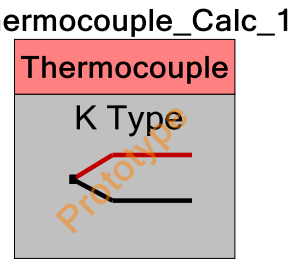
Thermocouple
v 1.0

Features



- Supports B, E, J, K, N, R, S, and T Type Thermocouples
- Provides functions for thermo-emf to temperature and temperature to voltage conversions
- Displays Calculation Error Vs Temperature graph

General Description

In thermocouple temperature measurement, the thermocouple temperature is obtained from thermocouple voltage. The voltage to temperature conversion is characterized by National Institute of Standards and Technology (NIST) and NIST provides tables and polynomial coefficients for thermo-emf to temperature conversion. The NIST tables and polynomial coefficients can be found in the link below.

http://srdata.nist.gov/its90/download/download.html

Thermocouple temperature measurement also involves measuring the thermocouple reference junction temperature and converting it into voltage. The Thermocouple component simplifies thermo-emf to voltage conversion by providing APIs for thermo-emf to temperature conversion and vice versa for all thermocouple types. The thermocouple component uses polynomials for temperature to thermo-emf conversion and vice versa. The thermocouple component implements the polynomial in an efficient way to reduce computation time. The polynomial order used for voltage to temperature conversion and vice versa in different temperature ranges for all thermocouple types are listed in tables 1 and 2.

See Cypress AN75511- Temperature measurement with Thermocouples to understand thermocouple temperature measurement in detail

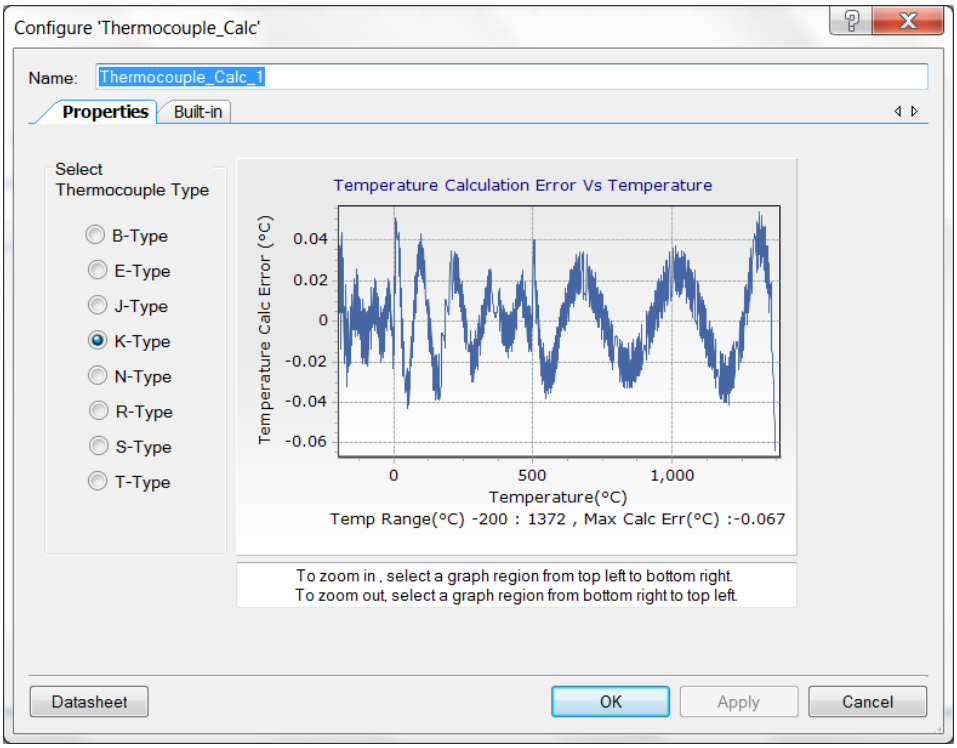
Input/Output Connections

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

Parameters

Drag a themocouple component onto your design and double-click it to open the Configure dialog.

Figure 1 Configure Thermocouple_Calc Dialog



The Thermocouple component provides the following parameter.

Select Thermocouple Type

This is the only option the user has to select. The user can select from one of the eight standard thermocouple types. By default, K-Type thermocouple is selected

Placement

There is no placement specific information for the thermocouple component

Resources

The thermocouple component is a software component and uses only flash and RAM space as shown in the table below.

Analog Blocks	Digital Blocks				API Memory (Bytes)	
	Datapaths	Macro cells	Status Registers	Control Registers	Flash	RAM
None	N/A	N/A	N/A	N/A	1200	16

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 "Thermocouple_Calc_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 " Thermocouple_Calc_1".

Function	Description
float Thermocouple_Calc_1_mVoltToTemp(float voltage)	Converts thermo emf to temperature
float Thermocouple_Calc_1_ColdJunctionTempTomVolt(float temperature)	Converts cold junction temperature to cold junction compensation voltage

float Thermocouple_Calc_1_mVoltToTemp(float voltage)

Description: This function converts thermo emf to temperature using polynomials. Multiple polynomials are used in different temperature (voltage) ranges

Parameters: float voltage, thermo-emf in millivolts

Return value: float, temperature in °C

Number of CPU cycles taken by the Thermocouple_Calc_1_mVoltToTemp() API for different input voltage (temperature) ranges for all thermocouple types. Each voltage (temperature) range uses a different polynomial and hence the computation time is different for different voltage ranges.

Table 1: Polynomial order and number of CPU cycles for different temperature ranges

Thermo-couple Type	Voltage Range (mV)	Temperature range (°C)	Polynomial order	No. of CPU cycles
B	0.033 to 0.291	100 to 250	7	4526
	0.291 to 2.431	250 to 700	8	5616
	2.431 to 13.820	700 to 1820	8	5160
E	-9.718 to -8.825	-250 to -200	9	5366
	-8.825 to 0	-200 to 0	8	5746
	0 to 76.373	0 to 1000	9	6288
J	-8.095 to 0	-210 to 0	8	5170
	0 to 42.919	0 to 760	7	4978
	42.919 to 69.553	760 to 1200	5	4190
K	-5.891 to 0	-200 to 0	8	5649
	0 to 20.644	0 to 500	9	5784
	20.644 to 54.886	500 to 1372	6	4498
N	-4.313 to -3.990	-250 to -200	9	5775
	-3.990 to 0	-200 to 0	9	5580
	0 to 20.613	0 to 600	7	5496
	20.613 to 47.513	600 to 1300	5	4303
R	-0.226 to 1.923	-50 to 250	10	6629
	1.923 to 13.228	250 to 1200	9	5962
	13.228 to 19.739	1200 to 1664.5	5	4644
	19.739 to 21.103	1664.5 to 1768.1	4	3864
S	-0.235 to 1.874	-50 to 250	9	6543
	1.874 to 11.950	250 to 1200	9	6404

	11.950 to 17.536	1200 to 1664.5	5	4080
	17.536 to 18.693	1664.5 to 1768.1	4	3687
T	-6.180 to -5.603	-250 to -200	7	3687
	-5.603 to 0	-200 to 0	7	4013
	0 to 20.872	0 to 400	6	3615

The actual thermocouple coefficients used for millivolt to temperature conversion different thermocouple types in the different voltage ranges can be found in Thermocouple_Calc_1.c file.

The following code snippet shows the polynomial coefficients used for thermo-emf to temperature conversion for a K-Type thermocouple in different temperature ranges. The comments in-line explain how to interpret the array elements.

Start Page | TopDesign.cysch | main.c | Thermoc..._Calc_1.c

```
157
158 /* Polynomial coefficients for converting thermo-emf to temperature.
159  * The first column corresponds to the polynomial coefficients
160  * for voltage range -5.891 to 0 mV(temp range -200 to 0 °C)
161  * The second column corresponds to the polynomial coefficients
162  * for voltage range 0 to 20.644 mV(temp range 0 to 500 °C)
163  * The third column corresponds to the polynomial coefficients
164  * for voltage range 20.644 to 54.886 mV(temp range 500 to 1372 °C)
165  * In each column the top entry is the constant term and the bottom
166  * entry is the highest order coefficient of the polynomial
167  */
168 CYCODE const float coeffV2T[][4] = {{ 0.0000000E+00,  0.000000E+00, -1.318058E+02},
169                                     { 2.5173462E+01,  2.508355E+01,  4.830222E+01},
170                                     {-1.1662878E+00,  7.860106E-02, -1.646031E+00},
171                                     {-1.0833638E+00, -2.503131E-01,  5.464731E-02},
172                                     {-8.9773540E-01,  8.315270E-02, -9.650715E-04},
173                                     {-3.7342377E-01, -1.228034E-02,  8.802193E-06},
174                                     {-8.6632643E-02,  9.804036E-04, -3.110810E-08},
175                                     {-1.0450598E-02, -4.413030E-05,  0.000000E+00},
176                                     {-5.1920577E-04,  1.057734E-06,  0.000000E+00},
177                                     { 0.0000000E+00, -1.052755E-08,  0.000000E+00}};
178
```

float Thermocouple_Calc_1_ColdJunctionTempTomVolt(float temperature)

Description: This function converts cold junction temperature to cold junction compensation voltage using a polynomial.

Parameters: float temperature, cold junction temperature in °C

Return value: float, cold junction compensation voltage in millivolts (float)

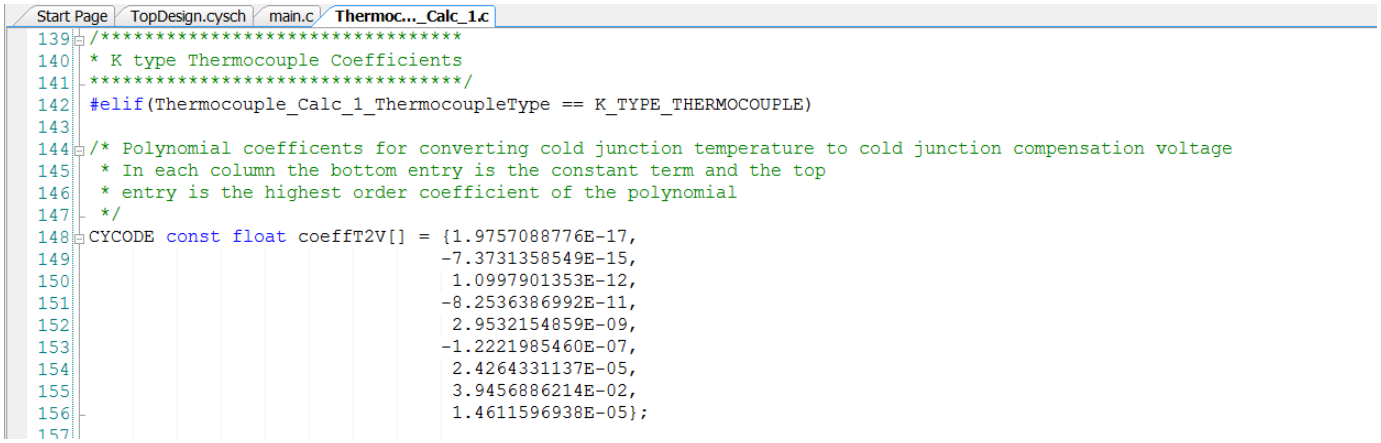
Number of CPU cycles taken by Thermocouple_Calc_1_ColdJunctionTempTomVolt() API for different thermocouple types is given in table 2. This API uses a single polynomial in -20°C to 100°C (0-100°C for B-Type) range for each thermocouple type. Ensure that the cold junction temperature is in -20°C to 100°C range.

Table 2: Polynomial order and CPU cycle for different temperature ranges

Thermocouple Type	Temperature range (°C)	Polynomial order	No. of CPU cycles
B	0 to 100	6	3550
E	-20 to 100	8	4133
J	-20 to 100	7	3746
K	-20 to 100	8	4128
N	-20 to 100	8	3922
R	-20 to 100	7	3643
S	-20 to 100	7	3643
T	-20 to 100	8	3830

The actual thermocouple coefficients used for different thermocouple types can be found in Thermocouple_Calc_1.c file

The following code snippet shows the polynomial coefficients used for temperature to millivolt conversion for a K-Type thermocouple. The comments in-line explain how to interpret the array elements.



Sample Firmware Source Code

The following is a C language example demonstrating the basic functionality of the Thermocouple component. This example assumes the component has been placed in a design with the default name "Thermocouple_Calc_1"

Note If you rename your component you must also edit the example code as appropriate to match the component name you specify.

```
void main()
{

      /* Insert code to obtain cold junction temperature */
          tcColdJnmVolt = Thermocouple_Calc_1_ColdJunctionTempTomVolt(coldJnTemp);
      /* Insert code to
       *    1. Obtain hot junction voltage.
       *    2. Obtain thermo emf from hot junction voltage and cold junction
             compensation voltage */
          tcTemp = Thermocouple_Calc_1_mVoltToTemp(tcmVolt);
}
```

Reference Component Changes

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

Version	Description of Changes
1.0	New component and datasheet

© 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 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.