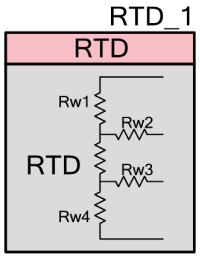
RTD Calculator Component
1.10



Features

- Calculation accuracy 0.01 °C for -200 °C to 850 °C temperature range
- Provides simple API function for resistance to temperature conversion
- Displays Error Vs Temperature graph

General Description

The Resistance Temperature Detector (RTD) Calculator component generates a polynomial approximation for calculating the RTD Temperature in terms of RTD resistance for a PT100, PT500 or PT1000 RTD. Calculation error budget is user-selectable, and determines the order of the polynomial that will be used for the calculation (from 1 to 5). A lower calculation error budget will result in a more computation intensive calculation. For example, a fifth order polynomial will give a more accurate temperature calculation than lower order polynomials, but will take more time for execution. After maximum and minimum temperatures and error budget are selected, the component generates the maximum temperature error, and an error vs. temperature graph for all temperatures in the range, along with an estimate of the number of CPU cycles necessary for calculation using the selected polynomial. Selecting the lowest error budget will choose the highest degree polynomial. For the whole RTD temperature range, -200 °C to 850 °C, the component can provide a maximum error of <0.01 °C using a fifth order polynomial.

When to use a RTD

The component has only one use case. The component provided API is used to calculate the temperature from RTD resistance.

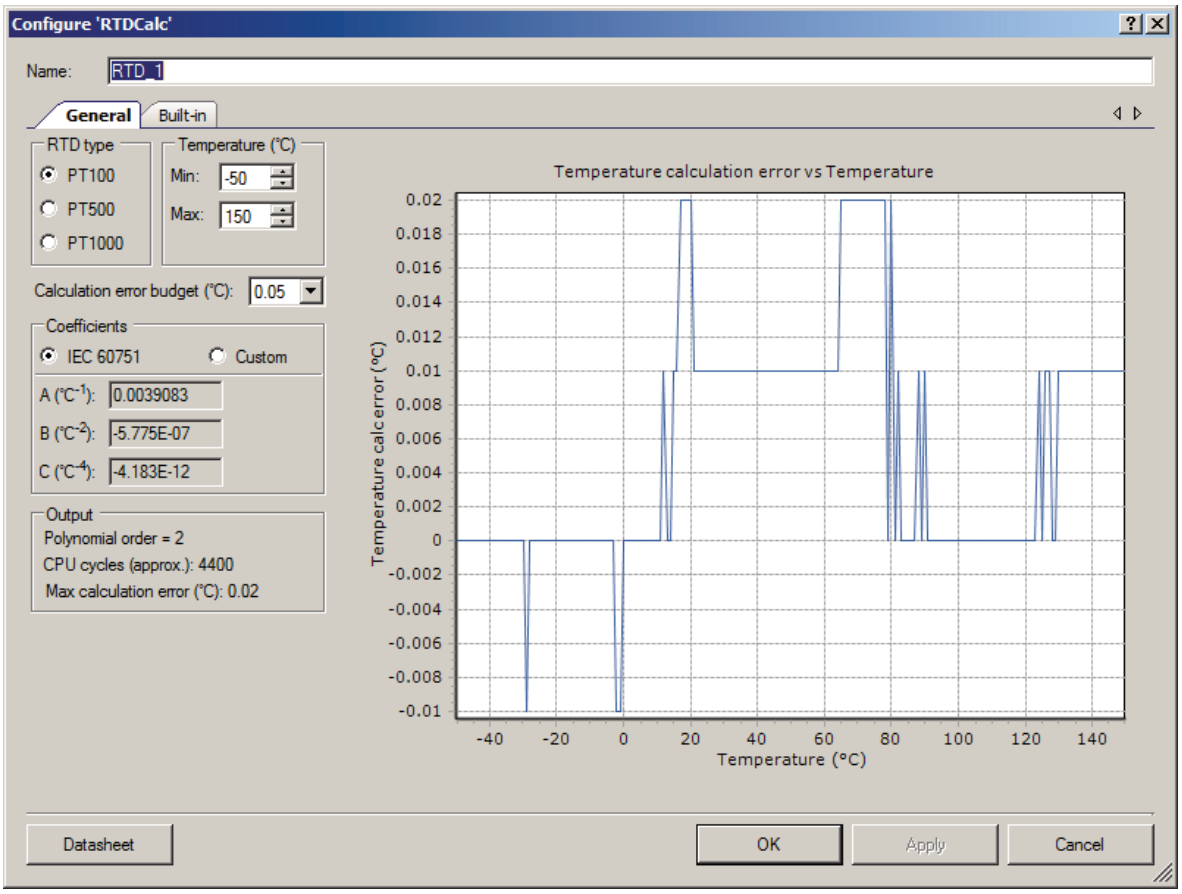
Input/Output Connections

This component is a software component and doesn't have any input/output connections.

Parameters and Settings

Drag an RTD Calculator component onto your design and double-click it to open the Configure dialog. This dialog has one tab to guide you through the process of setting up the RTD Calculator component.

General Tab



The **General** tab provides the following parameters.

RTD Type

Selection of the RTD Type - PT100, PT500 or PT1000. Default is PT100.

Temperature Range

The user selects the minimum and maximum temperature that the RTD is expected to measure. Temperature values should be in the range [-200, 850], both values inclusive.

Calculation Error Budget

The maximum error due to the polynomial approximation in the whole temperature range is given here. Options: 0.01, 0.05, 0.1, 0.5, 1. Default is 0.05.

Coefficients

When the "IEC 60751" radio button is selected, A, B and C coefficients are automatically filled with the Callendar–Van Dusen Coefficients from IEC 60751. If the "Custom" radio button is selected, A, B and C coefficients may be entered manually. The coefficients default to the standard values, but may be edited under the following restrictions:

A: Real value in the range (3.5E-3, 4.1E-3) with maximum 6 significant digits.

B: Real value in the range (-6.2E-3, 5.5E-3) with maximum 6 significant digits.

C: Real value in the range (-7E-12, -3E-12) with maximum 6 significant digits.

Temperature Error Vs Temperature graph

This graph shows the temperature error vs. temperature for the selected temperature range and accuracy. Whenever the temperature range or error budget is changed, this graph is refreshed.

Temperature error is computed by first forming a resistance vs. temperature table using the Callendar–Van Dusen equations directly, and then computing the temperature for the resistance values of the table using the desired polynomial approximation.

Polynomial order

The polynomial order is chosen such that the maximum error due to the polynomial is less than the selected Calculation error budget. This is achieved by evaluating maximum error for polynomials of degree 1 to 5 and selecting the lowest degree that satisfies the error budget requirements.

Max calc error

The maximum error due to the polynomial approximation across the whole temperature range.

No. of CPU cycles

An estimate of the total number of CPU cycles taken for computing the selected polynomial.

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name "RTD_1" to the first instance of a component in a given design. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is "RTD".

Function	Description
int32 RTD_GetTemperature(uint32 res)	Calculates the temperature from RTD resistance

int32 RTD_GetTemperature(uint32 res)

Description: Calculates the temperature from RTD resistance.

Parameters: res: Resistance in mΩ.

Return Value: Temperature in 1/100ths degrees C.

Side Effects: None

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the component. There are two types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator components
- specific deviations – deviations that are applicable only for this component

This section provides information on component-specific deviations. Project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The RTD Calculator component has not been verified for MISRA-C:2004 coding guidelines compliance.

Sample Firmware Source Code

PSoC Creator provides many example projects that include schematics and example code in the Find Example Project dialog. For component-specific examples, open the dialog from the Component Catalog or an instance of the component in a schematic. For general examples,

open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

Refer to the “Find Example Project” topic in the PSoC Creator Help for more information.

Functional Description

An RTD is a positive temperature coefficient (PTC - resistance increases as temperature increases) - sensor. The resistance-temperature relationship is not perfectly linear. Various standards approximate this non-linearity. Of them, IEC 60751 is one of the most widely used standards. The RTD resistance to temperature relationship is specified by the Callendar–Van Dusen equations. Equations 1 and 2 define the resistance to temperature relationship in IEC 60751. Above 0 °C, RTD temperature is specified by the RTD resistance at 0 °C (R0) and constants A and B.

For T>0,

$$R_T = R_0(1 + AT + BT^2) \qquad \text{Equation 1}$$

Where, R_T is the resistance at T °C.

Below 0 °C, in addition to the A and B, a third constant (C) is also involved, as shown in Equation 2:

For T<0,

$$R_T = R_0(1 + AT + BT^2 + C(T - 100)T^3 \qquad \text{Equation 2}$$

The values of A, B, and C for PT100 RTD are specified in IEC 60751 for standard industry-grade platinum and are:

$$A = 3.9083 * 10^{-3}\ °C^{-1}$$

$$B = -5.775 * 10^{-7}\ °C^{-2}$$

$$C = -4.183 * 10^{-12}\ °C^{-4}$$

These equations give resistance in terms of temperature. To obtain temperature in terms of resistance, the component customizer calculates a polynomial that best fits the set of resistance-temperature points computed using the above equations.

The polynomial coefficients are obtained by the least square fit method.

Resources

Component is implemented entirely in firmware. It does not consume any other PSoC resources.

API Memory Usage

The component memory usage varies significantly, depending on the compiler, device, number of APIs used and component configuration. The following table provides the memory usage for all APIs available in the given component configuration.

The measurements have been done with the associated compiler configured in Release mode with optimization set for Size. For a specific design the map file generated by the compiler can be analyzed to determine the memory usage.

Configuration	PSoC 3 (Keil_PK51)		PSoC 5 (GCC)	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
RTD [-200;850] with Calc Err 0.01	350	0	192	0
RTD [-200;850] with Calc Err 0.05	338	0	196	0
RTD [-200;850] with Calc Err 0.1	338	0	196	0
RTD [-200;850] with Calc Err 0.5	330	0	164	0
RTD [-200;850] with Calc Err 1	330	0	164	0
RTD [-150;700] with Calc Err 1	322	0	124	0
RTD [-50;100] with Calc Err 1	318	0	100	0

Performance

The performance of the component depends on the implementation method chosen in the customizer. The measurements below have been gathered using a CPU speed of 24 MHz, with the associated compiler configured in Release mode. These numbers should be treated as approximations and used to determine necessary trade-offs.

Polynomial order	No. of CPU cycles (PSoC 3)	No. of CPU cycles (PSoC 5)
1	3250	70
2	4400	110
3	5550	150
4	6700	190
5	7850	230

Component Changes

This section lists the major changes in the component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
1.10	Added MISRA Compliance section	The component was not verified for MISRA compliance.
	Updated Sample Firmware Source Code section	
1.0	Version 1.0 is the first release of the RTD Calculator component.	

© Cypress Semiconductor Corporation, 2012. 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.

PSoC® is a registered trademark, and PSoC® Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any 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 HALFICULAR 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.