PGA 使用指南

概述

本手册适用范围：

适用范围	
SPC1125 系列	SPC1125，SPC1128
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
SPC2188 系列	SPC1185，SPC2188

目录

图片列表

表格列表

版本历史

版本	日期	作者	状态	变更
C/0	2024-08-13	Hang Su	Released	1. 首次发布。

术语或缩写

术语或缩写	描述
MCU	Microcontroller Unit，微控制器单元

1 SPC1168/SPC2168/SPC2188 系列

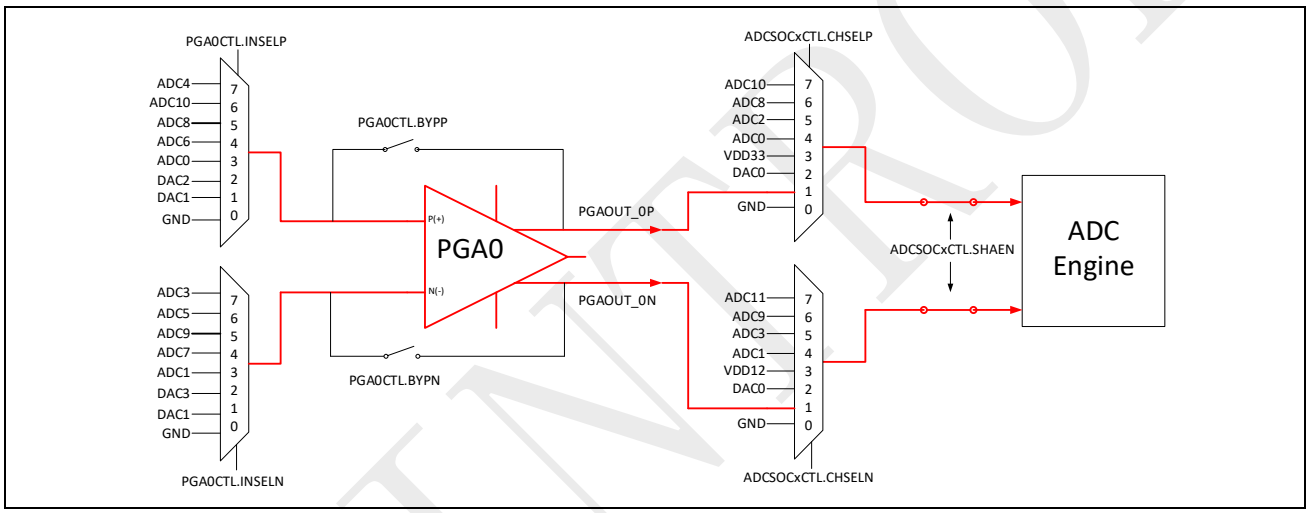
1.1 特性

每个 PGA 有两种工作模式：一种是差分模式，最大增益可到 64，另一种是单端模式，最大增益可到 32。在单端模式下，每个 PGA 可以支持两个单端信号的放大。

1.2 功能描述

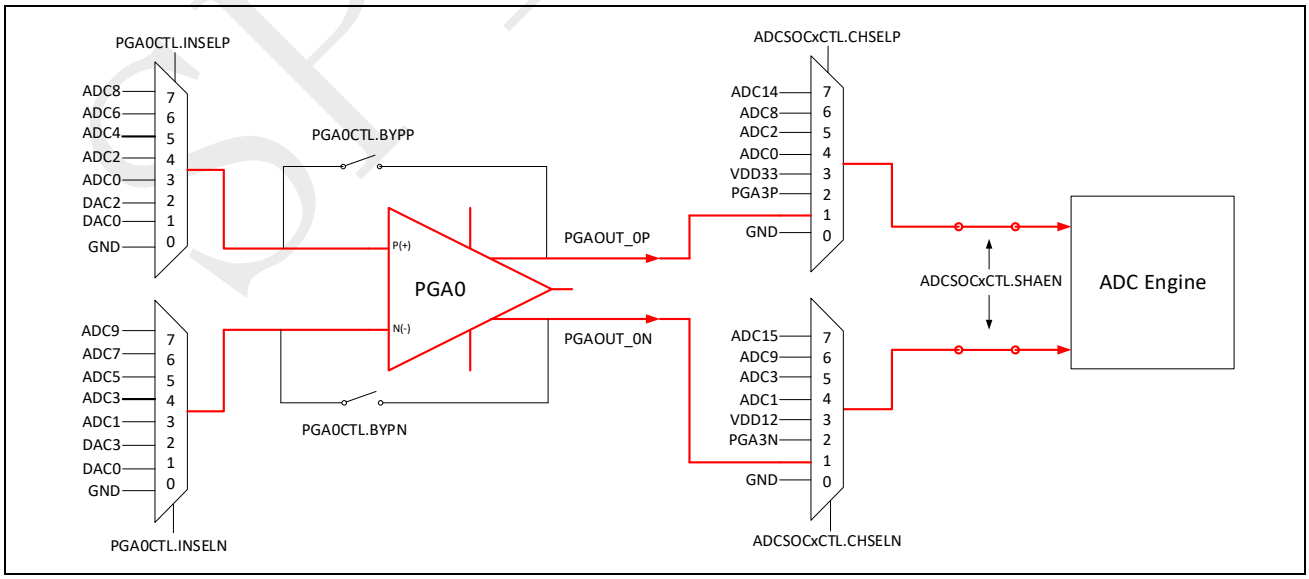
PGA0 结构框图如图 1-1 到图 1-3 所示。

图 1-1：PGA0 结构框图



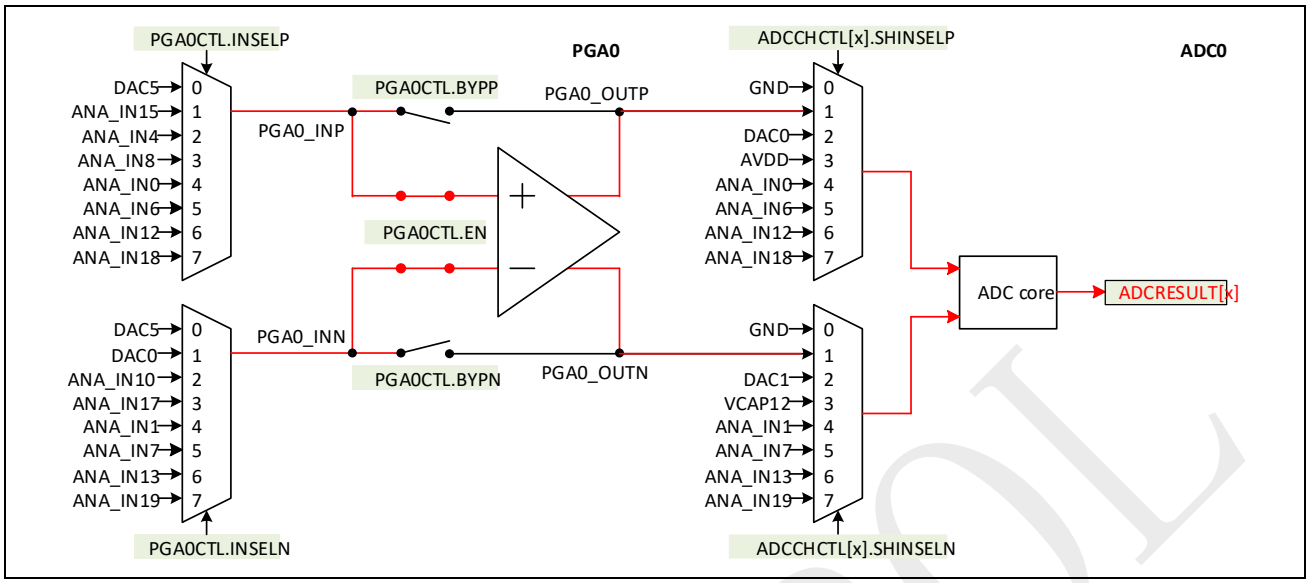
[1] SPC1168 PGA0 结构图。

图 1-2：PGA0 结构框图



[1] SPC2168 PGA0 结构图。

图 1-3：PGA0 结构框图



[1] SPC2188 PGA0 结构图。

1.3 功能实例

1.3.1 差分模式

1.3.1.1 功能需求

使用 ADC 对差分 PGA 电压进行采样。

1.3.1.2 功能实现

差分模式是指将 VPGAx_INP 与 VPGAx_INN 的差值作为输入，将 VPGAx_OUTP 与 VPGAx_OUTN 的差值作为输出。

$$V_{PGAx\_OUTP} - V_{PGAx\_OUTN} = (V_{PGAx\_INP} - V_{PGAx\_INN}) \times G$$

其中 VPGAx_OUTP 与 VPGAx_OUTN 各自输出为：

$$V_{PGAx\_OUTP} = V_{PGAx\_OUTCM} + (V_{PGAx\_INP} - V_{PGAx\_INN}) \times G / 2$$

$$V_{PGAx\_OUTN} = V_{PGAx\_OUTCM} - (V_{PGAx\_INP} - V_{PGAx\_INN}) \times G / 2$$

- VPGAx_INP 与 VPGAx_INN 本身电压必须为正（但是其差值可以为负）；
- VPGAx_OUTP 与 VPGAx_OUTN 电压必须为正。同时为了防止输出管饱和，使用时需要保证 VPGAx_OUTP 与 VPGAx_OUTN 电压在 0.3V 到 VAVDD - 0.3V 之间；
- VPGAx_OUTCM（输出共模电压）参照源取决于寄存器配置，参考对应芯片 TRM 文档 PGA 章节配置即可；
- 无论配置谁作为 VPGAx_OUTCM（输出共模电压）参照源，通常将其接入

$$\frac{0.3V + (VAVDD - 0.3V)}{2} = \frac{VAVDD}{2}$$

以获得最大的 VPGAx_OUTP 与 VPGAx_OUTN 电压范围；

- VPGAx_OUTP 与 VPGAx_OUTN 电压差在

$$\pm\big((\mathrm{VAVDD}-0.3\mathrm{V})-0.3\mathrm{V}\big)=\pm(\mathrm{VAVDD}-0.6\mathrm{V})$$

对应 VPGAx_INP 与 VPGAx_INN 电压差必须在$\frac{\pm(\mathrm{VAVDD}-0.6\mathrm{V})}{G}$范围内。

以下例子演示使用 ADC 对差分 PGA 进行采样，将差分 PGA 电压送给 ADC CH0，采用默认采样时间，默认转换时间。但在实际的工程中采样时间会受到外部电路的影响，其具体的设置数值，可参考《ADC 建立时间计算方法使用指南》，但转换时间则不会受到外部电路影响，通常保持 SDK 中设定的数值即可。

Example Code

```
/* As description below, ADC result convert to voltage can calculate as the
follow */
#define       ValueToVoltage(x)       ((x * 3657) / 8192) /1000

/*Variable for ADC result*/
int32_t       i32VSP;



int main()
{
    FLASH_WALLOW();

#if defined(SPC1158)
    FLASH_SetTiming(100000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_100MHZ);
#else
    FLASH_SetTiming(200000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);
#endif

    Delay_Init();

    /*
     * Init the UART
     *
     * 1.Set the GPIO34/35 as UART FUNC
     *
     * 2.Enable the UART CLK
     *
     * 3.Set the rest para
     */
    GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
    GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
    CLOCK_EnableModule(UART_MODULE);
    UART_Init(UART, 38400);

    /*
     * Set PGA in differential ended mode.
     *
     * 1.Set the GPIO8(ADC8) as the P-input of the PGA0.
```

Example Code

```
     *
     * 2.Set the GPIO9(ADC9) as the N-input of the PGA0.
     *
     * 3.Set the PGA0 amplitude scale as 4x.
     *
     * 4.Enable the PGA0.
     */
    GPIO_SetPinChannel(GPIO_8, GPIO8_ADC8);
    GPIO_SetPinChannel(GPIO_9, GPIO9_ADC9);
    PGA_DifferentialInit(PGA0, PGA0_CH_P_ADC8, PGA0_CH_N_ADC9, PGA_SCALE_4X);

    /*
     * ADC Init.
     *
     * 1.Set the ADC as differential mode.
     *
     * 2.Use SHA to sample the PGA0P and PGA0N which had been set as the P/N
channel.
     *
     * 3.Set software as the tigger model.
     */
    ADC_Init(ADC_SOC_0, ADC_SHA_P_PGA0P_OUT, ADC_SHA_N_PGA0N_OUT, SHA,
ADCTRIG_Software);

    while (1)
    {
        /* Use software to trigger ADC SOC0 to work */
        ADC_SoftwareTrigger(ADC_SOC_0);

        /* Wait until ADC conversion finished (Interrupt flag was set) */
        while (!ADC_GetIntFlag(ADC_SOC_0));

        /* Get trim result */
        i32VSP = ADC_GetTrimResult2(ADC_SOC_0);

        /* Clear ADC SOC0 INT flag */
        ADC_ClearInt(ADC_SOC_0);

        /* Print the difference voltage value of GPIO8/9 */
        printf("GPIO8/9 difference voltage is %2fV(Trim data is %d)\n",
(double)ValueToVoltage(i32VSP), i32VSP);

        Delay_Ms(3000);
    }
}
```

[1] 示例代码适用于 SPC1168。

Example Code

```
void PGA_InitExample2_2(void)
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  FLASH_WDIS();

  CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

  Delay_Init();

  /*
```

Example Code

```
   * Init the UART
   *
   * 1.Set the GPIO44/45 as UART FUNC
   *
   * 2.Enable the UART CLK
   *
   * 3.Set the rest para
   */
  GPIO_SetPinChannel(GPIO_44,GPIO44_UART_TXD);
  GPIO_SetPinChannel(GPIO_45,GPIO45_UART_RXD);
  CLOCK_EnableModule(UART_MODULE);
  UART_Init(UART,38400);

  /* Init ADC */
  ADC_EasyInit2(ADC_SOC_0,ADCx_PGA0P,ADCx_PGA0N,ADCTRIG_Software);

  /* Init PGA0 */
  PGA_DifferentialInit( PGA0,
            PGA0_CH_P_ADC4 /* Positive CH   */,
            PGA0_CH_N_ADC3 /* Negative CH   */,
            PGA_SCALE_8X   /* PGA Diff Gain */);
}
```

[1] 示例代码适用于 SPC2168。

Example Code

```
#include "spc2188.h"
#include <stdio.h>


/* ADC result convert to voltage can calculate as the follow */
#define             ValueToVoltage(x)           ((x*3339)/8192)


int32_t             i32VSP;
ErrorStatus          status;

int main(void)
{
    CLOCK_InitWithRCO(240000000U);

    Delay_Init();

    /*
    * Initial the UART
    */
    PIN_SetChannel(PIN_GPIO62, PIN_GPIO62_UART0_TXD);
    PIN_SetChannel(PIN_GPIO63, PIN_GPIO63_UART0_RXD);
    UART_Init(UART0, 38400);

    /*
    * Set PGA in  differential-ended mode.
    *
    */
    PGA_DifferentialInit(PGA0, PGA0_CH_P_ANA_IN6, PGA0_CH_N_ANA_IN7,
PGA_GAIN_DIFF_2X_SINGLE_1X);

    /*ADC Initial*/
    ADC_Init(ADC0, ADC_CH0, ADC0_SH0_P_PGA0_OUTP, ADC0_SH0_N_PGA0_OUTN,
ADC_SOC_TRIGGER_FROM_SOFTWARE);
```

Example Code

```
    /* Set Average Times */
    ADC_SetChannelResultAverageCount(ADC0, ADC_CH0, ADC_AVERAGE_COUNT_16);

    while(1)
    {
        /* Use software to trigger ADC SOC0 start to work */
        ADC_ForceChannelSOC(ADC0,ADC_CH0);

        /* Wait until ADC conversion finished (Interrupt flag was set) */
        while(ADC_GetChannelIntFlag(ADC0,ADC_CH0) == 0);

        /* Get result */
        i32VSP = ADC_GetChannelResult(ADC0,ADC_CH0);

        /* Clear ADC SOC0 INT flag */
        ADC_ClearChannelInt(ADC0,ADC_CH0);

        printf("ADC differential Result = %d\n", i32VSP);
        printf("voltage = %dmv\n", ValueToVoltage(i32VSP));

        Delay_Ms(500);
    }
}
```

[1] 示例代码适用于 SPC2188，其它系列产品的示例代码会根据实际需求进行补充。

1.3.2 将一个 PGA 拆分成两个单端 PGA

1.3.2.1 功能需求

使用 ADC 对两个单端 PGA 电压进行采样。

1.3.2.2 功能实现

拆分成两个单端 PGA 是指分别将 VPGAx_INP 作为输入，VPGAx_OUTP 作为输出；将 VPGAx_INN 作为输入，VPGAx_OUTN 作为输出。

$$V_{PGAx\_OUTP} = V_{PGAx\_INP} \times G_P$$

$$V_{PGAx\_OUTN} = V_{PGAx\_INN} \times G_N$$

- VPGAx_INP 与 VPGAx_INN 电压必须为正；
- PGA 在电气特性上的输入范围是轨到轨的。但是为了防止输出管饱和，使用时需要保证输出范围在 0.3V 到 VAVDD - 0.3V 之间，对应于输入范围为 0.3V / GP 到(VAVDD - 0.3V) / GP 以及 0.3V / G_N 到(V_{AVDD} - 0.3V) / G_N。

以下例子演示使用 ADC 对两个单端 PGA 进行采样，将两个单端 PGA 电压分别送给 ADC CH0 和 CH1，采用默认采样时间，默认转换时间。但在实际的工程中采样时间会受到外部电路的影响，其具体的设置数值，可参考《ADC 建立时间计算方法使用指南》，而转换时间则不会受到外部电路影响，通常保持 SDK 中设定的数值即可。

Example Code

```
void PGA_InitExample2_8_1(void)
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  /* Disable flash write access after flash operation had done */
  FLASH_WDIS();

  CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

  Delay_Init();

  /*
   * Init the UART
   *
   * 1.Set the GPIO34/35 as UART FUNC
   *
   * 2.Enable the UART CLK
   *
   * 3.Set the rest para
   */
  GPIO_SetPinChannel(GPIO_34,GPIO34_UART_TXD);
  GPIO_SetPinChannel(GPIO_35,GPIO35_UART_RXD);
  CLOCK_EnableModule(UART_MODULE);
  UART_Init(UART,38400);

  /* Select PGA input as analog pin */
  GPIO_SetPinAsAnalog(GPIO_0); /* ADC0 I+ */
  GPIO_SetPinAsAnalog(GPIO_1); /* ADC1 I- */

  /* Init PGA as single ending mode */
  /* Negative path gain */
  PGA->PGA0CTL.all = (PGA_SCALE_N_8X << PGA0CTL_ALL_GAINN_Pos);
  /* Positive path gain */
  PGA->PGA0CTL.all |= (PGA_SCALE_P_8X << PGA0CTL_ALL_GAINP_Pos);
  /* Negative input     */
  PGA->PGA0CTL.all |=  ((PGA0_CH_N_ADC1 & 0xF) << PGA0CTL_ALL_INSELN_Pos);
  /* Positive input     */
  PGA->PGA0CTL.all |= ((PGA0_CH_P_ADC0 & 0xF) << PGA0CTL_ALL_INSELP_Pos);
  /* Select Negative as Common mode */
  PGA->PGA0CTL.all |= PGA0CTL_ALL_CMSEL_NEGATIVE_AS_COMMON;
  /* single Mode */
  PGA->PGA0CTL.all |= PGA0CTL_ALL_MODE_SINGLE_BOTH;
  /* Enable PGA0 */
  PGA->PGA0CTL.all |= PGA0CTL_ALL_EN_ENABLE;

  /* Init ADC */
  ADC_EasyInit1(ADC_SOC_0,ADCx_PGA0P,ADCTRIG_Software);
  ADC_EasyInit1(ADC_SOC_0,ADCx_PGA0N,ADCTRIG_Software);
}
```

[1] 示例代码适用于 SPC1168。

Example Code

```
void PGA_InitExample2_8_1(void)
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  FLASH_WDIS();

  CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);
```

Example Code

```
  Delay_Init();

  /*
   * Init the UART
   *
   * 1.Set the GPIO44/45 as UART FUNC
   *
   * 2.Enable the UART CLK
   *
   * 3.Set the rest para
   */
  GPIO_SetPinChannel(GPIO_44,GPIO44_UART_TXD);
  GPIO_SetPinChannel(GPIO_45,GPIO45_UART_RXD);
  CLOCK_EnableModule(UART_MODULE);
  UART_Init(UART,38400);

  /* Init PGA as single ending mode */
  /* Negative path gain */
  PGA->PGA0CTL.all = (PGA_SCALE_N_8X << PGA0CTL_ALL_GAINN_Pos);
  /* Positive path gain */
  PGA->PGA0CTL.all |= (PGA_SCALE_P_8X << PGA0CTL_ALL_GAINP_Pos);
  /* Negative input     */
   PGA->PGA0CTL.all |=  ((PGA0_CH_N_ADC1 & 0xF) << PGA0CTL_ALL_INSELN_Pos);
  /* Positive input     */
   PGA->PGA0CTL.all |= ((PGA0_CH_P_ADC0 & 0xF) << PGA0CTL_ALL_INSELP_Pos);
  /* Select Negative as Common mode */
  PGA->PGA0CTL.all |= PGA0CTL_ALL_CMSEL_NEGATIVE_AS_COMMON;
  /* single Mode */
  PGA->PGA0CTL.all |= PGA0CTL_ALL_MODE_SINGLE_BOTH;
  /* Enable PGA0 */
  PGA->PGA0CTL.all |= PGA0CTL_ALL_EN_ENABLE;

  /* Init ADC */
  ADC_EasyInit1(ADC_SOC_0,ADCx_PGA0P,ADCTRIG_Software);
  ADC_EasyInit1(ADC_SOC_0,ADCx_PGA0N,ADCTRIG_Software);
}
```

[1] 示例代码适用于 SPC2168。

Example Code

```
#include "spc2188.h"
#include <stdio.h>


/* ADC result convert to voltage can calculate as the follow */
#define             ValueToVoltage(x)           ((x*3339)/8192)


int32_t             i32VSP;
ErrorStatus         status;

int main(void)
{
    CLOCK_InitWithRCO(240000000U);

    Delay_Init();

    /*
    * Initial the UART
    */
    PIN_SetChannel(PIN_GPIO62, PIN_GPIO62_UART0_TXD);
```

Example Code

```
    PIN_SetChannel(PIN_GPIO63, PIN_GPIO63_UART0_RXD);
    UART_Init(UART0, 38400);

    /*
    * Set PGA in  single-ended mode.
    */
    PGA_SingleEndedInit(PGA0, PGA0_CH_P_ANA_IN6, PGA0_CH_N_ANA_IN7,
PGA_GAIN_DIFF_4X_SINGLE_2X);

    /*ADC Initial*/
    ADC_Init(ADC0, ADC_CH0, ADC0_SH0_P_PGA0_OUTP, ADC0_SH0_N_GND,
ADC_SOC_TRIGGER_FROM_SOFTWARE);
    ADC_Init(ADC0, ADC_CH1, ADC0_SH0_P_GND, ADC0_SH0_N_PGA0_OUTN,
ADC_SOC_TRIGGER_FROM_SOFTWARE);

    /* Set Average Times */
    ADC_SetChannelResultAverageCount(ADC0,ADC_CH0,ADC_AVERAGE_COUNT_16);
    ADC_SetChannelResultAverageCount(ADC0,ADC_CH1,ADC_AVERAGE_COUNT_16);

    while(1)
    {
        /* Use software to trigger ADC SOC0 start to work */
        ADC_ForceChannelSOC(ADC0,ADC_CH0);

        /* Wait until ADC conversion finished (Interrupt flag was set) */
        while(ADC_GetChannelIntFlag(ADC0,ADC_CH0) == 0);

        /* Get result */
        i32VSP = ADC_GetChannelResult(ADC0,ADC_CH0);

        /* Clear ADC SOC0 INT flag */
        ADC_ClearChannelInt(ADC0,ADC_CH0);

        printf("ADC CH0 Result = %d\n", i32VSP);
        printf("voltage = %dmv\n", ValueToVoltage(i32VSP));

        /* Use software to trigger ADC SOC0 start to work */
        ADC_ForceChannelSOC(ADC0,ADC_CH1);

        /* Wait until ADC conversion finished (Interrupt flag was set) */
        while(ADC_GetChannelIntFlag(ADC0,ADC_CH1) == 0);

        /* Get result */
        i32VSP = ADC_GetChannelResult(ADC0,ADC_CH1);

        /* Clear ADC SOC0 INT flag */
        ADC_ClearChannelInt(ADC0,ADC_CH1);

        printf("ADC CH1 Result = %d\n", i32VSP);
        printf("voltage = %dmv\n", ValueToVoltage(i32VSP));

        Delay_Ms(500);
    }
}
```

[1] 示例代码适用于 SPC2188，其它系列产品的示例代码会根据实际需求进行补充。

1.3.3 使用 PGA 进行过电流保护

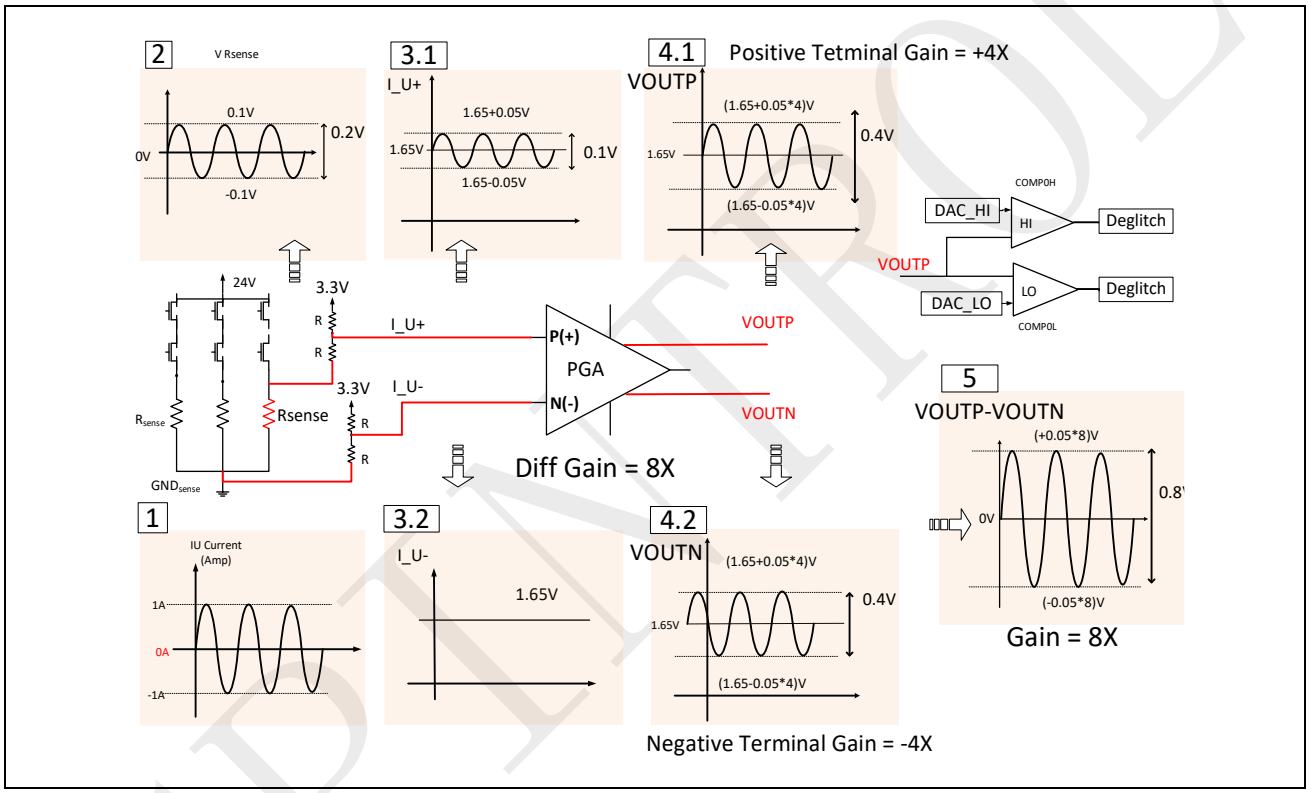
1.3.3.1 功能需求

对功率电路进行过电流保护。

1.3.3.2 功能实现

在实际使用场景中，通常会利用 PGA 放大 R_{sense} 上的电流，此时必须将信号的正端与负端均偏置到 VDD/2 的地方，图 1-4 所示。

图 1-4：PGA 放大信号示意图



以下以一个例子分析内部电压：

- 假设 R_{sense} 为 0.1 欧姆，U 相电流幅值为 1A；
- 设定的差分放大增益是 8X；
- R 建议为 10k 欧姆；
- 根据电路叠加原理，PGA 正端输入电压为 $\frac{3.3}{2}+\frac{0.1*\sin(wt)}{2}$ 负端输入电压为$\frac{3.3}{2}$；
- 最终输出信号$VOUTN$和$VOUTP$如下，因为寄存器 PGAxCTL.CMSEL 设置为 0，选择$VOUTN$做为输出共模电压：

$$VCM = VOUTN = 1.65$$

$$VOUTN = VCM - VIN * \frac{G}{2} = 1.65 - \frac{0.1 * sin(wt)}{2} * \frac{8}{2} = 1.65 - 0.2 * sin(wt)$$

$$VOUTP = VCM + VIN * \frac{G}{2} = 1.65 + \frac{0.1 * sin(wt)}{2} * \frac{8}{2} = 1.65 + 0.2 * sin(wt)$$

式中 VIN 为 PGA 输入端的差模电压$\frac{0.1*\sin(wt)}{2}$，G 为 PGA 的放大倍数 8X。

- COMP 选择$VOUTP$作为输入，因此其偏移为 1.65V，幅值为 0.2V；
- 假设 U 相过电流幅值 2A，是额定相电流幅值的 2 倍，因此 DAC 的参考电平偏移为 1.65V，幅值为 2*0.2V；

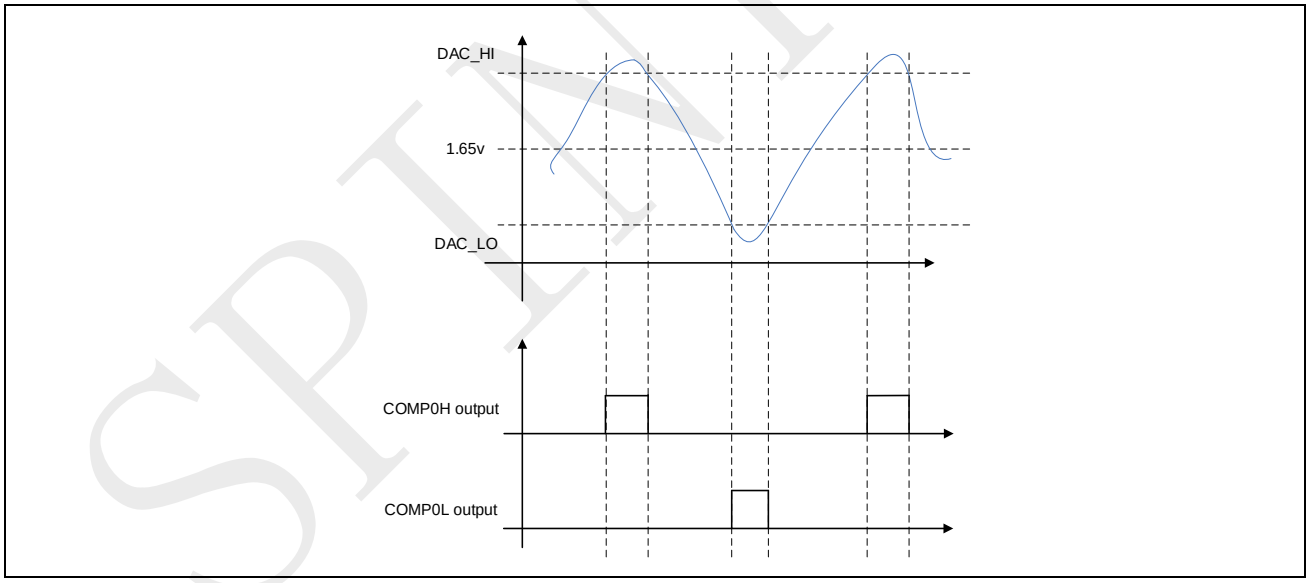
$$\mathrm{DAC_{HI}} = 1.65 + 2 * 0.2 = 2.05\mathrm{V}$$
$$\mathrm{DAC_{LO}} = 1.65 - 2 * 0.2 = 1.25\mathrm{V}$$

- 根据以下公式，可以决定$\mathrm{DAC_{HI}}$与$\mathrm{DAC_{LO}}$的设定数值；

$$\mathrm{DAC_{HI}} = \frac{\mathrm{DAC\_HI\_CODE(10bit)}}{1024} \times 3.3\mathrm{V}$$
$$\mathrm{DAC_{LO}} = \frac{\mathrm{DAC\_LO\_CODE(10bit)}}{1024} \times 3.3\mathrm{V}$$

- DAC_HI_CODE（寄存器数值）须设定为 636，DAC_LO_CODE（寄存器数值）须设定为 387。为了使用方便，Spintrol 提供之 COMP_Init()函数可直接输入 mV 值进行设定。
- 根据上一小节的设计，当输入电流过大或过小时，都会触发 COMP 输出；

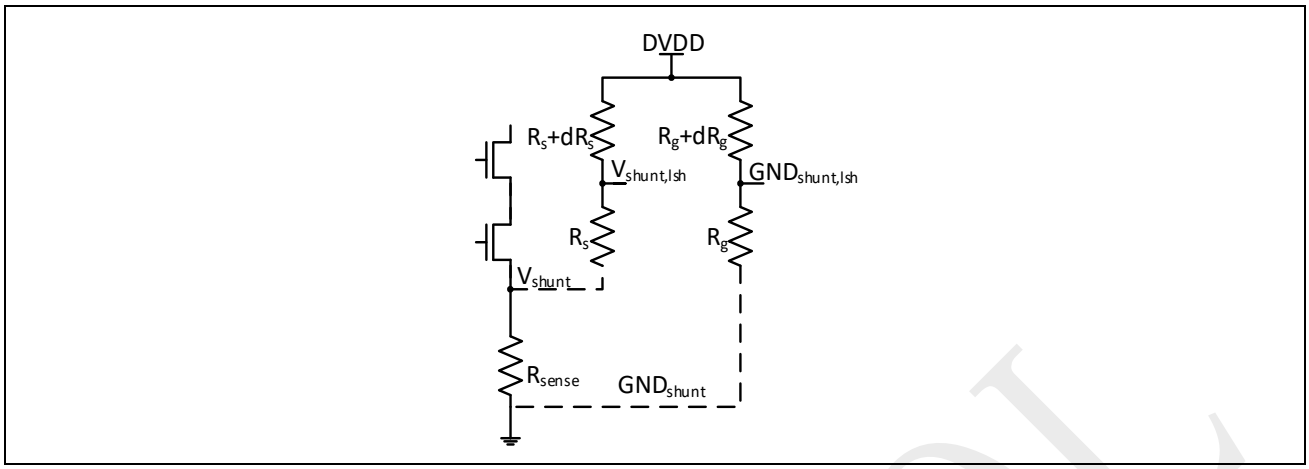
图 1-5：COMP 输出演示



PWM 控制 H 桥电路，在电机控制系统或是电源控制系统是非常常见的技术，但是在 PWM 开关的瞬间，往往会对电流回授电路有较大的干扰，或是开关的瞬间常有较大的电流瞬时（Current Transient）。解决方法参见 COMP 滤波器。

图 1-4 每个电阻电平移位器由两个电阻组成。两个电阻阻值应相等，以便使信号向上移位 AVDD/2。这两个电阻通常会有不匹配，这种不匹配需要校准，以便确定 Rsense 两端的电压。如图 1-6 所示。

图 1-6：电平移位器校准



为简单起见，但不降低讨论的一般性， Vshunt 和 GNDshunt 电压，理想中应该向上移位 AVDD/2，但是由于电阻不匹配，实际移位值会所差别。根据下面的表达式，可以计算出移位后的电压值：

$$V_{shunt,lsh} = V_{ddx}\frac{R_s}{2\cdot R_s + \delta R_s} + V_{shunt}\frac{R_s + \delta R_s}{2\cdot R_s + \delta R_s} = \frac{V_{ddx}}{2}\left[1 - \frac{\delta_s}{2}\right] + \frac{V_{shunt}}{2}\left[1 + \frac{\delta_s}{2}\right]$$

$$GND_{shunt,lsh} = V_{ddx}\frac{R_g}{2\cdot R_g + \delta R_g} + GND_{shunt}\frac{R_g + \delta R_g}{2\cdot R_g + \delta R_g} = \frac{V_{ddx}}{2}\left[1 - \frac{\delta_g}{2}\right] + \frac{GND_{shunt}}{2}\left[1 + \frac{\delta_g}{2}\right]$$

在这里，Rs/Rs=δs，Rg/Rg=δg。因此，移位后的差分输出电压为：

$$V_{ind} = V_{shunt,lsh} - GND_{shunt,lsh} = \frac{V_{ddx}}{2}\cdot\frac{\delta_g - \delta_s}{2} + \frac{V_{shunt} - GND_{shunt}}{2} + \frac{V_{shunt}}{2}\cdot\frac{\delta_s}{2}$$

这里，忽略了 GNDshuntδg/4，因为 GNDshunt 和δg 都是小量。我们发现失配误差可以分为增益误差（Vshunt 和δs /2 乘积）和偏移误差（Vshunt=0 时的 Vind）。增益误差项相当于 Rsense 电阻的变化。这两个项都可以在当电机不工作时通过测量电压来确定，具体步骤如下：

- 测量 AVDD。可以将 AVDD 通过 ADC 多路选择器送到 ADC 测量。
- 当电机不工作时，没有电流通过 Rsense，也就是 Vshunt=0，测量 Vshunt,lsh。通过上面 Vshunt,lsh 的表达式可以计算出δs。
- 当电机不工作时，没有电流通过 Rsense，测量 Vind。根据 Vind 的表达式可以计算出 $\delta_g - \delta_s$。

Example Code

```
void PWMx_Set_Digiter_Compare(PWM_REGS *PWMx)
{
    /* Affected by results monitored from all three phases */
    PWM_EnableDCAHTripEvent(PWMx, DC_TRIP_COMP0H |
                          DC_TRIP_COMP1H |
                          DC_TRIP_COMP2H |
                          DC_TRIP_COMP0L |
                          DC_TRIP_COMP1L |
                          DC_TRIP_COMP2L);

    PWM_SetRawDCAEVT0(PWMx, DCH_HIGH_DCL_X);

    /* Set the filtered event as the final DCAEVT0, and trigger TZ sb-model
when DCH is high */
    PWM_SetDCAEVT0(PWMx, DCEVT_FILTERED);

    /*
     * 1.Configure DC event filter
     *
     * 2.Input is DCAEVT0, apply blank window with original polarity
     *
     * 3.Set the blank window position relative to TBCNT=0
     */
    PWM_SetDCFilter(PWMx, DCF_FROM_RAW_DCAEVT0, DCF_ALIGN_ON_ZERO);

    /* Set blank window size as 100 TBCLK and offset as 50 TBCLK */
    PWM_SetDCFilterBlankWindow(PWMx, PWM_BlankWIN_Size, PWM_Blank_Offset);

    /* Enable PWM DC filter blank function */
    PWM_EnableDCFilterBlank(PWMx);

    PWM_SetOneShotTripEvent(PWMx, TRIP_EVENT_DCAEVT, TRIP_OUTPUT_LATCH);

    /* Set output to tri-state upon DCAEVT0 trip event */
    PWM_SetCHAOutputWhenTrip(PWMx, TZU_TRIP_AS_TRI_STATE |
                           TZD_TRIP_AS_TRI_STATE   |
                           DCEVT0U_TRIP_DO_NOTHING |
                           DCEVT0D_TRIP_DO_NOTHING |
                           DCEVT1U_TRIP_DO_NOTHING |
                           DCEVT1D_TRIP_DO_NOTHING);

    PWM_SetCHBOutputWhenTrip(PWMx, TZU_TRIP_AS_TRI_STATE |
                           TZD_TRIP_AS_TRI_STATE   |
                           DCEVT0U_TRIP_DO_NOTHING |
                           DCEVT0D_TRIP_DO_NOTHING |
                           DCEVT1U_TRIP_DO_NOTHING |
                           DCEVT1D_TRIP_DO_NOTHING);

    PWM_EnableDCAEVT0TripInt(PWMx);
}

/*
 * This function output Trip-Phase waveform for BLDC(Brushless Direct Current)
 */
void PWM_Output_Trip_Phase(void)
{
    /*
     * Init PWM5/1/2 and output 20kHz waveform on both channel A and channel B.
     */
    PWM_ComplementaryPairChannelInit(PWM_U, PWM_FREQ, PWM_DB_NS);
    PWM_ComplementaryPairChannelInit(PWM_V, PWM_FREQ, PWM_DB_NS);
```

Example Code

```
    PWM_ComplementaryPairChannelInit(PWM_W, PWM_FREQ, PWM_DB_NS);

    /* Start counting */
    PWM_RunCounter(PWM_U);
    PWM_RunCounter(PWM_V);
    PWM_RunCounter(PWM_W);

    /* PWM5 set CMPA and loading the same value to PWM1/2 synchronously */
    PWM_LinkCMPA(PWM_V, LINK_PWM_U);
    PWM_LinkCMPA(PWM_W, LINK_PWM_U);
    PWM_SetCMPA(PWM_U, PWM_CMPA);
}



int main()
{
    FLASH_WALLOW();

#if defined(SPC1158)
    FLASH_SetTiming(100000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_100MHZ);
#else
    FLASH_SetTiming(200000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);
#endif

    Delay_Init();

    /*
     * Init the UART
     *
     * 1.Set the GPIO34/35 as UART FUNC
     *
     * 2.Enable the UART CLK
     *
     * 3.Set the rest para
     */
    GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
    GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
    CLOCK_EnableModule(UART_MODULE);
    UART_Init(UART, 38400);

    /*
     * As the title described, we suppose GPIO8/10/12(ADC8/10/12) can be used to
sample the voltage
     * of current sampling resistor in the BLDC circuit. so, we need to set the
GPIOs as ADCs.
     */
    GPIO_SetPinAsAnalog(PGA0_P_PIN);
    GPIO_SetPinAsAnalog(PGA1_P_PIN);
    GPIO_SetPinAsAnalog(PGA2_P_PIN);

    /*
     * Set PGA in differential ended mode.
```

Example Code

```
     *
     * 1.Set the GPIO8(ADC8)/GPIO10(ADC10)/GPIO12(ADC12) as the P-input of the
PGA0/PGA1/PGA2.
     *
     * 2.Set the DAC1 as the N-input of the PGA0/PGA1/PGA2.
     *
     * 3.Set the PGA0/PGA1/PGA2 amplitude scale as 4x.
     *
     * 4.Enable the PGA0/PGA1/PGA2.
     */
    PGA_DifferentialInit(PGA0, PGA0_CH_P, PGA0_CH_N_DAC1, PGA_SCALE_8X);
    PGA_DifferentialInit(PGA1, PGA1_CH_P, PGA1_CH_N_DAC1, PGA_SCALE_8X);
    PGA_DifferentialInit(PGA2, PGA2_CH_P, PGA2_CH_N_DAC1, PGA_SCALE_8X);

    /* Set DAC1 output 1.65V */
    COMP_SetDACVoltage(DAC1, SampleRegisterNVol);
    COMP_EnableDAC(DAC1);

    /*
     * 1. Initialize comparator with 300ns deglitch filtering window
.
     * 2. Set DAC 2 & 3 as the comparator too-High trigger input and too-low
trigger input.
     */
    COMP_Init(COMP_0_HI, COMP0_FROM_PGA0P_OUT, SampleRegisterNVol +
VolOffsetmV, COMP_FilterWIN);
    COMP_Init(COMP_0_LO, COMP0_FROM_PGA0P_OUT, SampleRegisterNVol -
VolOffsetmV, COMP_FilterWIN);
    COMP_Init(COMP_1_HI, COMP1_FROM_PGA1P_OUT, SampleRegisterNVol +
VolOffsetmV, COMP_FilterWIN);
    COMP_Init(COMP_1_LO, COMP1_FROM_PGA1P_OUT, SampleRegisterNVol -
VolOffsetmV, COMP_FilterWIN);
    COMP_Init(COMP_2_HI, COMP2_FROM_PGA2P_OUT, SampleRegisterNVol +
VolOffsetmV, COMP_FilterWIN);
    COMP_Init(COMP_2_LO, COMP2_FROM_PGA2P_OUT, SampleRegisterNVol -
VolOffsetmV, COMP_FilterWIN);

    /* Select GPIO28/29 as the channel A/B output of PWM5 respectively */
    GPIO_SetPinChannel(GPIO_PWM_U_H, GPIO_PWM_U_H_SEL);
    GPIO_SetPinChannel(GPIO_PWM_U_L, GPIO_PWM_U_L_SEL);

    /* Select GPIO20/21 as the channel A/B output of PWM1 respectively */
    GPIO_SetPinChannel(GPIO_PWM_V_H, GPIO_PWM_V_H_SEL);
    GPIO_SetPinChannel(GPIO_PWM_V_L, GPIO_PWM_V_L_SEL);

    /* Select GPIO22/23 as the channel A/B output of PWM2 respectively */
    GPIO_SetPinChannel(GPIO_PWM_W_H, GPIO_PWM_W_H_SEL);
    GPIO_SetPinChannel(GPIO_PWM_W_L, GPIO_PWM_W_L_SEL);

    /* Generate a tri-phase waveform */
    PWM_Output_Trip_Phase();

    /* Set the DC sub-module for PWM_U */
    PWMx_Set_Digiter_Compare(PWM_U);

    /* Set the DC sub-module for PWM_V */
    PWMx_Set_Digiter_Compare(PWM_V);

    /* Set the DC sub-module for PWM_W */
    PWMx_Set_Digiter_Compare(PWM_W);
```

Example Code

```
    NVIC_EnableIRQ(PWM5TZ_IRQn);

    while (1)
    {
        Delay_Ms(500);

        /* Restore the output of wave in oneshot mode*/
        PWM_ClearOneShotTripInt(PWM_U);
        PWM_ClearOneShotTripInt(PWM_V);
        PWM_ClearOneShotTripInt(PWM_W);
    }
}

void PWM5TZ_IRQHandler(void)
{
    uint32_t u32TZStatus = PWM5->TZSTS.all;

    if (u32TZStatus & TZSTS_ALL_DCAEVT0_OCCUR)
    {
        printf("Digital compare A event 0 one-shot trip event occurred\n");

        PWM5->TZSTSCLR.bit.DCAEVT0 = 1;
    }

    PWM_ClearTripGlobalInt(PWM5);
}
```

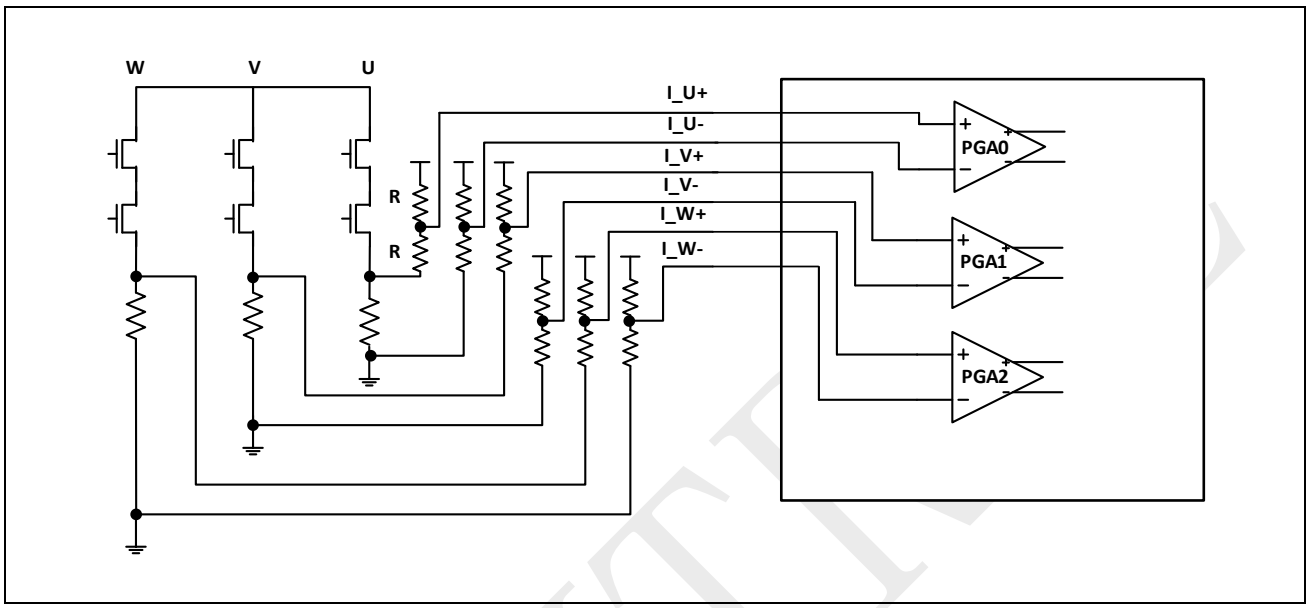
[1] 示例代码适用于 SPC1168，其它系列产品的示例代码会根据实际需求进行补充。

1.3.4 3 路差分 PGA 电阻采样电路

1. shunt 电阻采样，真正差分

每一个相电流采样电阻的负端讯号（GND）也需要一个 Pin 进行放大。

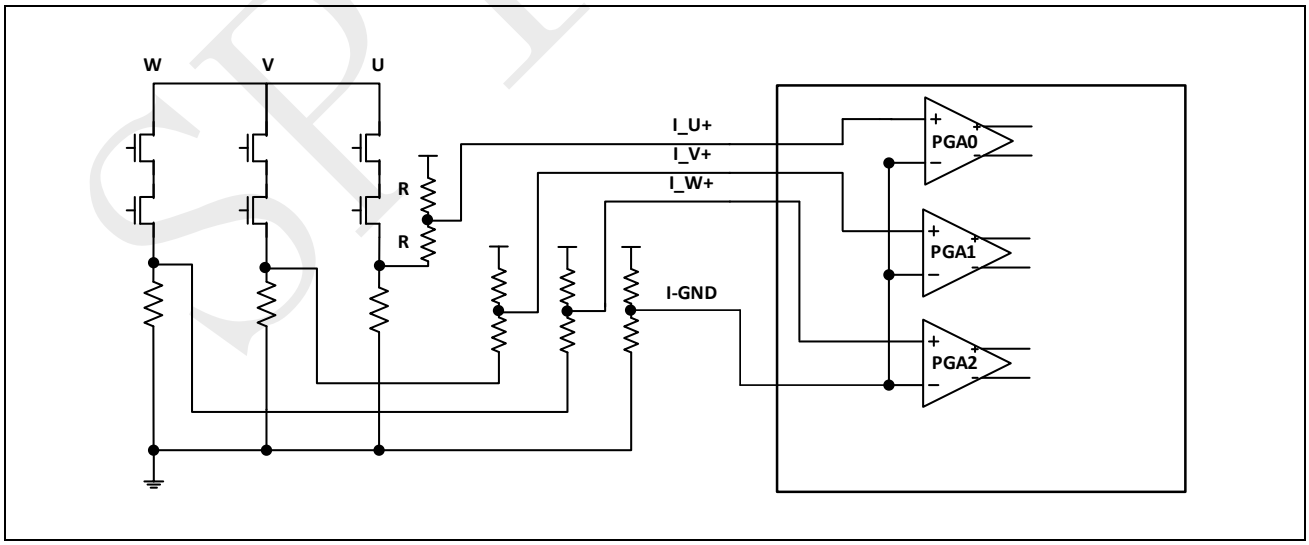
图 1-7：3 shunt 电阻差分采样示意图



若有滤波需求，可在进入 MCU 前加上一个电容。

2. 3 shunt 电阻采样，共享 GND

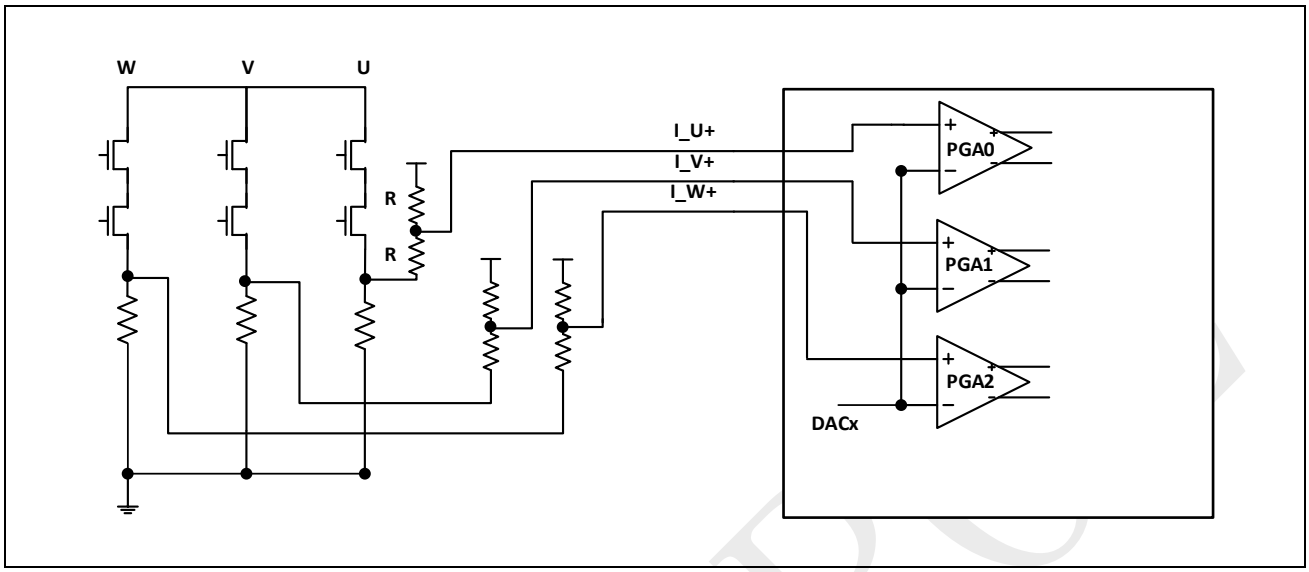
图 1-8：3 shunt 电阻共 GND 采样示意图



如果三个 PGA 负端输入有共同的端点，则其可作为三个讯号的共地。依照此接法可节省两个输入管脚与四个电阻，但是在 PCB Layout 上共地信号必须与三相电流信号尽可能走相同路径，才能有较佳的共模噪声滤除效果。

3. 3 shunt 电阻采样，使用内部 DAC 输出 VDD/2

图 1-9：3 shunt 电阻内部电阻分压采样



使用芯片内部 DAC 作为负端讯号源，设定此分压电阻输出 1.65V，与上一章节有同样效果，依照此接法只需要三个管脚与六个信号平移电阻，但是并无共模噪声滤除效果。

2 SPC1168/SPC2168 系列

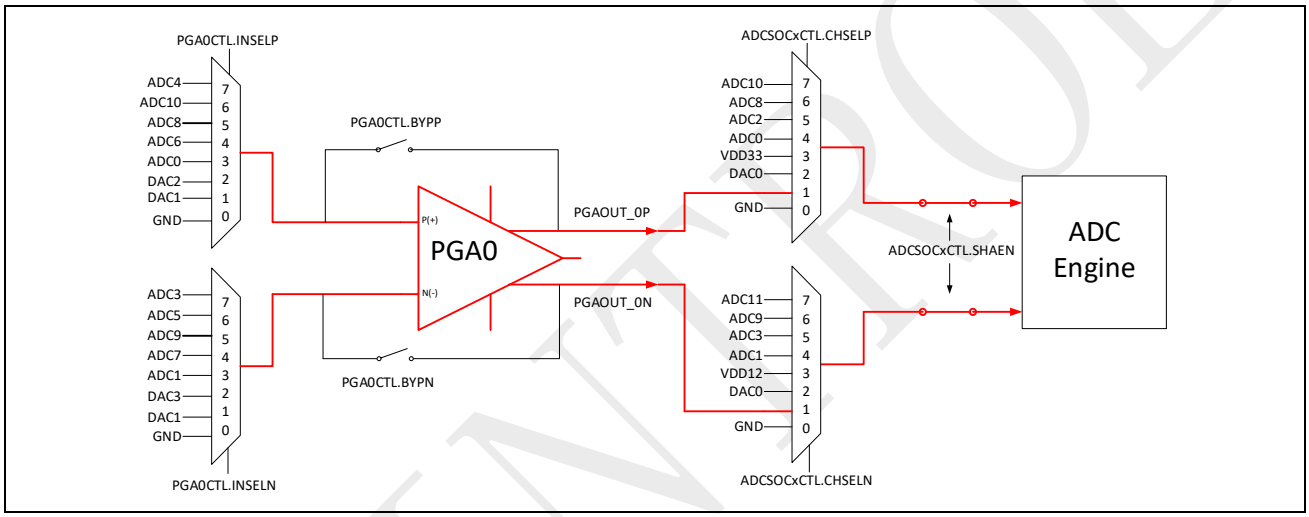
2.1 特性

每个 PGA 有两种工作模式：一种是差分模式，最大增益可到 64，另一种是单端模式，最大增益可到 32。在单端模式下，每个 PGA 可以支持两个单端信号的放大。

2.2 功能描述

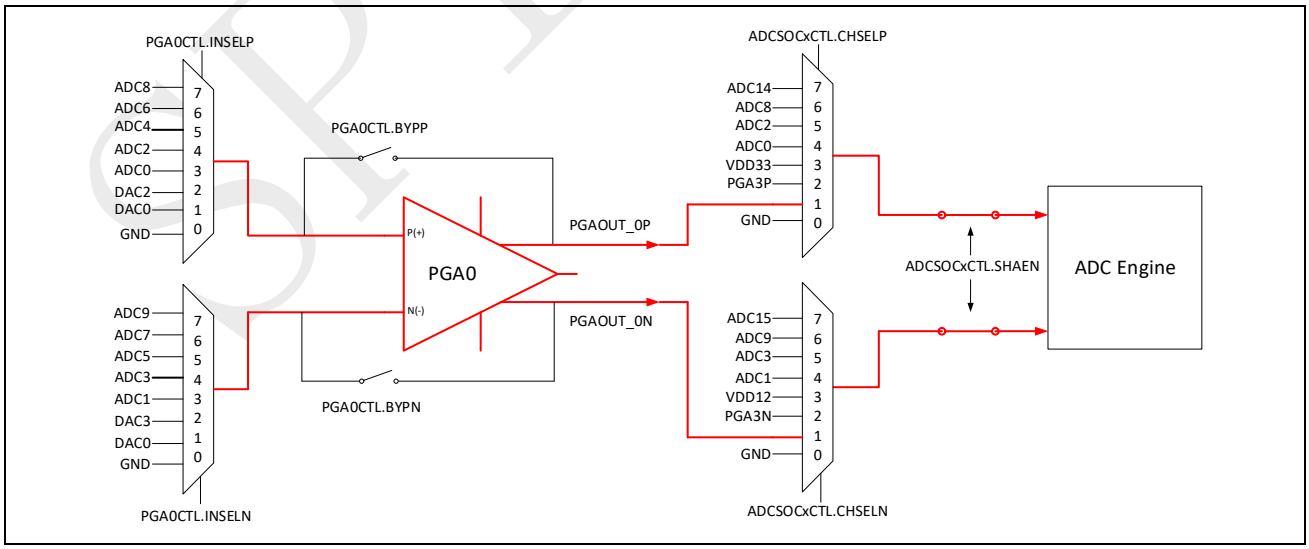
PGA0 结构框图如图 1-1 到图 1-3 所示。

图 2-1：PGA0 结构框图



[1] SPC1168 PGA0 结构图。

图 2-2：PGA0 结构框图



[1] SPC2168 PGA0 结构图。

2.3 功能实例

2.3.1 Sensor 模式下的 PGA 配置

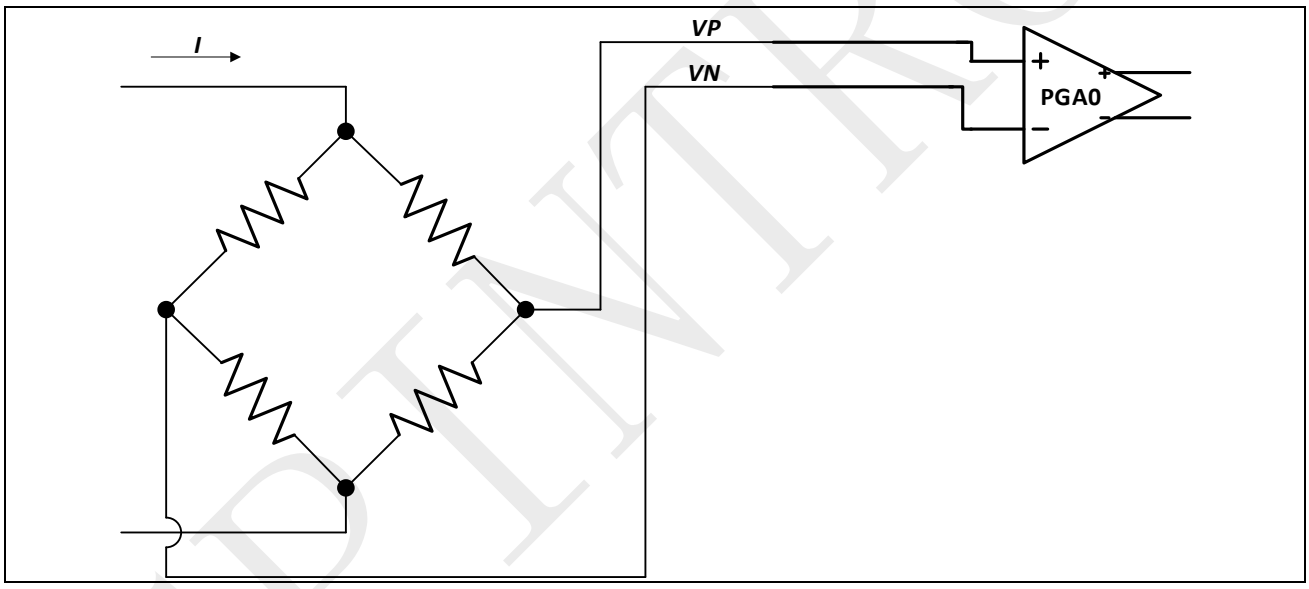
2.3.1.1 功能需求

对功率电路进行过电流保护。

2.3.1.2 功能实现

在某些应用中，如果输入是差分信号，并且输入的共模电平在 VDD/2，这种情况下不需要外置的电阻分压偏置电路，可以将 PGA 配置到 SENSOR 模式下，将信号直接送到 PGA 输入，则 PGA 输出的共模电平就是输入的共模电平，也就是 VDD/2。如果 PGA 配置到 SENSOR 模式下，输出的差分增益是正常差分模式下输出增益的一半，即输出的差分增益分别为 1X、2X、4X、8X、12X、16X、24X、32X。

图 2-3: Sensor 模式下 PGA 采样示意图



对应的范例代码如下：

Example Code

```
void PGA_InitExample2_5_1(void)
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  /* Disable flash write access after flash operation had done */
  FLASH_WDIS();

  CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

  Delay_Init();

  /*
   * Init the UART
   *
   * 1.Set the GPIO34/35 as UART FUNC
   *
```

Example Code

```
   * 2.Enable the UART CLK
   *
   * 3.Set the rest para
   */
  GPIO_SetPinChannel(GPIO_34,GPIO34_UART_TXD);
  GPIO_SetPinChannel(GPIO_35,GPIO35_UART_RXD);
  CLOCK_EnableModule(UART_MODULE);
  UART_Init(UART,38400);

  /* Select PGA input as analog pin */
  GPIO_SetPinAsAnalog(GPIO_0); /* ADC0 VP */
  GPIO_SetPinAsAnalog(GPIO_1); /* ADC1 VN */

  /* Init PGA */
  PGA_DifferentialInit( PGA0,
          PGA0_CH_P_ADC0 /* Positive CH   */,
          PGA0_CH_N_ADC1 /* Negative CH   */,
          PGA_SCALE_8X   /* PGA Diff Gain */);

  /* Enable PGA0 sensor mode and the actual PGA diff gain is 4x */
  PGA_EnableSensorMode(PGA0);

  /* Init ADC in 2 channel mode */
  ADC_EasyInit2(ADC_SOC_0,ADCx_PGA0P,ADCx_PGA0N,ADCTRIG_Software);
}
```

[1] 示例代码适用于 SPC1168。

Example Code

```
void PGA_InitExample2_5_1(void)
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  /* Disable flash write access after flash operation had done */
  FLASH_WDIS();

  CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

  Delay_Init();

  /*
   * Init the UART
   *
   * 1.Set the GPIO44/45 as UART FUNC
   *
   * 2.Enable the UART CLK
   *
   * 3.Set the rest para
   */
  GPIO_SetPinChannel(GPIO_44,GPIO44_UART_TXD);
  GPIO_SetPinChannel(GPIO_45,GPIO45_UART_RXD);
  CLOCK_EnableModule(UART_MODULE);
  UART_Init(UART,38400);

  /* Select PGA input as analog pin */
  GPIO_SetPinAsAnalog(GPIO_0); /* ADC0 VP */
  GPIO_SetPinAsAnalog(GPIO_1); /* ADC1 VN */

  /* Init PGA */
  PGA_DifferentialInit( PGA0,
          PGA0_CH_P_ADC0 /* Positive CH   */,
```

Example Code

```
            PGA0_CH_N_ADC1 /* Negative CH   */,
            PGA_SCALE_8X   /* PGA Diff Gain */);

  /* Enable PGA0 sensor mode and the actual PGA diff gain is 4x */
  PGA_EnableSensorMode(PGA0);

  /* Init ADC in 2 channel mode */
  ADC_EasyInit2(ADC_SOC_0,ADCx_PGA0P,ADCx_PGA0N,ADCTRIG_Software);
}
```

[1] 示例代码适用于 SPC2168，其它系列产品的示例代码会根据实际需求进行补充。

3 SPC1169，SPC1125 系列

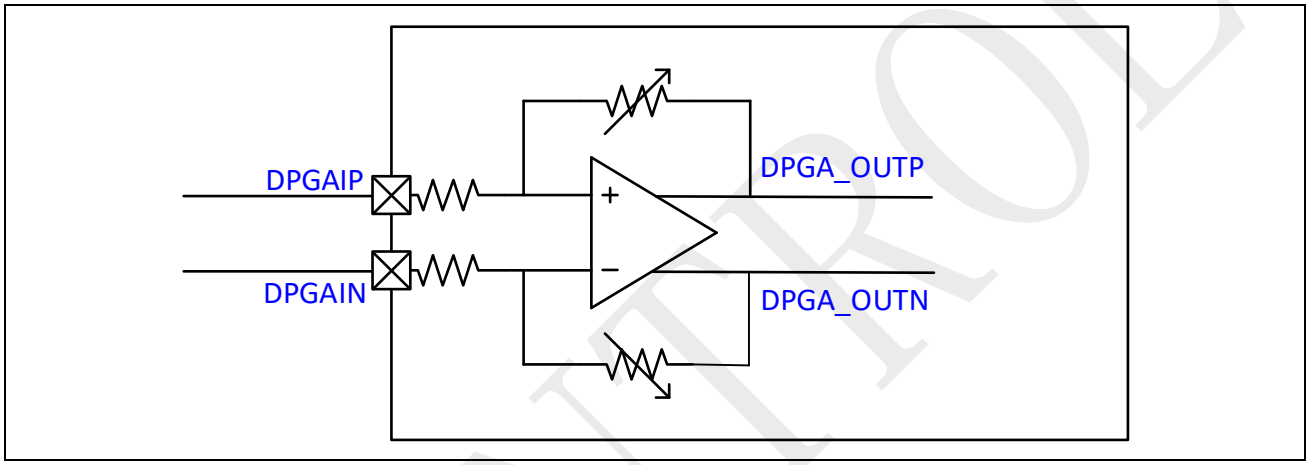
3.1 特性

提供一种带有两个专用引脚的差分可编程增益放大器（DPGA）。

3.2 功能描述

为 DPGAP_IN 与 DPGAN_IN 接入电平，其连接如图 3-1 所示。

图 3-1：ADC DPGA 采样



将 VDPGAIP 与 VDPGAIN 的差值作为输入，将 VDPGA_OUTP 与 VDPGA_OUTN 的差值作为输出。

$$V_{DPGA\_OUTP} - V_{DPGA\_OUTN} = (V_{DPGAIP} - V_{DPGAIN}) \times Gain_{DPGA}$$

其中 VDPGA_OUTP 与 VDPGA_OUTN 各自输出为：

$$V_{DPGA\_OUTP} = V_{CMOUT} + (V_{DPGAIP} - V_{DPGAIN}) \times Gain_{DPGA} / 2$$

$$V_{DPGA\_OUTN} = V_{CMOUT} - (V_{DPGAIP} - V_{DPGAIN}) \times Gain_{DPGA} / 2$$

- VDPGAIP 与 VDPGAIN 本身电压必须为正（但是其差值可以为负）；
- VDPGA_OUTP 与 VDPGA_OUTN 电压必须为正。同时为了防止输出管饱和，使用时需要保证 VDPGA_OUTP 与 VDPGA_OUTN 电压在 0.3V 到 3.0V 之间；
- VCMOUT 是输出的共模电压，设计上已经固定为 3.3V / 2；
- VDPGA_OUTP 与 VDPGA_OUTN 电压差在$\pm 2.7V$，对应 VDPGAIP 与 VDPGAIN 电压差必须在$\pm \frac{2.7V}{G}$范围内。

表 3-1：给定增益下 DPGA 输入输出范围

DPGACTL.GAIN	$Gain_{DPGA}$	Input range (V)
0	2	-1.35~1.35
1	4	-0.675~+0.675

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

3.3 功能实例

3.3.1 ADC 对 DPGA 采样

3.3.1.1 功能需求

使用 ADC 对 DPGA 电压进行采样。

3.3.1.2 功能实现

以下例子演示使用 ADC 对 DPGA 电压进行采样，将 DPGA 电压送给 ADC CH0 进行采样，默认采样时间，默认转换时间。但在实际的工程中采样时间会受到外部电路的影响，其具体的设置数值，可参考《ADC 建立时间计算方法使用指南》，而转换时间则不会受到外部电路影响，通常保持 SDK 中设定的数值即可。

Example Code

```
int main(void)
{
    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);

    /*
    * Set PGA in  differential-ended mode.
    *
    */
    PGA_InitDPGA(DPGA_GAIN_2X);

    /*ADC Init*/
    ADC_EasyInit2(ADC, ADC_CH0, ADC_IN_DPGAP_OUT, ADC_IN_DPGAN_OUT,
ADC_SOC_TRIGGER_FROM_SOFTWARE);

    /* Set Average Times */
    ADC_SetChannelResultAverageCount(ADC, ADC_CH0, ADC_AVERAGE_COUNT_16);
    while (1)
    {
```

Example Code

```
        /* Use software to trigger ADC SOC0 start to work */
        ADC_ForceChannelSOC(ADC, ADC_CH0);

        /* Wait until ADC conversion finished (Interrupt flag was set) */
        while (ADC_GetChannelIntFlag(ADC, ADC_CH0) == 0);

        /* Get result */
        i32VSP = ADC_GetChannelResult(ADC, ADC_CH0);

        /* Clear ADC SOC0 INT flag */
        ADC_ClearChannelInt(ADC, ADC_CH0);

        printf("ADC differential Result = %d\n", i32VSP);
        printf("ADC differential Voltage %dmv\n", ValueToVoltage(i32VSP));

        Delay_Ms(500);
    }
}
```

[1] 示例代码适用于 SPC1169，其它系列产品的示例代码会根据实际需求进行补充。

3.3.2 使用 DPGA 进行过电流保护

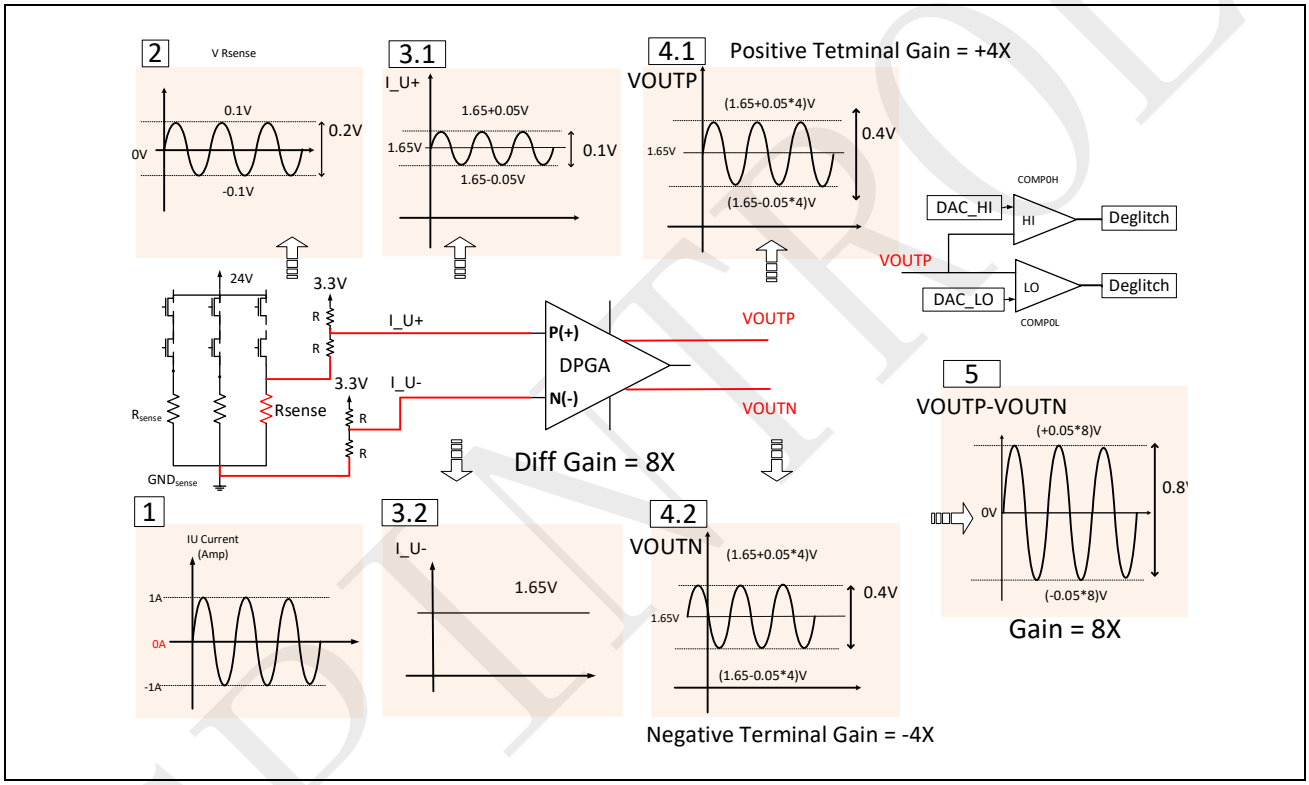
3.3.2.1 功能需求

对功率电路进行过电流保护。

3.3.2.2 功能实现

在实际使用场景中，通常会利用 DPGA 放大 R_{sense} 上的电流，此时必须将信号的正端与负端均偏置到 VDD/2 的地方，如图 3-2 所示。

图 3-2：DPGA 放大信号示意图



以下以一个例子分析内部电压：

- 假设 R_{sense} 为 0.1 欧姆，U 相电流幅值为 1A；
- 设定的差分放大增益是 8X；
- R 建议为 10k 欧姆；
- 根据电路叠加原理，DPGA 正端输入电压为 $\frac{3.3}{2}+\frac{0.1*\sin(wt)}{2}$ 负端输入电压为$\frac{3.3}{2}$；
- 最终输出信号$VOUTN$和$VOUTP$如下：

$$VCM=\frac{DVDD33}{2}=1.65$$

$$VOUTN = VCM - VIN * \frac{G}{2} = 1.65 - \frac{0.1 * sin(wt)}{2} * \frac{8}{2} = 1.65 - 0.2 * sin(wt)$$

$$VOUTP = VCM + VIN * \frac{G}{2} = 1.65 + \frac{0.1 * sin(wt)}{2} * \frac{8}{2} = 1.65 + 0.2 * sin(wt)$$

式中 VCM 为输出共模电压，芯片内部固定为$\frac{DVDD33}{2}$，VIN 为 DPGA 输入端的差模电压$\frac{0.1*\sin(wt)}{2}$，G 为 DPGA 的放大倍数 8X。

- COMP 选择$VOUTP$作为输入，因此其偏移为 1.65V，幅值为 0.2V；
- 假设 U 相过电流幅值 2A，是额定相电流幅值的 2 倍，因此 DAC 的参考电平偏移为 1.65V，幅值为 2*0.2V；

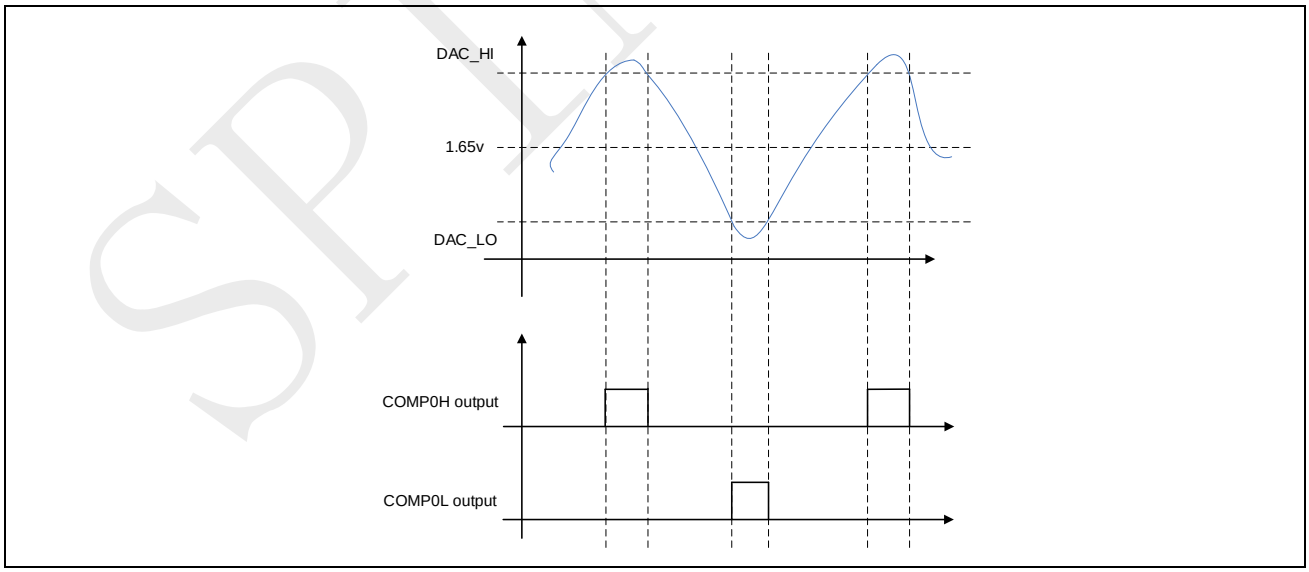
$$\mathrm{DAC_{HI}} = 1.65 + 2 * 0.2 = 2.05\mathrm{V}$$
$$\mathrm{DAC_{LO}} = 1.65 - 2 * 0.2 = 1.25\mathrm{V}$$

- 根据以下公式，可以决定$\mathrm{DAC_{HI}}$与$\mathrm{DAC_{LO}}$的设定数值；

$$\mathrm{DAC_{HI}} = \frac{\mathrm{DAC\_HI\_CODE(10bit)}}{1024} \times 3.3\mathrm{V}$$
$$\mathrm{DAC_{LO}} = \frac{\mathrm{DAC\_LO\_CODE(10bit)}}{1024} \times 3.3\mathrm{V}$$

- DAC_HI_CODE（寄存器数值）须设定为 636，DAC_LO_CODE（寄存器数值）须设定为 387。为了使用方便，Spintrol 提供之 COMP_Init()函数可直接输入 mV 值进行设定。
- 根据上一小节的设计，当输入电流过大或过小时，都会触发 COMP 输出；

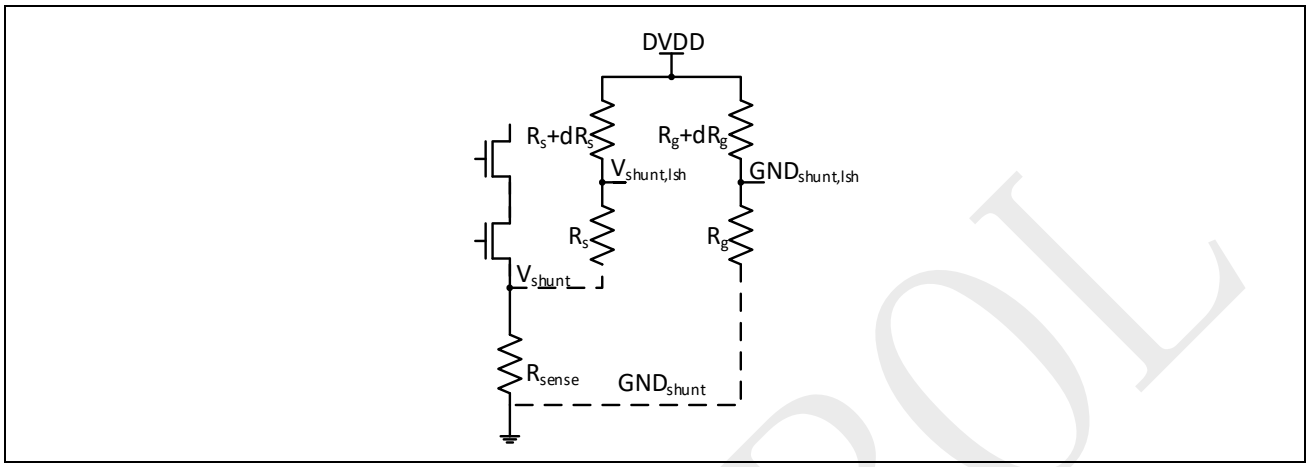
图 3-3：COMP 输出演示



PWM 控制 H 桥电路，在电机控制系统或是电源控制系统是非常常见的技术，但是在 PWM 开关的瞬间，往往会对电流回授电路有较大的干扰，或是开关的瞬间常有较大的电流瞬时（Current Transient）。解决方法参见 COMP 滤波器。

图 3-2 每个电阻电平移位器由两个电阻组成。两个电阻阻值应相等，以便使信号向上移位 AVDD/2。这两个电阻通常会有不匹配，这种不匹配需要校准，以便确定 Rsense 两端的电压。如图 3-4 所示。

图 3-4：电平移位器校准



为简单起见，但不降低讨论的一般性， Vshunt 和 GNDshunt 电压，理想中应该向上移位 AVDD/2，但是由于电阻不匹配，实际移位值会所差别。根据下面的表达式，可以计算出移位后的电压值：

$$V_{shunt,lsh} = V_{ddx}\frac{R_s}{2\cdot R_s + \delta R_s} + V_{shunt}\frac{R_s + \delta R_s}{2\cdot R_s + \delta R_s} = \frac{V_{ddx}}{2}\left[1 - \frac{\delta_s}{2}\right] + \frac{V_{shunt}}{2}\left[1 + \frac{\delta_s}{2}\right]$$

$$GND_{shunt,lsh} = V_{ddx}\frac{R_g}{2\cdot R_g + \delta R_g} + GND_{shunt}\frac{R_g + \delta R_g}{2\cdot R_g + \delta R_g} = \frac{V_{ddx}}{2}\left[1 - \frac{\delta_g}{2}\right] + \frac{GND_{shunt}}{2}\left[1 + \frac{\delta_g}{2}\right]$$

在这里，Rs/Rs=δs，Rg/Rg=δg。因此，移位后的差分输出电压为：

$$V_{ind} = V_{shunt,lsh} - GND_{shunt,lsh} = \frac{V_{ddx}}{2}\cdot\frac{\delta_g - \delta_s}{2} + \frac{V_{shunt} - GND_{shunt}}{2} + \frac{V_{shunt}}{2}\cdot\frac{\delta_s}{2}$$

这里，忽略了 GNDshuntδg/4，因为 GNDshunt 和δg 都是小量。我们发现失配误差可以分为增益误差（Vshunt 和δs /2 乘积）和偏移误差（Vshunt=0 时的 Vind）。增益误差项相当于 Rsense 电阻的变化。这两个项都可以在当电机不工作时通过测量电压来确定，具体步骤如下：

- 测量 AVDD。可以将 AVDD 通过 ADC 多路选择器送到 ADC 测量。
- 当电机不工作时，没有电流通过 Rsense，也就是 Vshunt=0，测量 Vshunt,lsh。通过上面 Vshunt,lsh 的表达式可以计算出δs。
- 当电机不工作时，没有电流通过 Rsense，测量 Vind。根据 Vind 的表达式可以计算出 $\delta_g - \delta_s$。

Example Code

```
void PWMx_Set_TZ_Event(PWM_REGS *PWMx)
{
    /* Affected by results monitored from COMP0 */
    PWM_EnableDCAHTripEvent(PWMx, DC_TRIP_COMP0H);
    PWM_EnableDCAHTripEvent(PWMx, DC_TRIP_COMP0L);

    /*DCAEVT0 event comes from DCAL=don't care, DCAH=high*/
    PWM_SetRawDCAEVT0(PWMx, DCH_HIGH_DCL_X);

    /* Set the filter, Filter trigger conditions is TBCNT=0*/
    PWM_SetDCFilter(PWMx, DCF_FROM_RAW_DCAEVT0, DCF_ALIGN_ON_ZERO);

    /* Enable PWM DC filter blank function */
    PWM_EnableDCFilterBlank(PWMx);

    /* Set blank window size as 100 TBCLK and offset as 50 TBCLK */
    PWM_SetDCFilterBlankWindow(PWMx, PWM_BlankWIN_Size, PWM_Blank_Offset);

    /* USe the filtered to deal the data from DCAEVT0 */
    PWM_SetDCAEVT0(PWMx, DCEVT_FILTERED);

    /* Set the DCAEVT0 as OneShot mode */
    PWM_SetOneShotTripEvent(PWMx, TRIP_EVENT_DCAEVT, TRIP_OUTPUT_LATCH);

    /* Set output to tri-state upon OST trip event */
    PWM_SetCHAOutputWhenTrip(PWMx, TZU_TRIP_AS_LOW |
                           TZD_TRIP_AS_LOW |
                           DCEVT0U_TRIP_DO_NOTHING |
                           DCEVT0D_TRIP_DO_NOTHING |
                           DCEVT1U_TRIP_DO_NOTHING |
                           DCEVT1D_TRIP_DO_NOTHING);

    PWM_SetCHBOutputWhenTrip(PWMx, TZU_TRIP_AS_LOW |
                           TZD_TRIP_AS_LOW |
                           DCEVT0U_TRIP_DO_NOTHING |
                           DCEVT0D_TRIP_DO_NOTHING |
                           DCEVT1U_TRIP_DO_NOTHING |
                           DCEVT1D_TRIP_DO_NOTHING);

    PWM_EnableTripInt(PWMx, TRIP_INT_OST);
}


int main(void)
{
    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("Enter the test\n");

    /*
    * Set PGA in  differential-ended mode.
    *
```

Example Code

```
    */
    PGA_InitDPGA(DPGA_GAIN_8X);

    /*
     * 1. Initialize comparator with 300ns deglitch filtering window.
     * 2. Set DAC voltage, COMP_HI's negative port is 2050mV,
     * COMP_LO's positive port is 1250mV
     */
    COMP_Init(COMP_H, COMP_FROM_DPGAP_OUT, SampleRgisterNVol + VolOffsetmV, \
            COMP_FilterWIN);
    COMP_Init(COMP_L, COMP_FROM_DPGAP_OUT, SampleRgisterNVol - VolOffsetmV, \
            COMP_FilterWIN);

    /* Select the channel A/B output of PWM1 respectively */
    PIN_SetChannel(PIN_GPIO12, PIN_GPIO12_PWM1A);
    PIN_SetChannel(PIN_GPIO13, PIN_GPIO13_PWM1B);

    PWM_InitComplementaryPairChannel(PWM1, PWM_FREQ, PWM_DB_NS);

    /* Set PWM1A output 25% duty waveform */
    u32PWMPeriod = PWMPeriod(PWM_FREQ);
    PWM_SetCMPA(PWM1, (u32PWMPeriod * 3) / 4);

    /* Start counting */
    PWM_RunCounter(PWM1);

    /* Set the DC sub-module for PWM1 */
    PWMx_Set_TZ_Event(PWM1);

    NVIC_EnableIRQ(PWM1TZ_IRQn);

    while (1)
    {
        Delay_Ms(1000);

        /* Restore the output of wave in oneshot mode*/
        PWM_ClearTripInt(PWM1, TRIP_INT_OST);
    }
}

void PWM1TZ_IRQHandler(void)
{
    if (PWM_GetOneShotTripEventFlag(PWM1, TRIP_EVENT_DCAEVT))
    {
        printf("one-shot trip event occurred\n");

        PWM_ClearOneShotTripEventFlag(PWM1, TRIP_EVENT_DCAEVT);
    }
    PWM_ClearTripInt(PWM1, TRIP_INT_GLOBAL);
}
```

[1] 示例代码适用于 SPC1169，其它系列产品的示例代码会根据实际需求进行补充。

除了使用 PGA 进行过电流保护外，SPD1179 内部集成了针对三相电流桥上下桥臂外部 Mos 管的过流保护电路（Vds-monitor），相比 PGA 的实现方式，其代码配置大大简化，详细说明见《预驱使用指南》。

4 SPC1169 系列

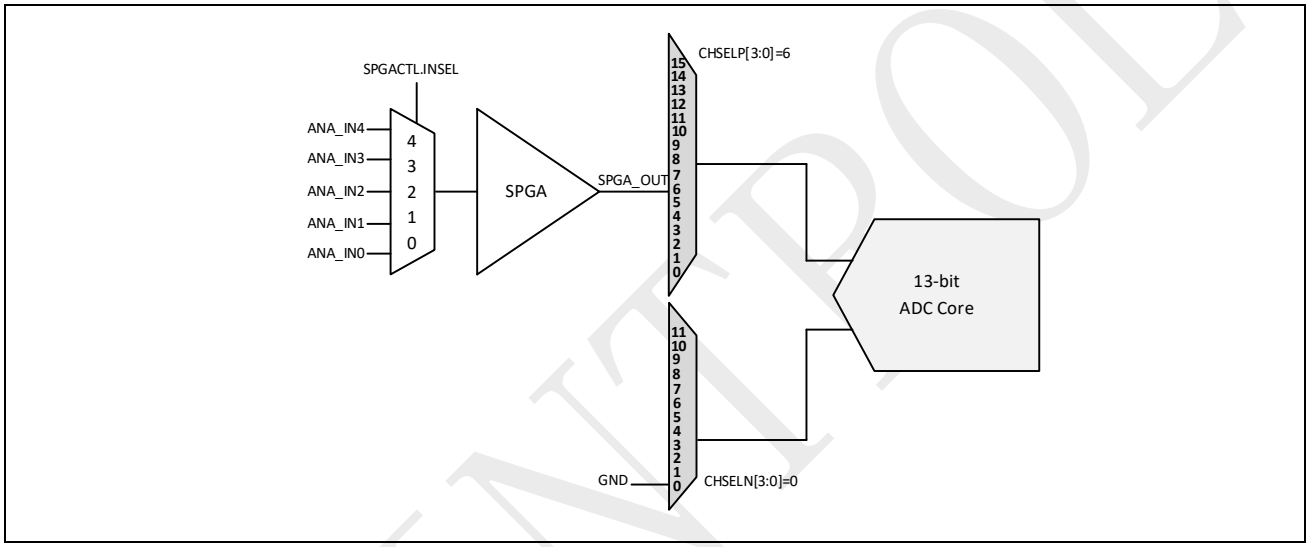
4.1 特性

实现了一个单端可编程增益放大器（PGA）。

4.2 功能描述

使用 ADC 对 SPGA 采样，其连接如图 4-1 所示。

图 4-1：ADC SPGA 采样



SPGA 的输出可以通过以下公式计算：

$$OUT = GAIN \times IN$$

- IN 电压必须为正；
- 为了避免 SPGA 处于饱和区域，SPGA 的输出范围需要在 0.3V 到 VDD3-0.3V 之间，并且输入范围在 0V 到 VDD3-1.3V 之间。下表显示了不同增益设置下的输入和输出范围（其中 VDD3 通常为 3.3V）。

表 4-1：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

4.3 功能实例

4.3.1 ADC 对 SPGA 采样

4.3.1.1 功能需求

使用 ADC 对 SPGA 电压进行采样。

4.3.1.2 功能实现

以下例子演示使用 ADC 对 SPGA 电压进行采样，将 SPGA 电压送给 ADC CH0 进行采样，默认采样时间，默认转换时间。但在实际的工程中采样时间会受到外部负载的影响，其具体的设置数值，可参考《ADC 建立时间计算方法使用指南》，而转换时间则不会受到外部负载影响，通常保持 SDK 中设定的数值即可。

表 4-2：代码路径

产品类型	代码路径
SPC1169 系列	0_Examples\ 10_9_ADC_With_SPGA