SPC1169 ADC PGA 校准使用指南

版本 A/0 – 2023 年 5 月

概述

默认芯片出厂都会校准 ADC，其校准值写入到 NVR 中，并在芯片启动的时候通过 boot 程序自动加载到 ADC SH0 对应寄存器 SHOFFSET[0]和 SHGAIN[0]中，此时 ADC 的性能符合设计指标，如无特殊需要，无需校准。

目录

图片列表

表格列表

版本历史

版本	日期	作者	状态	变更
A/0	2023 年 5 月 6 日	Hang Su	Released	首次发布。

术语或缩写

术语或缩写	描述

1 ADC 校准

ADC 的设计参数如表 1-1 所示，芯片出厂时已经过相关参数的校准。需要注意的是，如无特殊需要，不需要再次校准。

表 1-1：13 位模数转换器特性

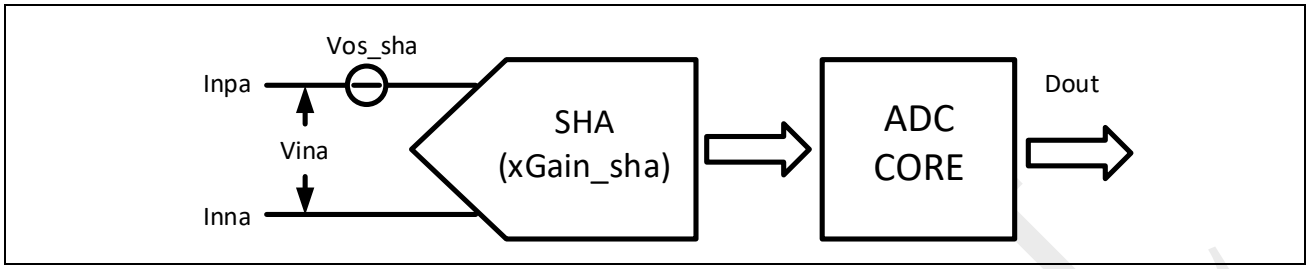
符号	参数	条件	最小	典型	最大	单位
N_R	分辨率	无码值丢失；单调的；	13	-	-	bits
f_S	转换速度[1]	-	-	-	2.5	MSPS
V_{in}	输入电压范围	-	0	-	V_{DDA}	V
V_{ref}	参考电压	-	1.194	1.2	1.206	V
I_{on}	工作电流	V_{DDA} = 3.3 V	-	8.5	11.5	mA
INL	积分线性误差	-	-3.0	-	3.0	LSB
DNL	微分线性误差	-	-1.0	-	1.0	LSB
E_{offset}	偏移误差[2]	已校准	-2	-	2	LSB
E_{gain}	增益误差[2]	已校准	-4	-	4	LSB
$E_{offset2}$	通道间偏移误差	-	-3	-	3	LSB
E_{gain2}	通道间增益误差	-	-5	-	5	LSB
T_{coef}	基于内部参考的 ADC 温度系数	-	-	30	-	ppm/°C
t_{settle}[3]	启动时间	-	-	-	100	us
$ENOB_{DC}$	有效位数（直流输入）	-	-	11.5	-	bits
SNR	信噪比	f_{in} = 100kHz, V_{in} = 0.94FS, N = 8192	-	71	-	dB
THD	总谐波失真		-	-84	-	dB
ENOB	有效位数		-	11.4	-	bits
SFDR	无杂散动态范围		-	80	-	dB

（1）采样时间 = 200ns，转换时间 = 200ns。
（2）偏移和增益可通过硬件自动校准。
（3）该项目不在量产中测试，设计保证。

1.1 增益以及输入失调电压

ADC 输出如图 1-1 所示。

图 1-1：ADC 采样单元示意图



对应公式为：

$$V_{out} = K(V_{in} + V_{os})$$

$$V_{out} = K * V_{in} + K * V_{os}$$

K为增益，V_{os}为输入失调电压，K增益和V_{os}输入失调电压均受到芯片个体差异以及温度影响。但在芯片给定，温度给定的条件下，K增益和V_{os}输入失调电压均为常量，因此，令：

$$V_{offset} = K * V_{os}$$

V_{offset}为输出失调电压，从而将原式化简为：

$$V_{out} = K * V_{in} + V_{offset}$$

符合线性方程的基本形式，从而可以被校准。

1.2 增益以及输出失调电压校准电路

在 ADC 未校准的时候，实际 ADC 的采样结果为：

$$V_{out} = K * V_{in} + V_{offset}$$

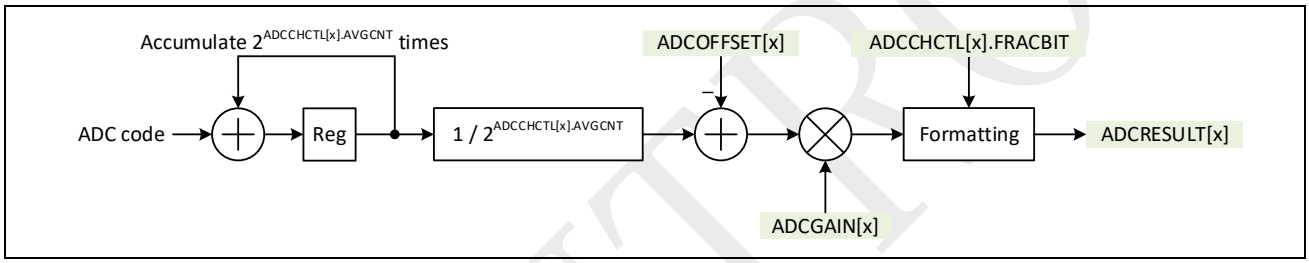
V_{in}表示为：

$$(V_{out} - V_{offset}) * \frac{1}{K} = V_{in}$$

也就是将 ADC 的采样结果V_{out}减去输出失调电压V_{offset}再乘以$\frac{1}{K}$，就可以得到V_{in}，如图 1-2 所示。

注意：校准电路减去的是输出失调电压V_{offset}，不是输入失调电压V_{os}。

图 1-2：ADC 码值处理



注意：ADCOFFSET（V_{offset}）和 ADCGAIN（$\frac{1}{K}$）都是按照定点数存储的。

其中 ADCOFFSET 末 7 位对应小数部分，因此在校准前，可以使能对应 CH 的 FRACBIT 位，从而保持 ADCRESULT（V_{out}）与 ADCOFFSET（V_{offset}）两个定点数小数点位的一致性，从而可以免去计算过程中的移位操作。

ADCGAIN 对应 VAL 为$32768 * \frac{1}{K}$。

1.3 校准过程

根据章节 1.1，ADC 的校准是在芯片给定，温度给定的条件下进行的，要控制变量。

校准前一定要确保采样通道 CH 对应的 ADCOFFSET 为0，对应的$ADCGAIN$为32768。即原校准电路不起任何作用。

可以使能对应 CH 的 FRACBIT 位，从而保持 ADCRESULT（V_{out}）与 ADCOFFSET（V_{offset}）两个定点数小数点位的一致性，从而可以免去计算过程中的移位操作。

将 ADC 正负端都接地($V_{in1}=0$)，得到V_{out1}；

将 ADC 正端接 3.3V，负端接地($V_{in2}=3.3V$)，得到V_{out2}；

ADCOFFSET计算如下：

将$V_{in1}=0$带入$V_{out1}=K*V_{in1}+V_{offset}$，得：

$$V_{out1}=V_{offset}$$

因为使能对应 CH 的 FRACBIT 位，V_{out1}末 7 位对应小数，从而直接满足 ADCOFFSET 的定义，所以

$$\text{ADCOFFSET}=V_{out1}$$

否则，V_{out1}要左移 7 位才能写入 ADCOFFSET 寄存器。

$ADCGAIN$计算如下：

$$ADCGAIN=32768*\frac{1}{K}$$

$$ADCGAIN=32768*\frac{1*V_{in2}}{K*V_{in2}}$$

将$V_{out2}-V_{offset}=K*V_{in2}$，$V_{out1}=V_{offset}$，代入得：

$$ADCGAIN=32768*\frac{1*V_{in2}}{V_{out2}-V_{out1}}$$

而$V_{in2}=(\frac{3.3}{3.657143}*4096)*2^7$，代入得：

$$ADCGAIN=32768*\frac{1*(\frac{3.3}{3.657143}*4096)*2^7}{Vout2-Vout1}$$

$$ADCGAIN=32768*\frac{3696*128}{Vout2-Vout1}$$

$$ADCGAIN=\frac{15502147584}{Vout2-Vout1}$$

最后，将计算出的$ADCGAIN$和ADCOFFSET写入到对应 CH 的$ADCGAIN$和ADCOFFSET，即可实现该 CH 校准后的输出。也可将计算出的$ADCGAIN$和ADCOFFSET写入到所有 CH 的$ADCGAIN$和ADCOFFSET，即可实现所有 CH 校准后的输出。

由于，

$$ADCGAIN = 32768 * \frac{1 * (\frac{3.3}{3.657143} * 4096) * 2^7}{Vout2 - Vout1}$$

因此$ADCGAIN$的测定依赖于 3.3V 电源的精度，如果其向下偏移到 3.29V，$Vout2$偏低，$ADCGAIN$的测定会偏大，反之$ADCGAIN$的测定会偏小。

Example Code

```
#define REF3V3_CODE      15502147584  // Ideal code for 3.3V input related to
+-3.657143V range. 3696(ideal_code)*32768*128

int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("Enter the test\n");

    old_adc_sha_offset = ADC_GetSHOffset(ADC,ADC_SH0);
    old_adc_sha_gain = ADC_GetSHGain(ADC,ADC_SH0);

    printf("old_adc_sha_offset %d\n", old_adc_sha_offset);
    printf("old_adc_sha_gain %d\n", old_adc_sha_gain);

    printf("\n************** ADC Calibration Test Start *****************\n");

    /* Power up ADC */
    ADC_PowerUp(ADC);

    /* Div ADC clk, the more lower clock the more accurate the sample
result ,now the clock frequency = systemclk / 2*/
    CLOCK_SetModuleDiv(ADC_MODULE, 2);
    UART_Init(UART0, 38400);

    /* Enable ADC Interrupt Flag */
    ADC_EnableChannelInt(ADC, ADC_CH0);
    ADC_EnableChannelInt(ADC, ADC_CH1);

    Delay_Us(100);

    ADC_EnableChannelFractionalResult(ADC, ADC_CH0);

    /* CH0, Set sample time and conversion time */
    ADC_SetSampleAndConvertTime(ADC, ADC_CH0, ADC_DEFAULT_SAMPLE_TIME_NS,
ADC_DEFAULT_CONVERSION_TIME_NS);

    /* trigger source selection */
    ADC_SetChannelSOCTriggerSource(ADC, ADC_CH0,
ADC_SOC_TRIGGER_FROM_SOFTWARE);

    /* Set averaging 16 times */
    ADC_SetChannelResultAverageCount(ADC, ADC_CH0, ADC_AVERAGE_COUNT_16);

    /* positive terminal select AGND, negative terminal select AGND */
```

```
    ADC_SetChannelSHPositiveInput(ADC, ADC_CH0, ADC_SH0_P_GND);
    ADC_SetChannelSHNegativeInput(ADC, ADC_CH0, ADC_SH0_N_GND);

    /* sampler enable */
    ADC_SetChannelSH(ADC, ADC_CH0, ADC_SH_SEL_0);

    /* clear INT0 */
    ADC_ClearChannelInt(ADC, ADC_CH0);

    /* Software trigger CH0 */
    ADC_ForceChannelSOC(ADC, ADC_CH0);

    /* wait for ADC conversion done */
    while (ADC_GetChannelIntFlag(ADC, ADC_CH0) == 0);

    ADC_EnableChannelFractionalResult(ADC, ADC_CH1);

    /* CH1, Set sample time and conversion time */
    ADC_SetSampleAndConvertTime(ADC, ADC_CH1, ADC_DEFAULT_SAMPLE_TIME_NS,
ADC_DEFAULT_CONVERSION_TIME_NS);

    /* trigger source selection */
    ADC_SetChannelSOCTriggerSource(ADC, ADC_CH1,
ADC_SOC_TRIGGER_FROM_SOFTWARE);

    /* Set averaging 16 times */
    ADC_SetChannelResultAverageCount(ADC, ADC_CH1, ADC_AVERAGE_COUNT_16);

    /* positive terminal select ANA_IN0(3.3V), negative terminal select
ANA_IN1(0V) */
    ADC_SetChannelSHPositiveInput(ADC, ADC_CH1, ADC_SH0_P_ANA_IN0);
    ADC_SetChannelSHNegativeInput(ADC, ADC_CH1, ADC_SH0_N_ANA_IN1);

    /* sampler enable */
    ADC_SetChannelSH(ADC, ADC_CH1, ADC_SH_SEL_0);

    /* clear INT1 */
    ADC_ClearChannelInt(ADC, ADC_CH1);

    /* Software trigger CH1 */
    ADC_ForceChannelSOC(ADC, ADC_CH1);

    /* wait for ADC conversion done */
    while (ADC_GetChannelIntFlag(ADC, ADC_CH1) == 0);

    printf("Measure AGND  Code is %d \n", ADC->ADCRESULT[0]);
    printf("Measure VDD33 Code is %d \n", ADC->ADCRESULT[1]);

    adc_sha_offset = ADC->ADCRESULT[0];
    adc_sha_gain = REF3V3_CODE / (ADC->ADCRESULT[1] - ADC->ADCRESULT[0]);


    printf("ADC Offset: %d \n", adc_sha_offset);
    printf("ADC Gain:   %d \n", adc_sha_gain);

    printf("\n************** ADC Calibration Test End ******************\n");

    printf("offset change percent %f %%\n", (float)(adc_sha_offset -
old_adc_sha_offset)/old_adc_sha_offset*100);
    printf("gain change percent %f %%\n", (float)(adc_sha_gain -
old_adc_sha_gain)/old_adc_sha_gain*100);
    while (1)
    {
```

```
        }
}
```

2 PGA 校准

PGA 设计指标，如表 2-1，表 2-2 所示，如无特殊需要，不需要再次校准。

注意：表格中的偏移指的都是输入失调电压。其为$\frac{输出失调电压}{PGA放大倍数}$。

表 2-1：差分可编程增益放大器（DPGA）特性

符号	参数	条件	最小	典型	最大	单位
V_{in}	差分输入电压范围		-2.7/G	-	+2.7/G	V
V_{out}	输出电压范围	-	0.3	-	V_{DVDD33}-0.3	V
R_{in}	输入阻抗	-	-	4	-	kΩ
G	增益	差分模式	2, 4, 8, 16, 24, 32, 48, 64			-
E_{gain}	增益误差	差分增益= 2	-1	-	1	%
		差分增益= 16	-1	-	1	%
		差分增益= 64	-1	-	1	%
V_{offset}	偏移	差分增益= 2	-3	-	3	mV
		差分增益= 16	-1.3		+1.3	mV
		差分增益= 64	-1		+1	mV
V_{CM}	共模输入电压范围	差分增益= 2	-1.5	-	2	
		差分增益= 4	-0.9	-	2	V
		差分增益= 8	-0.75	-	2	V
		差分增益= 16	-0.6	-	2	V
		差分增益= 24	-0.57	-	2	V
		差分增益= 32	-0.55	-	2	V
		差分增益= 48	-0.52	-	2	V
		差分增益= 64	-0.5	-	2	V
t_{settle}	建立时间[1]	差分增益= 2	-	313.7	441.6	ns
		差分增益= 4		303.1	425.5	ns
		差分增益= 8		268.4	423.8	ns
		差分增益= 16		376.7	604.7	ns
		差分增益= 24		355.8	598.9	ns
		差分增益= 32		442.9	742.3	ns
		差分增益= 48		628.0	1061.0	ns
		差分增益= 64	-	813.3	1378.0	ns
GBW	单位增益带宽[2]	差分增益= 2	6	9.68	-	MHz
		差分增益= 4	3.8	6.07	-	MHz
		差分增益= 8	2.56	3.45	-	MHz
		差分增益= 16	1.37	1.85	-	MHz
		差分增益= 24	1.23	1.63	-	MHz
		差分增益= 32	0.93	1.23	-	MHz
		差分增益= 48	0.63	0.83	-	MHz
		差分增益= 64	0.4	0.63	-	MHz

SR	压摆率[3]	差分增益= 2	15	21	30	V/us
		差分增益= 4	15	21	30	V/us
		差分增益= 8	14.5	21	30	V/us
		差分增益= 16	11.2	18.9	29.7	V/us
		差分增益= 24	12.57	21	33.4	V/us
		差分增益= 32	9.25	18.4	31.8	V/us
		差分增益= 48	6.1	11.5	25	V/us
		差分增益= 64	4.6	8.6	17.6	V/us
$ENOB_{DC}$	有效位数（直流输入）	差分增益= 2	-	11.93	-	bits
		差分增益= 16	-	11.57	-	bits
		差分增益= 64	-	10.94	-	bits
SNR	信噪比 f_{in} = 10kHz	差分增益= 2	-	72.2	-	dB
		差分增益= 16	-	71.1	-	dB
		差分增益= 64	-	64.8	-	dB
THD	总谐波失真 f_{in} = 10kHz	差分增益= 2	-	79.59	-	dB
		差分增益= 16	-	81.29	-	dB
		差分增益= 64	-	78.47	-	dB
$CMRR_{DC}$	共模抑制比（直流输入）	差分增益= 2	-	-59.7	-	dB
		差分增益= 16		-52.6	-	dB
		差分增益 = 64	-	-61	-	dB
$PSRR_{DC}$	电源抑制比（直流输入）	差分增益 = 2	-	-72.8	-	dB
		差分增益 = 16	-	-84.9	-	dB
		差分增益 = 64	-	-93.4	-	dB
I_{on}	电流消耗	差分增益 = 2	-	1.215	-	mA
		差分增益 = 16	-	1.096	-	mA
		差分增益 = 64	-	1.152	-	mA

（1）建立时间指阶跃输入到输出建立至 98%的时间。此时对应的差分输出从-2.7V 到 2.7V（V_{DVDD33} = 3.3V）。该指标由设计保证。
（2）GBW 数据由设计保证。
（3）SR 数据是指输出信号从 10%建立至 90%的压摆率，由设计保证。

表 2-2：单端可编程增益放大器（SPGA）特性

符号	参数	条件	最小	典型	最大	单位
V_{in}	输入电压范围	-	0.3	-	V_{DDA}-1.3	V
V_{out}	输出电压范围	-	0.3	-	V_{DDA}-0.3	V
R_{in}	输入阻抗	-	-	High-Z	-	Ω
G	增益	单端	1, 2, 4, 8, 16, 32, 48, 64			-
E_{gain}	增益误差	增益 = 2	-1	-	1	%
		增益 = 32	-1	-	1	%
		增益 = 64	-2	-	2	%
V_{offset}	偏移	-	-5	-	5	mV
t_{settle}	建立时间	增益 = 2	-	300	-	ns
		增益 = 32		1000	-	ns
		增益 = 64	-	2000	-	ns
ENOB	有效位数	增益 = 2	-	11	-	bits
		增益 = 32	-	9	-	bits

		增益 = 64	-	8	-	bits
I_{on}	电流消耗	-	-	0.5	-	mA

2.1 增益以及输入失调电压

PGA 的偏差模型和 ADC 一致，同样可以用$V_{out} = K(V_{in} + V_{os})$表示，同时 PGA 和 ADC 是串联关系，PGA 的输出作为 ADC 的输入，因此，串联系统的偏差模型为：

$$V_{pga\_out} = K_{pga} * (V_{pga\_in} + V_{pga\_os})$$

$$V_{adc\_out} = K_{adc} * (V_{adc\_in} + V_{adc\_os})$$

$$V_{adc\_in} = V_{pga\_out}$$

其中K_{pga}，K_{adc}为增益，V_{pga_os}，V_{adc_os}为输入失调电压，V_{pga_out}，V_{adc_out}为输出电压，V_{pga_in}，V_{adc_in}为输入电压。

整理得：

$$V_{adc\_out} = K_{adc} * K_{pga} * V_{pga\_in} + K_{adc} * K_{pga} * V_{pga\_os} + K_{adc} * V_{adc\_os}$$

其中K_{pga}，K_{adc}增益和V_{pga_os}，V_{adc_os}输入失调电压均受到芯片个体差异以及温度影响，K_{pga}还单独受到软件设定的放大倍数影响（2X, 4X, 8X…）

但在芯片给定，温度给定，PGA 放大倍数给定的条件下，K_{pga}，K_{adc}增益和V_{pga_os}，V_{adc_os}输入失调电压均为常量，令：

$$V_{adcpga\_offset} = K_{adc} * K_{pga} * V_{pga\_os} + K_{adc} * V_{adc\_os}$$

得：

$$V_{adc\_out} = K_{adc} * K_{pga} * V_{pga\_in} + V_{adcpga\_offset}$$

符合线性方程的基本形式，从而 PGA 和 ADC 的串联系统可以被校准。

2.2 增益以及输出失调电压校准电路

在 PGA 和 ADC 的串联系统未校准时，实际 ADC 的采样结果为：

$$V_{adc\_out} = K_{adc} * K_{pga} * V_{pga\_in} + V_{adcpga\_offset}$$

V_{pga_in}表示为：

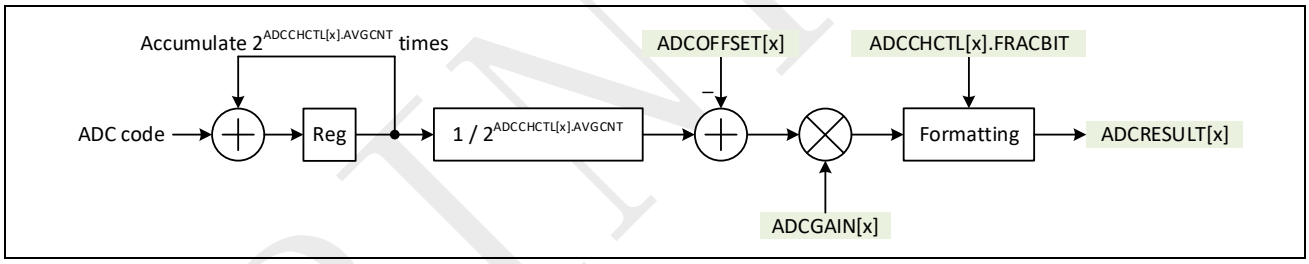
$$\left(V_{adc\_out} - V_{adcpga\_offset}\right) * \frac{1}{K_{adc} * K_{pga}} = V_{pga\_in}$$

$$\left(V_{adc\_out} - V_{adcpga\_offset}\right) * \frac{K_{pga理论}}{K_{adc} * K_{pga}} = V_{pga_{in}} * K_{pga理论}$$

也就是将 ADC 的采样结果V_{adc_out}减去输出失调电压V_{adcpga_offset}再乘以$\frac{K_{pga理论}}{K_{adc}*K_{pga}}$，就可以得到$V_{pga_{in}} * K_{pga理论}$，如图 2-1 所示。

注意：校准电路减去的是输出失调电压V_{adcpga_offset}，既不是 PGA 输入失调电压V_{pga_os}，也不是 ADC 输入失调电压V_{adc_os}。

图 2-1：ADC 码值处理



注意：ADCOFFSET（V_{adcpga_offset}）和 ADCGAIN（$\frac{K_{pga理论}}{K_{adc}*K_{pga}}$）都是按照定点数存储的。

其中 ADCOFFSET 末 7 位对应小数部分，因此在校准前，可以使能对应 CH 的 FRACBIT 位，从而保持 ADCRESULT（V_{adc_out}）与 ADCOFFSET（V_{adcpga_offset}）两个定点数小数点位的一致性，从而可以免去计算过程中的移位操作；

ADCGAIN 对应 VAL 为$32768 * \frac{K_{pga理论}}{K_{adc}*K_{pga}}$。

2.3 校准过程

根据章节 2.1，PGA 和 ADC 串联系统的校准是在芯片给定，温度给定，PGA 放大倍数给定的条件下进行的，校准时要控制变量。

假设 PGA 的放大倍数这里选为 4。

校准前一定要确保采样通道 CH 对应的 ADCOFFSET 为0，对应的$ADCGAIN$为32768。即原校准电路不起任何作用。

可以使能对应 CH 的 FRACBIT 位，从而保持 ADCRESULT（V_{adc_out}）与 ADCOFFSET（V_{adcpga_offset}）两个定点数小数点位的一致性，从而可以免去计算过程中的移位操作。

将 PGA 正负端都接地($V_{pga\_in1} = 0$)，得到V_{adc_out1}；

$V_{pga_{in2}}$的电压要尽量靠近最大输入，否则$ADCGAIN$的计算会有偏差，在 4 倍放大系数的条件下，SPGA 可以选到 0.7V 左右，DPGA 可以选到 0.6V 左右，如表 2-3，表 2-4 所示。

表 2-3：给定增益下 SPGA 输入输出范围

SPGACTL.GAIN	GAIN	Input range (V)			Output range (V)
0	1	0.3	~	2	0.3~2
1	2	0.15	~	1.5	0.3~3
2	**4**	**0.075**	**~**	**0.75**	**0.3~3**
3	8	0.0375	~	0.375	0.3~3
4	16	0.01875	~	0.1875	0.3~3
5	32	0.009375	~	0.09375	0.3~3
6	48	0.00625	~	0.0625	0.3~3
7	64	0.0046875	~	0.046875	0.3~3

表 2-4：给定增益下 DPGA 输入输出范围

DPGACTL.GAIN	$Gain_{DPGA}$	Input range (V)	Output range (V)
0	2	-1.35~1.35	0.3~3
1	**4**	**-0.675~+0.675**	**0.3~3**
2	8	-0.337~+0.337	0.3~3
3	16	-0.168~+0.168	0.3~3
4	24	-0.112~+0.112	0.3~3
5	32	-0.084~+0.084	0.3~3
6	48	-0.056~+0.056	0.3~3
7	64	-0.042~+0.042	0.3~3

下面以 SPGA 为例，进行校准演示

将 PGA 正端接0.7V，负端接地($V_{pga_{in2}} = 0.7V$)，得到V_{adc_out2}；

ADCOFFSET计算如下：

将$V_{pga_in1} = 0$带入$V_{adc_out1} = K_{adc} * K_{pga} * V_{pga_in1} + V_{adcpga_offset}$

得：

$$V_{adc\_out1} = V_{adcpga\_offset}$$

因为使能对应 CH 的 FRACBIT 位，V_{adc_out1}末 7 位对应小数，从而直接满足 ADCOFFSET 的定义，所以

$$\text{ADCOFFSET} = V_{adc\_out1}$$

否则，V_{adc_out1}要左移 7 位才能符合 ADCOFFSET 的定义。

$ADCGAIN$计算如下：

$$ADCGAIN = 32768 * \frac{K_{pga理论}}{K_{adc} * K_{pga}}$$

$$ADCGAIN = 32768 * \frac{K_{pga理论} * V_{pga\_in2}}{K_{adc} * K_{pga} * V_{pga\_in2}}$$

将$V_{adc_out2} - V_{adcpga_offset} = K_{adc} * K_{pga} * V_{pga_in2}$，$V_{adc\_out1} = V_{adcpga\_offset}$

代入得：

$$ADCGAIN = 32768 * \frac{K_{pga理论} * V_{pga\_in2}}{V_{adc\_out2} - V_{adc\_out1}}$$

而$V_{pga_in2} = (\frac{0.7}{3.657143} * 4096) * 2^7$，代入得：

$$ADCGAIN = 32768 * \frac{4 * (\frac{0.7}{3.657143} * 4096) * 2^7}{V_{adc\_out2} - V_{adc\_out1}}$$

最后，将计算出的$ADCGAIN$和ADCOFFSET写入到 PGA 和 ADC 串联系统对应 CH 的$ADCGAIN$和ADCOFFSET，即可实现 PGA 和 ADC 串联系统的校准。

注意：校准后的 CH 不能再用于单独的 ADC 采样。不要往任何其他 CH 写校准值，否则会损失其他 CH ADC 的采样精度。

由于

$$ADCGAIN = 32768 * \frac{4 * (\frac{0.7}{3.657143} * 4096) * 2^7}{V_{adc\_out2} - V_{adc\_out1}}$$

因此$ADCGAIN$的测定依赖于0.7V 电源的精度，如果其向下偏移到 0.69V，V_{adc_out2}偏低，$ADCGAIN$的测定会偏大，反之$ADCGAIN$的测定会偏小。

SPGA 校准过程如下：

- 初始状态设定 4 倍放大系数；

- 将 PGA 正负端都接地($V_{pga\_in1} = 0$)，得到V_{adc_out1}；
- 将 PGA 正端接0.7V，负端接地($V_{pga_{in2}} = 0.7V$)，得到V_{adc_out2}；
- 按照公式计算ADCOFFSET与$ADCGAIN$。

Example Code

```
int main(void)
{
    uint32_t j;

    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("\n\n************** SPGA Calibration Test Start
******************\n");

    ADC_EnableChannelFractionalResult(ADC, ADC_CH1);

    /* Init SPGA mode, signal from ANA_IN0 */
    PGA_InitSPGA(SPGA_FROM_ANA_IN0, SPGA_GAIN_4X);

    PIN_SetChannel(PIN_GPIO0, PIN_GPIO0_ANA_IN0);

    /* ADC Init, collect the signal from SPGA */
    ADC_EasyInit1(ADC, ADC_CH1, ADC_IN_SPGA_OUT,
ADC_SOC_TRIGGER_FROM_SOFTWARE);

    ADC_SetChannelGain(ADC,ADC_CH1, 32768);
    ADC_SetChannelOffset(ADC,ADC_CH1, 0);
    printf("old_adc_ch_offset %d\n", ADC_GetChannelOffset(ADC, ADC_CH1));
    printf("old_adc_ch_gain %d\n", ADC_GetChannelGain(ADC, ADC_CH1));

    /* Set sample and convert time */
    ADC_SetSampleAndConvertTime(
        ADC,
        ADC_CH1,
        8000U,
        ADC_DEFAULT_CONVERSION_TIME_NS
    );

    ADC_SetChannelResultAverageCount(ADC,ADC_CH1,ADC_AVERAGE_COUNT_128);


    for(j = 0; j < 5; j++)
    {
        Delay_Ms(1);

        /* Use software to trigger ADC CH1 to work */
        ADC_ForceChannelSOC(ADC, ADC_CH1);
```

```
        /* Wait until ADC conversion finished (Interrupt flag is set) */
        while (ADC_GetChannelIntFlag(ADC, ADC_CH1) == 0);

        /* Get the ADC CH result */
        i32result_out = (int32_t)(READ_REG( (ADC)->ADCRESULT[ADC_CH1] ));

        /* Clear channel Int */
        ADC_ClearChannelInt(ADC, ADC_CH1);

        printf("The out is %d\n", i32result_out);
    }

    while (1)
    {
    }
}
```

DPGA 校准过程如下：

- 初始状态设定 4 倍放大系数；
- 将 PGA 正负端都接地($V_{pga\_in1} = 0$)，得到V_{adc_out1}；
- 将 PGA 正端接0.7V，负端接地($V_{pga_{in2}} = 0.7V$)，得到V_{adc_out2}；
- 按照公式计算ADCOFFSET与$ADCGAIN$。

Example Code

```
int main(void)
{
    uint32_t j;

    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("\n\n*************** DPGA Calibration Test Start
******************\n");

    ADC_EnableChannelFractionalResult(ADC, ADC_CH1);
    ADC_EnableChannelFractionalResult(ADC, ADC_CH0);

    /* Init DPGA mode */
    PGA_InitDPGA(DPGA_GAIN_4X);

    /* Differential ADC Init, collect the signal from DPGAP and DPGAN */
    ADC_EasyInit2(ADC, ADC_CH1, ADC_IN_DPGAP_OUT, ADC_IN_DPGAN_OUT,
ADC_SOC_TRIGGER_FROM_SOFTWARE);
```

```
    ADC_SetChannelOffset(ADC, ADC_CH1, 0);
    ADC_SetChannelGain(ADC, ADC_CH1, 32768);

    printf("ChannelOffset %d\n", ADC_GetChannelOffset(ADC, ADC_CH1));
    printf("ChannelOffset %d\n", ADC_GetChannelGain(ADC, ADC_CH1));

    /* Set sample and convert time */
    ADC_SetSampleAndConvertTime(
        ADC,
        ADC_CH1,
        8000U,
        ADC_DEFAULT_CONVERSION_TIME_NS
    );

    ADC_SetChannelResultAverageCount(ADC,ADC_CH1,ADC_AVERAGE_COUNT_128);

    for(j = 0; j < 5; j ++)
    {
        PGA_DisableDPGABypass();
        Delay_Ms(10);

        /* Use software to trigger ADC CH1 to work */
        ADC_ForceChannelSOC(ADC, ADC_CH1);

        /* Wait until ADC conversion finished (Interrupt flag is set) */
        while (ADC_GetChannelIntFlag(ADC, ADC_CH1) == 0);

        /* Get the ADC CH result */
        i32result_out = ((int32_t)(READ_REG( (ADC)->ADCRESULT[ADC_CH1] )));

        /* Clear channel Int */
        ADC_ClearChannelInt(ADC, ADC_CH1);


        printf("The out is %d\n", i32result_out);
    }

    while (1)
    {
    }
}
```