

概述

适用范围	
SPC1125 系列	SPC1125, SPC1128, SPD1121
SPC1168 系列	SPC1155, SPC1156, SPC1158, SPC1168, SPD1148, SPD1178, SPD1188, SPD1163, SPM1173
SPC2168 系列	SPC2168, SPC2165, SPC2166, SPC1198
SPC1169 系列	SPC1169, SPD1179, SPD1176, SPD1177, SPD1179B
SPC2188 系列	SPC1185, SPC2188
SPC1198B 系列	SPC1198B

目录

1	概述	6
2	两点校准法	6
3	ADC 校准.....	8
4	PGA 与 ADC 系统校准	12
4.1	输入电压选择	12
4.1.1	PGA 输入电压选择	12
4.1.2	DPGA 输入电压选择	13
4.1.3	SPGA 输入电压选择	14
4.2	校准过程	15

图片列表

图 2-1: 两点法校准示例	6
图 3-1: ADC 校准过程图示	8

SPIN TROL

表格列表

表 4-1: SPC1168 系列给定增益下 PGA 双端模式时输入电压选择范围	13
表 4-2: SPC2168 系列给定增益下 PGA 双端模式时输入电压选择范围	13
表 4-3: SPC1128 给定增益下 DPGA 输入输出范围	14
表 4-4: SPC1169 系列给定增益下 DPGA 输入输出范围	14

版本历史

版本	日期	作者	状态	变更
A/0	2023-05-06	Hang Su	已过期	1. 首次发布。
C/0	2024-08-23	C.Chai	已过期	1. 修改为全系列通用文档。
C/1	2025-03-31	LemengZhou	已发布	1. 适用范围增加。 2. 添加对 SPC1198B 系列的描述。

1 概述

在实际应用中，为了补偿偏移和增益误差，我们需要进行两点校准。但 Spintrol 产品出厂时都会校准 ADC，其校准值写入到 Flash 的 NVR 扇区或者 OTP 扇区中，并在芯片启动的时候通过 Boot 程序自动加载到 ADCSH 对应寄存器 SHOFFSET 和 SHGAIN 中，换言之，Spintrol 产品在出厂之初 ADC 的精度已被校准到设计所要求的精度。

2 两点校准法

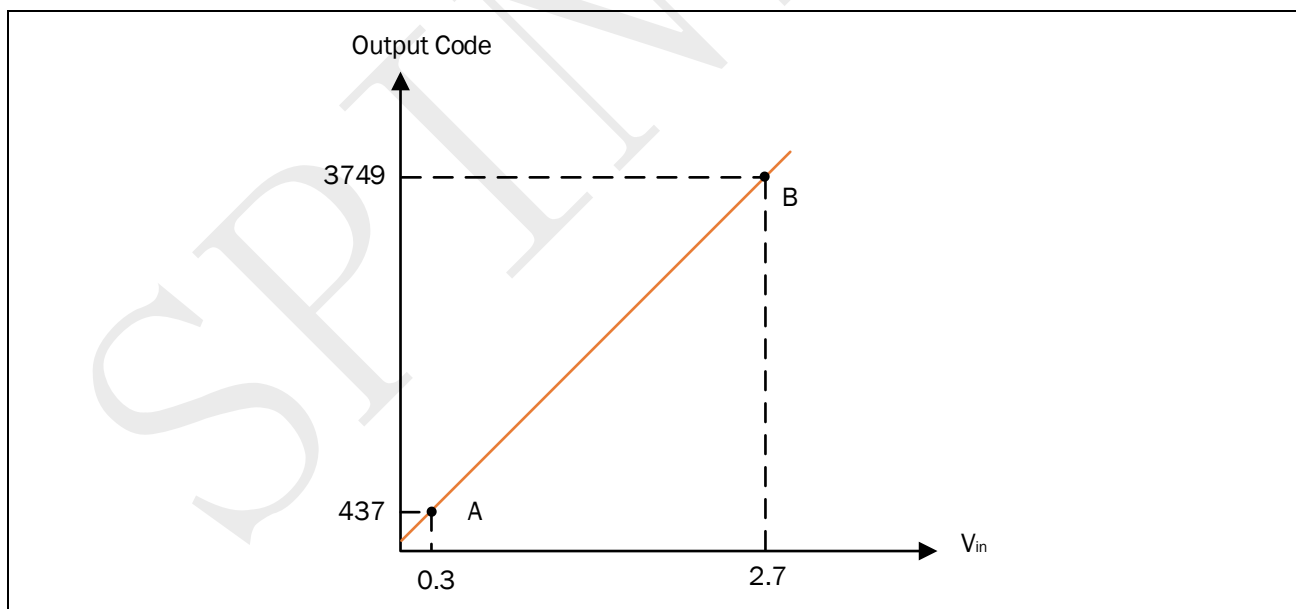
实际中 ADC 的输入与输出是线性关系，用数学表达式可以简单表示为：

$$y = KV_{in} + b$$

其中 y 可以理解为 ADC 的输出码值， K 可以理解为 Gain， V_{in} 为输入 ADC 的电压值， b 为 ADC Offset。对于 ADC 的 Offset 以及 Gain Error 校准过程，即为求解函数表达式中的 K 以及 b 的过程。很容易知道，使用两点法即可。

举例来说，如图 2-1 所示，展示了一个 12 位 ADC 输入的测量电压与输出码值之间的关系，且假设其满量程为 3.3V。为了测量这个 ADC 的 Offset 以及 Gain Error，我们选取 A，B 两点，假设 A，B 两点的输入至 ADC 的电压分别 0.3V，2.7V，且此时用此 ADC 测量出来的码值分别为 437，3749。

图 2-1：两点法校准示例



从图中可以看出，若使用测量数据计算函数斜率，其值为：

$$K_m = \frac{\text{Code}_B - \text{Code}_A}{V_{in, B} - V_{in, A}} = \frac{3749 - 437}{2.7 - 0.3} = 1380$$

由此斜率及 A 点的坐标值，可以知道此 ADC 任何输入电压下的输出码值为：

$$y_m = K_m(V_{in} - 0.3) + 437 = 1380(V_{in} - 0.3) + 437$$

若设 $V_{in} = 0$ ，则可计算出此时 ADC 的 Offset 为+23LSB。

在继续计算 ADC 的 Gain Error 之前，首先我们需要明白，理想 ADC 码值的计算模型为：

$$y_m = \frac{V_{in}}{FS} * 2^N$$

如上文假设所述，此 ADC 的位数为 12，且满量程为 3.3V，则可知 0.3V，2.7V 输入在理想情况下输出的码值应该分别为：409，3686，利用这两个理想输出码值，我们可以另外计算出理想斜率 K_i 为：

$$K_i = \frac{Code_{B,i} - Code_{A,i}}{V_{in,B} - V_{in,A}} = \frac{3686 - 409}{2.7 - 0.3} = 1365.42$$

现在我们可以计算 ADC 的 Gain Error：

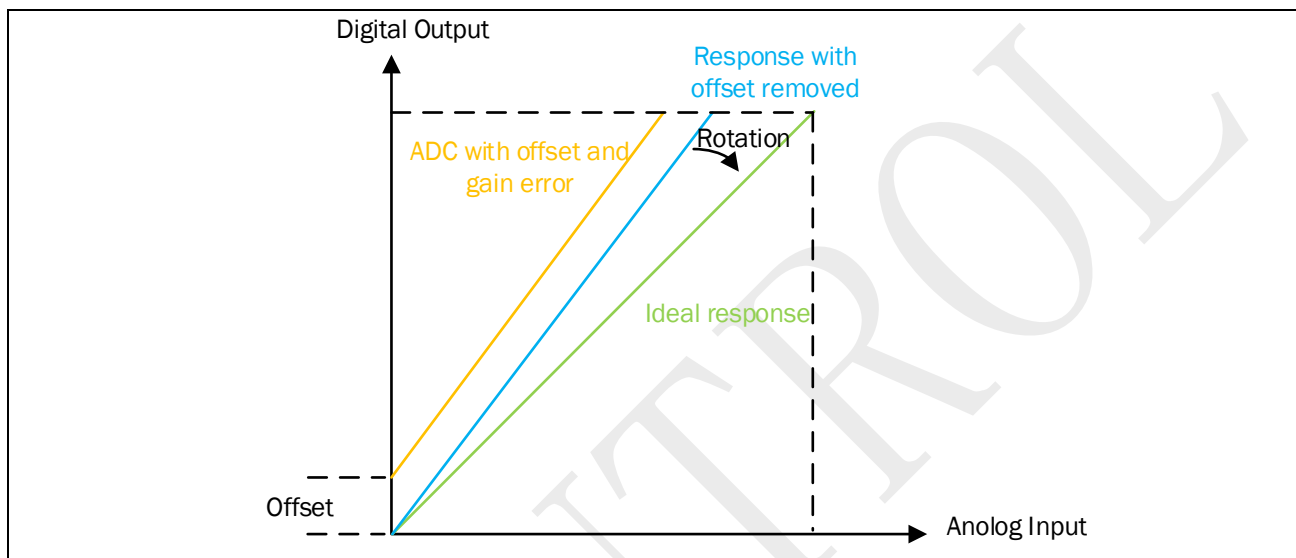
$$Gain Error = \frac{K_m - K_i}{K_i} = \frac{1380 - 1365.42}{1365.42} = 1.1\%$$

3 ADC 校准

若用户希望放弃原厂校准的数据，想通过自己的实验对 ADC 进行校准，用户可参考本文探讨的两点校准方法，并通过例子了解该技术的实现。

了解如[章节 2](#) 所述的实际响应后，现在可以轻松地在数字域中修正偏移和增益误差，[图 3-1](#) 可以直观的展示章节所述的整个过程。

图 3-1: ADC 校准过程图示



图中绿色线为 ADC 理想码值响应函数，黄色线为实际中 ADC 的码值响应函数，整个校准过程可以描述为：

1. 首先，从每个输出码中减去偏移量，以得到一个经过原点且斜率为 K_m 的响应（即图中蓝色响应函数）；
2. 接下来，将结果乘以 $\frac{K_i}{K_m}$ 会将得到的直线绕原点旋转，并生成一条斜率为 K_i 的直线（即图中绿色响应函数）。

Spintronic 产品 ADC 校准的具体实验操作描述如下：

1. 将 ADC 通道 CH 对应的 ADCOFFSET、ADCGAIN 设置为默认值；
2. 将 ADC 正负端都接地($V_{in1} = 0$)，得到 $Code_{out1}$ ；
3. 将 ADC 正端接 3.3V，负端接地($V_{in2} = 3.3V$)，得到 $Code_{out2}$ ；

如[章节 2](#) 所述，将 (V_{in1} , $Code_{out1}$) 以及 (V_{in2} , $Code_{out2}$) 代入到线性方程 $y = KV_{in} + b$ 可得此 Spintronic ADC 的响应函数为：

$$y_m = K_m(V_{in} - V_{in1}) + Code_{out1}, \text{ 其中 } K_m = \frac{Code_{out2} - Code_{out1}}{V_{in2} - V_{in1}}$$

将 $V_{in1} = 0$ 代入 ADC 影响方程可求得 ADC Offset:

$$\text{Offset} = \text{Code}_{\text{out1}}$$

- 当 ADC 无 FRACBIT 控制位时, $\text{ADCOFFSET} = \text{Code}_{\text{out1}}$;
- 当 ADC 有 FRACBIT 控制位, 且 FRACBIT 控制位为 0 时, $\text{ADCOFFSET} = \text{Code}_{\text{out1}}$;
- 当 ADC 有 FRACBIT 控制位, 且 FRACBIT 控制位为 1 时, $\text{ADCOFFSET} = \text{Code}_{\text{out1}} \ll 7$ 。

如章节 2 所述, ADC Gain Error 计算方式为:

$$K_i = \frac{\text{Code}_{\text{out2}, i} - \text{Code}_{\text{out1}, i}}{V_{\text{in2}} - V_{\text{in1}}}$$

在 Gain Error 的计算中, 在此选取 Spintrol SPD1179 产品 ADC 为例来进行说明。SPD1179 内置 13 位 ADC, 满量程为 3.657V。参考 SPD1179 TRM 手册中理想码值计算公式, 可以计算出实验操作中 $V_{\text{in1}} = 0$ 及 $V_{\text{in2}} = 3.3\text{V}$ 对应的理论数值为 (因为 SPD1179 ADC 带有 FRACBIT 控制位, 所以计算理论数值时需要分情况讨论):

- 当 FRACBIT=0 时, $\text{Code}_{\text{out2}, i} = 3696$, $\text{Code}_{\text{out1}, i} = 0$, 则 (此计算规则同样适用于 ADC 不带 FRACBIT 控制位的产品):

$$K_i = \frac{\text{Code}_{\text{out2}, i} - \text{Code}_{\text{out1}, i}}{V_{\text{in2}} - V_{\text{in1}}} = \frac{3696}{3.3} = 1120$$

- 当 FRACBIT=1 时, $\text{Code}_{\text{out2}, i} = 3696$, $\text{Code}_{\text{out1}, i} = 0$, 则:

$$K_i = \frac{\text{Code}_{\text{out2}, i} * 2^7 - \text{Code}_{\text{out1}, i} * 2^7}{V_{\text{in2}} - V_{\text{in1}}} = \frac{3696 * 2^7}{3.3}$$

如章节 2 所述, $\text{Gain Error} = \frac{K_m - K_i}{K_i}$, 但由于硬件设计上的不同, Spintrol 所有产品的 $\text{ADCGAIN} = \text{Gain Error} = \frac{K_m}{K_i} * 32786$ 。至此针对 ADC 的 Offset 以及 Gain Error 就已经校准完毕。最后, 将计算出数值分别写入到对应 CH 的 ADCGAIN 和 ADCOFFSET, 即可实现对该 CH 校准。

Example Code (以 SPD1179 为例)

```
#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);
```

Example Code (以 SPD1179 为例)

```

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);

```

Example Code (以 SPD1179 为例)

```
/* 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)
{
}
}
```

4 PGA 与 ADC 系统校准

在实际应用中，通常会将 PGA 的输出送到 ADC 进行转换，PGA 和 ADC 各自都有偏置和增益误差，此时就需要将 PGA 与 ADC 当成一个系统进行校准。

由于 PGA 和 ADC 的响应都是线性函数，所以通过一些简单的公式可以推算出 PGA 和 ADC 组成的系统中 ADC 的响应表达式。

由于，

$$V_{\text{pga_out}} = K_{\text{pga}} * V_{\text{pga_in}} + V_{\text{pga_offset}}$$

又由于，

$$V_{\text{adc_out}} = K_{\text{adc}} * V_{\text{adc_in}} + V_{\text{adc_offset}}$$

且，

$$V_{\text{adc_in}} = V_{\text{pga_out}}$$

整理可得：

$$V_{\text{adc_out}} = K_{\text{adc}} * K_{\text{pga}} * V_{\text{pga_in}} + K_{\text{adc}} * V_{\text{pga_offset}} + V_{\text{adc_offset}}$$

在给定温度条件下， K_{adc} 、 K_{pga} 、 $V_{\text{pga_offset}}$ 、 $V_{\text{adc_offset}}$ 均为常量，令：

$$V_{\text{adcpga_offset}} = K_{\text{adc}} * V_{\text{pga_offset}} + V_{\text{adc_offset}}$$

则，

$$V_{\text{adc_out}} = K_{\text{adc}} * K_{\text{pga}} * V_{\text{pga_in}} + V_{\text{adcpga_offset}}$$

从以上表达式可以看出，PGA 与 ADC 组成的系统与单独 ADC 一样，其响应也是类似的线性方程，可以使用两点校准法进行校准。

4.1 输入电压选择

在校准 PGA 和 ADC 组成的组合系统时，稍不注意所选择的 PGA 输入电压有可能会使得 PGA 的输出超过 PGA 的输出能力范围。为了避免在测试中发生这个问题，此章节将针对不同类型的 PGA 给出校准时输入电压的推荐值供用户参考。用户当然也可以自己自行选择输入电压，但是请谨记，PGA 的输出电压不要超过对应产品数据手册中给出的输出电压范围。

4.1.1 PGA 输入电压选择

在 Spintrol 产品中，SPC1168 平台及 SPC2168 平台产品具有 PGA，当 PGA 工作在单端模式下时，其输入电压选择较为简单，很容易就能选择一个不会导致超出输出范围的输入电压，在此就不对其做过多描述。但当 PGA 工作在双端时，此时想选择一个输入，且不要让 PGA 的输出不要超过范围，就需要通过一些计算，为了简便起见，提供如下表格供用户查询在不同 Gain 值下可选的输入电压范围。需要注意的是，表格中计算出来的数值是在共模端选为 N 端时计算得来，用户在实际中若按照表格选择输入电压，则也应该将共模端选择为 N 端。

表 4-1: SPC1168 系列给定增益下 PGA 双端模式时输入电压选择范围

PGAxCTL.GAINP/ PGAxCTL.GAINN	Gain _{PGA}	输入范围 (V)	输出范围 (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

[1] 表中输出范围 0.3~3 指的是 PGA 的 P 端和 N 端输出都应该在此范围。

[2] 表格中输入端的选择范围是基于工模端选择为 N 端计算。

表 4-2: SPC2168 系列给定增益下 PGA 双端模式时输入电压选择范围

PGAxCTL.GAINP (N)	增益	输入范围 (V)	输出范围 (V)
0	1	0.3~3	0.3~3
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	12	0.025~0.25	0.3~3
5	16	0.01875~0.1875	0.3~3
6	24	0.0125~0.125	0.3~3
7	32	0.009375~0.09375	0.3~3

[1] 表中输出范围 0.3~3 指的是 PGA 的 P 端和 N 端输出都应该在此范围。

[2] 表格中输入端的选择范围是基于工模端选择为 N 端计算。

输入电压要尽量靠近最大输入，否则ADCGAIN的计算结果偏差会较大，例如在 4 倍放大系数的条件下，输入电压应选择 0.6V 左右。

4.1.2 DPGA 输入电压选择

在 Spintrol 产品中，SPC1169 平台及 SPC1125 平台产品具有 PGA，当 PGA 工作在单端模式下时，其输入电压选择较为简单，很容易就能选择一个不会导致超出输出范围的输入电压，在此就不对其做过多描述。但当 PGA 工作在双端时，此时想选择一个输入，且不要让 PGA 的输出不要超过范围，就需要通过一些计算，为了简便起见，提供如下表格供用户查询在不同 Gain 值下可选的输入电压范围。需要注意的是，表格中计算出来的数值是在共模端选为 N 端时计算得来，用户在实际中若按照表格选择输入电压，则也应该将共模端选择为 N 端。

表 4-3: SPC1128 给定增益下 DPGA 输入输出范围

DPGACTL.GAIN	Gain _{DPGA}	输入范围 (V)	输出范围 (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

[1] 表中输出范围 0.3v~3v 指的是 PGA 的 P 端和 N 端输出都应该在此范围。

[2] 表格中输入端的选择范围是基于工模端选择为 N 端计算。

表 4-4: SPC1169 系列给定增益下 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

[1] 表中输出范围 0.3v~3v 指的是 PGA 的 P 端和 N 端输出都应该在此范围。

[2] 表格中输入端的选择范围是基于工模端选择为 N 端计算。

输入电压要尽量靠近最大输入，否则ADCGAIN的计算结果偏差会较大，例如在 4 倍放大系数的条件下，输入电压应选择 0.6V 左右。

4.1.3 SPGA 输入电压选择

对于 SPGA 而言，要选择一个不会让输出超出设计范围的输入值是较为简单直接的，在此就不再赘述。

4.2 校准过程

根据章节 0，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。

以 SPD1179 DPGA 为例，具体操作过程为：

1. 将 ADC 的 ADCOFFSET 及 ADCGAIN 寄存器设置为默认值；
2. 若产品 ADC 带有 FRACBIT 控制位，可将 FRACBIT 控制位使能，避免后续将最终校准数据写入 ADCOFFSET、ADCGAIN 寄存器时的移位操作；
3. 将 DPGA 正负端都接地($V_{pga_in1} = 0$)，得到 V_{adc_out1} ；
4. 将 DPGA 正端接 0.7V，负端接地($V_{pga_in2} = 0.7V$)，得到 V_{adc_out2} ；
5. 将 DPGA 共模端设置为负端。

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 FRACBIT 位使能，所以 V_{adc_out1} 末 7 位是小数部分，所以将计算结果直接填入 ADCOFFSET 寄存器即可：

$$ADCOFFSET = V_{adc_out1}$$

否则， V_{adc_out1} 要左移 7 位才能写入 ADCOFFSET 寄存器。

ADCGAIN 计算如下：

从章节 3 可知，在 Spintrol 的设计中，单独校准 ADC 时计算 ADC Gain Error 的公式为 $\text{Gain Error} = \frac{K_m}{K_i} * 32786$ ，从而可以推断出，在 ADC 与 PGA 组成的系统中，其 Gain Error 的计算公式为：

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

$$ADCGAIN = 32768 * \frac{K_{pga, i} * V_{pga_in2}}{K_{adc} * K_{pga} * V_{pga_in2}}$$

将章节 4 开始，推算过：

$$V_{adc_out} = K_{adc} * K_{pga} * V_{pga_in} + V_{adcpga_offset}$$

由此有, $V_{\text{adc_out2}} - V_{\text{adcpga_offset}} = K_{\text{adc}} * K_{\text{pga}} * V_{\text{pga_in2}}$

且有, $V_{\text{adc_out1}} = V_{\text{adcpga_offset}}$

代入得:

$$\text{ADCGAIN} = 32768 * \frac{K_{\text{pga}, i} * V_{\text{pga_in2}}}{V_{\text{adc_out2}} - V_{\text{adc_out1}}}$$

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

代入得:

$$\text{ADCGAIN} = 32768 * \frac{4 * (\frac{0.7}{3.657143} * 4096) * 2^7}{V_{\text{adc_out2}} - V_{\text{adc_out1}}}$$

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ADCGAIN和ADCOFFSET数值, 只能填入到对应的 CH 通道的对应 ADCGAIN和ADCOFFSET寄存器, 不要误写入其它 CH。
 - 从ADCGAIN的计算公式可以看出, 如果校准过程中, DPGA 输入电压 0.7V 需要较高的精度, 才能确保 ADCGAIN 的计算结果可靠。若输入的 0.7V 电压向下便宜到 0.69V, 则将导致 $V_{\text{adc_out2}}$ 结果偏低, ADCGAIN 结果将偏大。反之, ADCGAIN 的计算结果偏小。

以上操作步骤对应的代码操作步骤为:

- 初始状态设定 4 倍放大系数;
- 将 DPGA 正负端都接地($V_{\text{pga_in1}} = 0$), 得到 $V_{\text{adc_out1}}$;
- 将 DPGA 正端接0.7V, 负端接地($V_{\text{pga_in2}} = 0.7V$), 得到 $V_{\text{adc_out2}}$;
- 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
     */
```


Example Code

```

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("ChannelGain %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)
{
}

```

[1] 示例代码适用于 SPD1179，其它系列产品的示例代码会根据实际需求进行补充。