ANALOG DEVICES

AN-0970
应用笔记

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

使用ADuC706x微控制器实现RTD接口和线性化

作者：Mike Looney

简介

铂电阻温度检测器(RTD)是最精确的传感器之一，可用来测量–200°C至+850°C之间的温度。这种RTD的校准精度能够达到±0.02°C或更好。然而，为获得最高的精度，需要精确的信号调理、模数转换、线性化和校准。

ADI公司的MicroConverter®系列产品中有的产品在一个单芯片上集成有一个24位ADC和一个32位的ARM7 MCU，再配合信号调理电路，非常适合RTD传感器。

本应用笔记介绍如何通过使用ADuC706x和数个无源组件，实现一个完整的RTD传感器接口。本应用笔记基于AN–709应用笔记“使用ADuC8xx微转换器实现RTD接口和线性化”。

当使用微转换器实现RTD传感器接口时，强烈建议参考本应用笔记提供的软件工具和代码示例。这些工具和代码可从以下网址获取：www.analog.com/MicroConverter 。

目录

简介 1
硬件设计 3
根据ADC转换结果计算RTD电阻 4
RTD传递函数 4
线性化方法 5
直接的数学方法 5
单段线性逼近法 6
分段线性逼近法 7
RTD系数生成工具 9
校准 11
误差分析 13
噪声 13
温度漂移 13
RTD自热效应 14
其它误差源 14
软件和源代码 15

硬件设计

RTD传感器的阻值是以某一确定的方式随温度变化的函数。在试图理解RTD传递函数的阻值与温度关系(这是非线性的)之前，假设已经数字化的修正了非线性。然后，重点关注将RTD电阻转换为数字量。一个通常的方法如图1所示。

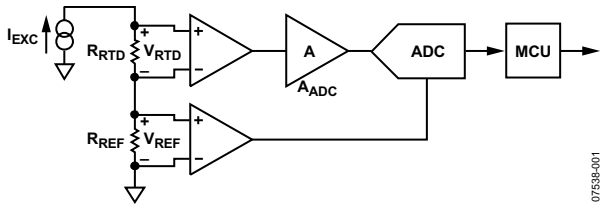

图1. RTD接口硬件配置

图1中，一个电流源(I_{EXC})同时激励串联在一起的RTD(R_{RTD})和一个精确的基准电阻(R_{REF})，分别产生ADC输入电压(V_{RTD})和基准电压(V_{REF})，如下所示：

$$V_{RTD} = I_{EXC} \times R_{RTD}$$

$$V_{REF} = I_{EXC} \times R_{REF}$$

ADC归一化的数字输出(零输入=0和满量程输入=1)是输入电压与基准电压的一个简单比值再乘以增益级A_{ADC}。

$$ADC_{norm} = A_{ADC} \times \frac{V_{RTD}}{V_{REF}} = A_{ADC} \times \frac{I_{EXC} \times R_{RTD}}{I_{EXC} \times R_{REF}} = A_{ADC} = \frac{R_{RTD}}{R_{REF}}$$

注意电流源(I_{EXC})是如何在等式中被消掉的。这意味着，即使激励电流发生变化或不精确，该ADC结果始终直接对应于RTD电阻和基准电阻的比值。选择一个高精度、低漂移基准电阻意味着即使使用精度很低的电流源也可以高精度地计算出RTD电阻值。

在一个微转换器上应用同一原理，图2给出了ADuC706x与一个4线RTD接口电路的连接。注意，这和图1是相同的整体拓扑结构，但所有有源组件(激励电流源、V_{RTD}和V_{REF}的差分输入级、增益级A_{ADC}、该ADC本身，和一个微控制器)都包括在ADuC706x芯片内部。二极管保护和100 Ω电阻仅用于保护ADuC706x在接线端子过压的情况下不受损。

还包括其它外设，如用于数字通信通道的串行通信端口。注意：为了实现信号R/C滤波和接线端子过压保护，电路还要添加一些无源组件。这是一个完整的执行电路，只需要一个电源和用于数字接口的任意特定外围芯片，如RS-232或RS-485线路驱动器/接收器。

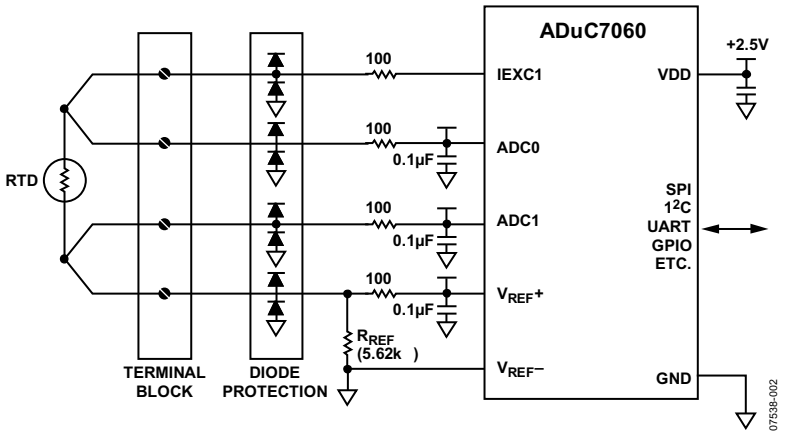

图2. 用ADuC706x设计完整RTD接口电路

根据ADC转换结果计算RTD电阻值

正如硬件设计部分介绍的那样

$$ADC_{norm} = A_{ADC} \times \frac{R_{RTD}}{R_{REF}}$$

可以写成

$$R_{RTD} = ADC_{norm} \times \frac{R_{REF}}{A_{ADC}} = ADC_{norm} \times scale$$

其中：

$$scale = \frac{R_{REF}}{A_{ADC}}$$

这个标度值(scale)是用在示例代码中的固定比例因子。进一步考虑，可以将一个固定的偏移值添加到方程，从而

$$R_{RTD} = ADC_{norm} \times scale + offset$$

这个偏移项offset表示一个固定的偏移量可以用来修正误差。这个偏移量，在“校准”部分有更深入的讨论。大多数情况下，偏移量设定为0已经足够了。请注意，RTD阻值作为ADC转换结果的函数可以用一个线性方程表示，这个方程用于标度值scale和偏移量offset就可以决定。

该应用笔记的其余部分考虑了最为通用的铂RTD，其标称电阻(R0)在0°C时为100 Ω。当使用该应用笔记时，假设基准电阻阻值为5.62 kΩ，该电阻可以很好的匹配这样一个RTD。使用这些元件值和ADuC706x，ADC内部增益最高可配置为32就可以满足RTD覆盖其全部的额定温度范围。

记住，ADCnorm被限制在0至1范围内，它在更高ADC增益的情况下定义了温度范围限制。增益32对应于ADC0CON寄存器的数值0x8415，或单极性输入37.5 mV的一个范围，其中

$$A_{ADC} = \frac{V_{REF}}{span} = \frac{1.2\,\text{V}}{37.5\,\text{mV}} = 32$$

为了对应于这个增益设置，标定值设定为175.625，因为

$$scale = \frac{R_{REF}}{A_{ADC}} = \frac{5.62\,\text{k}}{32} = 175.625$$

在示例代码中这是默认标定值。偏移量的默认值为0。这些方程假定IEXC0引脚的激励电流为200 μA。这是通过设置IEXCCON = 0x42进行配置的。

本节中求解RRTD的方程只是通过软件方法得出的，RTD阻值直接来源于一个给定的ADC转换结果。要确定RTD温度与其阻值变换之间的函数关系，需要了解RTD传递函数。

RTD传递函数

铂RTD传递函数是由是两个不同的多项式方程描述的：一个用于温度低于0°C时，另一个用于温度高于0°C时。

t ≤ 0°C时方程为：

$$R_{RTD}(t) = R_0\,(1 + At + Bt^2 + C(t - 100°C)t^3$$

t ≥ 0°C时方程为：

$$R_{RTD}(t) = R_0\,(1 + At + Bt^2)$$

对于上述两个方程：

t为RTD温度(°C)。

$R_{RTD}(t)$ = RTD电阻为RTD温度(t)的函数。

R_0是0°C时的RTD电阻(一般为100 Ω)。

$A = 3.9083 \times 10^{-3}\,°C^{-1}$。

$B = -5.775 \times 10^{-7}\,°C^{-2}$。

$C = -4.183 \times 10^{-12}\,°C^{-4}$。

注意：符号由R_{RTD}变为$R_{RTD}(t)$说明RTD电阻为温度的函数。图3给出了RTD传递函数(电阻为温度的函数)以及过0°C值的线性延伸(为了更形象地比较)。

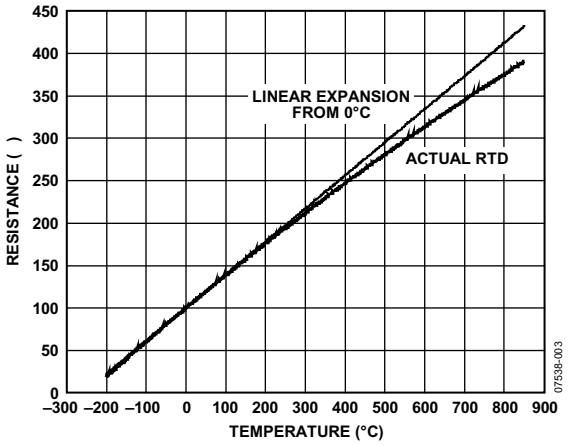

图3. RTD传递函数

前面的方程定义RTD电阻为温度的函数，$R_{RTD}(t)$。但是，为了实现一个RTD传感器接口电路，RTD温度必须是确定的，而不是电阻的函数，$T_{RTD}(r)$。对于非线性RTD传递函数，这可能不太易懂。在以下的章节里将探讨实现该任务的的实用技术。

线性化方法

给定了RTD传递函数，有多种方法可以确定作为RTD电阻的函数的温度值。本应用笔记考察了三种在嵌入式设计有用的方法。这些方法对基于微转换器的设计尤为适用。表1列出了每种方法的优缺点。这里的执行时间表示ADuC7060实证测量值，以10.24 MHz内核时钟速度执行下面的C子程序。

表1 线性化方法比较

方法	优点	缺点	小结
直接的数学方法	特别精确。 不需要查找表。	需要数学函数库(通常>1 kB)。 计算速度慢。	如果数学函数库已经存在，该方法有效。
单段线性逼近法	特别快(<56 µs)。 需要的代码空间较小。 对于狭窄温度带较为精确。 不需要查找表。 不需要数学函数库。	对于较宽温度范围不够精确。	适用于只有有限代码空间时的小温度范围测量。
分段线性逼近法	计算速度快(<1ms)。 代码大小及准确性可控。 可以相当精确。 不需要数学函数库。	相对于单段线性逼近法代码量较多。	对于大多数情况都适用。

直接的数学方法

在RTD传递函数那一节已给出了RTD电阻的详细数学方程，该方程可以作为其温度的一个函数$R_{RTD}(t)$。可不可以将方程变形为RTD温度为其阻值的函数$T_{RTD}(r)$这种表达式?。这是一个相当简单的任务，因为定义正温度特性的方程仅仅是二次的。解该二次方程得到两个表达式，只需代入几个已知量即可知道哪一个是正确的。结果为下述在0°C或者0°C以上时关于RTD温度的方程。

$$T_{RTD}(r) = \frac{\sqrt{A^2 - 4B\left(1 - \frac{r}{R_0}\right)}}{2B}$$

其中：

$A = 3.9083 \times 10^{-3}$°C^{-1}。

$B = -5.775 \times 10^{-7}$°C^{-2}。

$C = -4.183 \times 10^{-12}$°C^{-4}。

R_0是0°C时的RTD电阻(一般为100 Ω)。

r为RTD电阻。

由于方程式是实时求解的，故化为下面的形式较为有利：

$$T_{RTD}(r) = \frac{Z_1 \sqrt{Z_2 + Z_3 \times r}}{Z_4}$$

其中：

$Z_1 = -A = -3.9083 \times 10^{-3}$

$Z_2 = A2 - 4 \times B = 17.58480889 \times 10^{-6}$

$Z_3 = \frac{4 \times B}{R_0} = -23.10 \times 10^{-9}$

$Z_4 = 2 \times B = 1.155 \times 10^{-6}$

由于Z_1到Z_4都是常数且为绝对值，故只需较少的计算量，对于实时计算十分有利。上述关于$T_{RTD}(r)$的方程被称为正函数，这是因为它只是针对大于等于0°C时的情况。由于这是一个直接的数学解决方法，它是100%准确的。在解该方程时，ARM7 C代码的舍入误差为32位浮点型，精度可达到+0.0001°C/−0.0005°C。对于任何实际用途都可称之为接近100%准确。ADuC706x以10.24 MHz的内核时钟速度运行RTDmath.c中的C子程序，该方程的执行时间小于750 µs。

上述方程只在温度大于等于0°C时有效。定义负温度特性的$R_{RTD}(t)$方程是一个4阶多项式(将第三项展开)，由它解出单一的温度关于电阻的函数表达式是不现实的。但是，借助于计算机数学工具可以找出反向传递函数的近似解。

运用Mathematica®或其它类似的数学工具软件解出以下温度小于等于0°C时的最佳拟合多项式表达式。

$$T_{RTD}(r) = -242.02 + 2.2228 \times r + 2.5859 \times 10^{-3} \times r^2 - 4.8260 \times 10{-}6 \times r^3 - 2.8183 \times 10^{-8} \times r^4 + 1.5243 \times 10{-}10 \times r^5$$

$$T_{RTD}(r) = -241.96 + 2.2163 \times r + 2.8541 \times 10^{-3} \times r^2 - 9.9121 \times 10^{-6} \times r^3 - 1.7052 \times 10^{-8} r^4$$

$$T_{RTD}(r) = -242.09 + 2.2276 \times r + 2.5178 \times 10^{-3} \times r^2 - 5.8620 \times 10{-}6 \times r^3$$

$$T_{RTD}(r) = -242.97 + 2.2838 \times r + 1.4727 \times 10^{-3} \times r^2$$

这四个方程被称为负函数，这是因为它们每一个都只是针对温度小于等于0°C时的情况。最上面的方程(5阶)精度最高，但需要最长的计算时间；最下面的方程(2阶)精度最小，但计算速度最快。表2给出了这些负函数的一些特性，图4给出了每个方程作为温度函数的误差曲线，以及正函数扩展至负温度区域时的误差曲线(作为比较)。

由图4我们可以注意到，在0°C附近的负温度区域，正函数的误差比2、3、4阶负函数的误差要小。根据这一特性，在较小的负温度区域内使用正函数对RTDmath.c的示例代码将更为有利。实际中选用正函数还是负函数要取决于使用哪个负函数(二阶、三阶、四阶或五阶)，表二给出了各个阈值。大于这个阈值，使用正函数误差较小；小于这个阈值，使用负函数误差较小。表2中方程精确度那一栏给出的只是针对低于相应阈值温度的误差。

表2 最佳拟合多项式方程特性 (负函数)

方程 尺寸	最大值 执行时间[1]	方程 精度[1]	阈值
第5阶	2.16 ms	+0.0001°C/ −0.00005°C	0°C/ 100 Ω
第4阶	1.61 ms	+0.0022°C/ −0.001°C	−8.75°C/ +96.6 Ω
第3阶	1.13 ms	+0.0053°C/ −0.0085°C	−12.5°C/ +95.1 Ω
第2阶	800 µs	+0.075°C/ −0.17°C	−70.5°C/ +72.1 Ω

[1]在内核时钟速度为10.24MHz时，用一个ADuC706x芯片运行RTDmath.c中的C语言子程序示例来实际测量执行时间和方程精度。

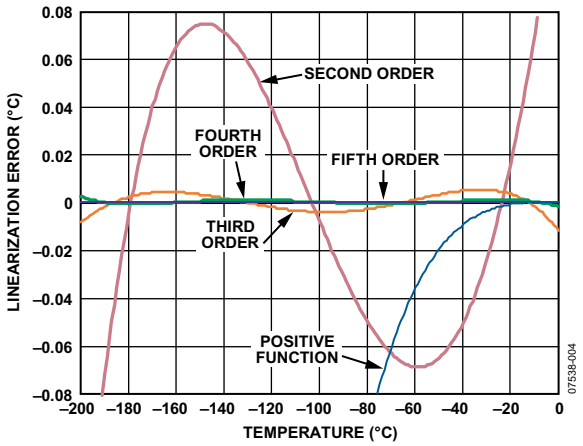

图4. 最佳拟合多项式方程误差曲线(负函数)

对于线性化，这种直接的数学方法的一个缺点是它需要浮点乘方和开平方根函数，这些函数可以在IAR系统编译器的数学公式库中找到。典型情况下，这些浮点数学函数会单独增加超过1kB的代码量。使用分段线性逼近的方法可以用更少的代码量获得相似的或更高的精度，该方法在“分段线性逼近方法”部分有所描述。尽管如此，如果程序中其它操作需要数学函数库中的函数，那么直接的数学方法或许是最好的解决方案，因为这些库函数已经可以使用。

单段线性逼近法

在图3中，可以看到在较小的温度范围RTD传递函数近似于一条直线。如果需要检测的温度范围仅仅是全部RTD温度范围的一部分，那么根本不需要对RTD信号进行线性化。在这些情况下，一个在所需温度测量范围的传递函数的最佳拟合线性逼近常常可以获得足够的精度。例如，在工业温度范围−40°C至+85°C时，一个最佳拟合线性逼近的精度为±0.3°C。

一般情况下，作为RTD电阻(r)的一个函数，温度线性方程形式如下：

$$Tlin(r) = A \times r + B$$

其中A和B为常量。

注意，正如在RTD传递函数部分描述的那样，没有相同的A和B。为A和B选择最优值来最小化误差范围需要一些数学公式，此处并未提及。尽管如此，本应用笔记附带的一个非常简单的软件工具可以自动找出A和B的最优值，来配合您具体的温度范围。这个工具的效果会在本应用笔记中进行验证，但是首先必须确定一个单段线性逼近是否适合具体的设计要求。

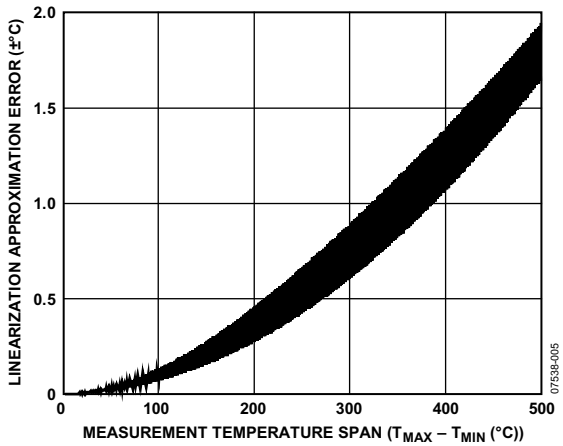

图5. 单段线性逼近误差曲线与温度测量范围

图5给出了500°C以下测量温度范围的全部逼近误差。对于500°C以上的范围，逼近误差会随着温度上升而减小。图5曲线的不精确特性(也就是数据轨迹的宽度范围)源于：即使是相同的温度范围，不同的绝对温度区域误差也是不同的。例如，−200°C至0°C以及+600°C至+800°C的温度范围，虽然这两个温度范围都为200°C，但是它们的精度却不同。

图5提供了误差的一个粗略概念，这样可以帮助判断是否考虑使用单段线性逼近。如果确定使用，那么RTD系数生成工具(在“RTD系数生成工具”部分有所描述)可以帮助确定一个具体温度范围的实际逼近误差，并且生成对于这个温度范围的最优源代码。

分段线性逼近法

将线性逼近方法延伸一步，设想用一定数量的直线段可以更好的逼近非线性RTD传递函数。产生这一系列直线段，那么每一个直线段与其相邻直线段之间都会有一个端点，这样看起来就像用一条直线连接一系列的点。可以一次计算出这些点(或系数)来最好的匹配RTD的非线性传递函数，然后将其永久存储在ROM或Flash存储器中。从这些系数表中，MCU可以实现简单的线性插值来确定基于RTD测量电阻的温度值。

为了理解这一过程在实际使用时是如何实现的，首先假设这个系数表已经存在。表中的每一个系数都是传递函数的一个简单的点，由一个电阻和一个温度表示。所以，表格形式如下：

{ro,T0; T1,T1; r2,T2;… rn,Tn}

有了这个表，MCU的实时任务(通过一个给定的电阻r来确定温度)就是，首先确定两个最接近所求点的系数({r_m,T_m}和{r_n,T_n})，然后在这两个点之间进行线性插值，来解出温度值。对于这个温度范围(只对r_m和r_n之间的阻值r有效)的实际线性插值公式具有以下这种形式

$$T_{SEG}(r) = T_m + (r - r_m)\frac{T_n - T_m}{r_n - r_m}$$

注意上述查询表中的每一个系数都包括两个值，一个是电阻，另一个是温度(实际就是传递函数的x和y值)。所以，对于N个直线段(*N*+1个系数)，总共有2*N*+2个值必须存储在存储器中。为了减小查询表的大小，考虑一个包括N个直线段的表，每一个直线段都有相同的阻值宽度。这样一个表就可以只存储一系列温度值点，如下所示：

{T0; T1; T2; ... tN}

所以，对于一个给定的系数{r_n,T_n}，可通过下式计算r_n的值：

$$r_n = r_0 + n \times r_{SEG}$$

其中：

r_0和r_{SEG}是固定值，除系数表之外，它们也存储在ROM中。

r_0是系数为0{r_0,T_0}时的电阻值。

r_{SEG}是电阻的固定范围，也就是相邻系数的范围。

对于一个给定直线段，线性插值公式变为

$$T_{SEG}\ r) = T_i + \left[r - \left(r_0 + i \times r_{SEG}\right)\right] \times \frac{T_{i+1} - T_i}{r_{SEG}}$$

其中i表明使用了哪一个直线段(即哪对系数)，并且可以通过*r*计算出来，如下所示：

$$i = trunc\left(\frac{r - r_0}{r_{SEG}}\right)$$

同样，上述表达式$T_{SEG}(r)$只是T_i和T_{i+1}这两个系数之间的一个线性插值。要想在实际应用时实现这个过程，MCU必须首先解出i(通过上面最后一个方程)，这样对于r来说，系数T_i 和T_{i+1}就是最接近输入值的两个系数。然后，MCU可以简单的求解$T_{SEG}(r)$方程，来确定在给定输入阻值情况下的温度。

这种分段线性逼近法产生的总误差，取决于分段的数量(系数的个数或查询表的大小)和总的温度范围。

图6给出了−200°C至+850°C这个温度测量范围内的线性逼近误差曲线，作为查询表大小的一个函数(用来优化RTD系数生成工具产生的系数)。注意，如果测量温度范围减小，相同大小的查询表会给出更好的误差结果，或者一个更小的查询表可以给出同样的误差结果。

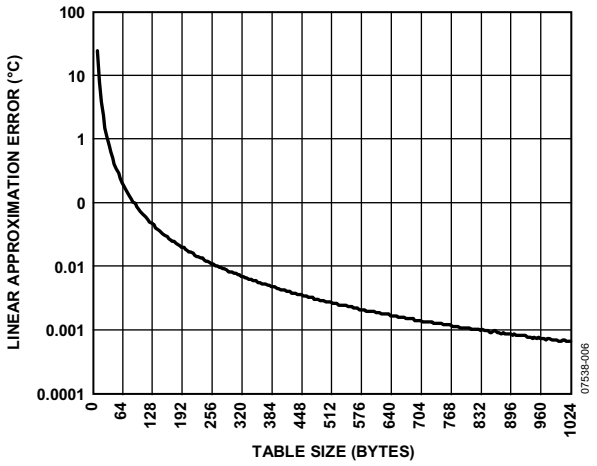

图6. 分段线性逼近误差曲线与查询表大小(范围−200°C~+850°C)

RTD系数生成工具

实现分段线性函数，最难的部分就是生成查询表。尽管如此，本应用笔记附带的RTD系数生成工具(coefRTD.exe)会自动地为铂RTD完成这项工作。这个可以在DOS环境下运行的工具助手(带有基于ARM7的RTD接口)，就是使用分段线性或单段线性逼近的方法设计的。它可以实现以下任务：

- 可以为一个给定的温度范围和查询表大小生成优化的查询表系数。
- 指明误差范围和查询表大小。
- 用ARM7 C源代码生成完整的RTD线性化函数)包括查询表)。
- 生成误差表作为已知查询表中温度的一个函数。

图7给出了一个用户输入的对话示例。注意，程序仅仅需要用户输入3个参数(T_{MIN}、T_{MAX}、和N_{SEG})。程序可以生成文件RTDpwl0.c，它是一个完整的C源程序文件(定制一个用户特定查询表)并且可以被包括在一个项目中。在这个项目中，T_rtd()函数可以直接从其它源程序的函数中调用。另外，也可以直接复制和粘贴RTDpwl0.c文件的任意部分到其它源文件中。

系数生成工具也可以输出一份误差分析文件(errorRTD.txt)，它是一个表格制式的文本文件，可以导入微软的Excel或任何其它空白表格软件，来检验由线性逼近程序产生的误差。

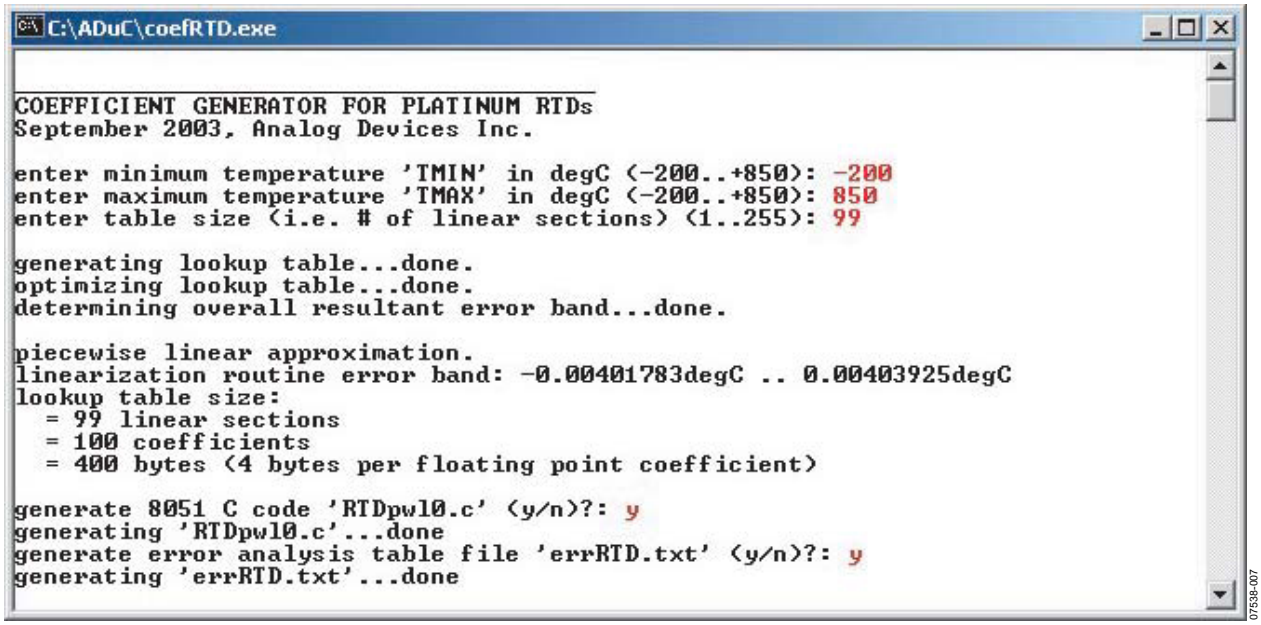

图7. 使用分段线性逼近法生成系数的会话示例（用户输入标记为红色）

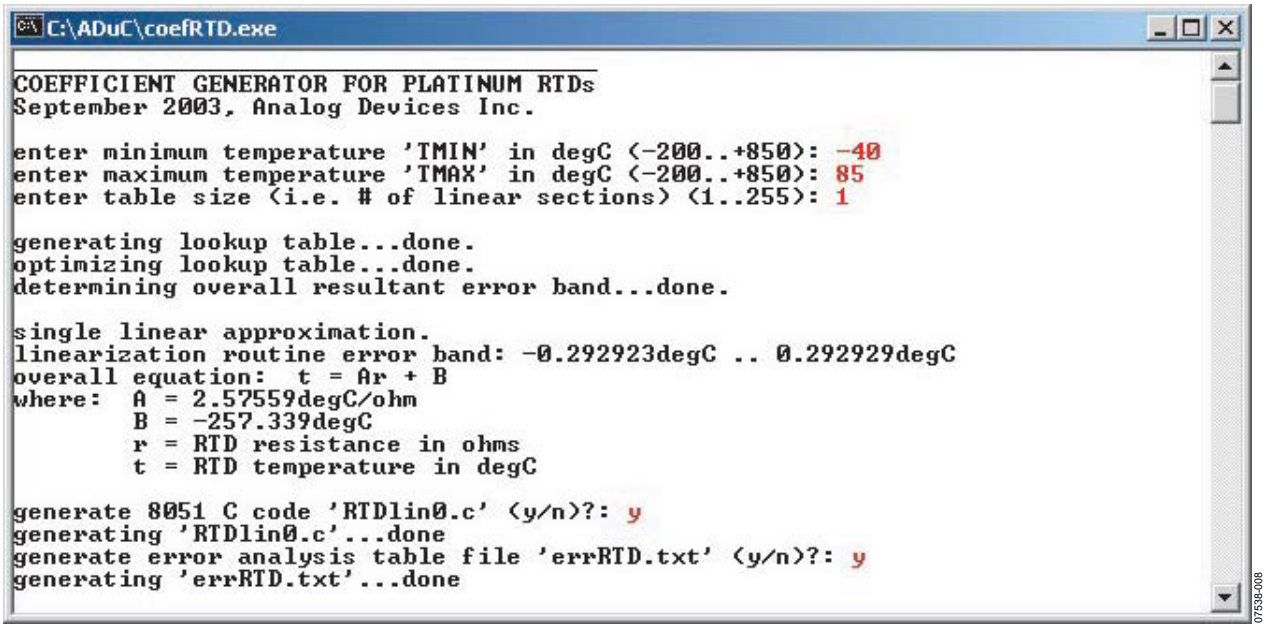

图8. 使用单段线性逼近法生成系数的会话示例(用户输入标记为红色)

系数生成程序不仅可为分段线性逼近生成线性化函数，也可为单段线性逼近生成线性化函数。为实现这些，表格大小输入1表示只有单根直线段。

程序可进行识别，输出结果适合单段线性化逼近法，而不是分段线性化逼近法，如图8所示。

校准

如产品数据手册所述，ADuC706x有一个内置的函数用于校准模数转换器端点误差(偏移和增益误差)。然而，如果在校准时考虑整个信号链，包括RTD本身，结果就会得到一个较低的总误差，在这种情况下，内置ADC校准没有额外的益处。本应用笔记首先研究了系统校准，随后列出一些内置ADC校准可能依然有用的情形。

到目前为止，一直都假设RTD本身是理想的。然而，实际的RTD不是理想的。就像现实世界里的其它事物，RTD与制造商数据手册所描述的内容会有误差。幸运的是，这些误差大都可以很容易地用软件校准。本应用笔记里所讨论的校准函数可用于单点或两点校准。这个函数可同任何线性化技术一起使用。

若要了解单点校准的工作原理，请参考"RTD传递函数"部分讨论的$R_{RTD}(t)$，注意它大部分是由RTD在0 时的电阻值R_0定义的。对于最常见的RTD，R_0一般为100Ω。然而，在一个RTD传感器中最主要的误差源就是R_0的值，因为不同器件之间该值差异较大。另外，$R_{RTD}(t)$表达式内传递函数的剩余部分需要乘以R_0，误差正是由于R_0容差是纯粹的乘法所导致的。所以，在将R_{RTD}作为ADC转换结果归一化的一个函数时，可以通过调整上述表达式中的标度因子来修正误差。

$$R_{RTD}=ADC_{norm}\times scale+offset$$

特殊情况下，如果RTD测量一个比较精确的已知温度，一次ADC转换已经完成，校正的标定值可由下式计算：

$$scale=\frac{R_{cal}}{ADC_{cal}}$$

其中ADC_{cal}是模数转换的实际归一化结果，R_{cal}是RTD温度理想的(期望的)电阻值。R_{cal}可以使用$R_{RTD}(t)$公式手动计算。通过这种方法(所谓的单点校准)可以得到一个校准标定值，通过它可以补偿RTD R_0容差，同时还有基准电阻的初始容差。从这里再进一步，可以执行两点校准，它不仅可以补偿这些尺度误差，也可以补偿任何可能存在的偏移误差。这样做不仅需要调整标定值，也需要调整偏移值。

假设已经实现了单点校准，RTD现在可以得到另一个非常准确的已知温度，ADC转换又进行了一次。标定值公式(也就是R_{RTD}/ADC_{norm}函数的斜率)

即

$$scale=\frac{R_{cal}-R_{precal}}{ADC_{cal}-ADC_{precal}}$$

其中：

R_{precal}和ADC_{precal}分别是前一校准点的电阻和ADC转换结果。对于当前校准点，R_{precal}和ADC_{precal}是一样的。

注意，这只是采用直线上两点计算R_{RTD}和ADC_{norm}关系函数曲线斜率的一种方法。这里只要考虑偏移值，因为标定值现在是已知的，偏移值可以用单点计算。下面偏移值的表达式通过解前面RRTD表达式的偏移，然后分别用R_{RTD}和ADC_{norm}替代R_{precal}和ADC_{precal}得到。

$$offset=R_{precal}-ADC_{precal}\times scale$$

注意，如果R_{precal}和ADC_{precal}全为0(就是没有前校准点)，那么标定值的表达式就同单点校准一样，只有在单点校准时偏移值表达式为0。因而，单点校准或两点校准可以使用同样的函数(示例代码中的Cal())。

如果使用示例代码，按照下列步骤完成两点校准：

1. 选择两个温度用于校准，确保温度点充分分开(理想情 况下至少有四分之一的整个测量跨度)以避免测量温度末端附近误差累积。
2. 使用RDT测量第一个温度点，等待以保证显示结果建立新值，然后按下终端(或终端仿真器)上的任何一个键，显示用户输入/输出菜单。
3. 按照菜单提示校准到已知温度点，然后再提示下输入温度。
4. 重复第2、3步较准第二个温度点。

注意：对于单点校准，转到第4步。

尽管这样校准有很多好处，但是在一些应用中，由于需要考虑许多系统因素，因此这种方法不太实用。如果不能实现如步骤1到4所描述的校准过程，那么就需要考虑用一个系统ADC校准来代替(ADuC706x数据手册中有所描述)。如果要这样做，可以简单的将RTD短路(0Ω)，并且进行一个系统零标度校准。然后，用一个719.36Ω的高精度电阻替代RTD，并且进行一个满量程校准。这是对于内部ADC误差和RREF电阻初始容差的补偿，但是并未考虑RTD本身的任何误差。

注意，ADuC706x(以及所有其它微转换器产品)的一个额外的好处就是，它内部有一个片内非易失性Flash存储器，这个存储器可以用来存储已校准标度以及偏移量。这样，每次系统上电启动时，芯片可以调出存储的校准值，而不需要每次都进行校准。

误差分析

在进行数据采集设计时会有很多误差源，诸如：ADC线性度、输入放大器噪声、电阻约翰逊噪声、放大器温漂以及电阻温漂。对于一项设计而言，确定主要误差源是比较困难的任务。幸运的是，ADuC706x将所有的有源级都集成到一个单独的完全工厂确定的器件中，这使得误差分析变得较为简单。但是在设计中仍然需要考虑非线性传感器元件带来的误差。对于迄今为止已讨论过的特定硬件和软件配置，本应用笔记给出了几个最为显著的误差因素。

对于一个具体的RTD，如果没有进行系统校准(用单点或两点校准)，那么RTD本身就已经是绝对误差的最显著来源。这个误差取决于所选RTD的具体型号，在RTD制造商的数据手册中，该误差应该很好的量化。本应用笔记重点讨论除RTD自身以外的误差源。

噪声

一类要研究的误差就是噪声。在本设计中有三个主要的噪声源需要考虑：电阻约翰逊噪声、放大器/ADC输入电压噪声和放大器/ADC输入电流噪声。这些噪声以均方根累积，从而当一个噪声源只是稍微比其它噪声源大时，较小的作用源是可以忽略的。在这种特殊情况下，主要的噪声源就发生在放大器/ADC输入电压噪声。具体而言，在增益设置讨论时，ADuC706x输入电压噪声规格为0.25μV均方根值，或大约1.65μV峰峰值。

由于电阻-温度传递函数的非线性，将输入电压噪声转换为输出温度噪声或许不够直观。作为RTD温度的一个函数，温度噪声的结果是多种多样的。结果如图9所示。

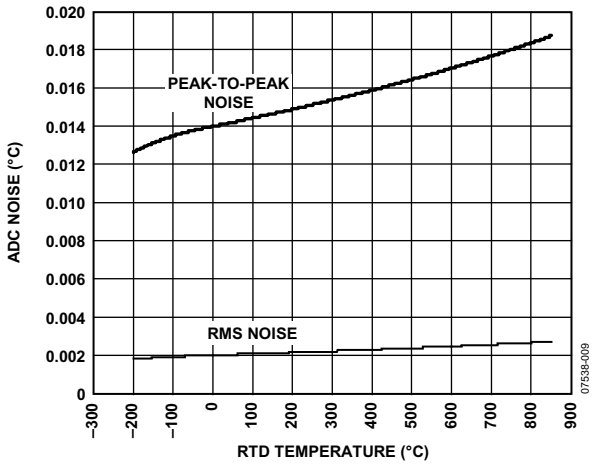

图9. ADC噪声与RTD温度

注意：即使在最高的RTD温度(也就是最坏噪声)峰峰噪声总是低于0.013 ；它甚至好于较低的测量温度。记住：这作为RTD温度函数的噪声变化不是ADC本身的函数，而是在数字域使用的非线性$T_{RTD}(r)$传递函数的直接结果。

温度漂移

另外需要考虑的误差源是温度漂移：特定ADC偏移、增益温度漂移和基准电阻温度漂移。这是直流误差的变化(偏移和增益误差)，可以作为ADC芯片或基准电阻温度变化的一个函数。这取决于RTD调理电路的环境温度，而不是实际测量的RTD温度。简单来说，此处这两个不同的温度分别指环境温度和RTD温度。另外，由于非线性$T_{RTD}(r)$传递函数，温度漂移值(也就是环境温度灵敏度)随着RTD温度函数变化。图10的结果需要几点解释。

图10的x轴表示RTD温度。y轴表示环境温度每°C变化测量误差温度漂移的度数。例如，如果RTD温度固定在100°C，V_{REF}漂移(使用5 ppm/°C基准电阻)约为±0.01°C/°C。因而，如果环境温度变化，例如50°C，读到的测量温度可能变化±0.5°C(忽略其它温度漂移的影响)。

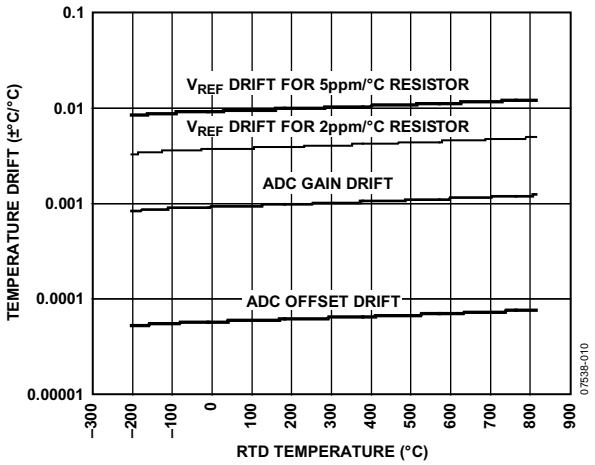

图10. 温度漂移与RTD温度

很显然，在工业环境下，环境温度经常在−40°C至+85°C或更宽的范围内变化，温度漂移就成了一个主要的误差源。可以直接使用ADuC706x的片内温度传感器测量片内温度(它紧随环境温度变化)，然后使用测量的片内/环境温度补偿温度漂移误差。

在制造时这需要一个额外的温度循环步骤，将环境温度设定为两个固定的值，然后在各环境温度下进行零标度和满量程ADC读数。无论如何，可以使用软件，在内部温度传感器精度范围内来补偿补偿温度漂移误差，以及补偿外部基准电阻和ADuC706x之间的温度梯度误差。本应用笔记没有进一步研究温度漂移补偿技术，然而请注意片内资源的存在使得仅仅改变软件就可以实现该选项。

RTD自热效应

RTD自热效应是另外一个需要考虑的误差源。简言之，电流通过RTD时会引起功率耗散，这会使RTD温度上升。幸运的是，RTD可以使用只有200 μA的激励，R_0为100 Ω时RTD总的功耗不超过8 μW。这么小的功耗所引起的自热量是变化的，取决于RTD所使用的特定型号，但是一般情况下导致的自热效应是可以忽略的。

其它误差源

其它误差源几乎是可以忽略的。直流端点误差(偏移和增益误差)可以使用在“校准”部分所讨论的校准技术充分校正。电阻约翰逊噪声远低于ADC输入电压噪声。值得考虑的唯一一个其它噪声源是模数转换器INL(积分非线性或相对精度)。ADuC706x数据手册中对于典型INL的具体规定是满量程的15 ppm，这会导致折合到输出端的INL误差大约是输出噪声峰峰值的两倍，如图9所示。

软件和源代码

本单元的所有参考程序和源代码包含在一个Zip文件里。可以访问以下网址获取该文件：

www.analog.com/MicroConverter 。Zip文件内容如下：

- coefRTD.exe。可执行系数生成工具。
- coefRTD.cpp。系数生成工具源代码。
- RTDdirect.c。使用直接数学线性化方法的线性化子程序。
- RTDpwl.c。使用分段线性逼近法的线性化子程序。可以使用coefRTD.exe程序生成该代码的定制版本。
- RTDlin.c。使用单段线性逼近法的线性化子程序。可以使用coefRTD.exe程序生成该代码的定制版本。
- RTDLinearMain.c。ADuC706x RTD接口程序的一个完整示例。该示例应用了所有三个线性化函数。
- RTDLinearMain.hex。RTDLinearMain.c和RTDpwl.c的完整编译版本。该版本已经可以下载并在ADuC706x上运行。
- Calibrate.c。示例代码为校准外部RTD提供子程序。
- ReadMe.txt。提供版本信息、介绍各文件功能的文本文件。
- 使用上述源代码的完整项目必须包括一个主程序(RTDLinearMain.c、Calibrate.c或自编程序)和一个线性化子程序文件(RTDmath.c、RTDpwl0.c、RTDlin0.c或coefRTD.exe工具生成的定制源文件)。许多具体内容在各C源文件的注解中提供。
- 一个IAR示例项目。

注释

AN07538-0-5/09(0)