FRONTGRADE

APPLICATION NOTE

UT32M0R500

Calculating and Converting Temperature Sensor Codes

3/15/2021

Version #: 1.0.0

Introduction

The 17th channel of the UT32M0R500 is connected to an internal temperature sensor. This appnote will describe how to read from this temperature channel, how to convert code into a usable value, and a very basic method of characterizing the temperature sensor using measurements taken at cold and hot temperatures.

Reading from the ADC Temperature Channel

The below code uses the Frontgrade Software Development Kit (SDK) to enable the ADC and Temperature channel. For this example, the ADC will be run in Single Sweep Mode.

```
ADC_TypeDef *ADC = (ADC_TypeDef *) ADC_BASE;
ADC_InitTypeDef ADC_InitStruct;
ADC_ChanCfgTypeDef ADC_ChanCfgStruct;

void ADC_Setup(void){
        ADC_StructInit(&ADC_InitStruct);
        ADC_InitStruct.SweepType = ADC_SWEEP_SINGLE;
        ADC_InitStruct.OscillatorDivisor = ADC_OSCDIV_BY_4;
        ADC_InitStruct.OBD_Delay = ADC_OBD_DELAY_0;
        ADC_InitStruct.ModulatorSamples = 127;
        ADC_InitStruct.SequenceDelay = 14;
        ADC_InitStruct.ModulatorResetClocks = 8;
        ADC_InitStruct.OverloadCounterThreshold = 30;
        ADC_Init(ADC, &ADC_InitStruct);
}
```

The temperature channel must have a gain setting of 16V/V for usable codes.

```
void ADC_TemperatureChannelSetup(void){
        ADC_ChanCfgStruct.UseDDF2 = FALSE; //COI3 for all!
        ADC_ChanCfgStruct.Gain = ADC_GAIN_16VperV;//always 16V/V for temp channel
        ADC_ChanCfgStruct.Enable = ENABLE;
        ADC_SetChannelConfig(ADC, ADC_TEMP_CHAN, &ADC_ChanCfgStruct);
}
```

To perform a sweep, simply call the ADC_Sweep function.

```
ADC_Sweep(ADC, ENABLE);//Trigger a sweep
```

Once the sweep has finished, call ADC_ReadChannel to read the data. In the below code 'ADC_TEMP_DATA_0' is the source of the temperature channel data, and 'Data_ADC' is an array of uint16_t halfwords.

```
ADCError = ADC_ReadChannel(ADC, ADC_TEMP_DATA_0, Data_ADC,&ADC_ChanDataStruct);
```

Finally, print the data to the terminal.

```
printf("\r\n TempChanData: 0x%0.4X  d%d  ADC Error: %X",
Data_ADC[0], Data_ADC[0], ADCError);
```

Example of the data printed to the terminal:

```
TempChanData: 0x09C7     d2503          ADC Error: 0
```

Converting Codes into Temperatures

Translating ADC codes into temperatures can be accomplished using a linear regression, although some users may choose to implement more complex conversion algorithms. To convert a decimal code into Celsius, use the following equation:

Temperature (°C) = (Slope (°C) * code) + Offset (°C)

The slope and the offset will vary between parts per the Datasheet's Temperature Monitor Characteristics Absolute Accuracy. Because there's a significant amount of variation between parts, users should perform characterization on each part for best accuracy.

Basic Two-Temp Characterization

To find the slope and offset of a specific part, users will need at least two data points. The below dataset samples from the temperature channel at -55°C and +105°C, but users could use two different temperatures to perform a basic characterization. Users can always take samples at additional temperatures for a more complete characterization.

For this appnote, five parts were used. Each part had 100 measurements taken at both -55°C and +105°C.

Table 1: Two-Temperature Characterization Data

Part	Temperature (°C)	Average Code	Standard Deviation	Minimum Code	Maximum Code
SN11	-55	2872	0.33	2871	2873
	105	2242	0.42	2241	2243
SN12	-55	2878	0.36	2878	2879
	105	2231	0.39	2230	2232
SN14	-55	2876	0.36	2875	2877
	105	2248	0.45	2248	2249
SN15	-55	2907	0.45	2906	2907
	105	2238	0.49	2237	2239
SN31	-55	2835	0.42	2834	2835
	105	2205	0.36	2204	2206

To calculate the slope, simply determine how much the temperature changes over how much the code changes:

Slope = (Temp2-Temp1)/(Code2 – Code1)

Slope (SN11) = (-55°C - 105°C)/(2872 – 2242)

Slope (SN11) = (-160°C)/630

Slope (SN11) = -0.2539°C

To calculate the offset, use the slope in conjunction with one of the data points.

Offset = Temperature (°C) – (Slope (°C) * code)
Offset (SN11) = 105(°C) – (-0.2539(°C) * 2242)
Offset (SN11) = -674.39°C

Yielding a linear regression for SN11:

Temperature_SN11 (°C) = (-0.2539 (°C) * Code_SN11) – 674.39(°C)

If SN11 had a code of 2557 (exactly halfway between codes 2872 and 2242), the above equation calculates a temperature of 25°C, providing a way to check the equation is approximately correct against real-world data.

Equations for the other four serial numbers are:

Temperature_SN12 (°C) = (-0.2472 (°C) * Code_SN12) – 656.71 (°C)
Temperature_SN14 (°C) = (-0.2547 (°C) * Code_SN14) – 677.73 (°C)
Temperature_SN15 (°C) = (-0.2391 (°C) * Code_SN15) – 640.24 (°C)
Temperature_SN31 (°C) = (-0.2539 (°C) * Code_SN31) – 665.00 (°C)

Using the above equations, we can gain insights into the slope and offset variation of the above five parts.

Part	Slope (°C)	Offset (°C)
SN11	-0.2539	-674.39
SN12	-0.2472	-656.71
SN14	-0.2547	-677.73
SN15	-0.2391	-640.24
SN31	-0.2539	-665.00
Average	-0.24976	-662.814
Standard Deviation	0.00598117	13.47674827

As you can see from the above data, the average slope is similar across the five parts, but the offset between parts varies significantly.

Conclusions

The Absolute Accuracy of the UT32M0R500’s Internal Temperature Sensor means users should characterize each part for accurate temperature readings.

Revision History

Date	Revision #	Author	Change Description	Page #
03/15/2021	1.0.0	OW	Initial Release	

Datasheet Definitions

	Definition
Advanced Datasheet	Frontgrade reserves the right to make changes to any products and services described herein at any time without notice. The product is still in the development stage and the **datasheet is subject to change**. Specifications can be **TBD** and the part package and pinout are **not final.**
Preliminary Datasheet	Frontgrade reserves the right to make changes to any products and services described herein at any time without notice. The product is in the characterization stage and prototypes are available.
Datasheet	Product is in production and any changes to the product and services described herein will follow a formal customer notification process for form, fit or function changes.