Cambridge CMOS Sensors

is now

Member of the ams Group

The technical content of this Cambridge CMOS Sensors (CCS) document is still valid.

Contact information:

Headquarters:
ams AG
Tobelbader Strasse 30
8141 Premstaetten, Austria
Tel: +43 (0) 3136 500 0
e-Mail: ams_sales@ams.com

Please visit our website at www.ams.com

NTC Thermistor Interface on CCS811

Introduction

The CCS811 digital gas sensor supports an external interface for connecting a negative thermal coefficient (NTC) thermistor. This can be integrated with minimal external discrete components, and provides a cost effective and power efficient means of calculating the local ambient temperature.

The thermistor enables the application to determine the ambient temperature in the location where the CCS811 is deployed. This temperature can subsequently be used for environmental compensation on the CCS811 MOX gas sensor.

This application note details the recommended hardware and software interface for connecting and controlling a NTC thermistor on the CCS811.

Hardware Configuration

The CCS811 signals in bold text in Table 1 are those that are required for NTC thermistor operation.

Pin	Function	Pin	Function
1	I^2C ADDR	**6**	**VDD**
2	nRESET	7	nWAKE
3	nINT	**8**	**AUX**
4	**PWM**	9	I^2C SDA
5	**Sense**	10	I^2C SCL
EP	GND		

Table 1 CCS811 Pins For NTC Thermistor

These signals are used in conjunction with a small number of external components (one resistor and one thermistor) to provide data that the host processor can use to calculate temperature. This temperature can then be used for environment compensation or for any other means required by the application.

Key Benefits

- Simple circuity for determining temperature
- Cost effective and minimal PCB footprint
- Integrated MCU with ADC
- I2C digital interface
- Optimised low-power modes
- Compact 2.7x4.0 mm LGA package
- Proven technology platform
- On-board processing to reduce requirement on host processor
- Fast time-to-market
- Extended battery life
- Reduced component count
- Suitable for small form factor designs
- Highly reliable solution

Applications

- IAQ monitoring for Smarthome, Smartphones and accessories

Voltage Divider and ADC

To enable the NTC circuit a voltage divider circuit is constructed between the VDD, AUX and Sense pins. Refer to Figure 1 below.

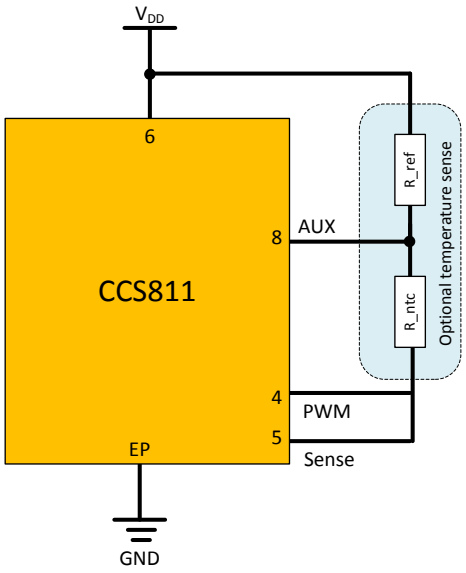

Figure 1 CCS811 NTC Voltage Divider Circuit

The CCS811 has an internal ADC that can measure the voltages at VDD, AUX and Sense, enabling the CCS811 to calculate the voltages across the Reference Resistor (R_{REF}) and the NTC Resistor (R_{NTC}).

For optimal usage of ADC resolution, the value of RREF and the nominal value of RNTC should be approximately the same; the suggested value is 100kΩ.

As temperature increases, RNTC decreases. This causes the voltage value sampled on the AUX signal to decrease. The opposite is true when temperature decreases, RNTC increases.

Hardware Connection When NTC Is Not Required

If the system where the CCS811 is deployed uses another means of obtaining temperature, such as combined temperature and humidity sensor, or if temperature is not measured in the system, then there is no requirement to connect the AUX and Sense pins to the thermistor. There is also no requirement to connect the reference resistor between AUX and VDD. Pins 4 and 5 must still be connected together for normal operation.

Equations

Determining Thermistor Resistance

The CCS811 retrieves samples on the Sense and AUX ADCs concurrently, thus obtaining the two voltages, V_{REF} and V_{NTC}. This allows the user to determine the resistance of the NTC thermistor. The following proof using Ohm's Law can be therefore be used to find the resistance, R_{NTC}, of the NTC thermistor.

$$Iref = \frac{Vref}{Rref}$$

$$Intc = \frac{Vntc}{Rntc}$$

$$As\ Iref\ =\ Intc$$

$$\frac{Vref}{Rref} = \frac{Vntc}{Rntc}$$

$$Therfore\ Rntc\ =\ Vntc\ *\frac{Rref}{Vref}$$

Equation 1 R_{NTC} Proof

The CCS811 NTC mailbox (ID = 6) provides the application with the V_{REF} and V_{NTC} values sampled by the AUX and Sense pins respectively. The format of this data is shown in Table 2 NTC Mailbox Data Encoding.

Byte 0	Byte 1	Byte 2	Byte 3
Voltage across R_{REF} (mV)		Voltage across R_{NTC} (mV)	
High Byte	Low Byte	High Byte	Low Byte

Table 2 NTC Mailbox Data Encoding

Both V_{REF} and V_{NTC} are in millivolts. As R_{REF} is a discrete component with a known magnitude of resistance, and as the CCS811 has provided the V_{REF} and V_{NTC} values it is therefore very simple to determine the R_{NTC} value. In the simplest case when V_{REF} and V_{NTC} are equal, R_{NTC} is equal to R_{REF}.

Using the Simplified Steinhart Equation to Determine Temperature

After determining the value of R_{REF} the thermistor's data sheet must be consulted in order to understand how this can be used to calculate the ambient temperature. The most common method for this is to use a simplified version of the Steinhart equation. In that case the data sheet will contain an equation of the form shown in Equation 2 Simplified Steinhart Equation.

$$B = \frac{\log\left(\frac{R}{Ro}\right)}{\frac{1}{T} - \frac{1}{To}}$$

Equation 2 Simplified Steinhart Equation

The equation contains a number of parameters that are found in the NTC thermistor's data sheet. It also contains some parameters that the user must provide. These are described below in Table 3 Simplified Steinhart Equation Parameter Descriptions.

Parameter	Description
B	A constant value, in Kelvin, provided by the thermistor's data sheet. This is used to describe how the thermistor's slope or curve reacts to changes in temperature. Viewed on a logarithmic scale this slope is almost linear in the range -20-85°C.
T_O	The temperature, in Kelvin, that yields a thermistor resistance of R_O. It is commonly 25°C (converted to Kelvin). This can be found in the thermistor's data sheet.
R_O	The resistance of the thermistor at a temperature equal to T_O. This can be found in the thermistor's data sheet.
R	The absolute resistance of the thermistor. Basically the thermistor's resistance at the current temperature T.
T	The temperature at resistance R.

Table 3 Simplified Steinhart Equation Parameter Descriptions

Note that B, T_O and R_O are constant values that can be found in the thermistor's data sheet. Also observe that R is the resistance of the thermistor at the current temperature. This value is available to the application after reading the data in the CCS811 NTC mailbox and using Equation 1. Therefore the only unknown value is the temperature, T. This allows Equation 2 to be solved for T by rearranging as shown in Equation 3 Temperature Calculation.

$$\frac{1}{T} = \frac{1}{To} + \frac{1}{B}\log\left(\frac{R}{Ro}\right)$$

Equation 3 Temperature Calculation

The temperature can then be calculated in software as described in the subsequent sections.

Calculating Temperature

The first step in calculating temperature is to perform a read to the NTC mailbox, this will look something similar to the following, adapt accordingly to the applications drivers and API:

```
i2c_write(CCS_811_ADDRESS, NTC_REG, i2c_buff, size=0);
i2c_read(CCS_811_ADDRESS, i2c_buff, size=4);
```

The first I^2C transaction is a setup write to the NTC mailbox (the argument NTC_REG has the value 6) with no data. This is followed by a 4 byte read, that access the NTC mailbox and stores these 4 bytes of data to a byte/character array called i2c_buff. Please see CC-000803 Programming and Interfacing Guide for more details on handling the CCS811 I^2C interface and timing requirements.

The V_{REF} and V_{NTC} in i2c_buff can then be passed to a function that implements Equation 1.

```
#define RREF 100000

rntc = calc_rntc((uint16_t)(i2c_buff[0]<<8 | i2c_buff[1]),
                      (uint16_t)(i2c_buff[2]<<8 | i2c_buff[3]));

uint32_t calc_rntc(uint16_t vref, uint16_t vntc)
{
        return (vntc * RREF / vref);

}
```

The value of RREF is the R_O taken from the thermistors data sheet. As i2c_buff is an array of chars in this example it will have to be converted to 2x16 bit scalars in order to be passed to the function. It is recommended to do the shifting as this will work on both a big and little endian host processor.

The returned rntc value can then be used to determine the temperature.

Application Software Running on a CPU with Floating Point Support

If the CPU running the application has floating point support and sufficient program memory for the floating point calculations and/or library functions then the c standard math library can be used to help implement Equation 3.

An example is shown below:

```
#define RNTC_25C            100000
#define BCONSTANT           4250
#define RNTC_TEMP           25

double calc_temp_from_ntc(uint32_t rntc)
{
        double ntc_temp;

        ntc_temp = log((double)rntc / RNTC_25C);    // 1
        ntc_temp /= BCONSTANT;                      // 2
        ntc_temp += 1.0 / (RNTC_TEMP + 273.15);     // 3
        ntc_temp = 1.0 / ntc_temp;                  // 4
        ntc_temp -= 273.15;                         // 5

        return ntc_temp;
}
```

Recall Equation 3:

$$\frac{1}{T} = \frac{1}{To} + \frac{1}{B}\log\left(\frac{R}{Ro}\right)$$

The application developer can extract from thermistors data sheet the constant values in order to solve for temperature. RNTC_25C, BCONSTANT and RNTC_TEMP correspond to R_O, B and T_O respectively. These can then be written to a c header file used in the application software.

Comments for each of the 5 steps in the software example above are as follows:

1. Calculate $\log(R/R_O)$ using the maths library, use the R_{NTC} value calculated from Equation 1
2. Divide $\log(R/R_O)$ by the thermistor's B constant
3. Add $1/T_O$ to the interim result, the equation requires all temperatures are in Kelvin. Adding 273.15 to T_O converts the value in the thermistor's data sheet, normally 25°C (RNTC_TEMP), to Kelvin
4. The result of 3 is the reciprocal of the temperature so this step yields the current temperature in Kelvin
5. Convert from Kelvin to °C

Application Software Running on a CPU With No Floating Point Support

If the application CPU does not have floating point support or there is insufficient program memory available for the library and/or floating point calculations, then the temperature can be determined using linear interpolation between two point on the thermistor's temperature versus resistance curve. Finding the two points can be done as follows:

1. The thermistors data sheet can be consulted to find the resistance at various points on the graph, normally in increments of 5°C
2. Pre-calculate the resistances at various temperatures required by the application in increments of x°C, where x is application specific.

The application software can store the resistance values in an array or lookup table and use the resistance, R_{NTC} calculated using Equation 1, as an input into the look up table. The look up operation must be programmed to return the two resistance points. Basically the goal is to determine which two resistance values R_{NTC} lies between in the lookup mechanism used by the application. Then linear interpolation can then be used to determine a reasonably accurate temperature value.

For example assuming that a 100kΩ thermistor has a resistance of 210k at 10°C and 270k at 5°C. Let's also assume Equation 1 yielded a value of 222000Ω for R_{NTC}. The following can be used to approximate the temperature.

1. The application must determine how many steps between the two points are required. For example assuming 100 steps then subtract the high and low resistance and divide by 100:

$$step\_size = (270000 - 210000) / 100 = 600$$

   Thus each 0.05°C increment in temperature corresponds to 600Ω decrease in resistance between these 2 points

2. Calculate how may steps R_{NTC} is from the higher resistance:

$$rntc\_steps = (270000 - 222000) / step\_size = 80$$

3. To avoid errors using integer types, use a few orders of magnitude greater: i.e. lower temp*1000 (5x1000 = 5000). As each step is 50 milli °C (0.05x1000) the temperature is therefore:

$$T = 5000 + rntc\_steps * 50 = 9000, i.e. 9\ degrees\ Celsius.$$

Using the NTC Thermistor for Temperature Compensation in the CCS811

The temperature determined using the NTC thermistor circuit and the equations in the sections above can be used for temperature compensation on the CCS811.

When writing the temperature to the ENV_DATA register it is necessary to also write the humidity. If the humidity is not known then the default value corresponding to 50% relative humidity must be written to ENV_DATA. For example if the temperature is 30°C and no RH data is available then the user must write four bytes as follows:

0x64, 0x00, 0x6E, 0x00

The first two bytes are the RH data in the format required by the CCS811 ENV_DATA register. The next two bytes are the temperature +25°C in the format required by ENV_DATA. Please consult the data sheet for more information. Additionally a full example of using ENV_DATA is available in application note CC-000803-AN Programming and Interfacing Guide.

Abbreviations

Abbreviation	Description
CCS	Cambridge CMOS Sensors
CMOS	Complementary Metal Oxide Semiconductor
RH	Relative Humidity
NTC	Negative Thermal Coefficient

References

Reference	Description
CC-000619-DS	Datasheet for CCS811
CC-000803-AN	Programming and Interfacing Guide