Interfacing Thermistors to TRI PLCs

Overview

I am working on a project that involves thermistors with a TRI PLC. I have used thermistors on several projects and have always relied on the use of look up tables and linear interpolation to convert from ADC values to temperature. The temperature measurement requirements for the new project are less demanding than previous projects.

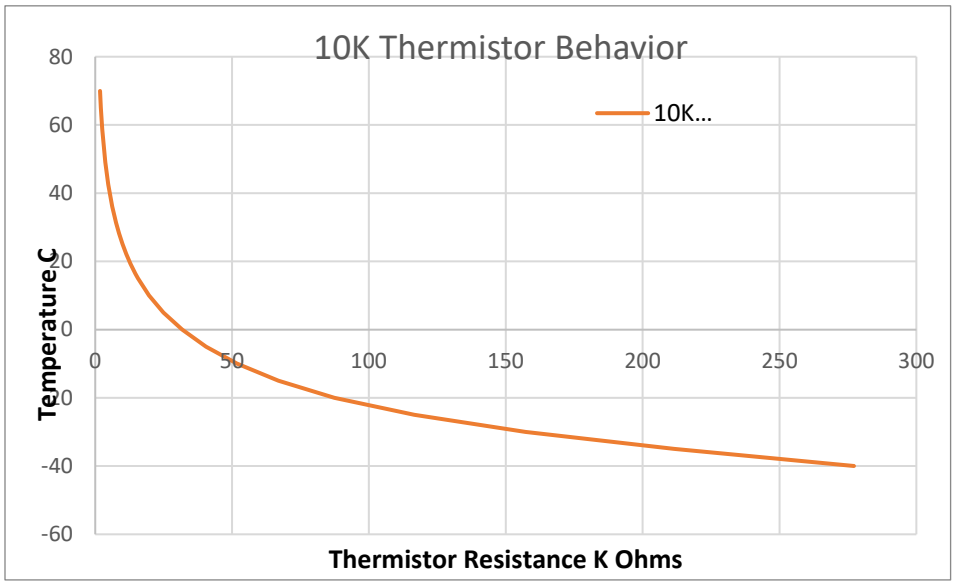
Because my measurement requirements are modest, I wanted to see if I could simplify the math to convert ADC readings to temperature. My hope was to eliminate the usages of lookup tables and linear interpolation.

NTC Thermistor Basics

This is the short version:

- A NTC thermistor is a temperature sensitive resistor. The resistance goes down as temperature goes up. NTC is short for negative temperature coefficient.
- The best math to describe the relationship between the thermistor's resistance and temperature is based on complex exponential equations. These equations are not 100% accurate but they are the best there is/
- It is not possible for the PLC's analog to digital converter, ADC, to directly measure resistance. It takes some external circuity to convert thermistor resistance a voltage that the ADCs to measure.

The following chart shows the relationship between the resistance of the typical NTC 10K thermistor and temperature:



The thermistor's resistance would be zero at some very high temperature, but it would catch fire first. The resistance decreases as the temperature goes down. At 25 ^{o}C the resistance of a 10K thermistor will by 10K Ohms. Note that the of curve above the 25 ^{o}C point just keeps climbing . The curve on the other

side of the 25 ^{o}C point slows down and flattens out at absolute 0. There is not a section in the resistance curve that could be described as linear or flat.

The behavior of thermistor is described by rather complex math. The most accurate approximation is the Steinhart-Hart equation:

$$\frac{1}{T} = a + b + lnR + c(lnR)^3$$

where: T is the thermistor temperature in Kelvin
R is the resistance of the thermistor in ohms
a, b, and c are the Steinhart-Hart constants that you must know

For extra credit, you are free to rewire the equation to solve for thermistor resistance for temperatures in the units of Celsius.

An older, simpler, and less accurate approximation is the β parameter equation:

$$R = R0e^{\beta(\frac{1}{\mathrm{T}}-\frac{1}{T0})}$$

Where: R is the thermistor resistance in Ohms
R0 is the thermistor resistance at the reference temperature, T0
β is the Beta parameter, like 3959
T is the current temperature in Kelvin
T0 is the reference temperature in Kelvin

If you see a thermistor described as 10K β(25/50) = 3950k the 3950k this is what they are trying to tell you

- is the thermistor resistance is 10K Ohms at 25 C^{o}
- B(25/50) tells you that the β parameter was calculated using measurements at 25 and 50 ^{o}C
- 3950k is the β parameter

This is the Excel version of the β parameter equation:

=R_25*EXP(Beta*(1/(Tc+273.15)-1/298.15))

where: R_25 is the resistance at 25 ^{o}C in ohms
Beta is the β parameter (something like 3950)
Tc is the thermistor temperature in ^{o}C
298.15 is 25 ^{o}C in Kelvin. (remember that the β parameter was calculated based on measurement at 25 ^{o}C)

If you do not have the manufacturer's temperature vs resistance chart for your thermistor, you are not 100% screwed. There are on-line calculators that can help you generate both the Steinhart-Hart constants and the β parameter for any NTC thermistor. You will need an accurate way to measure the thermistor resistance, a temperature measurement system, and a stable temperature. This sort of setup does not come cheaply if goal is to measure resistance to better than 0.1% and temperature accurate to 0.01 ^{o}C.

This is what I did to characterize an unknown thermistor with what I could find in my kitchen. I used a 4 digit DVM, an accurate digital thermometer, an immersion blender and an insulated container (ice bucket) filled with water heated 60 ^{o}C. I suspended the thermistor and digital thermometer in the middle of the water bath. Immersion blender stirred the water to ensure good thermal mixing. I covered the container and wrapped everything with towels to minimize heat loss. I recorded the resistance and temperature as the water temperature slowly dropped. I was extra careful to be accurate at the 50.0 and 25.0 ^{o}C points.

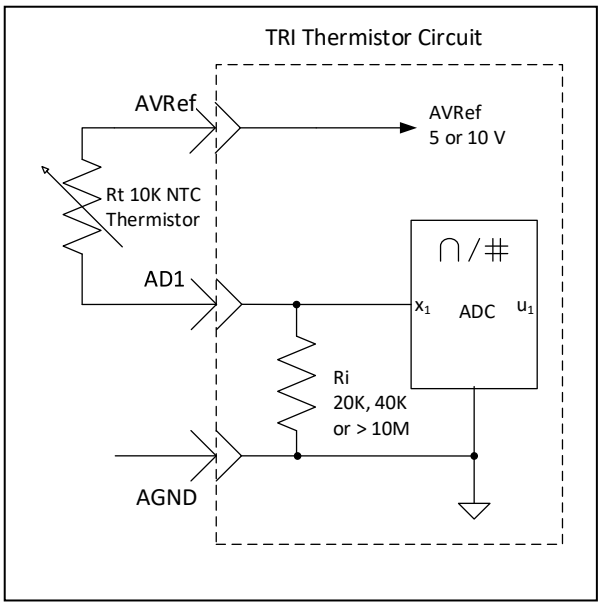
My calibration system is NOT accurate, but it is good enough for my applications. I can live with the errors in my characterization of the undocumented thermistor.

I used this online thermistor calculator to compute both the β parameter and the Steinhart-Hart constants and to help generate the thermistor characterization table values:

https://www.thinksrs.com/downloads/programs/therm%20calc/ntccalibrator/ntccalculator.html

TRI Suggested Thermistor Interface

All current of TRI PLCs support 12-bit analog to digital (ADC) inputs. The circuitry that is used to convert the thermistor resistance to a voltage for the ADC is known as a voltage divider. The following schematic is the interface suggested by TRI:

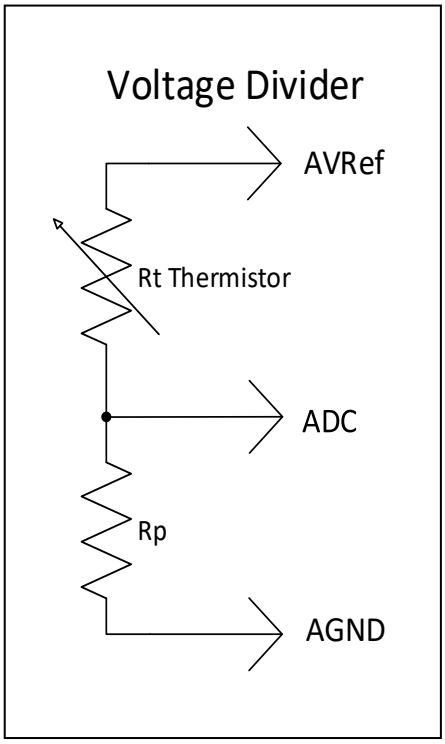


The thermistor is connected to the AVRef (analog voltage reference) of 5 or 10 V. the thermistor is in series with a 2nd resistor that has a value of 20000 or 40000 Ohms to AGND (analog ground). The AVRef voltage is divided across these 2 resistors. The ADC reads the voltage across the 2nd resistor.

Some of the ADC inputs on some of the TRI PLC's do not have either a 20K or 40K resistor in their input circuit. For those cases, TRI instructs you to add a external resistor from the ADC input to analog ground.

Rt is the thermistor and Rp is the total resistance from the ADC input to analog ground. The voltage source the analog voltage reference used by the ADC.

The TRI interface is a simple voltage divider. This is the equation for a voltage divider:



$$Vadc = \frac{AVRef * Rp}{Rt + Rp}$$

The resistance of the thermistor Rt can now be determined. We know the voltage across Rt because we know the voltage at the ADC input, VADC, and the total voltage supplied to the circuit:

$$VRt = AVRef - Vadc$$

The current that passes through the thermistor, Rt is exactly equal to the current that passes through the resistor Rp. We can calculate the current in the divider using Ohm's law:

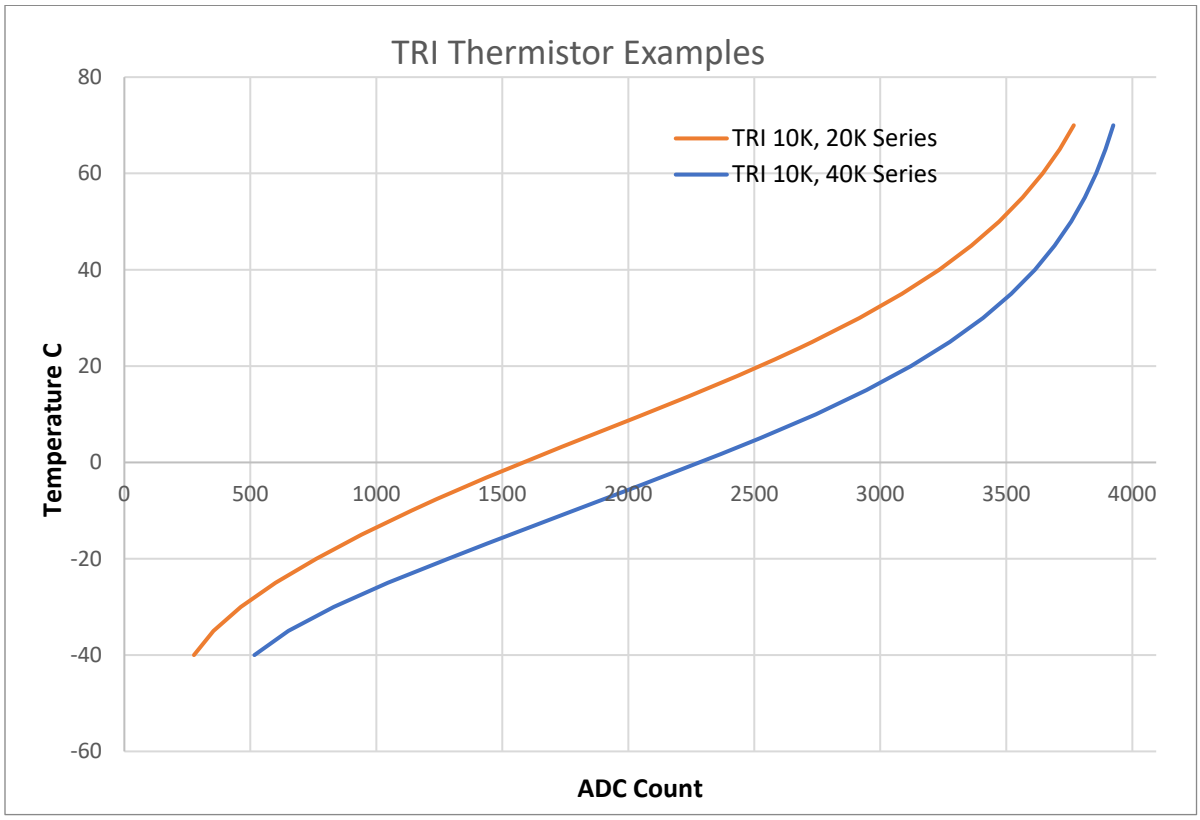
$$IRp = \frac{AVRef}{Rp}$$

Now we can calculate the thermistor resistance, Rt, knowing both the voltage and current:

$$Rt = Vadc/IRp$$

ADC values vs temperature for TRI Suggested Circuit

The following graph shows the relationship between ADC values and temperature using the TRI suggested thermistor interface. Curves are shown for both 20K and 40K circuits.



Compare these curves with the one that I showed earlier for thermistor resistance vs temperature.

- The ADC counts increase as temperature goes up. This is a result of placing the thermistor in the top ½ of the voltage divider and measuring the voltage across the bottom resistor in the divider.
- The traces have a big flat range in the middle. This is a result of the voltage divider.
- The traces curve in one direction on left side of the chart and curve in the opposite direction on the right side of the chart. Where the trace switches direction is the “inflection” point. The inflection point is in the middle of the flat spot for both curves.

Remember that TRI used 20K or 40K as the bottom resistance for the voltage divider. The inflection point for the 20K circuit is at 5 ^{o}C. The inflection point for the 40K circuit is at -10 ^{o}

Please look at the spreadsheet pages named “TRI Circuit Ri = 20K” and “TRI Circuit Ri = 40K” for my analysis on the TRI interfaces.

Improving on TRI's Thermistor Interface

It turns out that there are 3 things that you can do to improve the thermistor interface. They are as follows:

1. Oversample the ADC. The ADC is a 12-bit device. With 2 lines of TBASIC code you can get 14 bits out of the device. This gives you 4x the resolution and almost 0 ADC noise. This is a significant improvement in ADC behavior. And, since it cost you nothing, just do it.
2. Adjust the voltage divider so that the output curve is as flat as possible around the temperatures that you care about the most. Move the inflection point.
3. Look at Rt mW column at how much power is dissipated in the thermistor. The power that is lost in the thermistor turns ends up as heat and this can adversely affect the accuracy of the reading. You may have to pick a different a different thermistor that is not a 10K model. Thermistors are available in may values. I have sample spreadsheets for 10, 20 and 50K thermistors.

Increasing the resolution of the TRI ADCs from 12 bits to 14 bits

It is easy to increase the resolution of the ADC. Oversample, filter and then decimate (down sample).

The following is custom function is called every 0.1 second in my application. You should call this function at the lowest rate that makes sense for your application.

```
' Oversample ADC x16 (yes you can break TBASIC statements across multiple lines)
'
A = ADC(1) + ADC(1) + ADC(1) + ADC(1) + ADC(1) + ADC(1) + ADC(1) + ADC(1) +
      ADC(1) + ADC(1) + ADC(1) + ADC(1) + ADC(1) + ADC(1) + ADC(1) + ADC(1)

' Single pole low pass IIR filter ADC1Filter is used to filter out any remaining
'    ADC conversion noise. The filter is in units of ADC value x16.
'    ADC1Filter is one element in DM32[]. See the #Define table...
'
' The 7 and 8 constants determine the filter behavior. With these values
'    the next value of ADC1Filter will be composed of 7/8 of the current value
'    of the filter and 1/8 of the new value in 8.
'    Time constant for this filter will be 8 x the sample period.
'
 I have tested filter constants of 7-8, 15-16 and 31-32.
   The filter time constants will become to  8, 16 and 32 x the sample period.
'
' The "+ 4" term is not required for the filter. I added it because the "/ 8 " integer
'    division will round towards 0. Adding 4 (half of 8) will tend to make the integer
'    division round to the nearest integer. This just makes me happy.
'
ADC1Filter = ((ADC1Filter * 7) + A + 4) / 8
```

The next bit of code down samples the 16-bit data in the digital filter to 14 bits and then coverts this value to temperature. This code is usually in another custom function. I only call this code when necessary.

```
' Convert the 14-bit ADC value to temperature in the units of Cx100 ( scaled integer) in "T1C100"
'    T1C100 is one element in DM32[]. See the #Define table...
'
X = (ADC1Filter + 2)/4        ' down sample to 14 bit
y = ( x * 57848) / 100000 – 1611     ' y is in units of Cx100 (scaled integer)
```

Those 4 lines of code accomplish a lot or magic. This is what you get:

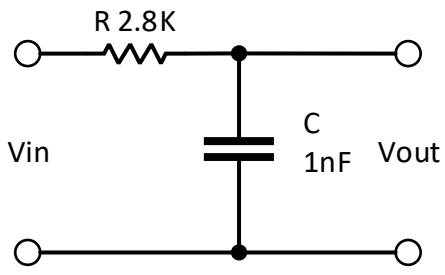
1. 14-bit ADC values from a 12-bit ADC. The 2 extra bits may allow you to see temperature changes as small as 0.01 degrees C around the inflection point of the thermistor curve.
2. Exceptionally low noise. The combination of oversampling and low pass filtering gets you very stable ADC readings. The approach is much simpler and much more effective than the moving average approach that TRI offers to solve the ADC noise problem.
3. Conversion of ADC measurements to temperature with a single line of TBASIC for many applications. This many eliminate the need for lookup tables and linear interpolation algorithms for many thermistor applications.
4. Easy field calibration. The "- 1611" is the intercept term and indicates that the line crosses the y axis at -16.11 ^{o}C. If you made this value a variable in EEPROM, you could change the value stored in the PLC and adjust the output of the equation. If the system was reading too high by 1.5 ^{o}C change the value to "-1761" to lower the output by 1.5 ^{o}C. This is a single point adjustment. Try that with a look up table approach.

If you what to learn more about oversampling, this is one of the best documents that I found on the web that explains the oversampling:

www.atmel.com/images/doc8003.pdf

Sorry, I do not have a good reference for IIR low pass filters. You will just have to trust me. This filter with 7/8 constants and a 10 Hz sample rate is the digital equivalent a single pole analog filter.

I use this filter as an exponential moving average of the ADC counts. It is simple to implement and does not use much memory to store old values. The Infinite part of IIR means that ever sample from the ADC affects the current value of the filter. The "exponential" part of this filter describes the fact that the newest values added to the filter have more weight that the values that went before.



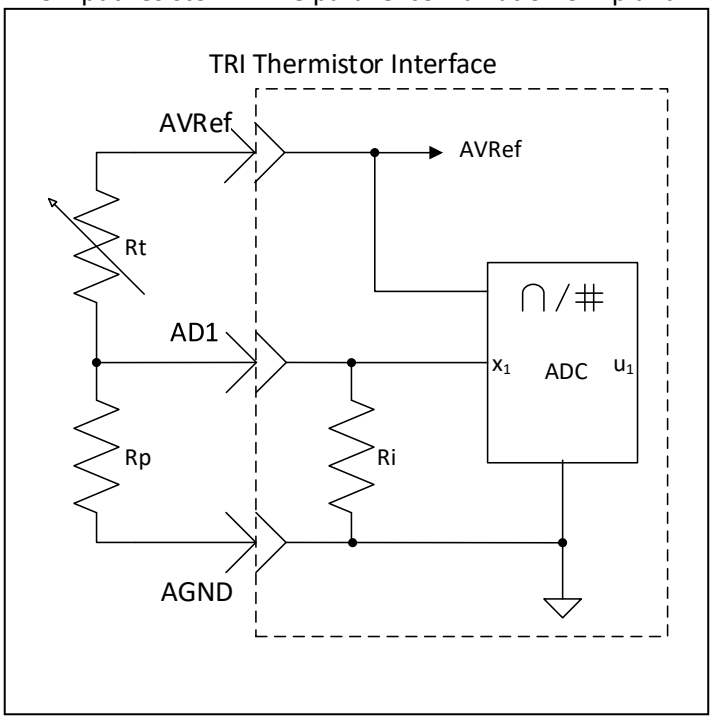
The time constant or the 7/8 filter at 10Hz sampling rage is 0.8 seconds. What this means is that if the input went for 0 to full ADC range, that the output of the filler will take 0.8 seconds to get to 62.3% of new value. Every time constant that passes will result in the filter output getting closer and closer to the input value.

This is classic low pass filter behavior. Every thermistor has a time constant value. Thermistor probes in free air may have time constants that approach 30 seconds. Stainless steel thermistor probes designed for immersion in water can have an 8 second time constant. If you are monitoring temperature in a 1000 liter cauldron of chicken soup, the temperature is not going to change all that fast. The cauldron may have a time constant that is measured in minutes or hours. As a result, 0.8 second time constant of this filter may not matter for your application.

Flattening the Curve.

The voltage divider circuit effects the shape of the voltage curve presented to the ADC. The voltage divider circuit with the lower resistor value of 20K results in an inflection point at 5 ^{o}C. The voltage curve is nearly flat on either side of the inflection point. Changing the resistance in the bottom ½ of the divider will move the inflection point.

The following circuit is what I am using to move the inflection point of the thermistor curve to a temperature that is important for my application. I just add a resistor, Rp, in parallel with the internal ADC input resistor Ri. The parallel combination of Rp and Ri will be lower than either resistor. You can find both voltage divider calculators and parallel resistance calculators on the internet. The trick is to pick the right value for Rp to move the inflection point to get the flattest response over the temperature range of interest.



The trick is to pick the right value for Rp to get the inflection point at the temperature that you are interested in. to be the most accurate. It is time to download and open the spread sheet.

Click on sheet named “Linearized Circuit 10K at 25C”. This spreadsheet is my optimization for a 10K thermistor with a β(25/50) = 3950k to get the inflection point at 25 ^{o}C.

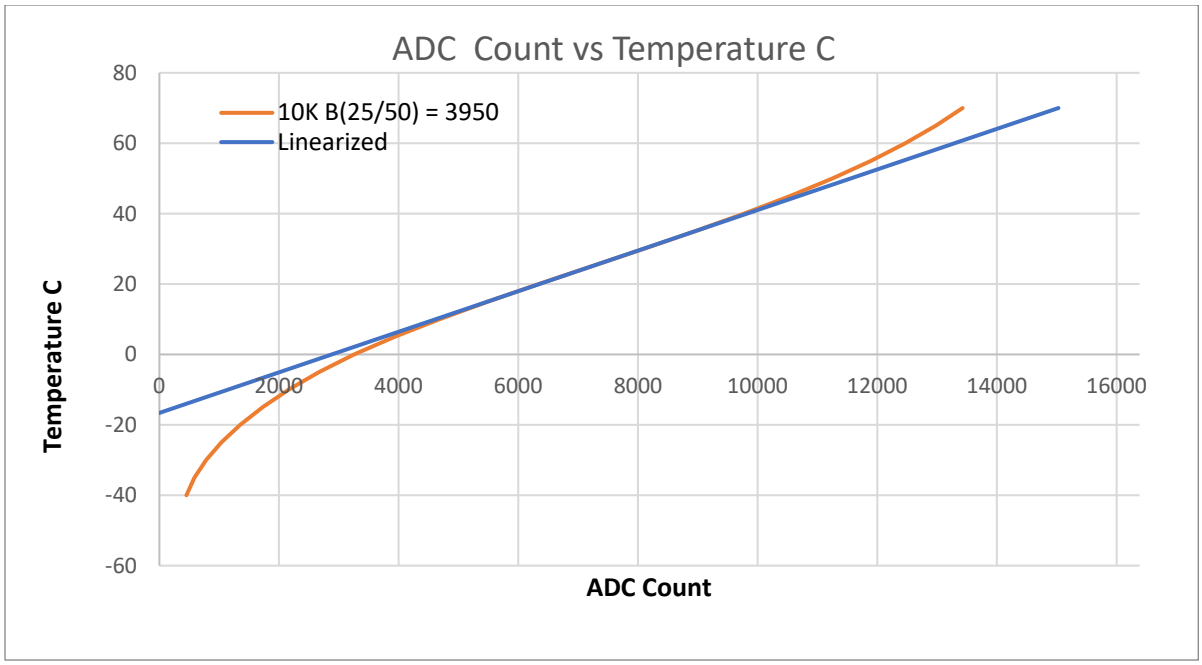
The thermistor modeled in this spreadsheet probably not the one that you are using. The flat spot may not be at the temperature that is most important for your application. But you need to start somewhere.

Spreadsheet Details

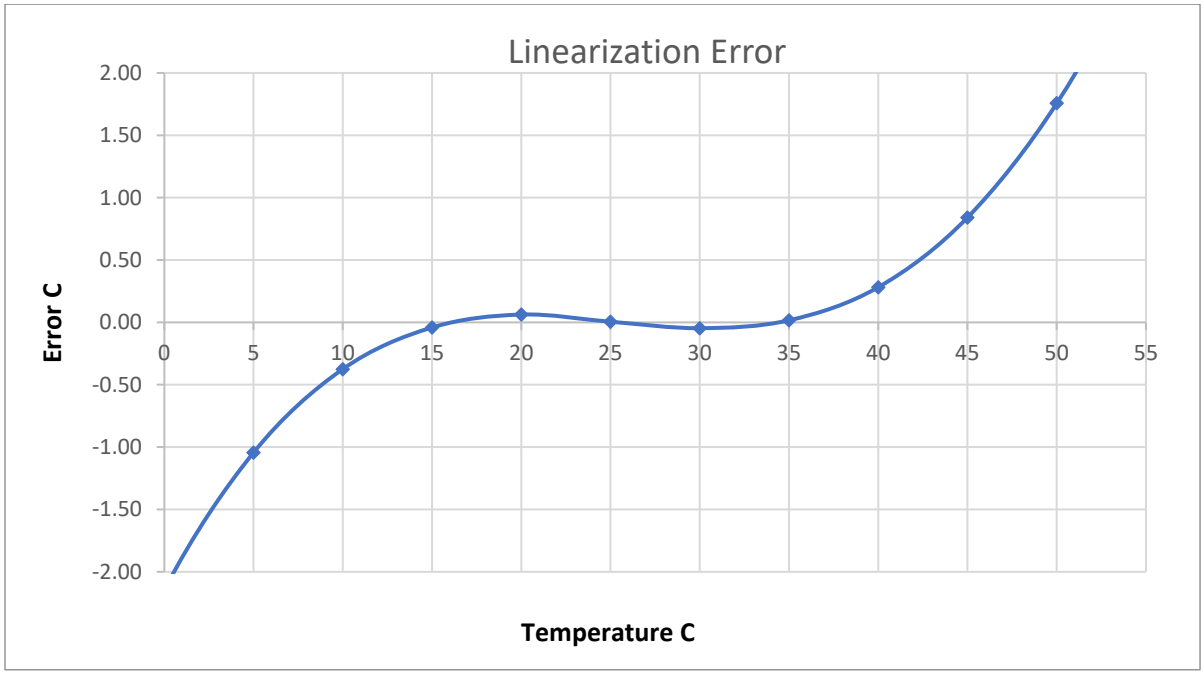
Open the sheet named “Linearized Circuit 10K at 25C”. This spreadsheet is my optimization for a 10K thermistor with a β(25/50) = 3950k to get the inflection point at 25 ^{o}C.

ADC Count vs Temperature C

This chart shows the thermistor curve in orange. The blue line is the calculated linear line that best matches the linear portion of the thermistor curve. Note that the thickness of the blue line covers over much of the orange curve.



The Linearization Error graph provides an amplified view the flat spot in the ADC Count vs Temperature C graph. The Linearization Error graph shows how closely the linear line matches the thermistor behavior. You see that from -10C to about 42C that the linear approximation is within +/- 0.5 ^{o}C from the actual thermistor curve. Please note that because the data points are 5 ^{o}C apart that this graph is a bit rough. Loook at the “Big 20K Chart” spereadsheet for an example that is based more data points.



ADC Interface Characteristics

This block describes your ADC and the linearization. The Blue fields are ones that you may need to change. The grey fields are things that the spread sheet calculates from your inputs.

ADC Interface Characteristics					
AVRef	ADC Bits	ADC Max	Ri	Rp	Rp || Ri
5	14	16383	20000	13000	7879

Definitions:

- AVRef is the Analog reference voltage. It is either 5 or 10 V
- ADC Bits are the number of bits that the ADC calculates. The native ADC is 12, the oversampled version is 14.
- Ri is the resistor from the ADC input to analog ground. This resistor is part of the TRI PLC board and is either 20K ohms, 40K ohms or high impedance. Enter 20000, 40000 or 0, respectively.
- RP is the external resistor value in ohms.
- Rp || Ri is the computed parallel resistance of Rp and Ri (external and internal resistors). The spread sheet does the calculation and uses the calculated value other calculations. I put this value here for your entertainment.

Thermistor Characteristics and Calculations

The thermistor characteristics are in a big table. The blue columns specify temperature and resistance values that you input. All the data in the chart are based on values that you entered. If you enter wonky (inaccurate) values, then the rest of the table will be wonky.

The yellow column, ADC Count, are the calculated values that the ADC should return for the value in the Temp C column. If you need to build a look up table for your application, this is the column of data that you need to copy into your lookup table mechanism.

The last 3 columns of the table starting with “Linear Count” are values used to generate the “Linearization Error” graph.

Linearized Circuit, Inflection @25C
Thermistor: 10K β(25/50) = 3950k

inflection

	Temp C	Rt	AD1 V	Rt V	Rt mA	Rt mW	ADC Count	dy/dx ADC/C	dy^2/dx^2 ADC/C	Linear Count	Error C	Linear C
	-40	277200	0.138	4.862	0.018	0.085	453			-4054	-25.98	-65.98
	-35	211500	0.180	4.820	0.023	0.110	588	32.90		-3187	-21.76	-56.76
	-30	157200	0.239	4.761	0.030	0.144	782	44.90	-2.43	-2320	-17.88	-47.88
	-25	116600	0.316	4.684	0.040	0.188	1037	57.20	-2.40	-1452	-14.35	-39.35
	-20	87430	0.413	4.587	0.052	0.241	1354	68.90	-2.36	-585	-11.18	-31.18
	-15	66920	0.527	4.473	0.067	0.299	1726	80.80	-2.56	282	-8.32	-23.32
	-10	51820	0.660	4.340	0.084	0.364	2162	94.50	-2.86	1150	-5.83	-15.83
	-5	40450	0.815	4.185	0.103	0.433	2671	109.40	-3.17	2017	-3.77	-8.77
	0	31770	0.994	4.006	0.126	0.505	3256	126.20	-3.34	2884	-2.14	-2.14
	5	24940	1.200	3.800	0.152	0.579	3933	142.80	-2.98	3752	-1.04	3.96
	10	19680	1.429	3.571	0.181	0.648	4684	156.00	-2.31	4619	-0.37	9.63
→	15	15620	1.676	3.324	0.213	0.707	5493	165.90	-1.67	5486	-0.04	14.96
→	20	12470	1.936	3.064	0.246	0.753	6343	172.70	-0.94	6354	0.06	20.06
→	25	10000	2.203	2.797	0.280	0.782	7220	175.30	-0.06	7221	0.01	25.01
→	30	8064	2.471	2.529	0.314	0.793	8096	173.30	0.75	8088	-0.05	29.95
→	35	6538	2.733	2.267	0.347	0.786	8953	167.80	1.42	8956	0.02	35.02
	40	5327	2.983	2.017	0.379	0.764	9774	159.10	1.99	9823	0.28	40.28
	45	4363	3.218	1.782	0.408	0.728	10544	147.90	2.39	10690	0.84	45.84
	50	3592	3.434	1.566	0.436	0.682	11253	135.20	2.62	11558	1.76	51.76
	55	2972	3.631	1.369	0.461	0.631	11896	121.70	2.69	12425	3.05	58.05
	60	2472	3.806	1.194	0.483	0.577	12470	108.30	2.61	13292	4.74	64.74
	65	2066	3.961	1.039	0.503	0.522	12979	95.60		14160	6.81	71.81
	70	1735	4.098	0.902	0.520	0.469	13426			15027	9.23	79.23

Definitions:

- The block of green arrows to the left of the chart is a digital post-it-note. It does nothing other than to remind me where the inflection point is supposed to be.
- **Temp C** is the list of temperatures for the system. For lookup table usage, the spacing between entries needs to be constant, every 1,2,5 degrees works.
- **Rt** Thermistor resistance at each temperature. I use this to calculate the power dissipated in the thermistor by the voltage divider circuit.
- **AD1 V** Calculated voltage at the input to the PLC ADC. Remember we are working with a voltage divider.
- **Rt V** the voltage that is dropped across the thermistor
- **Rt mA** current in mA that is flowing through the thermistor. This is the other bit of info that I need to calculate thermistor power dissipation. This is also, how much current that is being drawn from the Analog Reference Voltage provided by the PLC. If you are connecting multiple thermistors to the PLC be careful to not overload the AVRef supply.

- **Rt mW** power in milli-watts that is dissipated in the thermistor. This is also known as self-heating. Thermistors are rated in the units of mW/^{o}C. If the thermistor cannot dissipate this power to its surroundings, the thermistor will heat up and will report a temperature that is too high. If your thermistor is rated at 1 mW/^{o}C and you are dissipating 3 mW in the thermistor, your measurement may be 3 degrees to high.
- **ADC Count** this the calculated ADC count value based on Temp C. This column is highlighted in yellow. If you need to use a lookup table, these are the values that you will need.
- **dy/dx ADC/C** this is the 1st derivative of the temperature vs ADC function. The value represents the number of ADC counts per degree C. This gives you an idea of the resolution of the ADC at any given temperature. If you are expecting to read the thermistor to 0.1^{o}C and the dy/dx value is less than 10 then you will be disappointed.
- **Dy^2/dx^2 ADC/C** This is the 2nd derivative of the temperature vs ADC function. This is the rate of change of the rate of change of ADC count per degree squared. This value is a measure of how rapidly the ADC curve is changing shape. If the value is less than 0, the curve is slowing down or bending to the right. When this value is greater than 0 then the curve is accelerating or bending to the left. The magic point is when the 2nd derivative changes from negative to positive. This is the inflection point. You will see that the inflection point in this data table is at 25 ^{o}C.

The next 3 columns are based on drawing a straight line that is tangent to the inflection point. These values are used to figure out how accuracy of the tangent line.

- **Linear Count** This is an intermediate calculation that I use to measure the error between my straight line approximation of the thermistor's behavior and what the thermistor is doing.
- **Error C** This is the difference (error) between the linear model and the curve of the thermistor.
- **Linear C** This is the temperature in ^{o}C that you would get by using the linear equation.

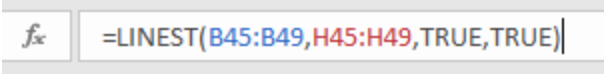
The linear equation for this tangent line may be sufficiently accurate for your application. If you can use the linear equation, then your application is easier to code and execute faster than what you'd get from table lookup and interpolation.

= LINEST() Table

The LINEST() table is a block of data produced by the Excel LINEST() function. LINEST() calculates the best fit line for a group of x and y values. I am using LINEST() to calculate a line that passes through the inflection point and is tangent to the ADC vs Temperature function at the inflection point.

=LINEST()		Values I care About	
0.005765	-16.6279483	Slope	Intercept
1.82E-05	0.133358097		
0.99997	0.049936866	R^2	
100250	3		
249.9925	0.007481072		

The blue square in the upper left side of the block is the calculated slope of this line. But this is where it gets messy, it is also where the Excel LINEST() function lives. Click on the blue square and then move your cursor up to the "Excel Name Box" and click on the equation. You should see something like this:



The Blue stuff is the range of known “y values”, the red stuff is the known “x values”. The point of LINEST if to compute the slope and intercept of the best straight line that runs through those data points.

Look at the big chart and you will see that the blue and red stuff has been highlighted:

	Temp C	Rt	AD1 V	Rt V	Rt mA	Rt mW	ADC Count	dy/ ADC
	-40	277200	0.138	4.862	0.018	0.085	453	
	-35	211500	0.180	4.820	0.023	0.110	588	32
	-30	157200	0.239	4.761	0.030	0.144	782	44
	-25	116600	0.316	4.684	0.040	0.188	1037	57
	-20	87430	0.413	4.587	0.052	0.241	1354	68
	-15	66920	0.527	4.473	0.067	0.299	1726	80
	-10	51820	0.660	4.340	0.084	0.364	2162	94
	-5	40450	0.815	4.185	0.103	0.433	2671	109
	0	31770	0.994	4.006	0.126	0.505	3256	126
	5	24940	1.200	3.800	0.152	0.579	3933	142
	10	19680	1.429	3.571	0.181	0.648	4684	156
→	15	15620	1.676	3.324	0.213	0.707	5493	165
→	20	12470	1.936	3.064	0.246	0.753	6343	172
→	25	10000	2.203	2.797	0.280	0.782	7220	175
→	30	8064	2.471	2.529	0.314	0.793	8096	173
→	35	6538	2.733	2.267	0.347	0.786	8953	167
	40	5327	2.983	2.017	0.379	0.764	9774	159
	45	4363	3.218	1.782	0.408	0.728	10544	147
	50	3592	3.434	1.566	0.436	0.682	11253	135
	55	2972	3.631	1.369	0.461	0.631	11896	121
	60	2472	3.806	1.194	0.483	0.577	12470	108
	65	2066	3.961	1.039	0.503	0.522	12979	95
	70	1735	4.098	0.902	0.520	0.469	13426	

If you need to change the focus of the optimization of the linear line you will need to slide the blue and red boxes up and down to pick the new center point. Just move your mouse over the top of bottom edge of a box and when the mouse pointer changes to cross with arrows click and hold and you can then move the selection box up and down the column. Move the green arrows so you will remember what you are trying to accomplish.

The blue and green cell in the LINEST() box are the calculated slope and offset for the best fit line through the inflection point. The green box on the 3rd row is the R^2 value that is a statistical measure of how accurately the computed line fit the x and y data. If the x,y data pairs are exactly on the computed line then the R^2 value would be 1.000, or a perfect match. If the x,y data are not a perfect match, then R^2

value will be less than 1.000. I get worried with the R^2 value is less than 0.990. The only reason I had to use LINEST() was to get the R^2 value, otherwise I would have used the SLOPE() and INTERCEPT() functions.

Line Equation Formulas for TBASIC

This area of the spreadsheet that generates TBASIC equations for converting ADC counts directly to temperature. stuff that you either need to modify for your application or may want to copy and paste into a custom function looks like this:

Value	Integer Cx10	Interger
Slope	5765	57648
Intercept	166	1663
Divisor	100000	100000

	Equations for TBASIC
Cx10	y = (x * 5765) / 100000 - 166
Cx100	y = (x * 57648) / 100000 - 1663
Float	y# = x * 5.7648E-3 - 1.6628E+1

In the box above the equation are two blue boxes. These contain an integer divisor value. These are used to scale the slope value up to something with enough digits to be accurate. I like to get 4 significant figures for the Cx10 Integer slope and 5 for the Cx100 equation. Change the value

The yellow stuff are the equations that can be copied and pasted into a TBASIC custom function. “x” is the ADC value. The equation will assign “y” with the temperature based x.

The Cx10 equation generates a scaled integer in the units of 0.1 ^{o}C. 25.7 ^{o}C is represented as the integer value 257. The Cx100 equation generates a scaled integer that is in the units of 0.01 ^{o}C.

The Float equation simply returns a floating point value without scaling. This works with the PLCs that support floating point math.

What Do I Recommend for Interfacing Thermistors to TRI’s PLCs?

- Collect the documentation for the thermistor. You will need an accurate resistance vs. temperature chart. You will need information on the dissipation factor (how much the thermistor’s in mW/^{o}C.
- Over sample the ADC to get 14 bits of resolution.
- Pick a value for the voltage divider Rp that moves the inflection point of the ADC count vs temperature curve into the temperature range that is most critical for your application.

What could go wrong?

Self-heating: The power dissipated in the thermistor may result in a significant measurement error. The dissipation factor for typical thermistors is from 1 to 2.7 mW/^{o}C. Please look carefully at the “Rt mW” column in the big table of computer stuff to see how much power is being dissipated across the temperature of interest.

There are a several things that you do if the power dissipated in the thermistor is too high:

- If you have the tools to determine the actual measurement error in your system, you could calculate correction information that could be baked into the ADC Count information that you are using in your PLC firmware.
- Pick a different thermistor. A 10K thermistor is called that because at 25 ^{o}C the thermistor has a resistance of 10K ohms. Thermistors come in families. 10K, 20K, 50K and 100K thermistors are common. I have spreadsheets for 10, 20 and 50K thermistors. Have a look at the spreadsheets to see if other families of thermistors will solve the problem.
- Add a solid state relay or analog switch between the thermistor and the AVRef voltage. The idea is to only connect the thermistor to power on the PLC scan before the analog measurement and then disconnect it immediately after the measurement. If the thermistor is powered only 10% of the time the power dissipated in the thermistor will be reduced by about 90%.
- Design a whole new circuit to power and linearize the thermistor. This can be done, but if you could do this sort of stuff, you would not be reading this document. The use of an external ADC module would allow you to use the voltage divider interface at a lower voltage. The I-7017 modules work with a 2.5V full scale ADC at 16 bits of resolution.
- Abandon trying to use thermistors for temperature measurement. Look at thermocouples and platinum RTDs and external electrons to go with them.
- Change your expectations. Just live with it.

Low ADC Resolution: If you are not getting enough ADC Counts/^{o}C for your application:

- Change the thermistor look at other thermistors such as 20..100K versions.
- You may have to move the inflection point in your linearization circuit.
- Abandon the PLC's ADC circuit and use an external higher resolution ADC. I have used the I-7017 data acquisition modules with the TRI PLCs with excellent results. These modules have 16-bit ADCs that will help with resolution. These modules also support differential inputs that will support full bridge linearization schemes.
- Abandon trying to use thermistors for temperature measurement. Look at thermocouples and platinum RTDs and external electrons to go with them.
- Change your expectations. Just live with it.

Suggested approach to using the spreadsheets

Practice on one of the existing spreadsheets.

Pick one of the spreadsheets that have be linearized around 25 ^{o}C and change the linearization to another temperature.

- Find the right value of Rp to move the inflection point. To your new target temperature.
- A good starting point is to make the parallel combination of Rp and Ri equal to the resistance of the thermistor at your new linearization temperature.
- Adjust Rp until your inflection point moves to your new temperature. There is a table of standard 1% resistor values on each sheet. You might as well use these values for Rp

At this point you have found the inflection point and it is time to adjust the x and y values used by the LINEST() function to generate a tangent line about the inflection point. This is described earlier in this document.

You will need to adjust the linearization graph. To get the inflection point centered on the graph. Hold your mouse over the temperature values running across the graph at right click. You should get a menu with the choice “Format Axis…”. Click on it. Now change the Minimum and Maximum values in the Bounds area to center the graph around the new inflection point.

Get to Work on your spread sheet

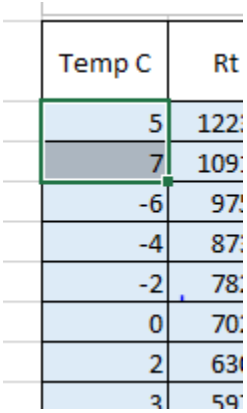
If you have made it this far, then you are ready to set up a spreadsheet that is focused on your application. I suggest that go to the “Big 20K Chart”. This chart is setup with over 64 entries spaced 2 degrees apart. 64 is more than enough entries for any practical usage, trust me.

Now start personalizing the chart for your usage:

Edit the ADC Interface Characteristics box to reflect the AVRef voltage and Ri (ADC input resistance) to match your application.

ADC Interface Characteristics					
AVRef	ADC Bits	ADC Max	Ri	Rp	Rp || Ri
5	14	16383	20000	80600	16024

Figure out the temperature range that you need. Say you want to start at 5 C and increment by 2 degrees. Enter 5 in place of -10 on the chart. Now replace -8 with 7. Now select your new 5 and 7 values and you should see a box with a square marker in the lower right corner:



Now move your mouse over the square until the cursor switches to a cross. Click and drag down to the bottom of the column and release the mouse when you get to the 110 value. Poof! You renumbered a bunch of stuff in about 20 seconds.

Now dig up your accurate thermistor data table for your thermistor. If you are either cleaver or lucky you may be able to get your data table into an Excel format. Copy from your table into the chart. Now right click over the Rt data and select the Paste option that just copies values (the clip board with “123”). This just pulls in the resistance values from your thermistor table without messing with the format of my spread sheet.

Now you need to adjust the inflection point for the curve:

- Move the green block or arrows to the left of the big table to center on your target temperature. This does not affect the calculations. It is just reminds about what my goal.
- Re-edit the LINEST() function to get the x and y values to line up with your green blob of arrows.
- Adjust Rp to get the inflection point for the big chart of data to center on the green block.

If everything looks right, you are ready to move on to the next step. You need to write some PLC code. You can use the yellow ADC Count values for a table look up approach or one of the linear equations.

Help

If you get stuck trying to figure this stuff out, you can ask questions on the forum. You can email me. I am stuck in my parent's basement until this COVID-19 stuff passes.

Gary Dickinson
gary.s.dickinson@gmail.com