Application Note

Document No.: AN1091

APM32F4xx Series TSensor demo Routine Application Note

Version: V1.0

1 Introduction

This application note provides a guide on how to use ADC1_CH16 on the APM32F4xx series for real-time detection of the temperature of the internal temperature sensors.

This application note mainly includes an introduction to internal temperature sensors, application routine design, and practical demonstration of application routine.

Contents

2 Introduction to temperature sensors

2.1 Introduction

The temperature sensor is a device for measuring the degree of hotness and coolness of objects, which provides temperature measurements in a readable form through electrical signals.

2.2 Classification of Temperature Sensors

Contact temperature sensors and non-contact temperature sensors.

2.3 Common Temperature Sensors

Contact temperature sensors: Thermal resistor, thermocouple, and thermostat sensors.

Non-contact temperature sensor: Infrared temperature sensor.

3 Introduction to Internal Temperature Sensor of Chip

3.1 Overview

This article mainly introduces the usage of ADC_TSensor routine in APM32F4xx_SDK_V1.2. This routine monitors the voltage value of the internal temperature sensor of the chip and converts it into a temperature value through a formula.

3.2 Introduction to Internal Temperature Sensor

1. APM32F40x integrates a temperature sensor internally to measure the CPU and surrounding temperatures.

2. The internal sensor is connected to ADC1_CH16 channel in the chip, and can convert the analog voltage value of the sensor into a digital value through the ADC1_CH16 channel, and then figure out the temperature value through a formula.

3. The recommended sampling time for temperature sensor analog input is 17.1μs.

4. The internal temperature sensor of APM32F40x supports a temperature range of -40~125 ℃.

3.3 Principle

The principle and conversion process are shown in Figure 1:

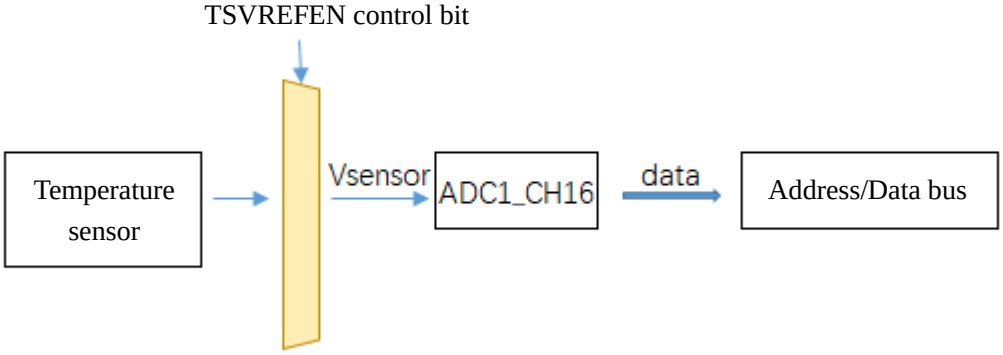

Figure 1

3.3.1 TSVREFEN Control Bit

The TSVREFEN bit in the ADC_CCTRL register is used to control the enabling status of the internal temperature sensor.

3.3.2 Calculation Formula

The analog voltage V_{sensor} of the temperature sensor is collected through ADC1_CH16 channel, and is converted into digital data, and then the temperature value of the temperature sensor can be calculated

according to the formula.

The formula is as follows:

$T = (V_{sensor} - V_{25}) / Slope + 25$

Formula description:

T, V_{sensor} are the temperature and output voltage of the temperature sensor respectively;

V_{25}, 25, is the reference output voltage value of the temperature sensor at 25 ℃;

Slope is the average slope of the temperature and V_{sensor} curve (in mv/ ℃ or uv/ ℃).

Because the process characteristics of each chip are not exactly the same, there may be errors between the temperature calculated according to the formula and the actual temperature.

The reference value and average slope value are obtained based on multiple batches of test data, and the final formula is:

$T = (V_{sensor} - 0.7782) / 0.0024 + 28$

The error is approximately within 4℃.

4 Application Routine Design

4.1 Programming Idea

1. Configure DMA peripheral and enable DMA transmission;
2. Configure ADC1 and select channel16;
3. Configure the sampling time to 480 cycles, with a sampling time greater than 17.1μs;
4. Set the TSVREFEN bit of ADC1_CCTRL to enable the internal temperature sensor;
5. Set the DMAEN bit of ADC1_CTRL2 to enable DMA mode;
6. Set the ADCEN bit of ADC1_CTRL2 to start conversion;
7. Read the ADC1 conversion results of DMA transmission;
8. Calculate the corresponding temperature value according to the formula.
9. Configure the serial port and use it to print out relevant information.

4.2 Hardware Design

This routine only needs to use a serial port line to connect the TX and RX of USART1 (TX: PA9, RX: PA10).

4.3 Software Design

4.3.1 USART1 Initialization

This routine uses USART1 to print information, so it needs to initialize the USART1.

The USART1 configuration is as follows:

```
USART_Config_T usartConfigStruct;


    /* USART configuration */
    USART_ConfigStructInit(&usartConfigStruct);
    usartConfigStruct.baudRate          = 115200;
    usartConfigStruct.mode              = USART_MODE_TX_RX;
    usartConfigStruct.parity            = USART_PARITY_NONE;
    usartConfigStruct.stopBits          = USART_STOP_BIT_1;
    usartConfigStruct.wordLength        = USART_WORD_LEN_8B;
    usartConfigStruct.hardwareFlow      = USART_HARDWARE_FLOW_NONE;


    /* COM1 init*/
    APM_MINI_COMInit(COM1, &usartConfigStruct);
```

4.3.2 DMA Initialization

This routine applies to DMA peripheral, so it is necessary to initialize the DMA peripheral.

The DMA configuration is as follows:

```
/*!
 * @brief       DMA Init
*
 * @param       None
 *
 * @retval      None
 */
void DMA_Init(uint32_t* Buf)
{
    /* DMA Configure */
    DMA_Config_T dmaConfig;
    /* Enable DMA clock */
    RCM_EnableAHB1PeriphClock(RCM_AHB1_PERIPH_DMA2);
```

```
    /* size of buffer*/
    dmaConfig.bufferSize = 1;
    /* set memory Data Size*/
    dmaConfig.memoryDataSize = DMA_MEMORY_DATA_SIZE_HALFWORD;
    /* Set peripheral Data Size*/
    dmaConfig.peripheralDataSize = DMA_PERIPHERAL_DATA_SIZE_HALFWORD;
    /* Enable Memory Address increase*/
    dmaConfig.memoryInc = DMA_MEMORY_INC_DISABLE;
    /* Disable Peripheral Address increase*/
    dmaConfig.peripheralInc = DMA_PERIPHERAL_INC_DISABLE;
    /* Reset Circular Mode*/
    dmaConfig.loopMode = DMA_MODE_CIRCULAR;
    /* set priority*/
    dmaConfig.priority = DMA_PRIORITY_HIGH;
    /* read from peripheral*/
    dmaConfig.dir = DMA_DIR_PERIPHERALTOMEMORY;
    /* Set memory Address*/
    dmaConfig.memoryBaseAddr = (uint32_t)Buf;
    /* Set Peripheral Address*/
    dmaConfig.peripheralBaseAddr = (uint32_t)&ADC1->REGDATA;


    dmaConfig.channel = DMA_CHANNEL_0;
    dmaConfig.fifoMode = DMA_FIFOMODE_DISABLE;
    dmaConfig.fifoThreshold = DMA_FIFOTHRESHOLD_FULL;
    dmaConfig.peripheralBurst = DMA_PERIPHERALBURST_SINGLE;
    dmaConfig.memoryBurst = DMA_MEMORYBURST_SINGLE;

    DMA_Config(DMA2_Stream0, &dmaConfig);
    /* Clear DMA TF flag*/
    DMA_ClearIntFlag(DMA2_Stream0, DMA_INT_TCIFLG0);
    /* Enable DMA Interrupt*/
    DMA_EnableInterrupt(DMA2_Stream0, DMA_INT_TCIFLG);
    DMA_Enable(DMA2_Stream0);

}
```

4.3.3 ADC Initialization

This routine applies to ADC peripheral, so it is necessary to initialize the ADC peripheral.

The ADC configuration is as follows:

```
/*!
 * @brief         ADC Init
 *
 * @param         None
 *
 * @retval        None
 */
void ADC_Init(void)
{
    ADC_Config_T    adcConfig;
    ADC_CommonConfig_T    adcCommonConfig;

    RCM_EnableAPB2PeriphClock(RCM_APB2_PERIPH_ADC1);

    /* ADC Configuration */
    ADC_Reset();
    ADC_ConfigStructInit(&adcConfig);
    /* Set resolution*/
    adcConfig.resolution = ADC_RESOLUTION_12BIT;
    /* Set dataAlign*/
    adcConfig.dataAlign = ADC_DATA_ALIGN_RIGHT;
    /* Set scanDir*/
    adcConfig.scanConvMode = DISABLE;
    /* Set convMode continous*/
    adcConfig.continuousConvMode = ENABLE;
    /* Set extTrigEdge*/
    adcConfig.extTrigEdge = ADC_EXT_TRIG_EDGE_NONE;

    /*Set ADC Clock Prescaler. ADC_Clock = APB2_Clock/4, 84/4=21MHz*/
    adcCommonConfig.prescaler = ADC_PRESCALER_DIV4;
    ADC_CommonConfig(&adcCommonConfig);
```

4.3.4 Functional Logical Subject of Application Routine

```
if (DMA_ReadStatusFlag(DMA2_Stream0, DMA_FLAG_TCIFLG0))
        {
            DataBuf = DMA_ConvertedValue;
            VSensorValue = (float)DataBuf/4095*3.3;

            /*!
             * According to actual test data of multiple bathes of chips,
             * V28 is adopted instead of V25 for this example. And 0.7782v is the voltage for 28℃
             */
            TSensorValue = (VSensorValue - 0.7782f)/0.0024f + 28;

            printf("\r\n");
            printf("ADC REGDATA = 0x%04X \r\n", DataBuf);
            printf("Volatage        = %f V \r\n", VSensorValue);
            printf("Temperature        = %f ℃ \r\n", TSensorValue);

            Delay(0xFFFFFF);
            DMA_ClearStatusFlag(DMA2_Stream0, DMA_FLAG_TCIFLG0);
        }
```

5 Practical Demonstration of Application Routine

The actual demonstration environment for this routine is:

1. Demonstration on the APM32F407IG MINIBOARD physical object;

2. The engineering is established on the IDE of Keil MDK-ARM V5.29.0.0, and the chip Device selects APM32F40IG

5.1 Application Routine Keil MDK Engineering Compilation

After completing the engineering codes, compile without errors or warnings, as shown in Figure 2:

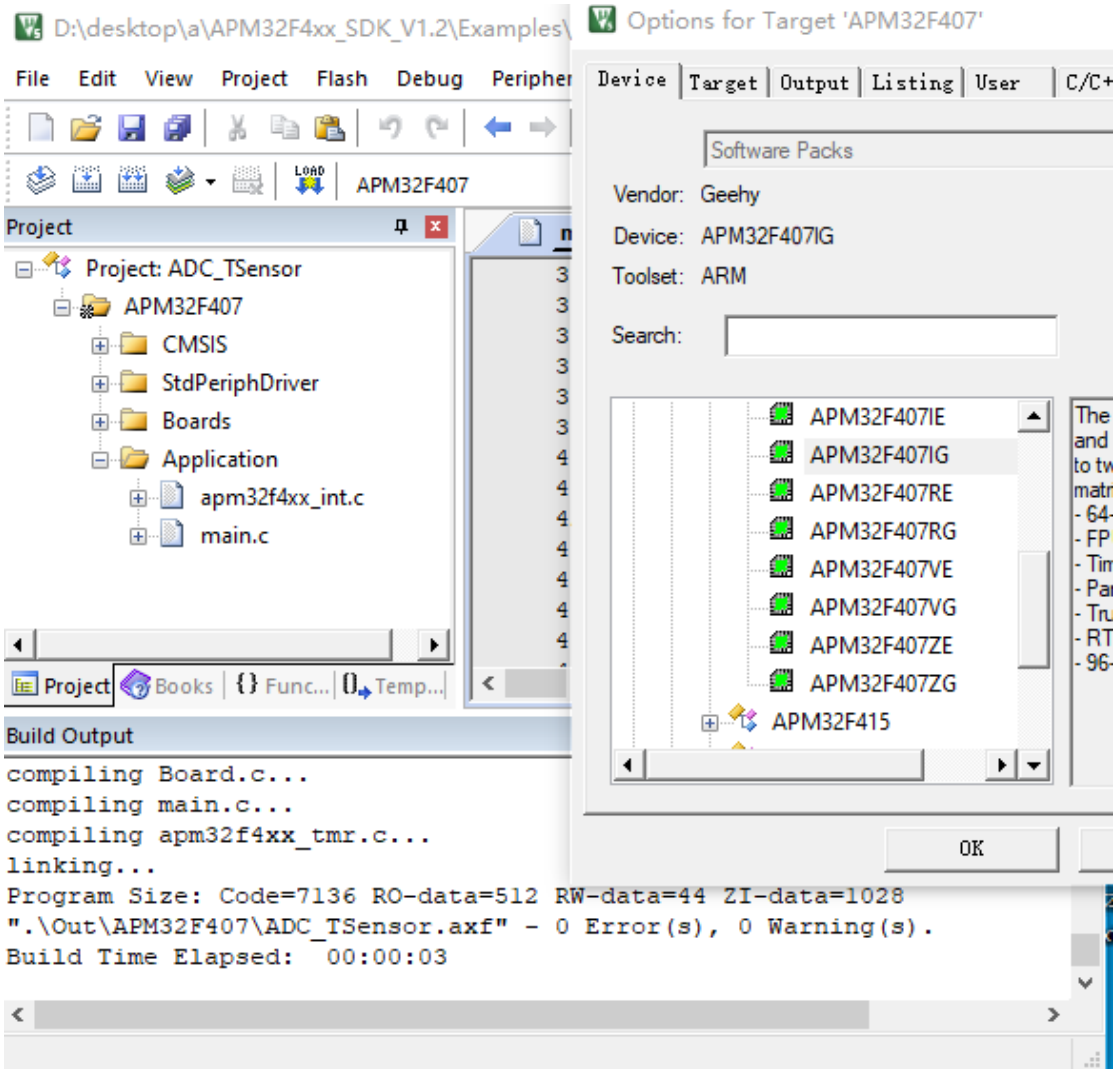

Figure 2

5.2 Burn the Program and Observe the Data Results

Successfully burn the program and display relevant information on the serial assistant tool, as shown in Figure 3:

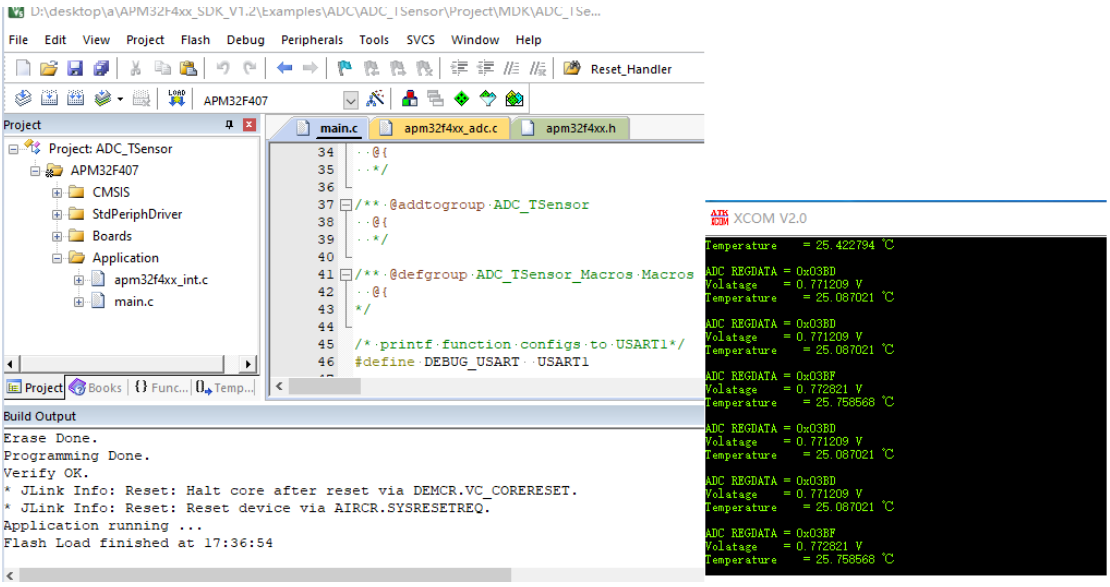
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Reset_Handler
APM32F407
Project
Project: ADC_TSensor
APM32F407
CMSIS
StdPeriphDriver
Boards
Application
apm32f4xx_int.c
main.c
main.c
apm32f4xx_adc.c
apm32f4xx.h
/**·@addtogroup·ADC_TSensor
/**·@defgroup·ADC_TSensor_Macros·Macros
/*·printf·function·configs·to·USART1*/
#define·DEBUG_USART··USART1
Build Output
Erase Done.
Programming Done.
Verify OK.
* JLink Info: Reset: Halt core after reset via DEMCR.VC_CORERESET.
* JLink Info: Reset: Reset device via AIRCR.SYSRESETREQ.
Application running ...
Flash Load finished at 17:36:54
XCOM V2.0
Temperature = 25.422794 ℃
ADC REGDATA = 0x03BD
Volatage = 0.771209 V
Temperature = 25.087021 ℃
ADC REGDATA = 0x03BD
Volatage = 0.771209 V
Temperature = 25.087021 ℃
ADC REGDATA = 0x03BF
Volatage = 0.772821 V
Temperature = 25.758568 ℃
ADC REGDATA = 0x03BD
Volatage = 0.771209 V
Temperature = 25.087021 ℃
ADC REGDATA = 0x03BD
Volatage = 0.771209 V
Temperature = 25.087021 ℃
ADC REGDATA = 0x03BF
Volatage = 0.772821 V
Temperature = 25.758568 ℃

图 3

6 Version History

Table 1 Document Version History

Date	Version	Change History
2023.03.01	1.0	New

Statement

This document is formulated and published by Geehy Semiconductor Co., Ltd. (hereinafter referred to as “Geehy”). The contents in this document are protected by laws and regulations of trademark, copyright and software copyright. Geehy reserves the right to make corrections and modifications to this document at any time. Please read this document carefully before using Geehy products. Once you use the Geehy product, it means that you (hereinafter referred to as the “users”) have known and accepted all the contents of this document. Users shall use the Geehy product in accordance with relevant laws and regulations and the requirements of this document.

1. Ownership

This document can only be used in connection with the corresponding chip products or software products provided by Geehy. Without the prior permission of Geehy, no unit or individual may copy, transcribe, modify, edit or disseminate all or part of the contents of this document for any reason or in any form.

The “极海” or “Geehy” words or graphics with “®” or “™” in this document are trademarks of Geehy. Other product or service names displayed on Geehy products are the property of their respective owners.

2. No Intellectual Property License

Geehy owns all rights, ownership and intellectual property rights involved in this document.

Geehy shall not be deemed to grant the license or right of any intellectual property to users explicitly or implicitly due to the sale or distribution of Geehy products or this document.

If any third party’s products, services or intellectual property are involved in this document, it shall not be deemed that Geehy authorizes users to use the aforesaid third party’s products, services or intellectual property, unless otherwise agreed in sales order or sales contract.

3. Version Update

Users can obtain the latest document of the corresponding models when ordering Geehy products.

If the contents in this document are inconsistent with Geehy products, the agreement in thesales order or the sales contract shall prevail.

4. Information Reliability

The relevant data in this document are obtained from batch test by Geehy Laboratory or cooperative third-party testing organization. However, clerical errors in correction or errors caused by differences in testing environment may occur inevitably. Therefore, users should understand that Geehy does not bear any responsibility for such errors that may occur in this document. The relevant data in this document are only used to guide users as performance parameter reference and do not constitute Geehy’s guarantee for any product performance.

Users shall select appropriate Geehy products according to their own needs, and effectively verify and test the applicability of Geehy products to confirm that Geehy products meet their own needs, corresponding standards, safety or other reliability requirements. If loses are caused to users due to the user's failure to fully verify and test Geehy products, Geehy will not bear any responsibility.

5. Legality

USERS SHALL ABIDE BY ALL APPLICABLE LOCAL LAWS AND REGULATIONS WHEN USING THIS DOCUMENT AND THE MATCHING GEEHY PRODUCTS. USERS SHALL UNDERSTAND THAT THE PRODUCTS MAY BE RESTRICTED BY THE EXPORT, RE-EXPORT OR OTHER LAWS OF THE COUNTIRIES OF THE PRODUCTS SUPPLIERS, GEEHY, GEEHY DISTRIBUTORS AND USERS. USERS (ON BEHALF OR ITSELF, SUBSIDIARIES AND AFFILIATED ENTERPRISES) SHALL AGREE AND PROMISE TO ABIDE BY ALL APPLICABLE LAWS AND REGULATIONS ON THE EXPORT AND RE-EXPORT OF GEEHY PRODUCTS AND/OR TECHNOLOGIES AND DIRECT PRODUCTS.

6. Disclaimer of Warranty

THIS DOCUMENT IS PROVIDED BY GEEHY "AS IS" AND THERE IS NO WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

GEEHY WILL BEAR NO RESPONSIBILITY FOR ANY DISPUTES ARISING FROM THE SUBSEQUENT DESIGN OR USE BY USERS.

7. Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL GEEHY OR ANY OTHER PARTY WHO PROVIDE THE DOCUMENT "AS IS", BE LIABLE FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, DIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE DOCUMENT (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY USERS OR THIRD PARTIES).

8. Scope of Application

The information in this document replaces the information provided in all previous versions of the document.