SPC1168 温度传感器使用指南

概述

在芯片使用过程中，可以将芯片内部的温度通过 ADC 采集出来。

注意： 本文档主要以 SPC1168 为例进行介绍。

目录

图片列表

版本历史

版本	日期	作者	状态	变更
A/0	2023-06-08	CanChai	Outdated	首次发布。
C/0	2024-03-26	Jiali Zhou	Released	修改排版格式。

术语或缩写

术语或缩写	描述
/	/

1 温度传感器简介

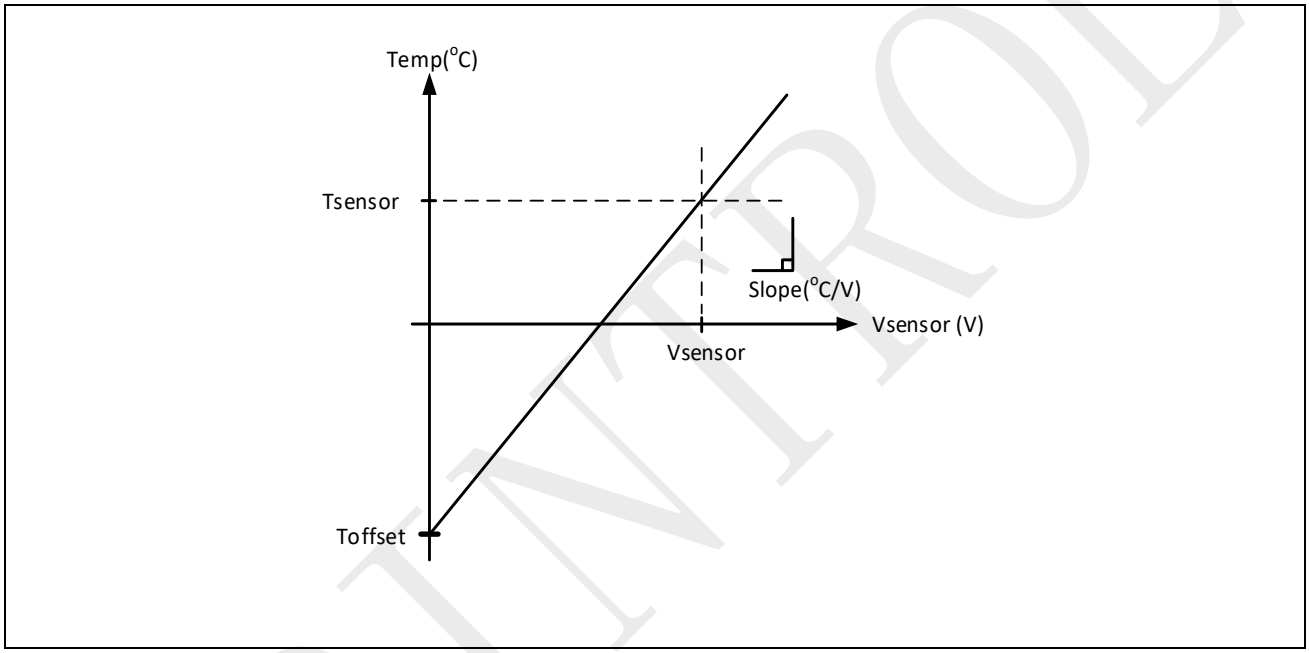
SPC1168 内部集成了温度传感器，其输出为电压信号，经由 14-bit ADC 量化为数字码，根据温度转化曲线可以得到。其具有如下特性：

温度检测范围-40°C~125°C；

温度检测分辨率为±1.9°C /LSB；

温度检测偏差为±10°C。

图 1–1：仿真器件信号流图



2 温度测量

2.1 温度计算公式

SPC1168 的计算公式如下：

$$\text{Temp} = \text{Slope} * V_{sens} + T_{offset}$$

其中，斜率$\text{Slope} = 4238.97℃/V$，失调$T_{offset} = -283.36℃$。

由于温度传感器的输出电压会用 ADC 进行转换，所以实际上，需要根据 ADC 的数字输出码推算温度。在这种情况下，考虑到计算精度，则 SPC2168 温度传感器的计算公式如下：

$$\text{Temp} = \text{Slope} * \text{i32ADCCode} * 3.657/8192 + T_{offset}$$

例如，在室温25℃下，温度传感器的输出电压为 72.7mV，则数字输出码为 162。实际温度为24.8℃，计算温度为23.20℃。相差±1.9℃ /LSB。

2.2 温度传感器动态元件匹配（DEM）

SPC1168 的温度传感器内部电路的元件失配会影响温度传感器的输出电压，进而影响曲线的精准度。为了改善这个问题，测量温度时，我们可以采用多次测量，每次测量时动态切换其内部元件的分配方式，再将这些多次测量的结果取平均，以减小元件失配带来的影响。Spintrol 提供的 ADC_CalculateTemperature()函数实现了此功能，用户可以直接使用。

2.3 用 PGA 提高 ADC 温度检测分辨率

可以通过用 PGA 放大温度传感器的电压输出，以此来提高 ADC 的测量精度。由于 Tsensor 电压输出 VSENS1 和 VSENS0 的绝对值在 0.7V 附近，他们的差值在−40℃~125℃范围内为 57mV~96mV。所以考虑 PGA 工作在差分模式，增益选择 8X。这样 ADC 测量分辨率可以提高到±0.24℃/LSB。

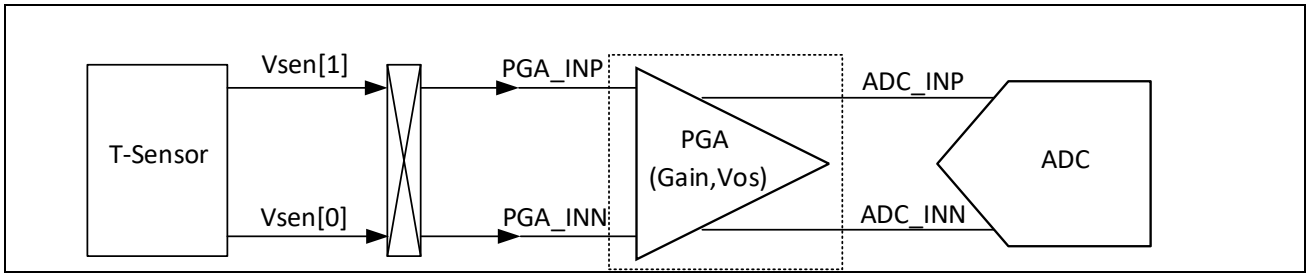
$$\text{Temp} = \text{Slope} * \text{i32ADCCode} * 3.657/8192/8 + T_{offset}$$

在室温25℃下，温度传感器的输出电压为72.7mV，PGA 的电压输出为581.6mV，则数字输出码为 1302。计算出来的温度为24.62℃，和理想情况的误差小于0.5℃。

2.4 消除 PGA 失调

PGA 存在失调误差，这个误差会随着温度而变。我们可以通过 Tsensor 的 CHOP 功能消除这个误差。如图 2–1 所示，需要测量两次，第二次和第一次的不同之处在于 Tsensor 输出交换了位置。

图 2–1：用 PGA 增加温度传感器测量分辨率



假设 PGA 的增益为 Gain，失调为 Vos。Tsensor 的输出为 Vsen[1]和 Vsen[0]。则，

对于第一次测量，PGA 的输出为

$$\mathrm{Vout1} = \mathrm{Gain} \times (\mathrm{Vsen}[1] - \mathrm{Vsen}[0] + \mathrm{Vos})$$

对于第二次测量，PGA 的输出为

$$\mathrm{Vout2} = \mathrm{Gain} \times (\mathrm{Vsen}[0] - \mathrm{Vsen}[1] + \mathrm{Vos})$$

将两次 PGA 的输出结果相减并除以 2，可以得到

$$\Delta V = \frac{Vout1 - Vout2}{2} = \mathrm{Gain} \times (\mathrm{Vsen}[1] - \mathrm{Vsen}[0])$$

因此 PGA 的 offset 就被消除了，这样只剩下 PGA 的增益误差，这个增益误差很小（<1%），在 150°C 时，只会贡献 4.3°C 的误差。

ADC_CalculatePreciseTemperature()代码如下：

Example Code

```
int32_t ADC_CalculatePreciseTemperature(ADC_SocEnum eSOC)
{
    volatile uint32_t i, j;
    uint32_t u32DivOld, u32DivNew;
    int32_t i32ADCCode;
    int32_t i32TempAvg;

    /* Power up ADC */
    ADC_PowerUp();

    /* Save Old divider */
    u32DivOld = CLOCK_GetModuleDiv(ADC_MODULE);

    /* Set new ADC Clock Divider - Set ADC Clock = 20MHZ */
    u32DivNew = CLOCK_GetModuleClock(ADC_MODULE) / 20000000U;
    if (u32DivNew == 0U)
    {
        u32DivNew = 1U;
    }
    CLOCK->ADCCLKCTL.bit.DIV = (u32DivNew - 1U);


    /* Init PGA2 */
    PGA_DifferentialInit(PGA2, PGA2_CH_P_TSENSOR_H, PGA2_CH_N_TSENSOR_L,
PGA_SCALE_8X);
    PGA_SelectPositiveChannelAsCommonInput(PGA2);


    /* Select T-Sensor as ADC input */
```

```
    ADC_SelectPinDifferetial(eSOC, ADCx_PGA2P, ADCx_PGA2N, ADCTRIG_Software);

    /* Set Sample and Convert time */
    ADC_SetSampleAndConvertTime(eSOC, 500, 150);

    /* Set S/H gain and offset */
    ADC_SetGainAndOffset(eSOC);

    /* Set Average Count */
    ADC_SetAverageCnt(eSOC, ADCSOCCTL0_BIT_AVGCNT_AVG_16);

    /* Enable SOC interrupt */
    ADC_EnableInt(eSOC);

    /* Clear interrupt flag */
    ADC_ClearInt(eSOC);


    /* Enable T-Sensor */
    ADC->TSENSCTL.bit.EN = 1;
    ADC->TSENSCTL.bit.OUTINV = 0;
    ADC->TSENSCTL.bit.DEMSEL = 0;
    ADC->TSENSCTL.bit.SWAPBJT = 0;

    i32ADCCode = 0;

    /* Use ADC to measure T-Sensor by DEM */
    ADC->TSENSCTL.bit.SWAPBJT = 0;
    for (j = 0; j < 16; j++)
    {
        /* Alter internal bias current */
        ADC->TSENSCTL.bit.DEMSEL = j;

        /* Delay 1ms */
#if DELAY_FOR_MEM != DELAY_FOR_FLASH
        i = ((SystemCoreClock / 1000000) * 1000 / 4);
#else
        i = ((SystemCoreClock / 1000000) * 1000 / (4 * SystemCoreClock /
32000000));
#endif
        while (i--);

        /* Software trigger SOC */
        ADC_SoftwareTrigger(eSOC);

        /* Wait SOC Finished */
        while (ADC_GetIntFlag(eSOC) == 0);

        i32ADCCode += (int32_t)ADC->ADCRESULT[eSOC].all;

        /* Clear interrupt flag */
        ADC_ClearInt(eSOC);
    }


    /* Alter current between ADCx_TSENSOR_H and ADCx_TSENSOR_L */
    ADC->TSENSCTL.bit.SWAPBJT = 1;
    for (j = 0; j < 16; j++)
    {
        /* Alter internal bias current */
        ADC->TSENSCTL.bit.DEMSEL = j;

        /* Delay 1ms */
```

```
#if DELAY_FOR_MEM != DELAY_FOR_FLASH
        i = ((SystemCoreClock / 1000000) * 1000 / 4);
#else
        i = ((SystemCoreClock / 1000000) * 1000 / (4 * SystemCoreClock /
32000000));
#endif
        while (i--);

        /* Software trigger SOC */
        ADC_SoftwareTrigger(eSOC);

        /* Wait SOC Finished */
        while (ADC_GetIntFlag(eSOC) == 0);

        i32ADCCode -= (int32_t)ADC->ADCRESULT[eSOC].all;

        /* Clear interrupt flag */
        ADC_ClearInt(eSOC);
    }

    /* Average ADC code */
    i32ADCCode /= 32 ;

    /* Calculate temperature */
    i32TempAvg = TSENSOR_SLOPE * (i32ADCCode * (float)3.657f / 8192 / 8) -
TSENSOR_OFFSET;

    /* Restore ADC Clock Divider */
    CLOCK->ADCCLKCTL.bit.DIV = u32DivOld - 1;

    /* Restore ADC SOC register */
    ADC->ADCSOCCTL[eSOC].all = 0;
    ADC_DisableInt(eSOC);
    ADC->TSENSCTL.all = 0;

    /* Restore PGA register */
    PGA->PGA2CTL.all = 0;

    return i32TempAvg;
}
```