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THIS SPEC IS OBSOLETE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: 001-26119

Spec Title: SENSING - THERMISTOR-BASED TEMPERATURE MEASUREMENT IN BATTERY PACKS - AN2314

Sunset Owner: Sachin Gupta (SGUP)

Replaced by: 001-40882

Sensing - Thermistor-Based Temperature Measurement in Battery Packs

AN2314

Author: Oleksandr Karpin
Associated Project: Yes
Associated Part Family: CY8C24xxxA, CY8C24794, CY8C27xxx, CY8C29xxx
Software Version: PSoC® Designer™ 5.1 SP1.1
Associated Application Notes: AN2017, AN2260, AN2267

OBSOLETE

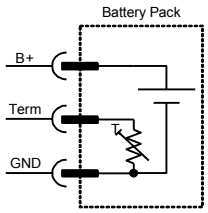
Application Note Abstract

The battery packs are widely used in notebooks, PDAs, mobile phones, and other consumer and industrials applications. This Application Note describes an accurate thermistor-based temperature measuring technique for battery packs. The proposed technique is implemented with minimal PSoC® device resources and external components.

Introduction

A battery pack consists of one or more batteries connected together and a thermistor that controls battery pack temperature during use. Figure 1 shows the internal schematic of an example battery pack.

Figure 1. Battery Pack Internal Schematic



The measurement of accurate battery temperature is very important during battery charge. For example, in the Li-Ion and Li-Pol batteries, the charge is allowed only when the temperature falls within the predefined range.

For Ni-Cd and Ni-MH batteries, the battery temperature is used to switch between rapid (large charge current) and trickle (small charge current) charges, to detect the completion of the charge cycle by checking the temperature slope. Therefore, it is important to measure battery temperature accurately.

Temperature Measuring Technique

Using a thermistor to measure temperature is described in Application Note AN2017, *A Thermistor-Based Thermometer, PSoC Style*.

A thermistor has a non-linear transfer function. The Steinhart-Hart equation describes the resistance change of a semiconductor thermistor that is related to its temperature. Equation 1 shows it to be a 3rd-order logarithmic polynomial using three constants (A, B, and C):

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

- A, B, and C are empirical constants.
- R is the thermistor's resistance.
- T_k is the temperature in Kelvin.

The next equation shows the temperature in Celsius:

$$T_C = \frac{1}{A + B \times \ln(R) + C \times (\ln(R))^3} - 273.15 \qquad \text{Equation 2}$$

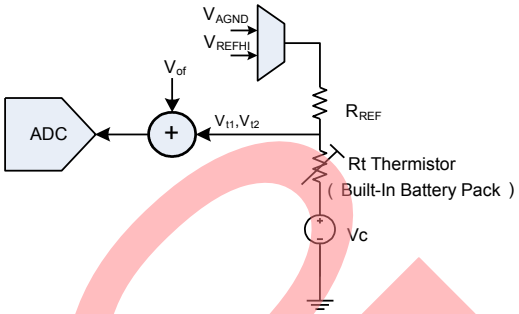
Equation 2 shows that for temperature calculations, knowing the thermistor resistance and its function approximation coefficients is required. These coefficients are often given in the thermistor datasheet or can be found from a resistance table by using any curve-fitting technique.

Several methods can be used to measure the resistance. For example, a simple resistive divider powered by constant DC source can be used. Thermistor resistance can be calculated from the voltage drop on the thermistor. But this method has disadvantages, such as the bottom lead of the thermistor (Figure 1) can be biased to some voltage caused by the voltage drop on the battery pack ground connector. This drop is proportional to the connector pins' resistance and battery current, and varies during battery pack usage. Note that some systems have a current sense resistor in the negative battery lead that additionally increases the voltage drop in the negative battery lead path. This condition decreases the accuracy of temperature measurement, especially at high temperatures when the thermistor resistances are low.

The drawback-free method is proposed to measure the temperature. The idea lies in applying the two different voltages to a resistive divider and then calculating the thermistor resistance based on the difference between

voltages at the upper thermistor lead. Figure 2 shows the proposed thermistor resistance measurement scheme.

Figure 2. Thermistor Resistance Measurement



V_c is the cumulative offset voltage caused by the voltage drop on battery pack ground lead resistance and the voltage drop on PCB tracks. V_{of} is the ADC offset voltage.

The measurement cycle consists of two stages. During the first stage, the V_{REFHI} reference voltage is applied to the resistive divider and ADC code n_{t1} is collected. During the second stage, the V_{AGND} reference voltage is applied and ADC code n_{t2} is collected. The thermistor resistance is calculated by subtracting the two ADC code values, n_{t1} and n_{t2}. The following equations show the resistance-measuring scheme:

$$V_{t1} = V_C + V_{of} + (V_{AGND} - V_C)\frac{R_t}{R_{REF} + R_t} \quad \text{Equation 3}$$

$$V_{t2} = V_C + V_{of} + (V_{REFHI} - V_C)\frac{R_t}{R_{REF} + R_t} \quad \text{Equation 4}$$

$$V_{t2} - V_{t1} = (V_{REFHI} - V_{AGND}) \times \frac{R_t}{R_{REF} + R_t} \quad \text{Equation 5}$$

$$V_{REFHI} = V_{AGND} + V_{AGND} \quad \text{Equation 6}$$

$$V_{t2} - V_{t1} = V_{AGND} \times \frac{R_t}{R_{REF} + R_t} \quad \text{Equation 7}$$

V_{t1}, the voltage level on the thermistor, is set to V_{AGND} (1.3-V) reference voltage. V_{t2} is the voltage level on the thermistor during application of the V_{REFHI} (2 V_{AGND} 2.6-V) reference voltage. V_C is the cumulative offset voltage on the bottom lead of the thermistor. V_{of} is the ADC offset voltage. R_t is the thermistor resistance. R_{REF} is the reference resistor value.

Bias voltages are formed by using the PSoC device's *TestMux* to apply V_{REFHI} and V_{AGND} levels. Taking into account that full-scale ADC code n_{max} corresponds to the 1.3 V input voltage (V_{REFHI}-V_{AGND}, V_{REFLOW} = 0, all voltages considered relative to PSoC's V_{SS} ground), then Equation 7 can be simplified as:

$$n_{t2} - n_{t1} = n_{max} \times \frac{R_t}{R_{REF} + R_t} \quad \text{Equation 8}$$

n_{max} = 2047 for a 12-bit ADC.

Because the measurement is the difference between the ADC code values, the ADC offset is automatically compensated.

Calculating the temperature directly from Equation 2 requires complex, float-point calculations that are not well suited to an 8-bit microcontroller. However, it can be done using integer arithmetic by applying a lookup table. The allowed temperature range for battery usage is only -20...60 °C, which reduces table size. The table is an array of calculated ADC code values from Equation 8 for a linear set of temperatures. The search algorithm looks at the table value that is nearest to the measured ADC code difference. The table index reflects a linear approximation to battery temperature between the measured points. If higher resolution is required, additional interpolation between adjacent table values can be applied to limit the size of the lookup table, within reason.

The overall accuracy of the temperature measurement technique is limited by resistor reference tolerances and ADC gain errors. It is easy to reach a 1-1.5 °C temperature measurement error within the battery operation temperature range with no calibration. For further details regarding the temperature measurement accuracy, see the Appendix: Temperature Measurement Error Analysis.

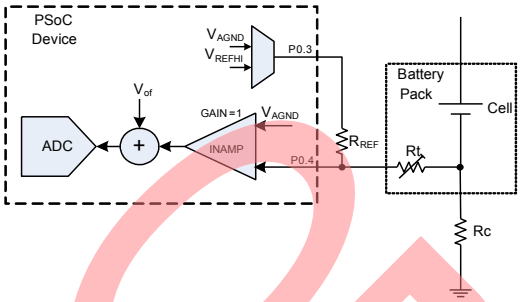
Schematic Examples

The variety of operational characteristic measurements (different cell voltages and charge/discharge currents for each type of battery) requires several connections to the analog capability on the PSoC's Port 0 inputs. Two methods of implementation are proposed. The first method uses two Port 0 pins and the second method uses one Port 0 pin and one Port 2 pin with analog input capability. The advantage of the first method is that the same PSoC internal structure for temperature measurement is used for cell voltage, charge/discharge current measurement. The advantage of the second method is the usage of fewer pins on Port 0. Users can choose among the method that is appropriate, depending on their own project specifics.

OBSOLETE

Figure 3 shows a temperature measurement circuit using the first proposed method of thermistor-to-PSoC connection.

Figure 3. Thermistor Resistance Measurement



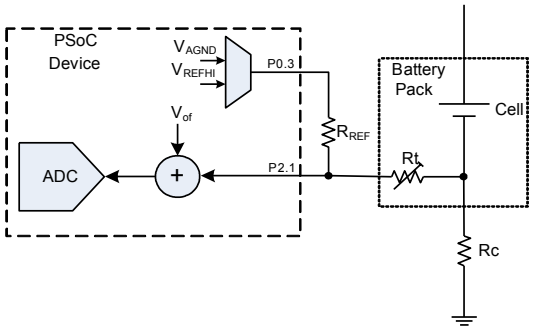
Similar to the Equations 3–8, the ADC code values for the schematic in Figure 3 must be calculated by Equation 9.

$$n_{t2} - n_{t1} = GAIN_{INA} \times n_{\max} \times \frac{R_t}{R_{REF} + R_t} \qquad \text{Equation 9}$$

$GAIN_{INA}$ is the instrumentation amplifier gain (equal to 1). Note that the unity gain PGA can be used instead of the INA. But the INA is useful for other functions, such as the battery voltage and current measurements.

Figure 4 shows a temperature measurement circuit using the second proposed method of thermistor-to-PSoC connection. It uses only one analog pin on Port 0 and one pin on Port 2.

Figure 4. Temperature Measuring using One Pin on Port 0 and One Pin on Port P 2



Note that this method provides a less accurate resistance measurement when working with high-impedance thermistors, especially at low temperatures. The primary error source is the ADC input impedance because the switched capacitor ADC input is connected directly to the resistive divider. The ADC input impedance is sum of the internal routing resistance (approximately 12 kΩ) and the switched capacitor module impedance:

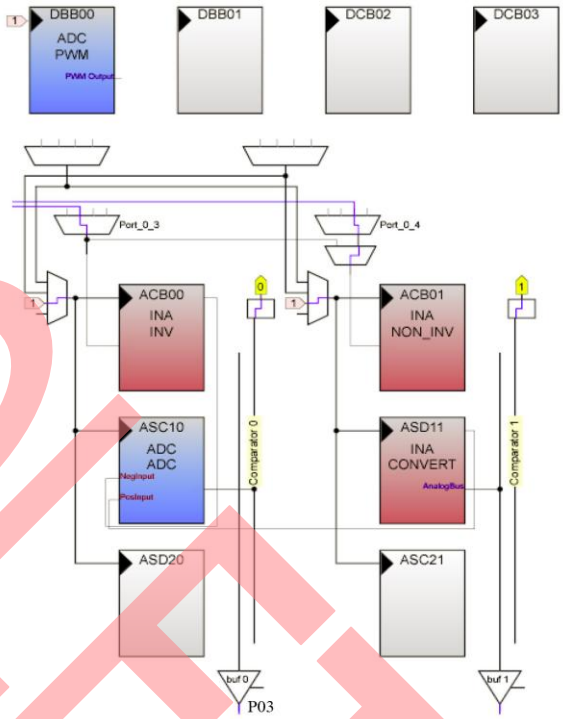
$$Z_{ADC} \approx \frac{5}{2}\frac{1}{F_{clk}} \qquad \text{Equation 10}$$

Z_{ADC} is the ADC input stage impedance. F_{clk} is the ADC sample clock frequency in Hz (1/4 of the column clock frequency).

Sample Project

The project based on the circuit in Figure 3 is described in this section. Figure 5 shows the PSoC device's internal structure.

Figure 5. PSoC Internal User Module Configuration



In the project, a three-opamp topology of an instrumental amplifier (INA) is used. This INA can be used in the battery charger or for other tasks, such as battery current/voltage handling. The INA is placed on blocks ACB00, ACB01, and ASD11. The incremental ADC is placed on ASC10 and DBB00. The ADC resolution is set to 12 bits. The resolution and the sample rate can be increased if the application requires a higher temperature measurement resolution and faster conversion. In this project, the analog ground bias is set to BandGap reference value, or 1.3 V (*RefMux* is set *BandGap* ± *BandGap*). However, a different set of reference voltages can be used if they are more suitable to the balance of the design.

The control software is written in 'C' and shown in example Code 1.

OBSOLETE

Code 1. Temperature Measuring using Two Pins on Port 0

```
#include <m8c.h>
#include "PSoCAPI.h"
#include "globdefs.h"

INT iTb;//voltage on thermistor (in ADC counts)
CHAR cTemperature =0; // temp (in Celsius)

//temperature lookup table from -20C to 60C
//with step 1C by thermistor temperature-
//resistance relations measured in ADC counts
const WORD anwLookTable[TEMP_DATA_COUNT] ={
ADC_R20C_MINUS, ADC_R19C_MINUS,
 … , … ,…
ADC_R1C_MINUS, ADC_R0C, ADC_R1C, … ADC_R60C};

INT iGetADCData(void) {
   INT iData = 0;

   ADC_GetSamples(1);
   while (!ADC_fIsDataAvailable());
   iData = ADC_iClearFlagGetData();
   return iData;
}

// Convert temp ADC code to temp value
CHAR cADCtoTemp(INT iTb){
   CHAR cIndex, cTemperature;
   INT iForward, iBack;

   // find position (cIndex) in temperature
   // lookup table
   // where the value is greater then measured
   for (cIndex = 0; cIndex < TEMP_DATA_COUNT-1;
        cIndex++) {
      if (iTb > anwLookTable[cIndex]) {
         break ;
      }
   }

   // establish what value in temperature
   // lookup table is near
   // to measured  (back or forward)
   if (0!=cIndex) {
      iForward = anwLookTable[cIndex]-iTb;
      iBack = iTb - anwLookTable[cIndex-1];
      if (iBack < iForward) cIndex--;
   }

   // as the index in lookup table shows
   // the real temperature value with
   // shift in maximum negative temperature
   // value (if temperature step is 1 degree)
   // then the real temperature value is:
   cTemperature = cIndex + TEMP_MIN_VALUE;
   return cTemperature;
}

void main() {
   INA_Start(INA_HIGHPOWER);  // init INA
   ADC_Start(ADC_HIGHPOWER);  // init ADC
   // configure INA inputs to measure temp
   AMX_IN = MUX_THERMISTOR_SETTING;
   INA_Set2StageGain(INA_INGAIN_1,
       INA_OUTGAIN_1_00);

   M8C_EnableGInt;

   ACB00CR1 |=0x03; //Set INA-in to AGND (PMUX)
   //ACB00CR1 &=0xFD; //Ret INA-in to Port-in

   while (1) {
      // measure battery temp in ADC counts
      ACB00CR2 |= 0x1C; // Set P0[3] = RefHI
      iTb =  iGetADCData();
      ACB00CR2 &= 0xF7; // Set P0[3] = AGND
      iTb -=  iGetADCData();

      // real temperature value calculation
      // in battery charger is recommended
      // to use only for Fuel Gauge function
      cTemperature = cADCtoTemp(iTb);
   }
}
```

During initial configuration, one of the INA analog inputs is configured directly to V_{AGND} on the ACB00 continuous time block via PMux. Next, the routine sets the V_{REFHI}, retrieves the ADC code value for this bias, sets the V_{AGND} level, obtains the ADC code value, and calculates the difference. The algorithm searches the lookup table to find the nearest table index. The temperature is now proportional to the table index. If Celsius is the required temperature value, the index is scaled during the table temperature step and biased to the start of the lookup table.

All global resources for this project are located in the header file *globdefs.h*, that is in the project folder. An example for the thermistor temperature-resistance relationship at 25 °C and the ADC code calculation is shown in Code 2. Equation 9 describes the calculation method for this example.

Code 2. ADC Code of Thermistor Temperature-Resistance Relationship at 25 °C Calculation

```
#define TEMPERATURE_GAIN 1.0
#define TEMPERATURE_R_REF 10000
#define ADC_MAX 2047

//Thermistor Temperature-Resistance Relations
#define R25C 10000 //Ohms

//Thermistor Parameters in ADC counts
#define ADC_R25C
(INT)((ADC_MAX*R25C/
(TEMPERATURE_R_REF+R25C))*TEMPERATURE_GAIN)
```

An additional project based on Figure 4 is similar to this project. All these projects are available on the Cypress website at http://www.cypress.com/.

Project Adaptation to Selected Thermistor

To set the project up to accommodate the selected thermistor, it is only necessary to modify the temperature-resistance relations in the header file *globdefs.h*. All other tasks are performed by the pre-designed macros (see Code 2). The thermistor temperature-resistance relationship can be taken directly from the thermistor datasheet or calculated by hand. For example, for this application, a BetaTHERM's 10K3A1A thermistor is used. It is a precision thermistor with the following parameters:

- 10,000 Ω at 25 °C
- Tolerance = ±0.1 °C from 0 °C to 70 °C
- -80 °C to 150 °C operating range

OBSOLETE

Table 1 gives the thermistor temperature-resistance relationship taken directly from the thermistor datasheet.

Table 1. 10K3A1A Thermistor Temperature-Resistance Relationship (from -20 °C to 60 °C)

Temp (°C)	R (Ω)	Temp (°C)	R (Ω)
-20	96974	51	3466.9
-19	91525	52	3338.6
-18	86415	53	3215.6
-17	81621	54	3097.9
-16	77121	55	2985.1
-15	72895	56	2876.9
-14	68927	57	2773.2
-13	65198	58	2673.9
-12	61693	59	2578.5
-11	58397	60	2487.1

Note: Table Example (Not all Values Shown)

In the sample projects associated to this Application Note these exact temperature-resistance values are used. If the thermistor datasheet is not available, the user can measure the resistance of at least three different temperature points with values distributed in the expected operation range for better temperature-resistance curve interpolation. For example, for our thermistor we used the following points shown in Table 2.

Table 2. Three Data Points for the 10K3A1A Thermistor

Temperature (°C)	Resistance (Ω)
-20	96974
25	10000
60	2487.1

The three data points in Table 2 are used to calculate the Steinhart-Hart coefficients A, B, and C in Equation 2. The results are shown in Table 3. If more than three data points are known or the user wants to increase the resolution of the known thermistor-resistance relationship, then the Steinhart-Hart coefficients A, B, and C should be calculated by using the least squares fitting method.

Table 3. Steinhart-Hart Coefficients for 10K3A1A

Steinhart-Hart Coefficient	Value
A	0.001129676798
B	0.0002340323705
C	8.808445665*10-8

Using these coefficients and Equation 2, the thermistor temperature-resistance relationship can be easily calculated for all necessary temperature points.

Summary

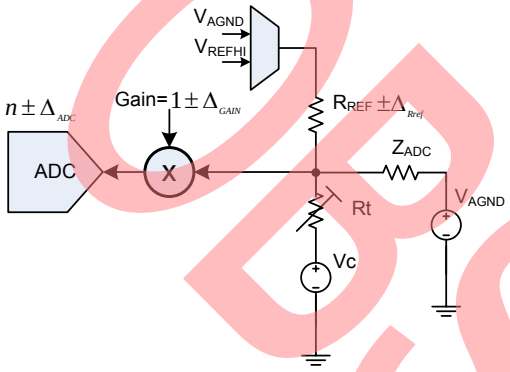
The thermistor-based temperature measurement technique for the battery packs is discussed in this Application Note. The proposed technique and its modifications have been used in the Application Notes AN2260, *Rapid NiCd/NiMH Battery Charger and DC Brushed Motor Controller for Autonomous Appliances*, and AN2267, *Single Cell Li-Ion Battery Charger* along with others.

OBSOLETE

Appendix: Temperature Measurement Error Analysis

The general circuit for temperature measurement with possible error sources for the two proposed methods of thermistor connection (Figure 3 and Figure 4, respectively) is shown in Figure 6.

Figure 6. General Circuit for Temperature Measurement with Possible Error Sources



- Δ_{ADC} is the ADC cumulative error (ADC noise, nonlinear and others). It is closest to the LSB.
- Δ_{Rref} is the absolute error of R_{REF} tolerance.
- Δ_{GAIN} is the absolute error of the INA/ADC gain error.
- Z_{ADC} is the ADC input stage impedance (using the first method, it tends toward infinity, using the second method, it can be evaluated as shown in Equation 10.

Equation 2 shows that the temperature is a function of thermistor resistance R_t in Equation 11.

$$T = F(R_t) \qquad \text{Equation 11}$$

From Equation 9, the R_t value can be represented as:

$$R_t = \frac{R_{REF}}{\frac{GAIN_{INA} \times n_{\max}}{n} - 1} \qquad \text{Equation 12}$$

n is the resultant ADC code difference ($n = n_{t2} - n_{t1}$).

Taking into account Figure 6 and Equation 12, it follows that the measured thermistor resistance value determined by this system is dependent on several factors with possible error sources as shown in Equation 13:

$$R = R(a_1, a_2, a_3, ..., a_k) \qquad \text{Equation 13}$$

- a_1 is the ADC value n with cumulative error Δ_{ADC}.
- a_2 is the R_{REF} with tolerance Δ_{Rref}.
- a_3 is the INA/ADC gain with error Δ_{GAIN}.
- a_4 is the ADC value with the difference between the calculated n value for the specific temperature in the firmware (see Code 2) and the measurement through the Z_{ADC} influence error.
- a_5 is the thermistor resistance error.
- $a_6 - a_k$ are other, but less significant, errors.

The Δ_{ADC}, Δ_{Rref}, Δ_{INA} are well known. The absolute error between the calculated n value and measurement through the Z_{ADC} influence can be evaluated using Equation 14:

$$\Delta_n = \left| n_{Zadc} - n_{calc} \right| \qquad \text{Equation 14}$$

n_{calc} is the calculated n value for the specific temperature in the firmware (see Code 2). n_{Zadc} is the measured n value with the Z_{ADC} influence.

Taking into consideration the Z_{ADC} influence (see Figure 6 schematic), the value of n_{Zadc} is found:

$$n_{Zadc} = \frac{GAIN_{INA} \times n_{\max}}{R_{REF} \times \left(\frac{1}{R_t} + \frac{1}{Z_{ADC}} + \frac{1}{R_{ref}} \right)} \qquad \text{Equation 15}$$

Next, the summary absolute error of the temperature calculation by the proposed method can be evaluated with a sensitivity analysis with respect to each of the parameters. As the error sources are not correlated, an RMS evaluation (Equation 16) yields a relatively accurate result:

$$\Delta T = \left| \frac{\partial T}{\partial R} \right| \times \left[\sum_{i=1}^{k} \left(\frac{\partial R}{\partial a_i} \times \Delta_i \right)^2 \right]^{\frac{1}{2}} \qquad \text{Equation 16}$$

Δ_i is the absolute error of parameter a_i.

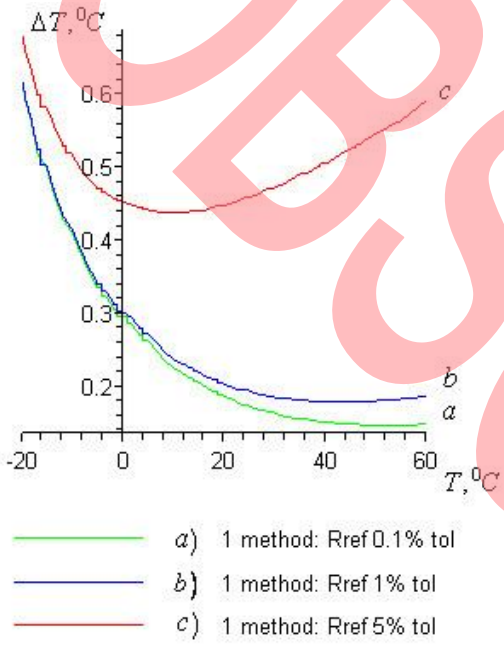
$$\Delta_i = a_i^{\max} \times \varepsilon_i \qquad \text{Equation 17}$$

OBSOLETE

a_i^{max} is the maximum value of parameter a_i. ε_i is the relative error of a_i.

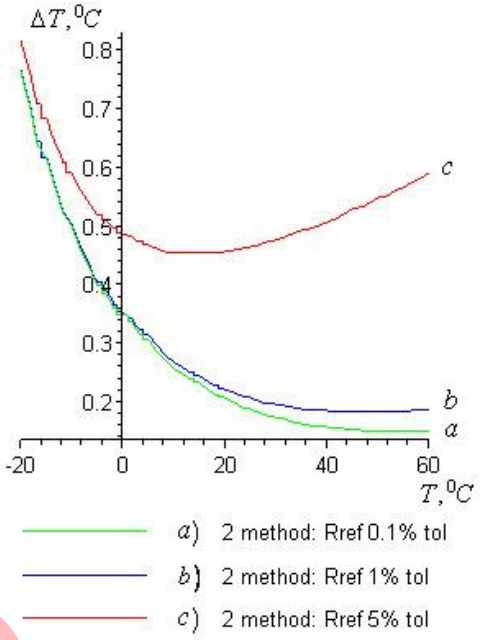
Figure 7 shows the absolute error for the temperature measurement using the first proposed method of thermistor connection (Figure 3) and Figure 8 shows the absolute error for the temperature measurement using the second proposed method of thermistor connection (Figure 4). The calculation technique for the two methods is identical except that using the first method, the Z_{ADC} value tends toward infinity and using the second method, it is evaluated as in Equation 10.

Figure 7. Temperature Measurement Absolute Error Comparison Relative to Rref Tolerance ε_{Rref}



(Figure 3 Schematic: Δ_{ADC} =1 LSB, ε_{GAIN} = 1%, $Z_{ADC} \to \infty$)

Figure 8. Temperature Measurement Absolute Error Comparison Relative to Rref Tolerance ε_{Rref}



(Figure 4 Schematic: Δ_{ADC} =1 LSB, ε_{GAIN} = 1%, $Z_{ADC} = 1.2MOhm$)

Temperature Measurement Error Analysis Conclusion

The research shows that the absolute error of the proposed temperature measurement scheme is lower than 1 °C across all temperature measurement ranges. This level of error is suitable for all battery pack charger operations that are temperature based. Therefore, the temperature measurement technique proposed in this Application Note can be used in battery pack chargers without calibration of individual PSoC parameters.

About the Author

Name: Oleksandr Karpin
Title: Post-Graduate Student
Background: Oleksandr earned his computer-engineering diploma in 2001 from National University "Lvivska Polytechnika" (Lviv, Ukraine). He is furthering his studies at this university. His interests include embedded systems design and new technologies.
Contact: kool_ukr@cypress.com

OBSOLETE

Document History

Document Title: Sensing - Thermistor-Based Temperature Measurement in Battery Packs – AN2314

Document Number: 001-26119

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1448764	KOOL_UKR	09/07/2007	New spec.
*A	3338109	ANBI_UKR	08/09/2011	Updated project to support the latest version PSoC designer
*B	4051197	SGUP	07/05/2013	Obsolete application note

OBSOLETE

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.

PSoC is a registered trademark of Cypress Semiconductor Corp. "Programmable System-on-Chip," PSoC Designer, and PSoC Express are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

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

© Cypress Semiconductor Corporation, 2007-2013. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.