Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

THIS SPEC IS OBSOLETE

Spec No: 001-41448

Spec Title: THERMISTOR LOOKUP TABLE GENERATION TOOL - AN2395

Sunset Owner: M Ganesh Raaja (graa)

Replaced by: NONE

Thermistor Lookup Table Generation Tool

AN2395

Author: Petro Sasnyk
Associated Project: Yes
Associated Part Family: All
Software Version: PSoC® Designer™ 4.2 or PSoC Express™ 2.0
Associated Application Notes: AN2017, AN2107, AN2260, AN2314
PSoC Application Notes Index

Application Note Abstract

This application note describes a PC GUI tool for generating thermistor lookup tables based on datasheets or experimentally measured data. The tool allows you to use arbitrary expressions to generate the lookup table. Example PSoC® projects are included to show how to use these tables to measure temperature.

Introduction

A thermistor is an electronic component that exhibits a large change in resistance with a change in its temperature. The word *thermistor* is a contraction of the words *thermal resistor*. The thermistors discussed in this application note are semiconductors that have either a large positive temperature coefficient (PTC) or a large negative temperature coefficient (NTC) of resistance. Both PTC and NTC thermistors have features and advantages that make them ideal for certain sensor applications. Thermistors are widely used for accurate temperature measurements.

Cypress has several other application notes based on the resistance-temperature characteristic, such as:

- AN2017, “A Thermistor-Based Thermometer, PSoC Style“
- AN2107, “A Multi-Chemistry Battery Charger“

These applications use some method of resistance measurement with an ADC. The result is some ADC code that corresponds to a certain temperature. Unlike those documents, this application note focuses on how the temperature is calculated.

Theory of Operation

To calculate temperature using thermistor resistance, you need a mathematical description of the thermistor’s resistance-temperature dependence characteristic. There are a few possible analytical description methods of thermistor resistance versus the temperature dependency.

As a first-order approximation, we can assume that the relationship between the resistance logarithm and temperature is inversely proportional using a linear approximation:

$$T_k = \frac{1}{A + B \ln R}$$ Equation 1

R is the thermistor resistance. T_k is thermistor body temperature in kelvins. *A*, *B* are coefficients that characterize the thermistor.

In practice, the linear approximation shown in Equation 1 works accurately only over a small temperature range. A real thermistor has a more complex temperature-to-resistance relationship. The first-order approximation will not be accurate enough for many applications. A more complicated analytical expression must be used to describe the thermistor behavior with better precision. There are several methods that can be used, but in practice, the Steinhart-Hart equation is the most usable.

Steinhart-Hart Equation

The Steinhart-Hart equation is a third-order approximation formula (expression) that uses three approximation coefficients A, B, and C. These coefficients are the empirical constants. They must be specified for each device type:

$$T_k = \frac{1}{A + B\ln R + C(\ln R)^3}$$ Equation 2

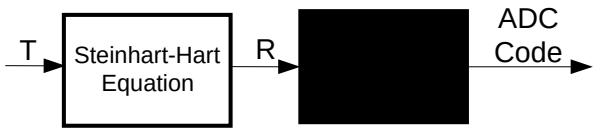
A, *B*, and *C* are Steinhart-Hart coefficients. T_k is temperature in kelvins. R is resistance in Ohms.

Equation 2 shows that to calculate temperature you must know the thermistor resistance and its approximation function. Approximation coefficients are often given in the thermistor manufacturer's data sheet. If not, the coefficients can be found from a resistance table by using a curve-fitting technique. When approximation coefficients are properly defined, the error of the Steinhart-Hart equation for good quality thermistors is generally less than 0.1°C over a wide temperature range.

Temperature calculation using an analytical formula such as Equation 2 requires floating-point arithmetic that is resource inefficient for an 8-bit PSoC microcontroller. The lookup table conversion method is better for PSoC devices. A lookup table can be built in several ways. One way is to store the temperature values in the table as a function of resistance.

A second way is to store the resistance values as a function of temperature. The table index reflects the temperature value. For most applications the temperature index is a constant step. For example, if your application is an outdoor thermometer, accuracy of one-half degrees Celsius is enough. You should keep the same accuracy over the whole temperature operation range. For this example, a table of resistance as a function of temperature is better. The resistance is measured using an ADC in most cases. Therefore, the table should be built as a table of ADC codes as a function of temperature. This is the approach used in this application note. The process is shown in Figure 1.

Figure 1. Temperature to ADC Code Conversion Process Flowchart



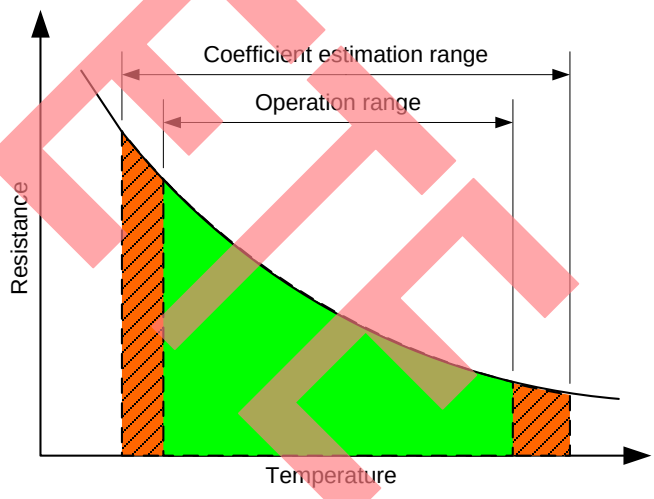
To build the table, you must calculate the resistance of the thermistor for a certain temperature and then take into account the measurement function that converts this resistance into a measured value (ADC code, frequency, period, and so on). You want to know what the measurement function is so that you know what ADC codes correspond to various resistance values. This allows you to create a table of ADC codes as a function of temperature so that you do not have to do any calculation to determine temperature.

The measurement function can be calculated from the measuring circuit and the resistance-temperature dependence can be found by solving the Steinhart-Hart equation.

To solve this equation you need to find the approximation coefficients A, B, and C. Some manufactures specify these coefficients in technical documentation.

If the coefficients cannot be located in the data sheets, they can be found experimentally by measuring the resistance of the thermistor at a series of known temperatures. The PC tool described in this application note allows you to enter these values from technical documentation or find them from experimental measurements. Coefficients found experimentally in a range of temperatures that represents your target operation range are better tailored to your application. If you calculate approximation coefficients from the values in the data sheet, the calculated data temperature range should enclose the device operation range as shown in Figure 2.

Figure 2. Parameter Estimation Temperature Range Needs to Cover the Target Device Operation Range



Estimating Steinhart-Hart Coefficients

To find the Steinhart-Hart coefficients from experimental data, use a least-squares curve fitting Levenberg-Marquardt algorithm (Reference [1]). The parameters are found by minimizing the goal function shown in Equation 3:

$$F(A,B,C) = \sum_i (T_{ie} - T_{ic})^2 \rightarrow \min \qquad \text{Equation 3}$$

T_{ie} is the temperature value for the resistance subscript i.

T_{ic} is the temperature calculated by Equation 2.

A least-squares Levenberg-Marquardt algorithm has two stages. In the first stage coefficient C is set equal to 0 (the first-order Equation 1 is used), and the A_0 and B_0 coefficients are calculated.

$$C_0 = 0$$

$$T_{i0} = \frac{1}{A_0 + B_0 x} - 273, x = \ln R, \qquad \text{Equation 4}$$

In the second stage, the A_0 and B_0 coefficients previously calculated are used as initial approximations. The temperature is calculated by Steinhart-Hart Equation 2. This two-stage procedure provides a reliable method of finding approximation coefficients for different thermistors.

Strategy of Selecting the Optimal Root

To get the relationship between temperature and resistance you must rewrite Equation 2 in another form and denote the value $\ln R$ as *x*. This results in the following representation of Equation 2:

$$Cx^3 + Bx + A - \frac{1}{T_k} = 0; \qquad \text{Equation 5}$$

$$x = \ln R$$

This can be rewritten as:

$$x^3 + 0 \cdot x^2 + \frac{B}{C}x + \left(A - \frac{1}{T_k}\right)\frac{1}{C} = 0 \qquad \text{Equation 6}$$

Equation 6 is a cubic equation of the form:

$$x^3 + a \cdot x^2 + b \cdot x + c = 0 \qquad \text{Equation 7}$$

Where:

$$a = 0,\ b = \frac{B}{C},\ c = \left(A - \frac{1}{T_k}\right)\frac{1}{C}.$$

To find a thermistor's resistance, you must solve the cubic Equation 5 relative to the $\ln R$ value. If you divide the Steinhart-Hart equation by coefficient C, you get a cubic equation in its standard form shown in Equation 6. The roots of this equation are found using Cardano's formula. In general, cubic equations can have up to three roots (see Figure 3 and Figure 4).

Figure 3. One Possible Root

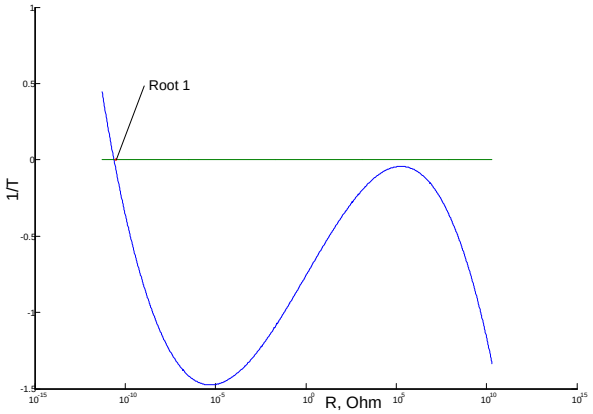
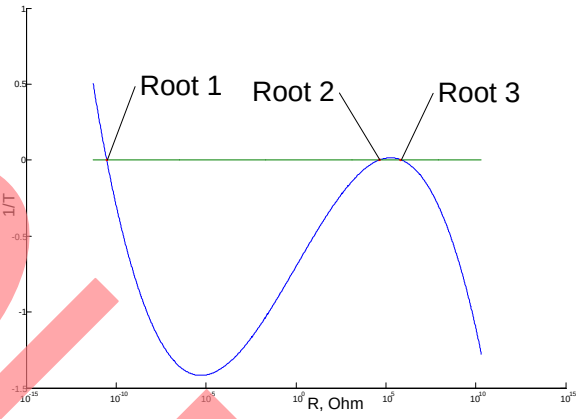


Figure 4. Up to Three Possible Roots



If there is just one root, then that root is used for resistance calculation. If there are three roots, you must discard the two extra roots. Use a two-step analysis for this. First, analyze a derivative of the $F(A,B,C)$ function (see Equation 6). As seen in Figure 3, near the valid root area the derivative of $F(A,B,C)$ function is negative for NTC thermistors and positive for PTC thermistors. For the given case, roots one and three are going to have the proper sign (see Figure 4). Because the type of thermistor is defined using B_0 coefficient, the first step results in discarding root two.

The second step is to decide which of the two remaining roots corresponds to the real thermistor's resistance. If you found the Steinhart-Hart coefficients analytically, then choose the maximum root. If you found the Steinhart-Hart coefficients by table approximation, then you will find the valid root by comparing the calculated resistance values for a given temperature with the value calculated by the simplified approximation formula, as shown in Equation 4. Choose the root that is closest to this value.

Selecting Thermistor Input Data Format

The PC tool allows you to enter thermistor data in the analytical (Steinhart-Hart) or table (resistance versus temperature) formats. If you enter the data in table form, the coefficients are calculated by considering all temperature experimental points. Therefore, the table method is usually more accurate than the analytical method.

Steinhart-Hart coefficients provided by thermistor manufacturers are typically calculated for three calibration points depending on temperature values that are more representative of the thermistor's usage temperature range:

- 20°C, 0°C, 50°C
- 0°C, 25°C, 70°C
- 25°C, 100°C, 150°C

If you use approximation coefficients from data sheet tables, the device operation range should be enclosed in the range where thermistor parameters are calculated. For example, if your charger should operate from 0°C to 60°C, then the thermistor resistance table of approximation coefficient estimates should be from –10°C to +85°C.

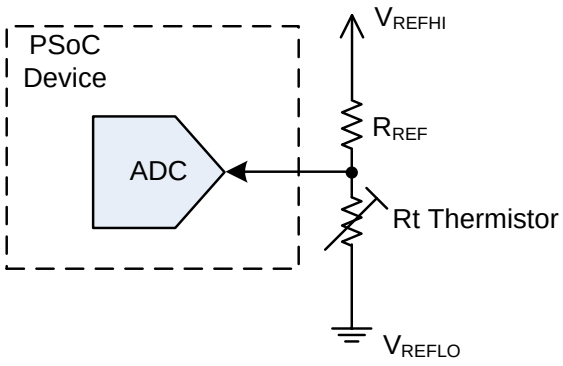
Thermistor Resistance Measurement Scheme Examples

The PC GUI tool has a built-in expression parser that allows you to enter arbitrary expressions for lookup table calculations. The following examples illustrate possible variants for thermistor resistance measurement using an ADC. They can be adapted for your application demands. The equations are written so that they can accept signed or unsigned ADC output formats of varying resolution by setting the ADC characterization constants NADCMax and NADCMin. For example, for an 8-bit ADC with unsigned output data format, set NADCMin to '0', NADCMax to '255'.

The PC tool can generate tables even if another thermistor resistance measurement principle is used, for example, using the frequency measurement of an RC oscillator.

The simplest measurement scheme is shown in Figure 5. This is the method used in AN2017, "A Thermistor Based Thermometer, PSoC Style." The thermistor is connected between the Vrefhi and Vreflow sources.

Figure 5. Simple Resistive Divider Circuit

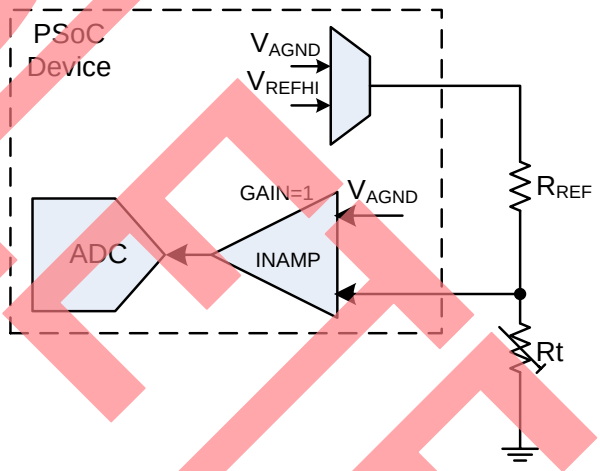


$$n_{ADC} = (n_{max} - n_{min}) \times \frac{R_t}{R_{REF} + R_t} + n_{min}$$ Equation 8

The formula for this example is given below:

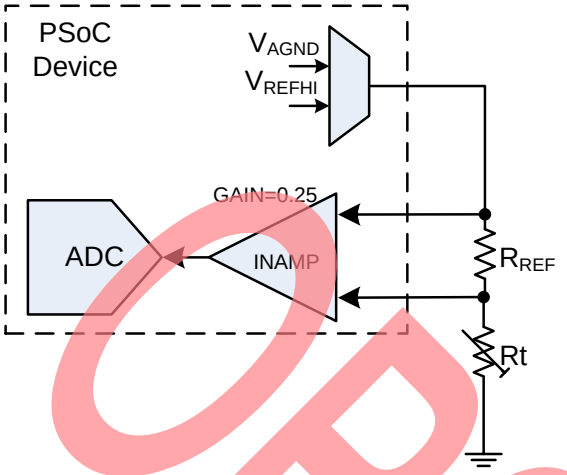
The following schemes were used in AN2107, "A Multi-Chemistry Battery Charger," AN2260, "Rapid NiCd/NiMH Battery Charger and DC Brushed Motor Controller for Autonomous Appliances," AN2314, "Thermistor-Based Temperature Measurement in Battery Packs," and other application notes. They all use correlated double sampling methods to cancel the offset errors.

Figure 6. Measurement Scheme with Offset Error Cancellation, Variant One



$$n_{ADC2} - n_{ADC1} = G_{INA} \times \frac{(n_{max} - n_{min})}{2} \times \frac{R_t}{R_{REF} + R_t}$$ Equation 9

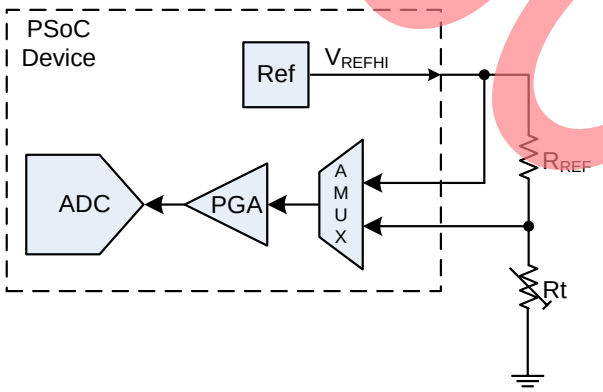
Figure 7. Measurement Scheme with Offset Error Cancellation, Variant Two



$$n_{ADC2} - n_{ADC1} = G_{INA} \times \frac{(n_{\max} - n_{\min})}{2} \times \times \frac{R_{REF}}{R_{REF} + R_t}$$

Equation 10

Figure 8. Measurement Scheme with Offset Error Cancellation, Variant Three



$$n_{ADC1} - n_{ADC2} = (n_{\max} - n_{\min}) \times G_{PGA} \times \times \frac{R_{REF}}{R_{REF} + R_t} + n_{\min}$$

Equation 11

Using the Lookup Tables

The tool generates lookup tables as C or ASM project header files. The following code fragments show examples of algorithms that can be used to search the tables and convert ADC codes into temperature.

Code 1. *ADCLookUpTable.h*

```
//---------------------------------------
#ifndef ADC_LOOKUP_TABLE_DEF_H
#define ADC_LOOKUP_TABLE_DEF_H
#include "m8c.h"
//---------- Start of Lookup Table -------
//--------------Part Number: PART-NUM
//---------------------------------------
#define LOOKUP_TABLE_SIZE 34
#define TEMP_MIN_VALUE -40
#define TEMP_STEP  5
extern const BYTE ResistanceLookupTable
[LOOKUP_TABLE_SIZE];
//---------------------------------------
#endif
```

Code 2. *ADCLookUpTable.c*

```
#include "ADCLookUpTable.h"
//---------- Start of Lookup Table -------
//--------------Part Number: PART-NUM
//---------------------------------------
const BYTE ResistanceLookupTable
[LOOKUP_TABLE_SIZE] = {
// Temp Min: -40 - Temp MAx: 125, Temp Step: 5
       229,   222,   213,   203,   191,   178,
165,   150,
       136,   122,   108,    95,    84,    73,
63,    55,
        47,    41,    35,    31,    27,    23,
20,    17,
        15,    13,    12,    10,     9,     8,
7,     6,
         6,     5

};
//---------- End of Lookup Table  -------
```

Code 3. Fragment of *main.c*

```
#include <stdlib.h>
#include "ADCLookUpTable.h"

INT iTb = 100;
CHAR iTemperature =0;  // thermistor temperature
INT iADCtoTemp(INT iTb);

/// Temperature ADC code to temperature value
INT iADCtoTemp(INT iTb)
{
    INT i, iTemperature, iDistance,
iMinDistance, iIndex;

    iMinDistance = abs(ResistanceLookupTable[0]-
iTb);
    iIndex = 0;
    // find position (iIndex) in temperature
lookup table
    // where the value is nearest to measured

    for (i = 0; i < LOOKUP_TABLE_SIZE; i++)
    {
        iDistance = abs(ResistanceLookupTable[i]
- iTb);
        if (iDistance <= iMinDistance)
        {
            iMinDistance = iDistance;
            iIndex = i;
        }
    }
    // the real temperature value is:
    iTemperature = iIndex * TEMP_STEP  +
TEMP_MIN_VALUE;
```

```
    return iTemperature;
void main()
{
    iTemperature = iADCtoTemp(iTb); // iTb - ADC
code, cTemperature - calculated temperature

…..
}
```

To use the lookup tables generated by the PC GUI tool in your own application, do the following:

- Add ADCLookUpTable.h and ADCLookUpTable.*c* to your project. You can change the names of the files if you want.
- Include the temperature calculating function iADCtoTemp in the file where you will use the lookup tables; *main.c*, for example.
- Include the header file with the lookup table declaration.
- Call the function and pass it an ADC code for which you want to find a corresponding temperature, as shown in the fragment of main() shown in **Code 3**.

Summary

This application note describes a PC GUI tool that generates lookup tables. It allows you to automate the table generation process and speed up your design cycle.

References

1. Press, William H., et al. Numerical Recipes in C: The Art of Scientific Computing, Second Edition. Cambridge, NY: Cambridge University Press, 1988.

About the Author

Name: Petro Sasnyk

Title: Engineering Manager

Background: Petro graduated from Lviv Polytechnic National University (Lviv, Ukraine) in 2003. He is currently a post-graduate student at this university. His interests include embedded systems design and signal processing.

Contact: stoune@gmail.com

Appendix

Figure 9. Window for Entering Approximation Parameters

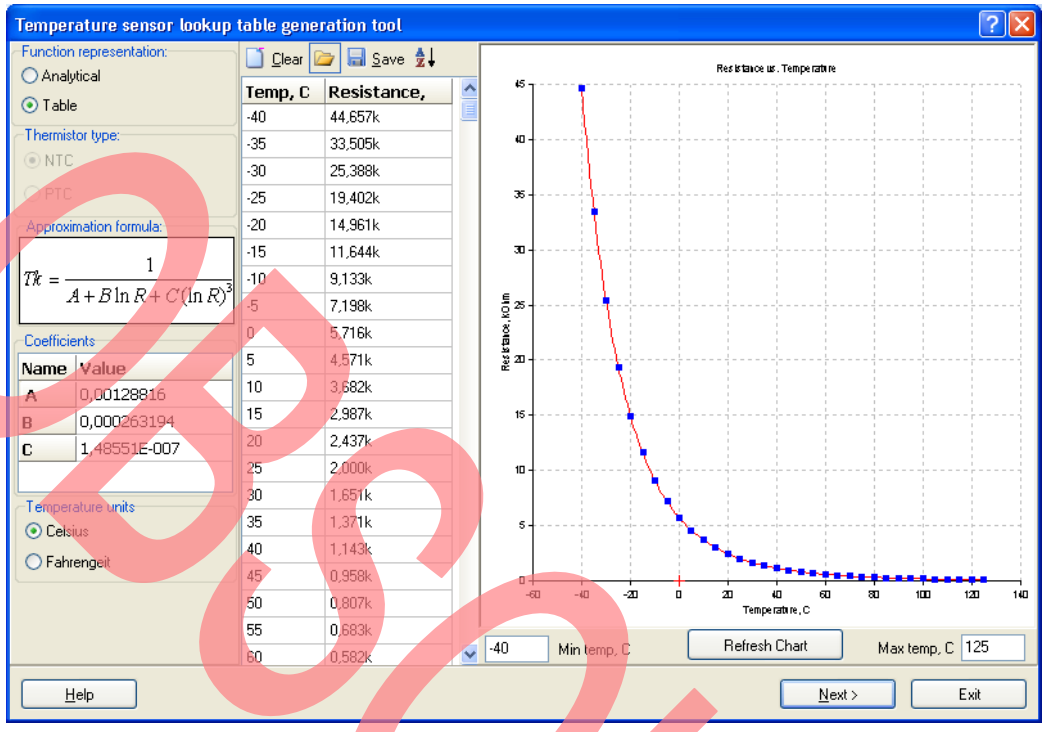


Figure 10. Window for Entering Circuit Parameters

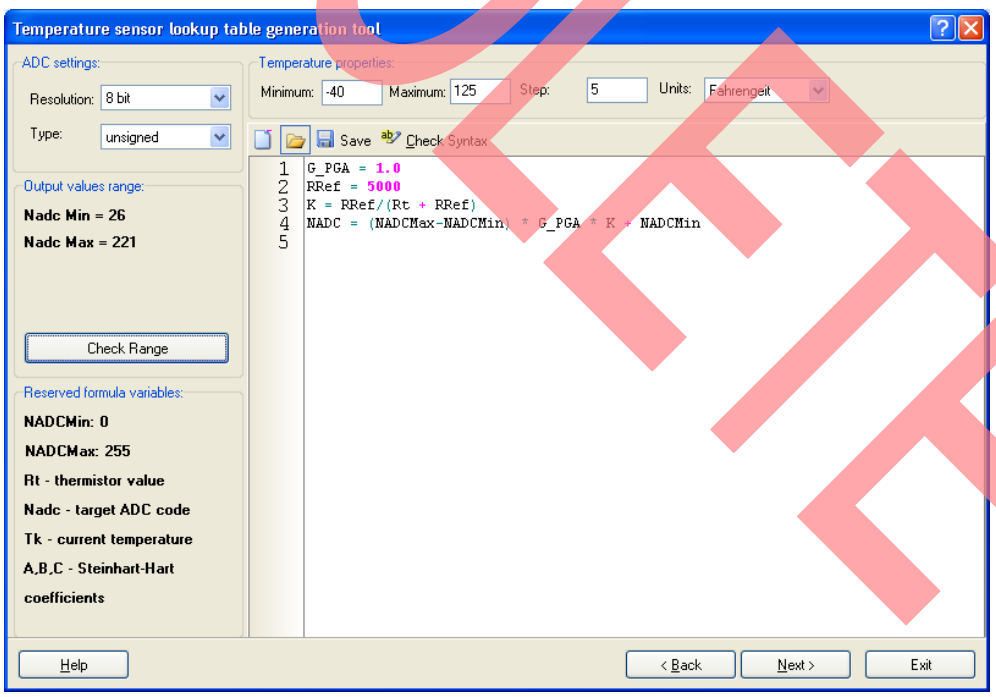


Figure 11. Preview of Generated Sequences

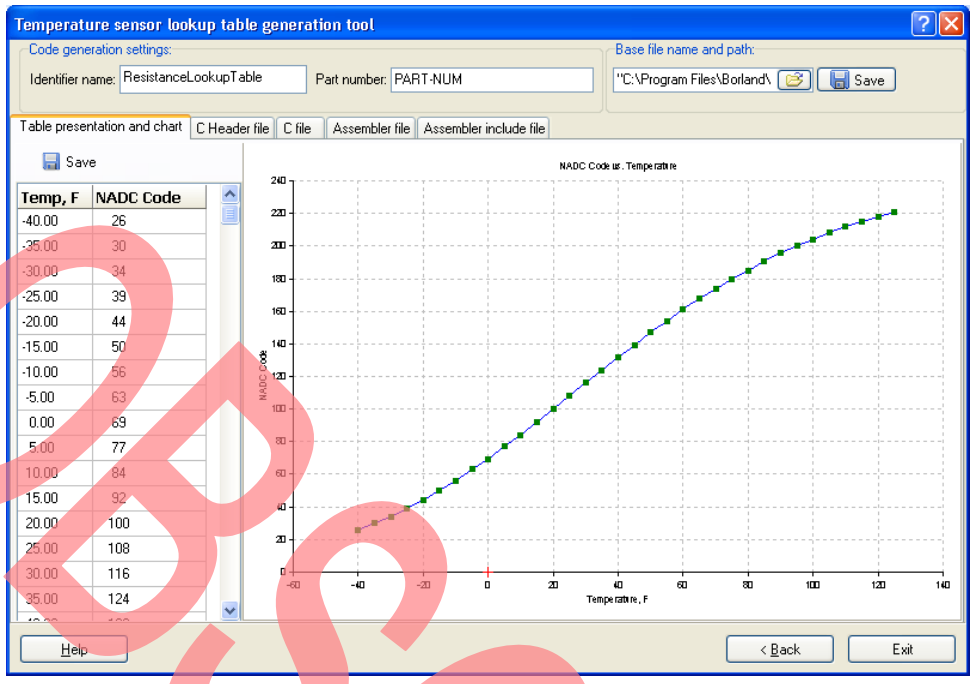
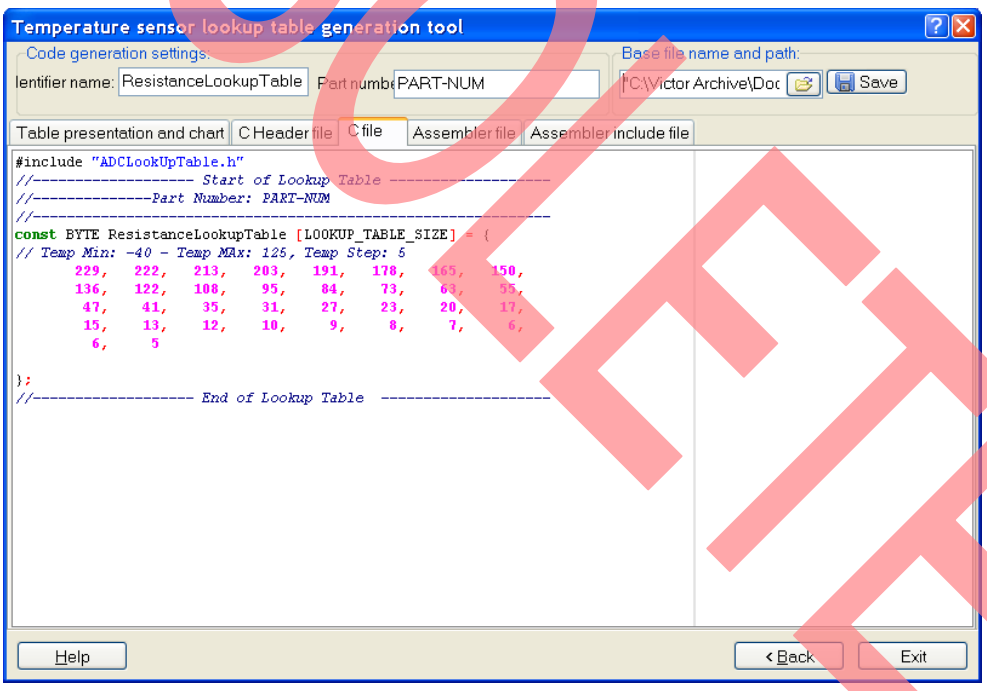


Figure 12. Generated C Code



Document History

Document Title: Thermistor Lookup Table Generation Tool – AN2395

Document Number: 001-41448

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1541809	XSG	10/04/2007	OLD APP. NOTE: Obtained spec. # for note to be added to spec system.
*A	3196511	BIOL	03/15/2011	Template update
*B	4313216	GRAA	03/18/2014	Obsolete document.

In March of 2007, Cypress recataloged all of its Application Notes using a new documentation number and revision code. This new documentation number and revision code (001-xxxxx, beginning with rev. **), located in the footer of the document, will be used in all subsequent revisions.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com/