resistance to get the temperature
Temperature	26.6	°C	Calculated temperature from the 4th order polynomial regression above
	79.8	°F	
	299.7	°K	

图 3-1. 4 阶多项式

这里提供了可直接实施到系统设计软件的 C 代码，以计算所选 TI TMP6 线性热敏电阻的温度。

EXAMPLE OF C CODE Copy the content of the box below, then paste into your C code

```
// 4th order polynomial equations to calculate the temperature of the thermistor----------
// C code examples only  (NOTE: this code example is based on floating point math)

unsigned int RBias =                    10000 ;     // set the value of the top resistor
float VBias =                            3.30 ;     // set the VBIAS voltage
unsigned int ADC_BITS =                  4096 ;     // set the number of bits based on you ADC (2^# of ADC Bit Value)
float VTEMP = 0;                                    // set up the variable for the measured voltage
float THRM_RES = 0;                                 // setup the variable for the calculated resistance
float THRM_TEMP = 0;                                // setup the variable for the calculated temperature


float Thermistor(int raw_ADC)                       // send the ADC bit value to the calculation function
{
  // THRM calculations - 4th order polynomial regression
  VTEMP = 0;                                        // reset these variables to zero in order to recalculate the new factors
  THRM_RES = 0;                                     // reset these variables to zero in order to recalculate the new factors
  THRM_ADC = raw_ADC

  float THRM_A0 =          -2.694795E+02 ;
  float THRM_A1 =           5.094962E-02 ;
  float THRM_A2 =          -3.143945E-06 ;
  float THRM_A3 =           1.182160E-10 ;
  float THRM_A4 =          -1.810821E-15 ;

  VTEMP = (VBias/ADC_BITS) * THRM_ADC;              // calculate volts per bit then multiply that times the ADV value
  THRM_RES = VTEMP/((VBias - VTEMP)/RBias);         // calculate the resistance of the thermistor
  THRM_TEMP = (THRM_A4 * powf(THRM_RES,4)) + (THRM_A3 * powf(THRM_RES,3)) + (THRM_A2 * powf(THRM_RES,2)) + (THRM_A1 * THRM_RES) + THRM_A0;   // 4th order regression to get temperature
  return THRM_TEMP;

}
```

图 3-2. 4 阶多项式 C 代码

3.2 过采样

按先进先出 (FIFO) 的顺序对温度测量进行过采样和求均值可以提高测量分辨率和信噪比。虽然此做法也适用于 12 位或 14 位 ADC，但建议在采用 12 位以下的 ADC 时使用。在 ADC 读取位值而且您的代码计算温度之后，您可以将该值存储在一个数组中。当新值进入数组时，最老的样本被丢弃，而所有其他样本都被移动到下一个对应的单元，从而形成 FIFO。均值法可应用于温度转换中使用的任何值，例如温度、ADC 位值、分压器电压甚至计算得出的电阻。每 8 次过采样，分辨率将增加 2 位。16 次过采样会将 10 位 ADC 提高到 14 位的分辨率。下图展示了两种如何实现过采样的方法。有关更多信息，请参见 T 热敏电阻设计工具 的 *平均值*选项卡。

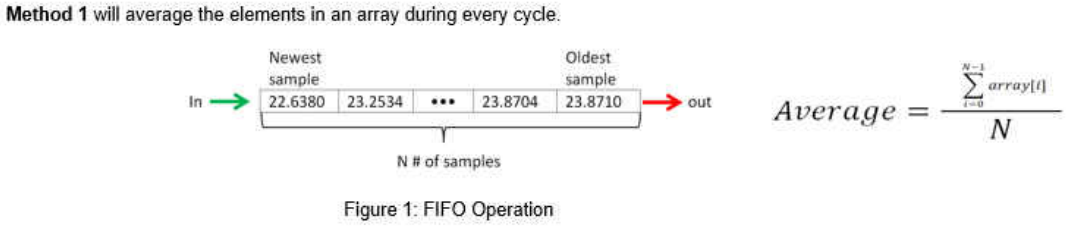

图 3-3. 过采样方法 1

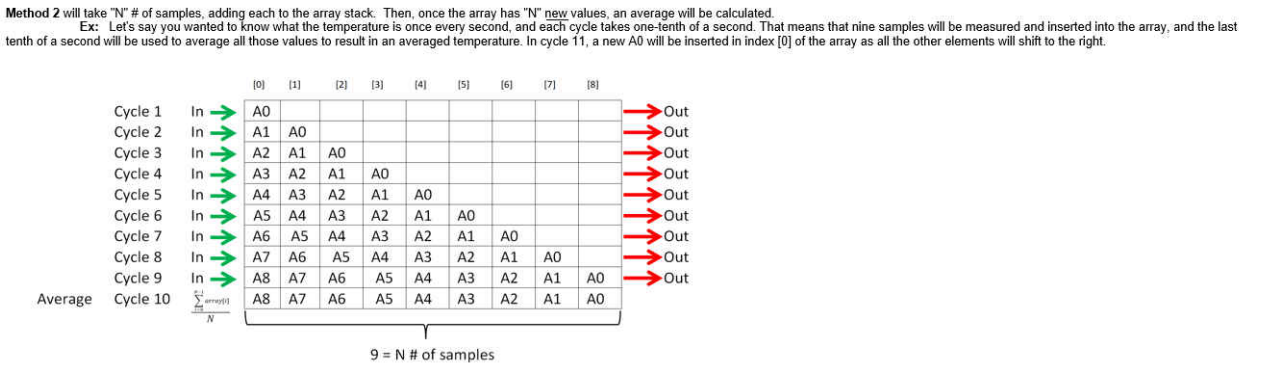

图 3-4. 过采样方法 2

Thermistor Design Tool 中提供的方法 1 的 C 代码示例可参见图 3-5。

EXAMPLE OF C CODE **Copy the content of the box below, then paste into your C code**

```
// FIFO averaging of the data sample at predetermined intervals and averaged across the last (x) samples----------
// C code examples only

// (1) Method one will read the ADC and average the last "N" values as set in the "#define Tmp_1_length"
//  FIFO setup of the temporary arrays and define the filter depth (samples to average)
#define Tmp_1_length 16                    // Sample length for the averaging filter for oversampling
float Tmp_1_array[Tmp_1_length];           // The FIFO arrays for averaging the ADC value

float ADC_AVG = 0;                         //  This is the averaged ADC value over (x) samples
float ADC_Value = 0;                       // this is the most recent ADC value captured
int i = 0;                                 //  set to 0
float sum_array_1 = 0;                     //  set to 0

void FIFO_AVG(void)
{
    // FIFO to average thermistor temperature //
    i = 0;                                 // reset to 0
    sum_array_1 = 0;                       // reset to 0
    for(i = 0; i < Tmp_1_length - 1; i++)  // shift the array as a FIFO and drop the last data value
    {
      Tmp_1_array[i] = Tmp_1_array[i + 1];  // Makes all the array indexes equal to the number after them
    }
    Tmp_1_array[Tmp_1_length - 1] = ADC_Value;      // add the new value to the beginning of the array
    for(i = 0; i < Tmp_1_length; i++)      // sum the array
    {
      sum_array_1 += Tmp_1_array[i];       // add all of the array elements
    }
    ADC_AVG = sum_array_1 / Tmp_1_length;  // divide the sum of the array to get an average
}

//  Read the ADC and place the bit value into ADC_Value
// Call the ADC_AVG function to get the last ADC value added and averaged into the array
FIFO_AVG();
//  The ADC average value will be placed into ADC_AVG register
```

图 3-5. 过采样 C 代码示例

在下方图 3-6 和图 3-7 中，可看到使用 12 位 ADC 过滤 TMP6331 热敏电阻原始数据的影响。经过 32x 过采样后，数据与参考探头更加吻合。

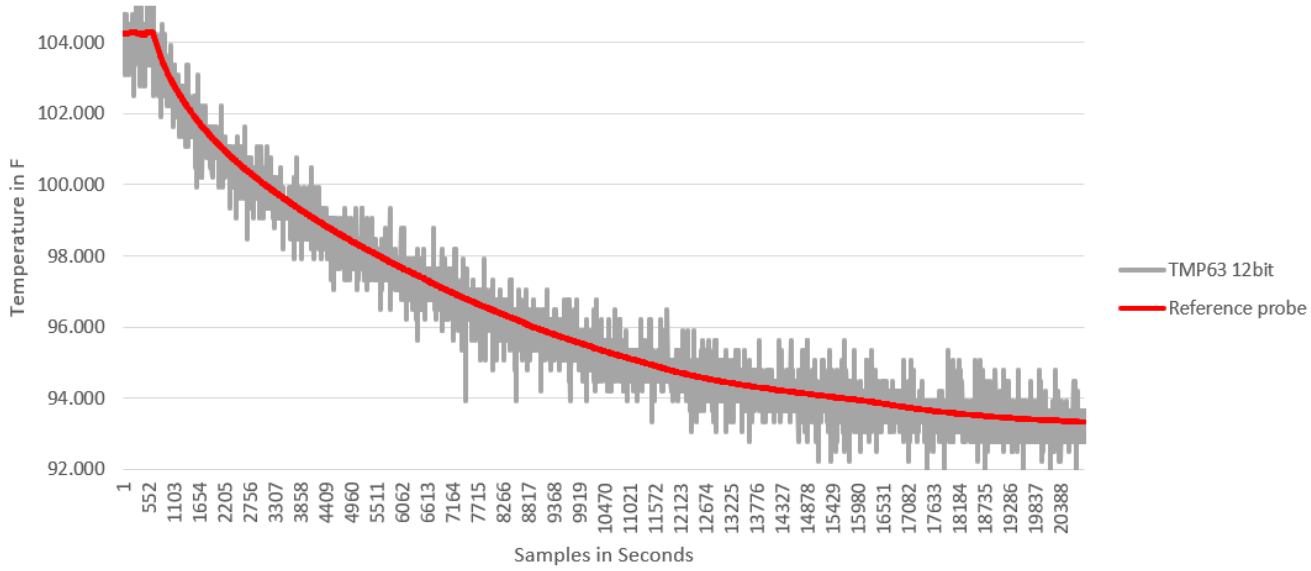

图 3-6. 无过采样的 TMP6331 热敏电阻数据

图 3-7. 应用 32x 过采样的 TMP6331 热敏电阻数据

3.3 硬件和软件中的低通滤波

噪声可导致温度测量错误，因此许多设计人员选择在硬件中添加 RC 滤波器来滤除系统产生的噪声。但是请勿在硬件中进行滤波，您可以使用此方法来避免增加额外的电阻和电容，从而增大电路板并节省成本。采用软件滤波器可以通过实时改变 Alpha 值来更好地控制滤波器的响应。此外，具有设置过滤温度的能力能够极大地缩短启动时间。

基于固件的低通滤波器需要三个变量：

1. Alpha
2. 测量温度
3. 滤后温度

Alpha：此变量控制过滤掉的噪声量。

测量温度：此变量存储计算出来的滤波前的温度读数。

滤后温度：此变量存储温度值经过滤波器后得出的温度。

固件低通滤波的执行方程如下：

低通滤波器方程：

$$Y(n) = (1 - \alpha) \times Y(n-1) + (\alpha \times X(n)) \quad (4)$$

其中

- Y = 滤后温度
- α = Alpha
- X = 测量温度

简化...

$$Y(n) = Y(n-1) - (\alpha \times (Y(n-1) - X(n))) \quad (5)$$

再简化...

$$Y(n) = \text{是} - (\alpha \times (\text{是} - X)) \text{ 表示 Filtered\_Temp = Previous\_Filtered\_Temp - (Alpha * (Previous\_Filtered\_Temp - Meas\_Temp))} \quad (6)$$

在 Thermistor Design Tool 的 *Low-Pass Filter* 选项卡上，您可以调整 Alpha 和每秒采样量的值，以更改滤波器。在图 3-8 中，您可以看到 Alpha 设置为 0.8。图 3-9 中的结果显示，实施低通滤波后得到的温度数据与原始数据相比变化不大。

TI Device TMP6131QDYARQ1

	Adjust the Alpha to change the filter		
Alpha (α)	0.8	0.001<α<1	Enter the Alpha value (The smaller the Alpha value the more the filter reduces variations)
Meas_Temp	22.714 C		preset value To increase accuracy and reduce start up time of the filter set the Meas_Temp variable to the first temperature value measured.
Filtered_Temp	22.714 C		preset value To increase accuracy and reduce start up time of the filter set the Filtered_Temp variable to the first temperature value measured.
SPS (Samples Per Second)	1		Enter the sample rate per second
LPF (Low Pass Filter)	0.8 Hz		

图 3-8. Alpha 值为 0.8 的低通滤波器设置

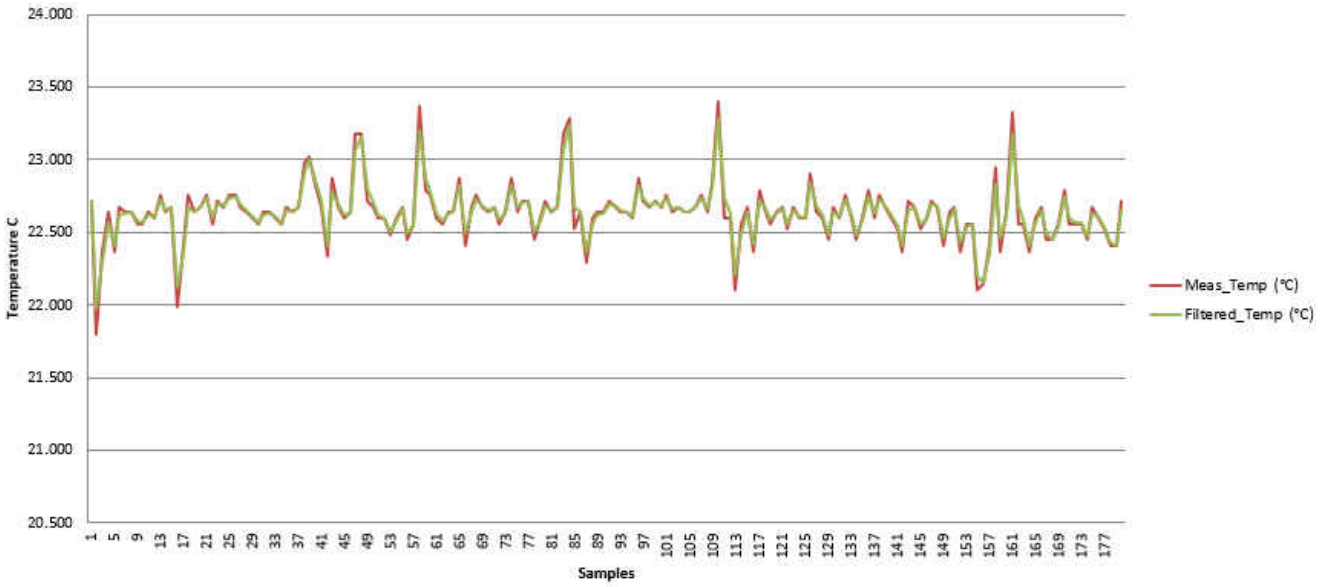

图 3-9. Alpha 值为 0.8 的低通滤波器响应

调整 Alpha 值后，Alpha 值为 0.2 时的滤后响应如下所示：

图 3-10. Alpha 值为 0.2 的低通滤波器响应

由此可见，滤后温度数据远比原始数据更加平滑。

Thermistor Design Tool 中的低通滤波器 C 代码示例可参见图 3-11。

EXAMPLE OF C CODE **Copy the content of the box below, then paste into your C code**

```
// Low Pass Filter using an Alpha and preset values to create a filtered or smoothed temperature response ----------
// C code examples only

//Floating Point

float Meas_Temp = 2! // It's assumed that you have read the ADC and calculated your temperature
float Filtered_Temp = // Set a default filtered temperature value equal to the first measured value to reduce the ramp time
float Alpha = 0.1;       // .001<α<1, adjust as required

// Insert this line of code after the temperature calculation (remember to add the offset to your temperature values)
Filtered_Temp = Filtered_Temp - (Alpha * (Filtered_Temp - Meas_Temp));
```

图 3-11. 低通滤波器 C 代码示例

经过采样并实施低通滤波算法后，TMP6 热敏电阻的性能得到了改善。校正后的 TMP6 热敏电阻和典型的 NTC 热敏电阻的对比如下：

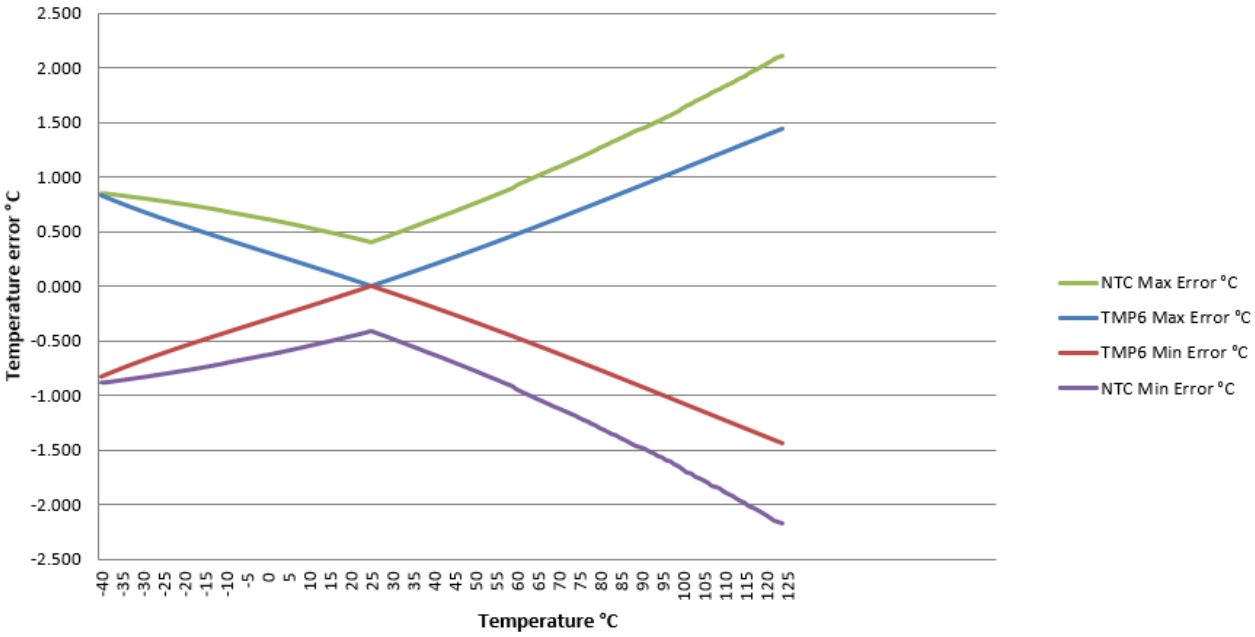

图 3-12. 校正后的热敏电阻精度比较

有关更多信息，请参见 热敏电阻设计工具 的低通滤波器选项卡。

3.4 校准

图 3-13 显示多个 TMP6 热敏电阻器件在温度范围内的精度。从图中可看出，每个器件的精度各不相同，但都具有线性度。由于 TMP6x 热敏电阻具有高线性度而且在不同器件中有相似性，因此我们无需额外的成本，即可消除热敏电阻、VCC、VREF、ADC LSB 和偏置电阻的容许误差，并在整个温度范围内得到非常一致的精度。您将需要高精度的温度参考。TMP117 是高精度、低功耗的数字温度传感器，在 -55°C 到 +150°C 的范围内 NIST 可追踪精度为 ±0.3°C（最大值）。固件编程到 UUT（被测器件）后，需要在该过程中添加一些自动化操作。加入校准后，现在各器件在整个温度范围的±0.3°C 内排列（参阅图 3-14）。

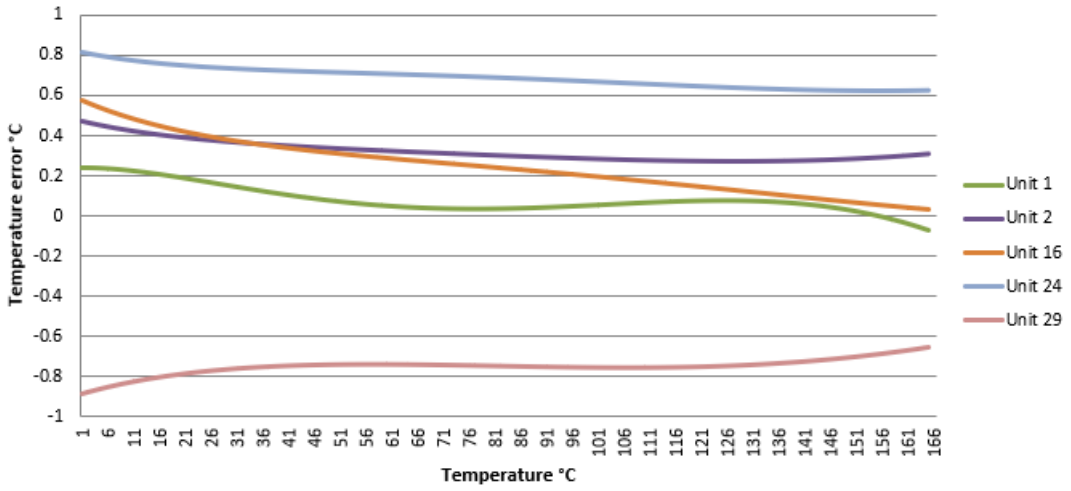

图 3-13. TMP6 热敏电阻潜在温度误差

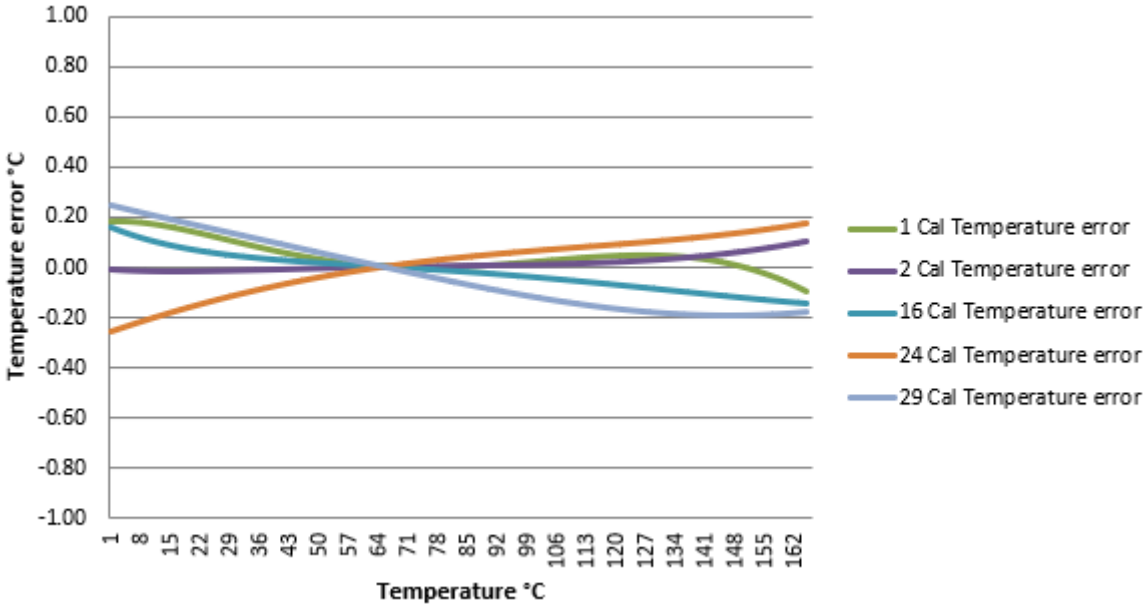

图 3-14. TMP6 热敏电阻潜在温度误差（校正后）

过程：生产编程器件将固件编程到 UUT。UUT 首次上电后，UUT 将使用其 ADC 测量 PCB 上 TMP6x 热敏电阻的 VSensevoltage，计算温度,并将温度写入温度寄存器。此时，无论是生产编程器件还是 UUT 都将读取 UUT 中的温度寄存器并读取外部温度参考，从外部温度参考中减去测量的温度，并将此值写入偏移寄存器。对于所有的后续温度测量，UUT 会将测量的 TMP6 热敏电阻温度值与偏移寄存器值相加，以得出校正后的最终温度。

假定：UUT 和温度参考处于环境温度下。UUT 将在固件编程期间低功率运行。在上电后立即测量 TMP6x 热敏电阻的温度。根据上电后测量的第一个温度和温度参考计算偏移。

实施单点校准后，NTC 热敏电阻和 TMP61 线性热敏电阻的精度对比显示在图 3-15 中。

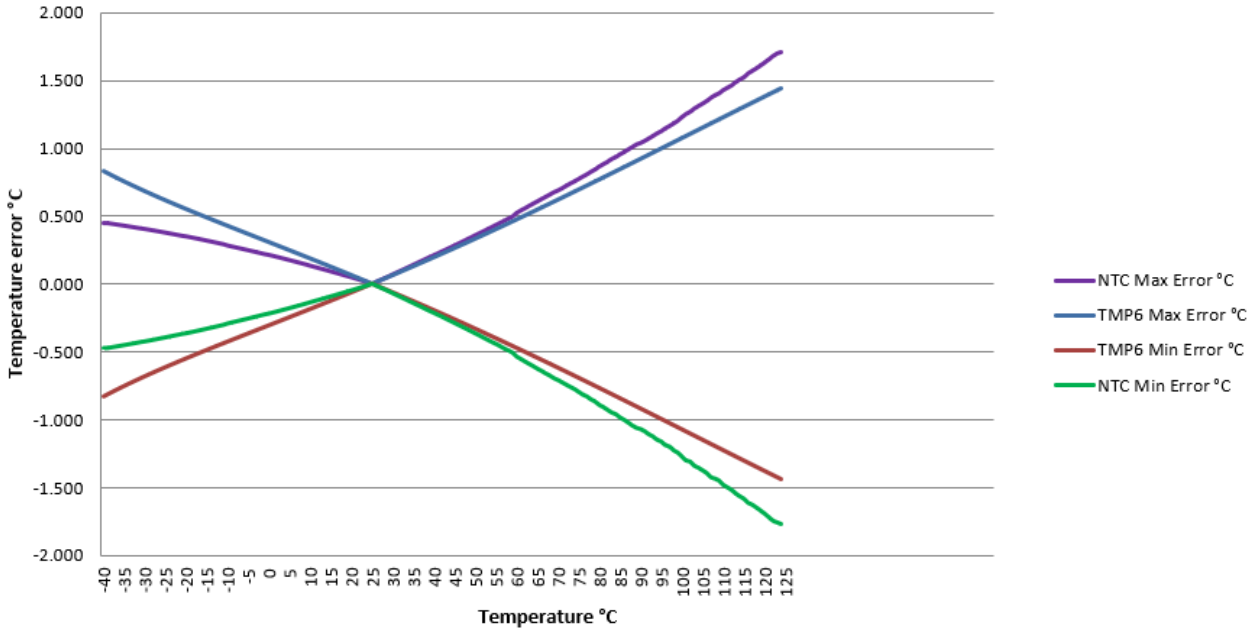

图 3-15. 偏移校正后的热敏电阻温度总误差对比

4 满量程电压输出的设计注意事项

虽然带 TMP61 热敏电阻的分压器电路转换简单，但其没有利用 ADC 输入的满量程优势。有两种推荐的设计方法来增加热敏电阻分压器电路的满量程范围。一种方法是使用电流源，另一种方法是在设计电路时加入运算放大器。后续部分将有助于设计这些电路。

4.1 简单的电流偏置

TMP61 线性热敏电阻的一种高级配置是使用电流偏置网络。与上述设计步骤类似，电流偏置的 TMP61 线性热敏电阻网络的最简单模型如下所示 图 4-1。对于 ADC，将使用 12 位的分辨率和 5V 的基准电压 V_{BIAS} 。

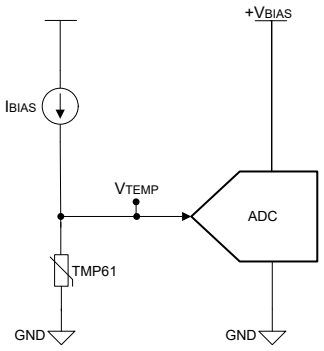

图 4-1. TMP6 线性热敏电阻电流源电路

在 ADC 参考电压为 5V 的情况下，使用 200µA 电流源对 TMP61 热敏电阻进行偏置是一个不错的选择。产生的 Vtemp 电压摆幅在 -40°C 到 125°C 为 1.3226V 到 3.58V，如下方模拟图所示。

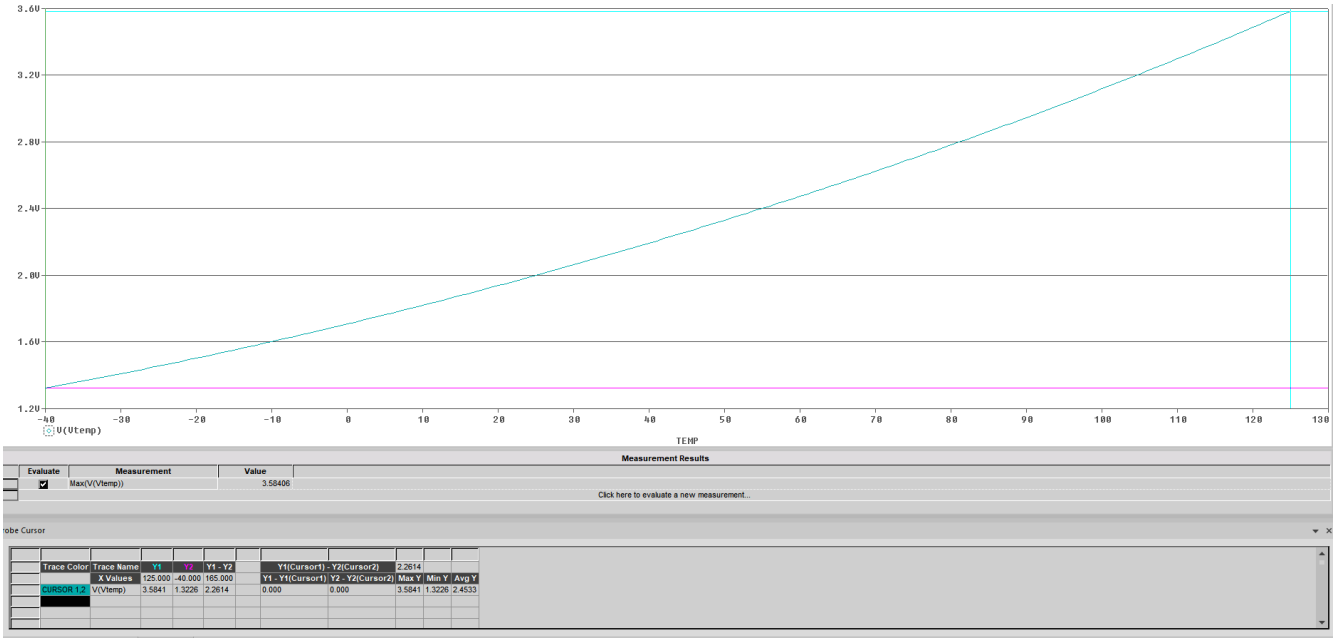

图 4-2. 使用 200µA 电流源的 TMP61 热敏电阻电压摆幅

在采用 TMP61 热敏电阻时，改变电流将增加输出响应的动态范围。当采用 50μA 至 400μA 之间的电流对 TMP61 热敏电阻进行偏置时，其输出响应可如下所示：

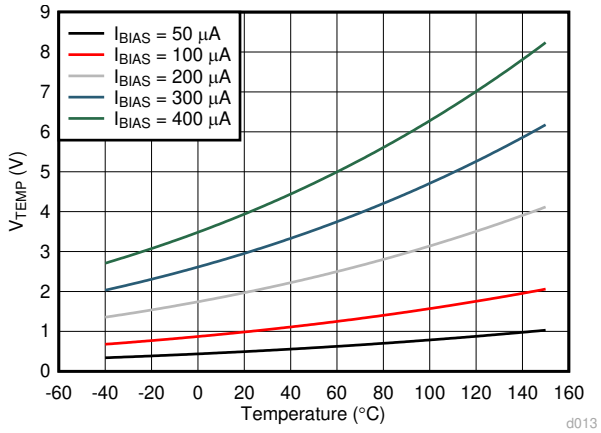

图 4-3. 不同电流源下的 TMP61 热敏电阻温度电压

偏置电流的适当值取决于系统中 ADC 的基准电压。您需要选择该值，以便在 ADC 输入的满量程内优化动态范围。大多数情况下，推荐 200μA。对于较低的系统电流，可使用标称电阻较高的 TI TMP6 线性热敏电阻，例如 TMP64 (47kΩ) 和 TMP63 (100kΩ) 热敏电阻。 TMP64 (47kΩ) 和 TMP63 (100kΩ) 热敏电阻的最佳电流源分别是 42.533μA 和 20μA。

4.2 有效电压偏置

有源热敏电阻网络的硬件改动还涉及其他几个步骤。与上面的简单设计类似，第一处硬件改动是交换 NTC 和 TMP61 热敏电阻，R_{BIAS} 可以保持原样。

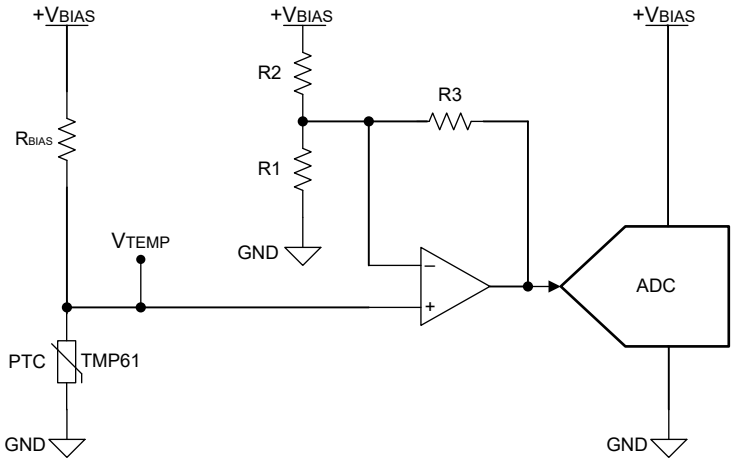

图 4-4. 带有运算放大器的 TMP6 线性热敏电阻原理图

按照指南的设计步骤，最终的电阻值 R_{BIAS} = 10kΩ，R1 = 6.84kΩ，R2 = 6.25kΩ，R3 = 10kΩ。生成的 V_OUT 范围为 0.129V 至 4.86V，处于运算放大器的线性工作范围内而且分辨率更高。这些设计决策导致图 4-5 如下所示的电压响应。

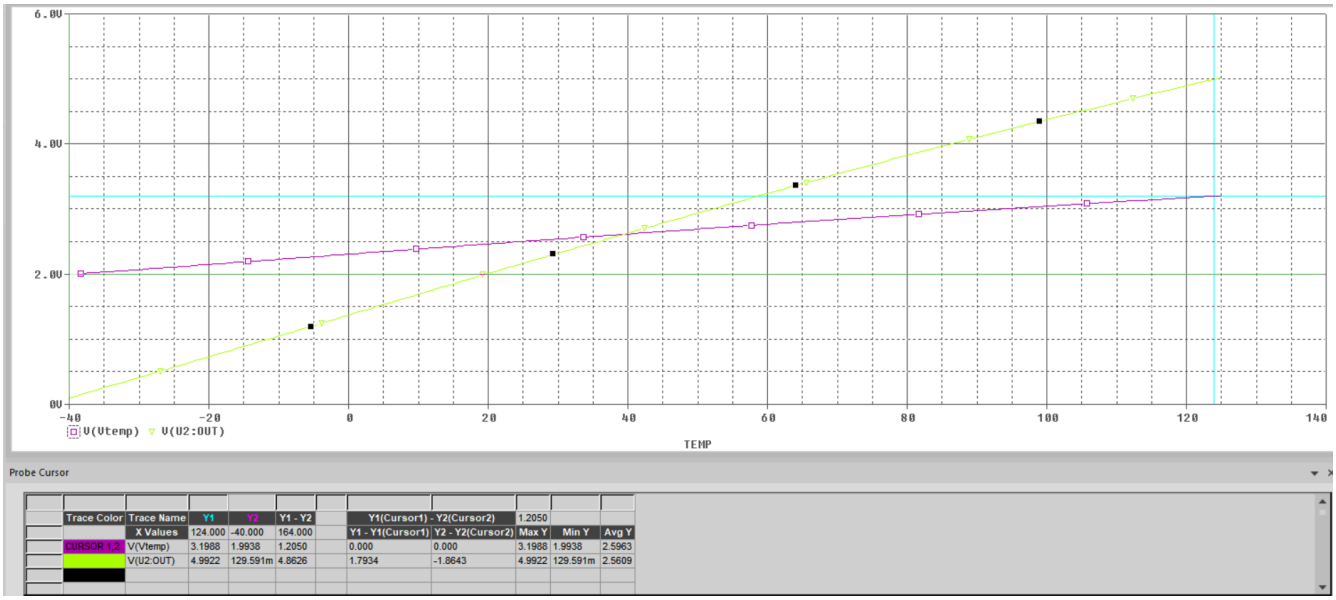

图 4-5. 带有运算放大器的 TMP6 线性热敏电阻电压响应

5 结论

总之，对比 NTC 热敏电阻和线性热敏电阻时，NTC 热敏电阻在室温下似乎具有更高的分辨率。然而，深入研究时可以发现使用线性热敏电阻相比 NTC 热敏电阻有许多额外的好处，例如 TI 的 TMP61 线性热敏电阻系列。由于器件是以引脚对引脚方式替代的，所以可切换 NTC 热敏电阻和 TI TMP61 线性热敏电阻的元件。当考虑过采样、低通滤波、校准等考虑事项，还可以使用 TI 的 TMP6 热敏电阻系列在整个温度范围实现更高的精度。

6 其他资源/注意事项

6.1 恒流源设计

使用 TI 的 Thermistor Design Tool，我们可以设计适合检测热敏电阻温度的恒流源。

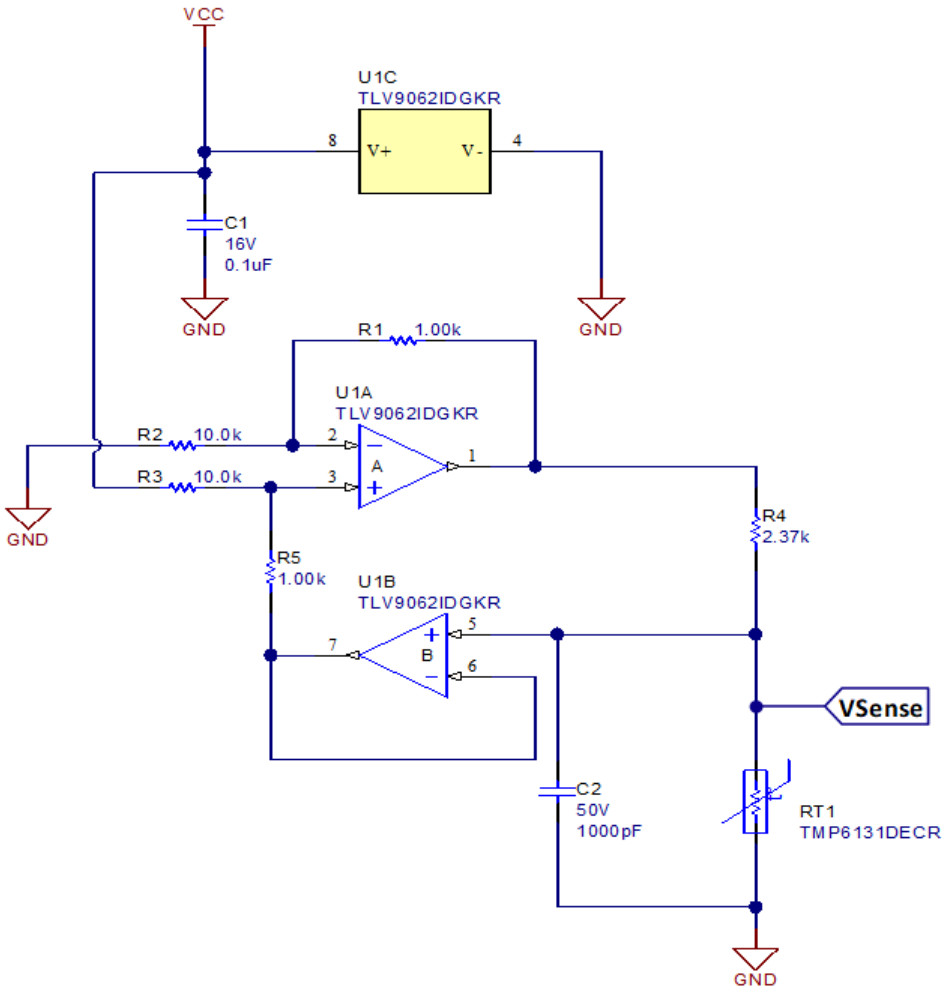

图 6-1. 恒流源设计

TLV9062 是具有轨至轨输入和输出摆幅功能的双通道运算放大器。

在此处的产品文件夹中可找到 TMP6131 热敏电阻的 Spice 模型。

6.2 TMP6 热敏电阻标准元件封装

TMP6 热敏电阻当前采用的封装有 X1SON (DEC)、SOT-5X3 (DYA) 和 TO-92S (LPG)。DYA 封装置于 IPC-782A 0603 封装之上，不存在适配或焊接质量问题。TI 建议尽可能减少焊盘，以尽量减少多余的铜，从而充分提高器件

的灵敏度。但是，改变封装是完全没必要的。封装尺寸对于器件的精度没有影响。另请注意，IPC 建议增加焊盘尺寸，以提高波峰焊的可靠性。使用标准的 0603 焊盘尺寸时，这对于此器件是有益的。

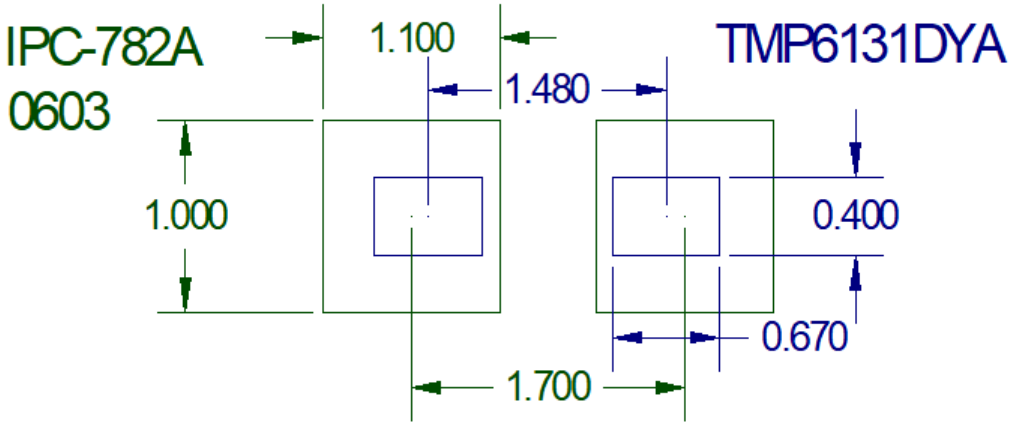

图 6-2. IPC-782A 0603 封装上的 TMP6 DYA 封装

虽然数据表强调 SOT-5X3/DYA 封装与 0603/1608 封装兼容，但由于其独特的引线框尺寸，所以也与 0805/2012 封装兼容。

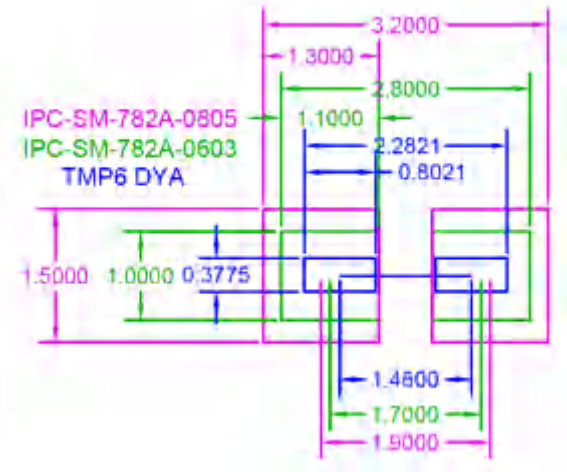

图 6-3. IPC-782A-0805 封装上的 TMP6 热敏电阻 DYA

当 DYA、IPC 0603 和 IPC 0805 PCB 封装叠放时，如下图所示，可以看到为 DYA 推荐的 PCB 封装的建议空间略大于 0603/1608 或 0805/2012。在此对比中，务必要注意，确保落在 0805/2012 焊盘上的 TMP6 热敏电阻引脚柱跟允许按照 IPC-A-610G 标准（关于焊接质量）的要求进行柱跟填角。与 0603/1608 或 DYA 封装相比，0805/2012 封装具有更大的焊盘，因此端部和侧面填角很容易满足相同的 IPC-A-610G 要求。

有关更多详细信息，请参阅此处 e2e 论坛中的 TMP61 热敏电阻常见问题解答。

6.3 用于 TMP6 和 NTC 热敏电阻的双电源方法

一项常见的系统要求是在 BOM 上设置多源元件。本节提供了一种使用 TMP6 和 NTC 热敏电阻进行多源的方法。这里主要介绍根据初始压差 (ΔV) 确定板载器件，然后使用正确的温度转换代码。

第一步是，预先确定启动期间初始温差 (ΔT) 的方向。组装期间，由于电源、处理器等自发热，电路板本身可升温约 5°C。如果您愿意，可以使用热灯 (+ΔT) 或冷冻喷雾 (-ΔT) 增加温差。

第二步是，通过软件确定启动期间初始压差 (ΔV) 是正 (+) 还是负 (-)。使用下表作为参考，可通过软件确定板载热敏电阻类型，并使用正确的温度转换代码。

	Increase in temperature (+ΔT)	Decrease in temperature (−ΔT)
Change in voltage (ΔV) for TMP6	+	−
Change in voltage (ΔV) for NTC	−	+

图 6-4. 用于 TMP6 和 NTC 热敏电阻的双电源

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265