Application Note 011
Version 005
14 Apr 2020

A Simple & Economical Method for Reading a Thermistor with an EtherMeter®.

Along with its two meter inputs and three digital I/O channels, the EtherMeter also features two analog inputs; and a useful application for one (or both) of these inputs is to collect raw voltage signals from a thermistor.

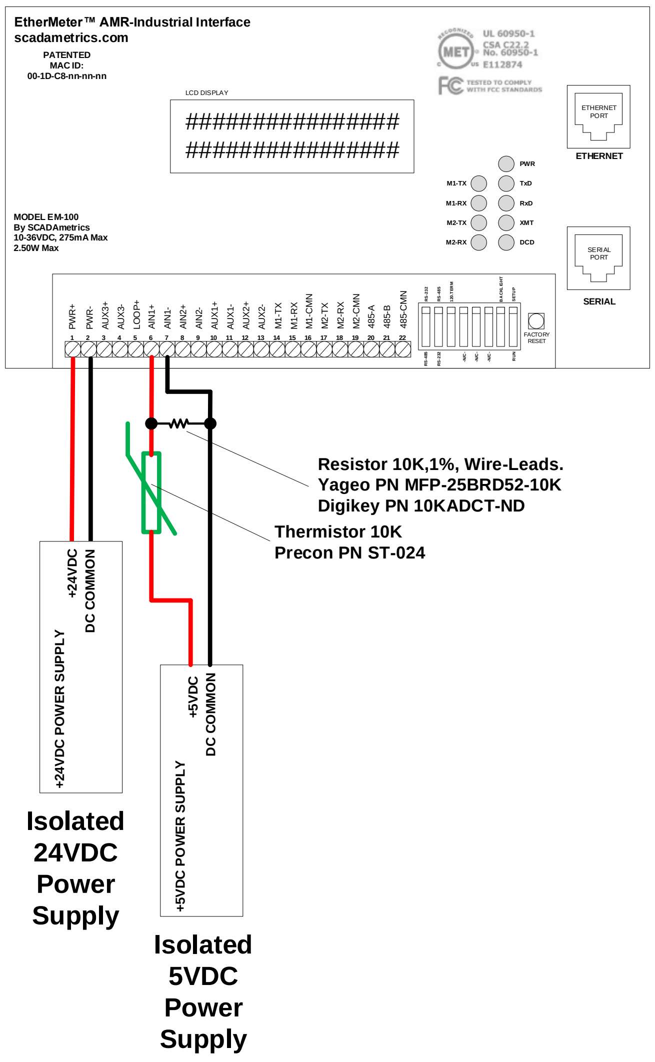
This Application Note details economical wiring and setup procedures for a Model ST-O24 NTC Thermistor (10Kohm@77degF, Manufactured by Precon). This document also presents the math formulas required to convert the raw voltage signal into a temperature value.

1. For this application, the internal 240 ohm precision resistor jumper JP1 should be removed, and a precision 10K ohm resistor should be inserted across the analog input channel terminals 6 and 7. In this way, the 10K precision resistor will form a voltage divider with the thermistor.

2. At the EtherMeter Setup Menu, the following command should be entered: "SET AIN1 V"
This modifies the configuration of AIN1 from a MA to a VOLTAGE input.

3. It is required that the EtherMeter and the thermistor be powered by separate voltages: The EtherMeter should be powered by +24VDC (10-36VDC), and the thermistor should be powered by +5VDC. It is imperative that both power supplies feature galvanically-isolated outputs. The output on the +5VDC power supply should be adjusted using a voltmeter to be as close to 5.000VDC as possible.

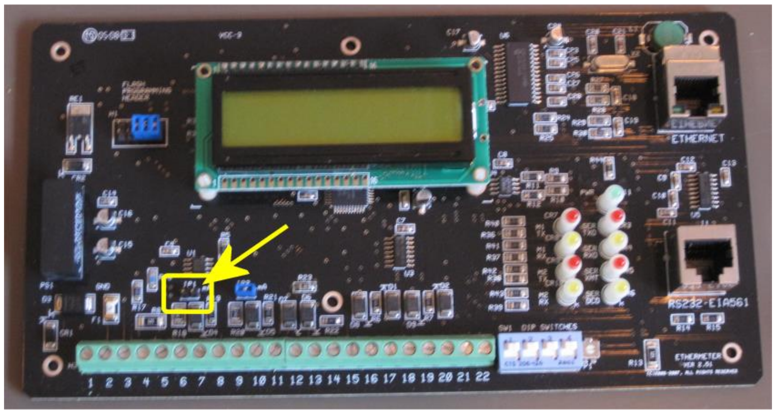
4. AIN1 Wiring should be as illustrated in the Figure to the right. Alternatively, Channel AIN2 could also be used.

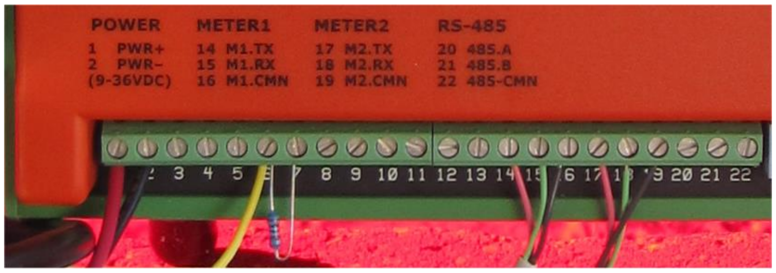


5. At very low temperatures, the thermistor resistance is very high, and therefore the input voltage to the EtherMeter will be very close to zero. At increasingly higher temperatures, though, the thermistor resistance will decrease, thereby increasing the input voltage to the EtherMeter.

6. The raw analog reading for AIN1 will then be stored within the EtherMeter's MODBUS holding register 40011. The contents of this register will be an integer value between 0 and 10,000, where 0 corresponds to +0.000V and 10,000 corresponds to +5.000V.

7. Taking into account the 10-bit analog input resolution of the EtherMeter and the resistance range of the ST-O24 thermistor, the described configuration can be used to gather temperature readings within a -35degF to 185degF range (and beyond) at approximately ¼ degF increments.


JP1 Removed (Converts AIN1 Input from Milliamp to Voltage)



10K Ohm Precision Resistor Installed Across Terminals 6 and 7.

Master (Client) Device Programming

At this point, the wiring and configuration of the EtherMeter is complete. Now, the remote MODBUS client device should be programmed to convert the EtherMeter's 40011 holding register into a useful temperature reading.

In order to develop the correct formula for the PRECON ST-O24, the resistance versus temperature chart was downloaded from kele.com: link. (The chart is also attached to this document below.)

And with this table in hand, the data was curve-fit into the Steinhart-Hart Formula, which is described by the following equation:

$$\frac{1}{T} = A + B\ln(R) + C(\ln(R))^3$$

… where T is the temperature in Kelvin, and the constants A, B, & C are as follows:

A:	1.1296E-3
B:	2.3408E-4
C:	8.7710E-8

Conversion from Kelvin to Celsius is straightforward… $T_C = T_K - 273.15$

And conversion from Celsius to Fahrenheit is also straightforward… $T_F = (1.8 \times T_C) + 32.0$

A sample computer program was written to demonstrate the formula, and it is attached below. This program was compiled using Microsoft Visual C++ Version 6.0 under the Windows XP Professional Operating System.

SAMPLE ‘C’ PROGRAM
CONVERSION OF ETHERMETER VOLTAGE DATA INTO TEMPERATURE

```
#include <stdio.h>
#include <math.h>
float tempF( int input_value ) ;

// Main program that demonstrates the tempF()
// voltage-temperature conversion function.

int main( int argc , char *argv[] )
   {
   int    input_value ;
   float  tC , tF ;

   if ( argc == 2 )
      {
      sscanf( argv[1] , "%d" , &input_value ) ;
      tF = tempF( input_value ) ;
      printf( "Input Value        : %d\r\n" , input_value ) ;
      printf( "Temperature (degF) : %f\r\n" , tF ) ;
      }
   else
      {
      printf("Usage: temp.exe nnn\r\n”);
      printf(“where nnn is a value between 0 and 10000)\r\n") ;
      }

   return(1);
   }

// Function To Convert EtherMeter MODBUS Register 40011 to Temperature
// For A PRECON ST-O24 Thermistor.

float tempF( int input_value )
   {
   double a , b , c ;
   double finput , volts, current, total_r , therm_r , temp_out ;

   a = (double)1.1296e-3 ;
   b = (double)2.3408e-4 ;
   c = (double)8.7710e-8 ;

   finput = (float)input_value ;

   volts   = 5.0 * finput / 10000.0 ;  // 0=0V, 10000=5V
   current = volts / 10000.0 ;         // 10K Ohm Precision Res. Across Terms 6,7
   total_r = 5.0 / current ;           // Total Resistance (Thermistor + 10K Ohm)
   therm_r = total_r - 10000.0 ;       // Thermistor Resistance Only.

   printf("Volts   : %f\r\n",volts);
   printf("Current : %f\r\n",current);
   printf("Rtot    : %f\r\n",total_r);
   printf("Rtherm  : %f\r\n",therm_r);

   if ( therm_r > 0.0 )
      {
      temp_out = ( 1.0 / ( a + b*log(therm_r) + c*pow(log(therm_r),3) ) ) - 273.15 ;
      temp_out = temp_out * 1.8 + 32.0 ;
      }
   else
      {
      // If thermistor resistance zero or less, then avoid a divide-by-zero
      // or other overflow error by assigning a very low temperature and exiting.
      temp_out = -328.0 ;
      }
   return((float)temp_out) ;
   }
```

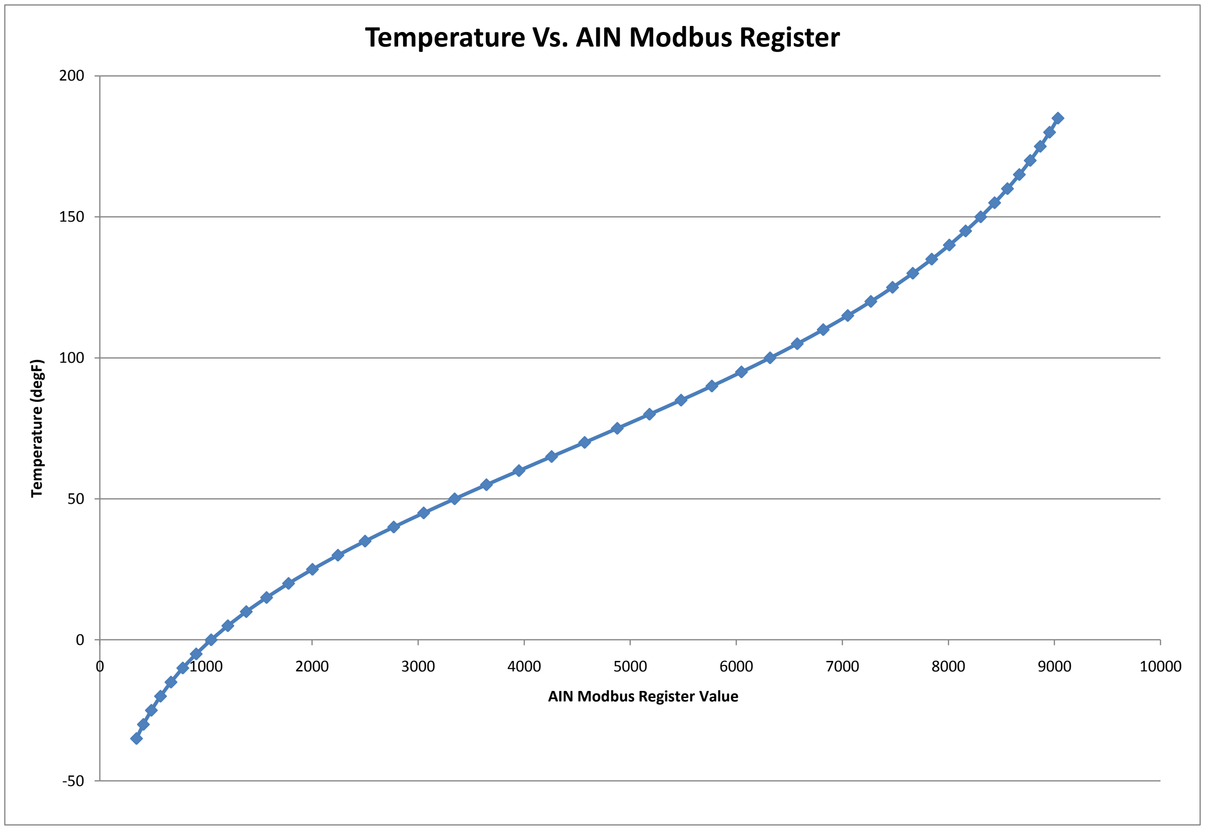
RESISTANCE VERSUS TEMPERATURE
PRECON ST-O24 THERMISTOR

degF	degC	Resistance (Ohms)
-35	-37.2222	280100
-30	-34.4444	234100
-25	-31.6667	196300
-20	-28.8889	165100
-15	-26.1111	139300
-10	-23.3333	118000
-5	-20.5556	100200
0	-17.7778	85350
5	-15	72910
10	-12.2222	62480
15	-9.44444	53640
20	-6.66667	46230
25	-3.88889	39910
30	-1.11111	34560
35	1.666667	30000
40	4.444444	26100
45	7.222222	22760
50	10	19900
55	12.77778	17440
60	15.55556	15310
65	18.33333	13480
70	21.11111	11880
75	23.88889	10500
80	26.66667	9298
85	29.44444	8250
90	32.22222	7331
95	35	6532
100	37.77778	5826
105	40.55556	5209
110	43.33333	4663
115	46.11111	4182
120	48.88889	3757
125	51.66667	3381
130	54.44444	3047
135	57.22222	2750
140	60	2486
145	62.77778	2251
150	65.55556	2041
155	68.33333	1854
160	71.11111	1686
165	73.88889	1535
170	76.66667	1400
175	79.44444	1278
180	82.22222	1168
185	85	1070

TEMPERATURE VERSUS INPUT VOLTAGE
(MODBUS REGISTER 40011 = 2000 x INPUT VOLTAGE)
TABULAR FORM

AIN Voltage	AIN Register	degF
0.172	345	-35
0.205	410	-30
0.242	485	-25
0.286	571	-20
0.335	670	-15
0.391	781	-10
0.454	907	-5
0.524	1049	0
0.603	1206	5
0.690	1380	10
0.786	1571	15
0.889	1778	20
1.002	2004	25
1.122	2244	30
1.250	2500	35
1.385	2770	40
1.526	3053	45
1.672	3344	50
1.822	3644	55
1.976	3951	60
2.129	4259	65
2.285	4570	70
2.439	4878	75
2.591	5182	80
2.740	5479	85
2.885	5770	90
3.024	6049	95
3.159	6319	100
3.288	6575	105
3.410	6820	110
3.526	7051	115
3.635	7269	120
3.737	7473	125
3.832	7665	130
3.922	7843	135
4.004	8009	140
4.081	8163	145
4.152	8305	150
4.218	8436	155
4.279	8557	160
4.335	8669	165
4.386	8772	170
4.433	8867	175
4.477	8954	180
4.517	9033	185

TEMPERATURE VERSUS CONTENTS OF MODBUS REGISTER 40011 GRAPHICAL FORM


Temperature Vs. AIN Modbus Register
Temperature (degF)
AIN Modbus Register Value