Application Note

SPC11X8/SPD11X8 PGA 和 COMP 使用指南

2021 年 11 月 – 版本 2

目录

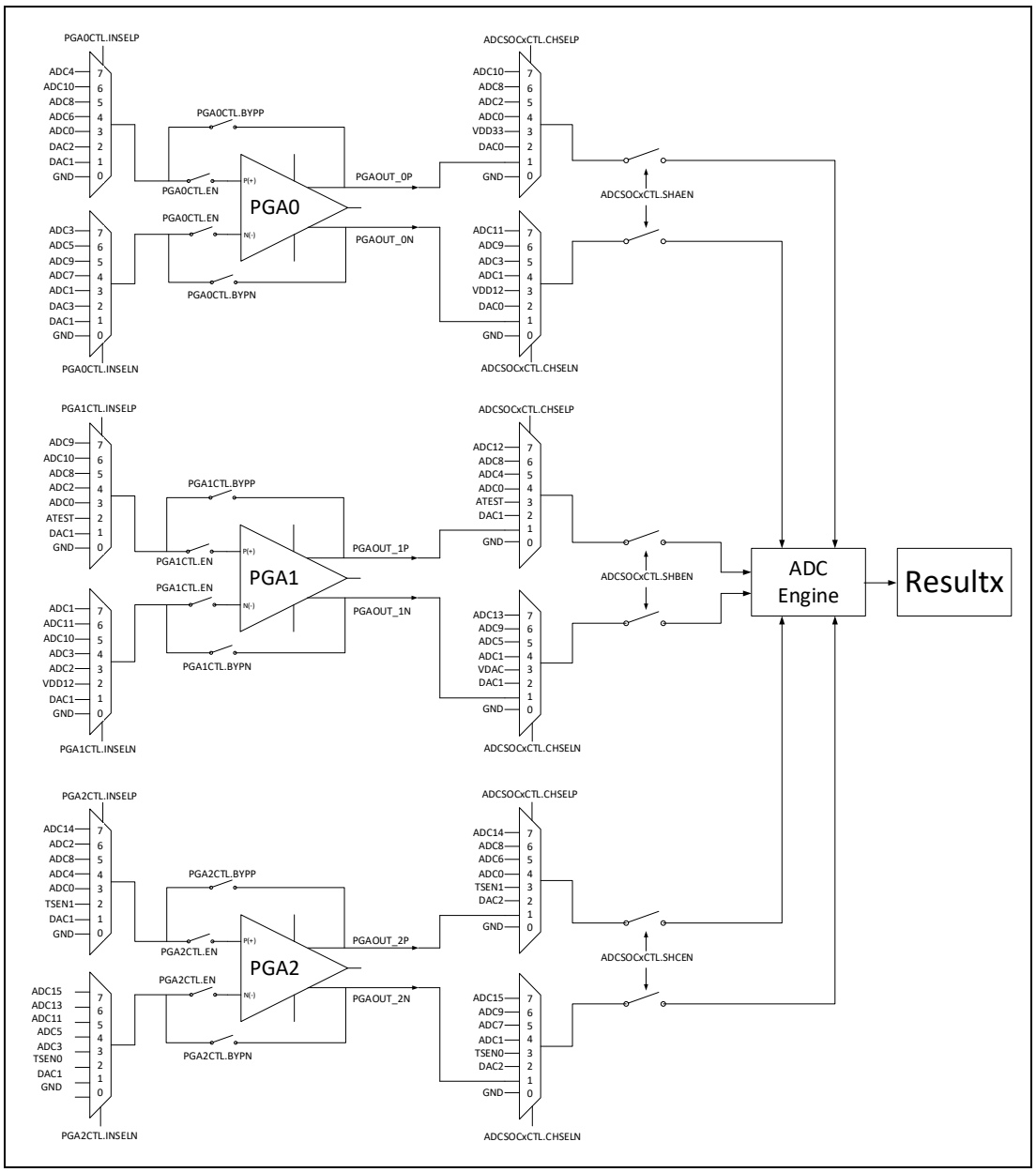
表格列表

图片列表

1 概述

SPC11X8/SPD11X8 内部集成了 3 组可编程增益的内部运放（PGA），每一组 PGA，配置两个比较器（Comparator），提供讯号 Too High 与 Too Low 时的两种保护。每一个比较器又可以致使 PWM 输出停止（PWM trip zone），可轻易达成过电流防护功能，下图集成了 SPC11X8/SPD11X8 中 PGA 和 ADC 两个重要的仿真器件之讯号流图（Signal Flow）。

图 1-1: 仿真器件信号流图



在阅读完本章后，您将可以了解：

- 如何利用差分 PGA 放大电流讯号；
- 如何使用 COMP 侦测过电流讯号，以及如何利用此过电流讯号停止 PWM。

2 PGA 使用教学

差分 ADC 可配置成常见的单端 ADC 使用，建议采用 SDK 内提供的 ADC_EasyInit1（）函数进行设定。SPC11X8/SPD11X8 集成了 3 组可调增益的差分运放（Differential PGA），可抗拒共模噪声干扰，亦可仿真为单端 PGA 使用。SPC11X8/SPD11X8 支持将其中一组 PGA 拆分成两个独立的单端 PGA 使用。SPC11X8/SPD11X8 还支持 SENSOR 模式下的差分放大。

PGA 的基本特点如下：

- 差分模式时放大增益
  2X 4X 8X 16X 24X 32X 48X 64X
- 单端模式时放大增益
  1X 2X 4X 8X 12X 16X 24X 32X
- 弹性的信道选择
- 输出直接链接 ADC 采样通道
- 输入与输出均可输入比较器 COMP
- 输出范围为 0.3V ~ VDDX-0.3V
- 在差分模式增益为 64 倍下，有效位数(ENOB)能达到 9 比特。

在本章节中，将介绍：

- 实际放大电流范例，利用差分模式放大电流
- MCU 建议的脚位排法
- PGA SDK API 参考表

2.1 PGA 单元概述

PGA 为可变增益运放，其仿真电路构架图见上图 1-1: 仿真器件信号流图。

2.2 PGA 驱动函数

SPC11X8/SPD11X8 提供了操作 PGA 存储器的驱动函数，如表 2-1 和表 2-2 所示。

表 2-1: PGA 相关宏定义

宏名	功能及说明
PGA_WALLOW()	使能 PGA 寄存器写操作
PGA_WDIS()	禁止 PGA 寄存器写操作

表 2-2: PGA 驱动函数

函数名	功能及说明
Void PGA_SelectPositiveChannelAsCommonInput (PGA_NumEnum ePGA)	选择正极作为共模电压 ePGA：PGA 模组
void PGA_SelectNegativeChannelAsCommonInput (PGA_NumEnum ePGA)	选择负极作为共模电压 ePGA：PGA 模组
void PGA_DifferentialInit (PGA_NumEnum ePGA, PGA_CH_P ePositiveCH, PGA_CH_N eNegativeCH, PGA_SCALE eDiffGain)	将 PGA 初始化为差分模式 ePGA：PGA 模组 ePositiveCH：PGA 正极输入 eNegativeCH：PGA 负极输入 eDiffGain：增益倍数

2.3 利用差分 PGA 放大电流信号

Spintrol 提供了 PGA 驱动函数，只需定义好通道选择就可以使用。下面是一个利用 PGA 放大 shunt resistor 上电阻的实际范例。

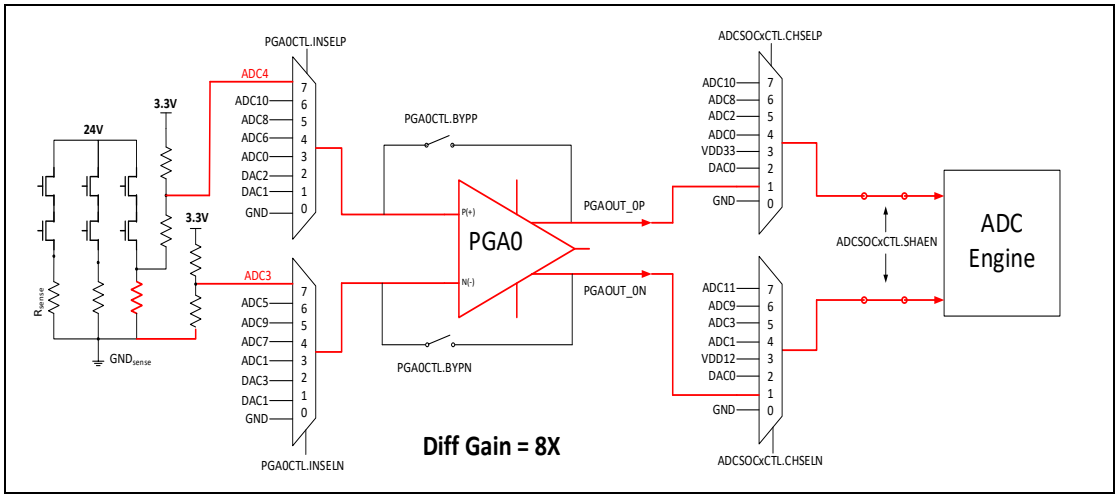
Example Code

```
void PGA_InitExample2_3(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_4); /* ADC4 */
  GPIO_SetPinAsAnalog(GPIO_3); /* ADC3 */
  /* Init PGA0 */
  PGA_DifferentialInit( PGA0,
            PGA0_CH_P_ADC4 /* Positive CH   */,
            PGA0_CH_N_ADC3 /* Negative CH   */,
            PGA_SCALE_8X   /* PGA Diff Gain */);
  /* Init ADC in 2 channel mode */
  ADC_EasyInit2(ADC_SOC_0,ADCx_PGA0P,ADCx_PGA0N,ADCTRIG_Software);
}
```

请注意，使用 PGA 之前请先将管脚初始化为模拟管脚。上述代码对应于实际硬件配置如图 2-1

所示。必须将信号的正端与负端均偏置到$\frac{VDD}{2}$的地方，请注意以下的电路，信号在偏置时会变成$\frac{1}{2}$倍，请见后方信号流程细节。

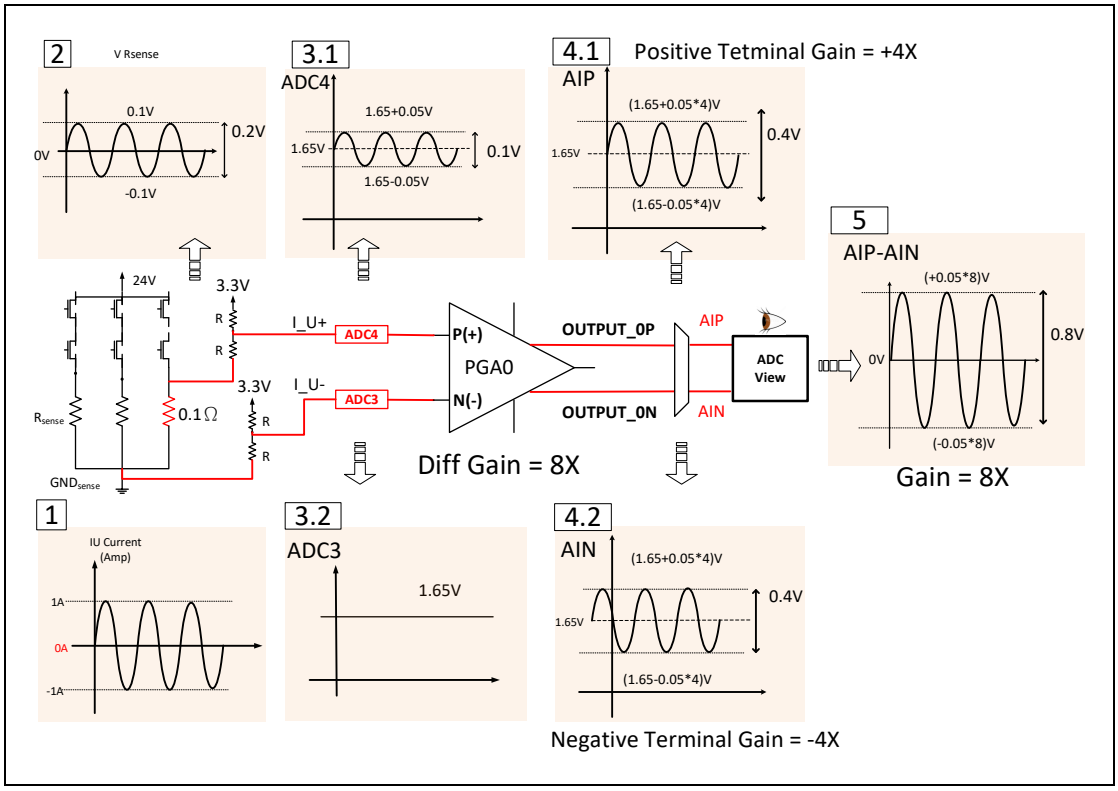
图 2-1: PGA 放大信号示意图



以下以一个例子分析内部电压。请注意：

- 假设 shunt resistor 为 0.1 欧姆，电流振幅为 1A
- 设定的差分放大增益是 8X
- 请注意电压在不同节点的摆幅
- R 建议为 10k 欧姆

图 2-2: PGA 差分模式放大信号细节流程图



以下讲解讯号放大之细部流程，编号对应上图。

（1） 假设 U 相电流为 1A 振幅之 sine 波

（2） 此电流通过采样电阻（0.1Ω）后，可得到 0V 附近摆动，振幅为0.1V的电压 sine 波。电流讯号（V Rsense）与地（GND）在进入 MCU 前，务必抬升到$\frac{VDD}{2} = 1.65V$。

（3） 1. 在进入芯片前，请用两个电阻将电压抬升到$\frac{VDD}{2}$，也就是说在进入芯片前，必须要将正端讯号拉升到$\frac{VDD}{2}$附近，否则 PGA 无法正常以差分模式进行放大，推荐的拉升电阻为 10K 欧姆。注意此时的电压摆幅为0.05V。

2. 因为本范例是差分讯号放大，所以必须将 GND 讯号也做一个拉升，如上图所示。也就是说在进入芯片前，负端讯号也必须拉升到$\frac{VDD}{2}$，以本例来说，即在1.65V。

（4） 1. 本范例设定的 PGA 差分增益为 8X，实际上芯片内部是以两个 OP 实现之，内部两个 OP 的增益都是 4X，负责放大正端讯号的 OP 是 4X，负责放大负端讯号的 OP 是-4X。必须注意的是正端与负端的放大倍率会差一个负号。可以发现讯号在 PGA 正端的输出仍是在1.65V附近摆荡，但是幅值已经达到了 4 倍，也就是$0.05\text{V} * 4 = 0.2\text{V}$，峰峰值为 0.4V。

2. 注意负责放大负端讯号的 OP 是-4X。所以他会是一个输出在1.65V附近摆荡，振幅放大四倍的讯号，但是他的相位会跟正端讯号刚好相反。

（5） ADC 选择输入为 PGA 的正端与负端讯号，因此以 ADC 的角度来看，其实是看到

$$\text{AIP} - \text{AIN} = (1.65 + 0.05\text{V} * 4) - (1.65 - 0.05\text{V} * 4) = 0.05\text{V}(4 + 4) = 0.4\text{V}$$

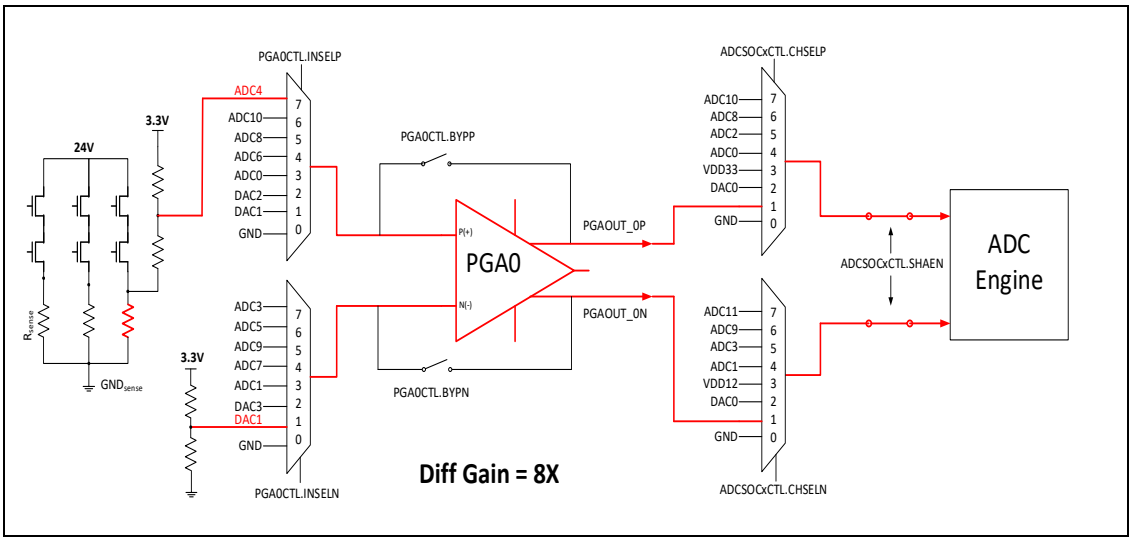
的讯号，也就是说在 ADC 端，看的到讯号是一个在 0V 附近摆荡，但是振幅已经来到了 8 倍。

（6） 从 Shunt resistor 的信号出发，到 ADC 所看到的讯号，实际上是放大了 4 倍，这是因为偏置到$\frac{VDD}{2}$之后，信号会变为$\frac{1}{2}$。

2.4 差分 PGA 模拟为单端 PGA 使用

由上一章节可知，差分 PGA 的负端讯号其实是需要一个 1.65V 的讯号，但若是共模噪声在应用中不明显，使用者可以透过芯片内部的 DAC 制造此 1.65V，可以节省 PIN 脚的使用。

图 2-3: PGA 单端模式示意图



Example Code

```
void PGA_InitExample2_4(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_4); /* ADC4 */
  /* Init PGA0 */
  PGA_DifferentialInit( PGA0,
            PGA0_CH_P_ADC4     /* Positive CH          */,
            PGA0_CH_N_DAC1 /* Negative CH :
                              DAC1 output */,
            PGA_SCALE_8X       /* PGA Diff Gain        */);
  /* Config DAC1 */
  COMP->DAC1CTL.bit.EN = 1;  //enable DAC1
  COMP->DAC1CODE.bit.VAL = 512;  // set DAC1 output to 1.65V
  /* Init ADC in 2 channel mode */
  ADC_EasyInit2(ADC_SOC_0,ADCx_PGA0P,ADCx_PGA0N,ADCTRIG_Software);
}
```

请注意 PGA 的负端输入通道，以改为 DAC1 之输出，DAC CODE 设定为 512（满幅为 10bit=1024），可输出 1.65V ($DAC_{output} = \frac{CODE}{1024} * AVDD$)。

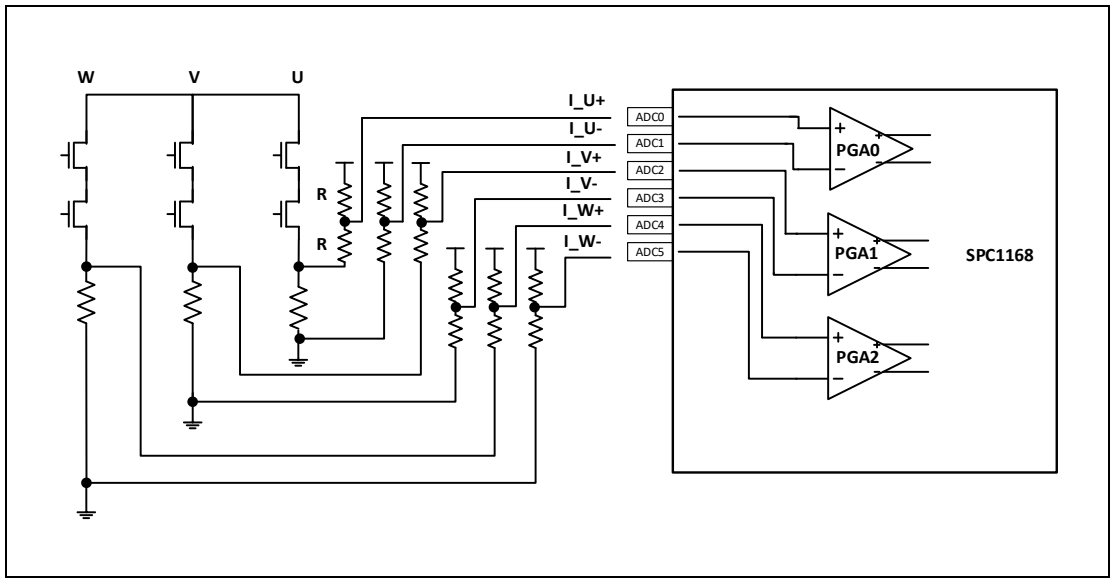
2.5 MCU 管脚建议排法

SPC11X8/SPD11X8 的 PGA 信道配置弹性，用户可根据需求自行调配，但建议可根据以下配置进行不同采样应用的搭配。

■ **3 shunt 电阻采样，真正差分讯号**

在这个范例中，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 2-4: 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。图 2-4 对应的范例代码如下：

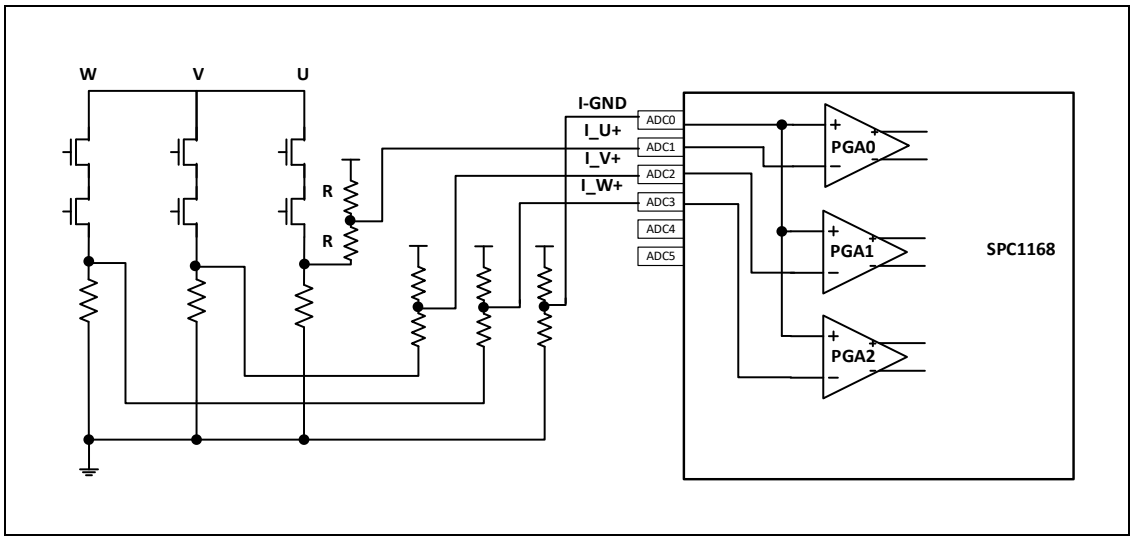
Example Code

```
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
   *
   * 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_U+ */
  GPIO_SetPinAsAnalog(GPIO_1); /* ADC1 I_U- */
  GPIO_SetPinAsAnalog(GPIO_2); /* ADC2 I_V+ */
  GPIO_SetPinAsAnalog(GPIO_3); /* ADC3 I_V- */
  GPIO_SetPinAsAnalog(GPIO_4); /* ADC4 I_W+ */
  GPIO_SetPinAsAnalog(GPIO_5); /* ADC5 I_W- */

/* Init PGA */
  PGA_DifferentialInit( PGA0,
           PGA0_CH_P_ADC0 /* Positive CH   */,
           PGA0_CH_N_ADC1 /* Negative CH   */,
           PGA_SCALE_8X   /* PGA Diff Gain */);
  PGA_DifferentialInit( PGA1,
           PGA1_CH_P_ADC2 /* Positive CH   */,
           PGA1_CH_N_ADC3 /* Negative CH   */,
           PGA_SCALE_8X   /* PGA Diff Gain */);
  PGA_DifferentialInit( PGA2,
           PGA2_CH_P_ADC4 /* Positive CH   */,
           PGA2_CH_N_ADC5 /* Negative CH   */,
           PGA_SCALE_8X   /* PGA Diff Gain */);
  /* Init ADC in 2 channel mode */
  ADC_EasyInit2(ADC_SOC_0,ADCx_PGA0P,ADCx_PGA0N,ADCTRIG_Software);
  ADC_EasyInit2(ADC_SOC_1,ADCx_PGA1P,ADCx_PGA1N,ADCTRIG_Software);
  ADC_EasyInit2(ADC_SOC_2,ADCx_PGA2P,ADCx_PGA2N,ADCTRIG_Software);
}
```

■ **3 shunt 电阻采样，共享 GND 讯号**

图 2-5: 3 shunt 电阻共 GND 采样示意图



ADC0 是三个 PGA 负端输入都共有的端点，可作为三个讯号的共地使用。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 V_GND 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。图 2-5 对应的范例代码如下：

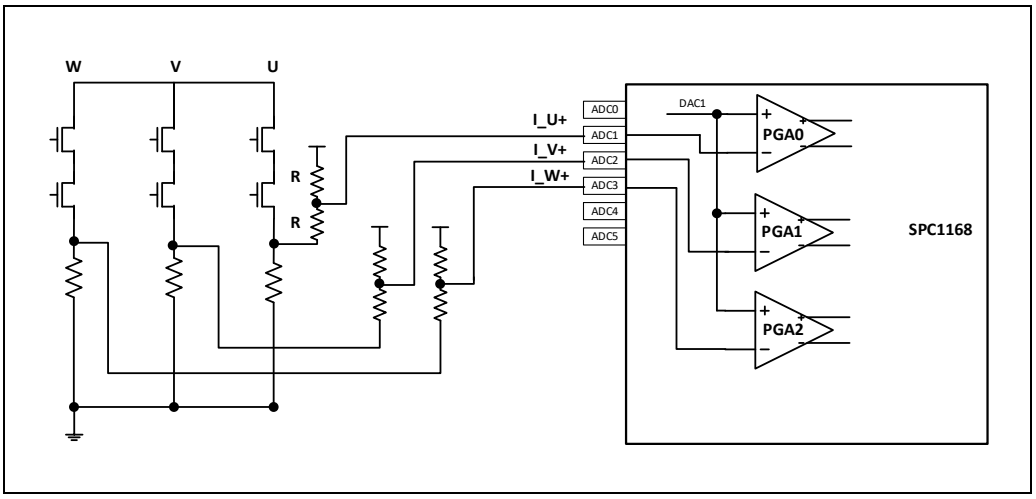
Example Code

```
void PGA_InitExample2_5_2(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 Common Positive */
  GPIO_SetPinAsAnalog(GPIO_1); /* ADC1 I_U+ */
  GPIO_SetPinAsAnalog(GPIO_2); /* ADC2 I_V+ */
  GPIO_SetPinAsAnalog(GPIO_3); /* ADC3 I_W+ */
  /* Init PGA */
  PGA_DifferentialInit( PGA0,
          PGA0_CH_P_ADC0 /* Positive CH   */,
          PGA0_CH_N_ADC1 /* Negative CH   */,
          PGA_SCALE_8X   /* PGA Diff Gain */);
  PGA_DifferentialInit( PGA1,
          PGA1_CH_P_ADC0 /* Positive CH   */,
          PGA1_CH_N_ADC2 /* Negative CH   */,
          PGA_SCALE_8X   /* PGA Diff Gain */);
  PGA_DifferentialInit( PGA2,
          PGA2_CH_P_ADC0 /* Positive CH   */,
          PGA2_CH_N_ADC3 /* Negative CH   */,
          PGA_SCALE_8X   /* PGA Diff Gain */);
  /* Init ADC in 2 channel mode */
  ADC_EasyInit2(ADC_SOC_0,ADCx_PGA0P,ADCx_PGA0N,ADCTRIG_Software);
  ADC_EasyInit2(ADC_SOC_1,ADCx_PGA1P,ADCx_PGA1N,ADCTRIG_Software);
  ADC_EasyInit2(ADC_SOC_2,ADCx_PGA2P,ADCx_PGA2N,ADCTRIG_Software);
}
```

■ **3 shunt 电阻采样，使用内部 DAC 输出 VDD/2**

图 2-6: 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。以上配置请参考如下范例码：

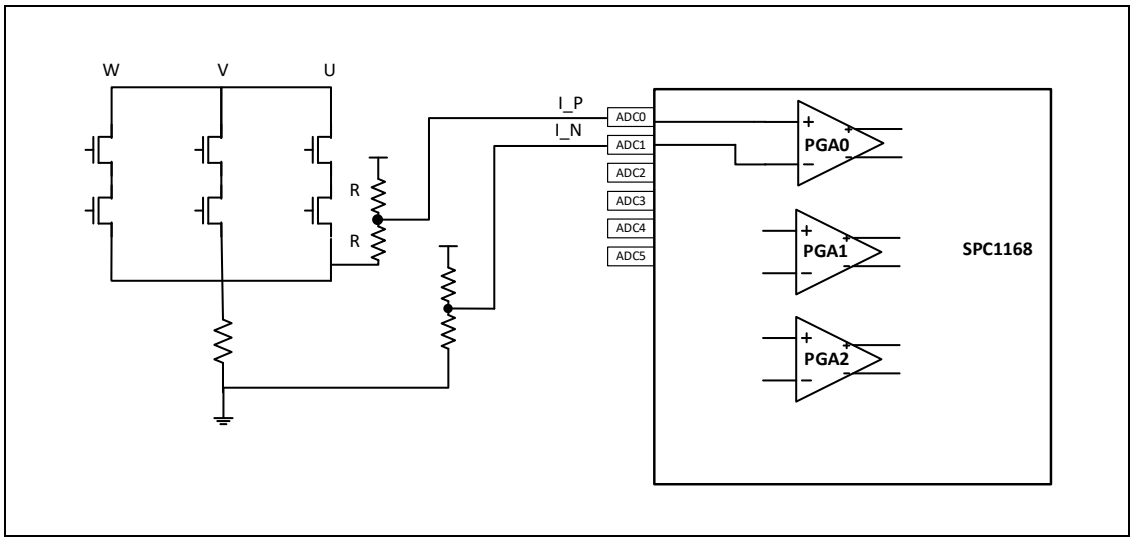
Example Code

```
void PGA_InitExample2_5_3(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_1); /* ADC1 I_U+ */
  GPIO_SetPinAsAnalog(GPIO_2); /* ADC2 I_V+ */
  GPIO_SetPinAsAnalog(GPIO_3); /* ADC3 I_W+ */
  /* Init PGA */
  PGA_DifferentialInit( PGA0,
          PGA0_CH_P_DAC1    /* Positive CH   */,
          PGA0_CH_N_ADC1 /* Negative CH   */,
          PGA_SCALE_8X      /* PGA Diff Gain */);
  PGA_DifferentialInit( PGA1,
          PGA1_CH_P_DAC1    /* Positive CH   */,
          PGA1_CH_N_ADC2 /* Negative CH   */,
          PGA_SCALE_8X      /* PGA Diff Gain */);
  PGA_DifferentialInit( PGA2,
          PGA2_CH_P_DAC1    /* Positive CH   */,
          PGA2_CH_N_ADC3 /* Negative CH   */,
          PGA_SCALE_8X      /* PGA Diff Gain */);
  /* Config DAC1 */
   COMP->DAC1CTL.bit.EN = 1;  //enable DAC1
     COMP->DAC1CODE.bit.VAL = 512;  // set DAC1 output to 1.65V
  /* Init ADC in 2 channel mode */
  ADC_EasyInit2(ADC_SOC_0,ADCx_PGA0P,ADCx_PGA0N,ADCTRIG_Software);
  ADC_EasyInit2(ADC_SOC_1,ADCx_PGA1P,ADCx_PGA1N,ADCTRIG_Software);
  ADC_EasyInit2(ADC_SOC_2,ADCx_PGA2P,ADCx_PGA2N,ADCTRIG_Software);
}
```

2.6 单电阻采样配置

图 2-7: 单电阻采样电路图



上图对应的范例代码如下：

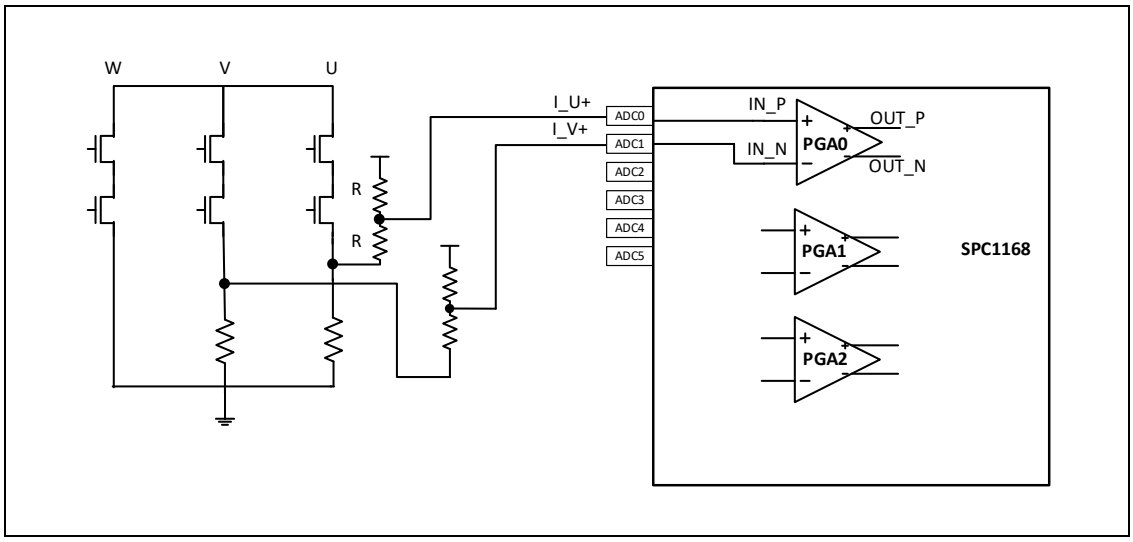
Example Code

```
void PGA_InitExample2_6_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 */
  PGA_DifferentialInit( PGA0,
          PGA0_CH_P_ADC0    /* Positive CH   */,
          PGA0_CH_N_ADC1    /* Negative CH   */,
          PGA_SCALE_8X      /* PGA Diff Gain */);

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

2.7 将一个 PGA 拆分成两个单端 PGA 配置

图 2-8: 单端 PGA 采样电路图



对于精度要求不高的应用，可以将一个 PGA 拆分成两个单端 PGA 使用，比如本例中将 PGA0 用作两个单端的 PGA，其中$OUT_P = IN_P * Gain_P, OUT_N = IN_N * Gain_N$.

上图对应的范例代码如下：

Example Code

```
void PGA_InitExample2_7_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);
}
```

3 Comparator 使用教学

3.1 Comparator 单元概述

SPC11X8/SPD11X8 共有五组模拟比较器（COMP0~COMP4），特点如下：

- 其中有三组 COMP 分别与三个 PGA 绑定，每个 PGA 搭配一组 COMP
  一组 COMP 中包含两个 COMP，一个作为信号 Too High 检测，一个作为 Too Low 检测
  另外两组 COMP 作为扩展，支持更多应用，比如双电机。
- 可利用内部 DAC (共有四个 10bit 的 DAC 可选择) 调整 COMP 比较阀值
- 支持 phase comparator 功能，可以将 PGA 的正端输出和负端输出分别送到 COMP 的正端输入和负端输入进行比较。
- COMP 输出具数字滤波功能（Deglitch or debounce）可避免 PWM 开关之噪声造成误动作。请注意最长滤波时间限制。
- 具磁滞功能，输出可反转
- 任意一个 COMP 输出可同步关断所有 PWM 输出信号

3.2 Comparator 驱动函数

表 3-1: Comparator 宏定义

宏名	功能及说明
COMP_ClearFilterOutputStatus(eCOMP)	清除对应 COMP 通道的滤波器锁存状态 eCOMP：通道号
COMP_ClearAllFilterOutputStatus()	清除所有滤波器锁存的状态
COMP_GetFilterOutputStatus(eCOMP)	获取对应通道滤波器锁存的状态 eCOMP：通道号
COMP_GetRawFilterOutputStatus(eCOMP)	获取对应通道未经锁存的状态 eCOMP：通道号
COMP_EnableDACBuffer()	使能 DAC Buffer
COMP_DisableDACBuffer()	关闭 DAC Buffer
COMP_WALLOW()	使能 Comparator 寄存器写操作
COMP_WDIS()	禁止 Comparator 寄存器写操作

表 3-2: Comparator 驱动函数

函数名	功能及说明
void COMP_Enable(COMP_ComparatorSelEnum eCOMP)	使能比较器通道 eCOMP：要使能的比较器通道
void COMP_Disable(COMP_ComparatorSelEnum eCOMP)	关闭比较器通道 eCOMP：需要关闭的比较器通道
void COMP_ResetFilter(COMP_ComparatorSelEnum eCOMP)	复位比较器通道 eCOMP：需要复位的比较器通道
void COMP_EnableOutputInvert(COMP_ComparatorSelEnum eCOMP)	使能比较器事件触发时输出反转（默认输出 1，表示比较器事件触发，使能后，输出 0） eCOMP：比较器通道
void COMP_DisableOutputInvert(COMP_ComparatorSelEnum eCOMP)	关闭比较器事件触发时输出反转 eCOMP：比较器通道
void COMP_EnablePWMSyncClearFilterOutputStatus(COMP_ComparatorSelEnum eCOMP)	使能 PWM 同步信号清除比较器的锁存状态 eCOMP：比较器通道
void COMP_DisablePWMSyncClearFilterOutputStatus(COMP_ComparatorSelEnum eCOMP)	关闭 PWM 同步信号清除比较器的锁存状态 eCOMP：比较器通道

函数名	功能及说明
void COMP_SetSyncEvent(COMP_ComparatorSelEnum eCOMP, uint8_t ePWMIndex)	设置清除比较器的锁存状态的 PWM 同步信号来源 eCOMP：比较器通道 ePWMIndex： PWM 通道号，参见 PWM_SelEnum 类型定义
void COMP_EnableDAC(COMP_DACSelEnum eDACSel)	使能 DAC 作为比较器输入 eDACSel：需要使能的 DAC 通道
void COMP_DisableDAC(COMP_DACSelEnum eDACSel)	关闭 DAC 作为比较器输入 eDACSel：需要关闭的 DAC 通道
void COMP_SetDACValue10Bit(COMP_DACSelEnum eDACSel, uint32_t u32Code)	设置 DAC 输出的电压数值 eDACSel：DAC 通道 u32Code：输出电压对应的码值
void COMP_SetDACVoltage (COMP_DACSelEnum eDACSel, int32_t i32ValueMV)	设置 DAC 输出电压值 eDACSel：DAC 通道 i32ValueMV：需要输出的电压值
void COMP_SetDACSyncEvent(COMP_DACSelEnum eDACSel, uint8_t ePWMIndex)	设置 DAC 数值加载模式的事件触发源 eDACSel：DAC 通道 ePWMIndex： PWM 通道号，参见 PWM_SelEnum 类型定义
void COMP_SetDACCodeLoadTiming(COMP_DACSelEnum eDACSel, uint8_t eLoadMode)	设置 DAC 数值加载时机 eDACSel：DAC 通道 eLoadMode：加载模式
void COMP_Init(COMP_ComparatorSelEnum eCOMP, COMP_CHSelEnum eCH, int32_t i32DACVoltageMV, uint32_t u32DeglicthTimeNs)	初始化比较器 eCOMP：比较器模组 eCH：比较器输入通道 i32DACVoltageMV：DAV 要输出的电压值 u32DeglicthTimeNs：滤波器窗口长度
void COMP_SelectChannel(COMP_ComparatorSelEnum eCOMP, COMP_CHSelEnum eCH)	初始化比较器 eCOMP：比较器模组 eCH：比较器输入通道
void COMP_SetOutputType(COMP_ComparatorSelEnum eCOMP, COMP_OutputSelEnum eOutputSel)	设置事件触发时输出给 PWM 的信号模式 eCOMP：比较器模组 eOutputSel：比较器输出模式
void COMP_SetFilterWindow(COMP_ComparatorSelEnum eCOMP, uint8_t u8Size,	设置过滤器的窗口长度及阈值 eCOMP：比较器模组 u8Size：滤波器窗口长度

函数名	功能及说明
uint8_t u8Threshold)	u8Threshold：滤波器阈值
void COMP_SetFilterWindowTimeNs(COMP_ComparatorSelEnum eCOMP, uint32_t u32SizeTimeNS, uint32_t u32ThresholdTimeNS)	设置过滤器窗口时长及阈值 eCOMP：比较器模组 u32SizeTimeNS：滤波器窗口时长 u32ThresholdTimeNS：滤波器阈值（以时间为单位）
void COMP_SetFilterClockDiv(COMP_ComparatorSelEnum eCOMP, uint16_t u16ClkDiv)	设置过滤器的工作时钟分频系数 eCOMP：比较器模组 u16ClkDiv：时钟分频系数
void COMP_SetHysteresis(COMP_ComparatorSelEnum eCOMP, COMP_HystSelEnum eHystSel)	设置比较器延滞数值 eCOMP：比较器模组 eHystSel：延滞数值
void COMP_DACBufferInit(COMP_DACSelEnum eDACSel, BOOL bIsOut10, BOOL bIsOut13)	使能 DAC Buffer eDACSel：DAC 模组 bIsOut10：是否将 Buffer 输出到 GPIO10 bIsOut13：是否将 Buffer 输出到 GPIO13

3.3 COMP 模拟架构

COMP0 的模拟架构如图 3-1 所示，其余 COMP1~COMP4 架构和 COMP0 相同，其输入端选择见表 3-3 所示。

图 3-1: COMP0 模拟架构

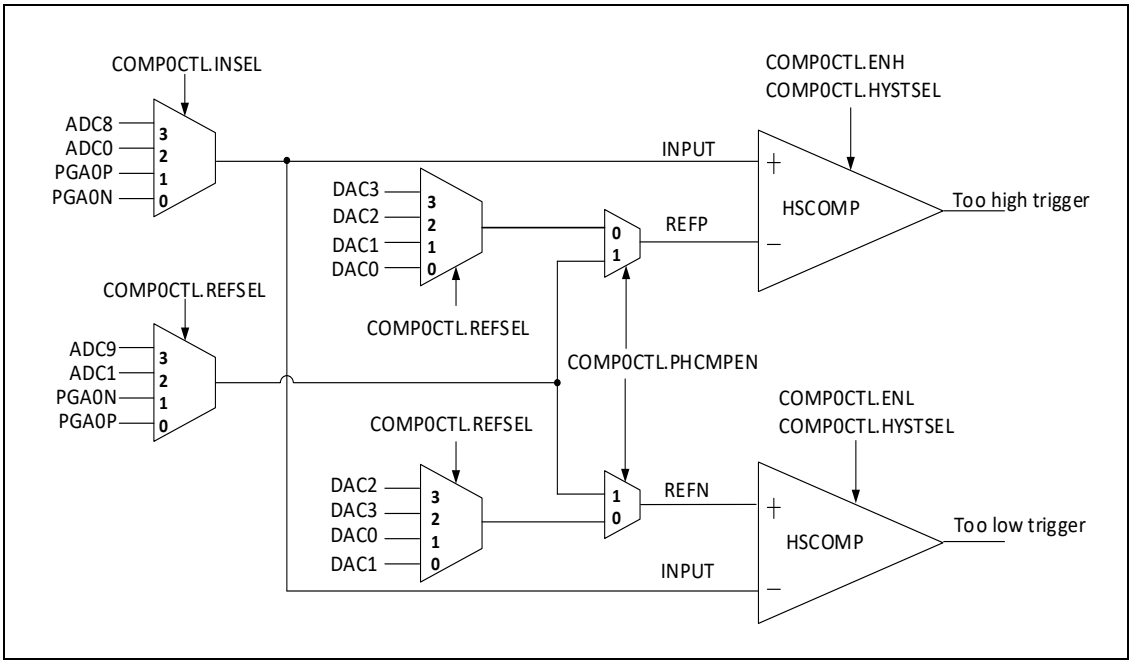


表 3-3: Comparator 0~4 输入选择

Comparator0 MUX								
INSEL	**INPUT**	**PHCOMPEN**	**REFSEL**	**REFP/N**	**PHCOMPEN**	**REFSEL**	**REFP**	**REFN**
3	ADC8	1	3	ADC9	0	3	DAC3	DAC2
2	ADC0	1	2	ADC1	0	2	DAC2	DAC3
1	PGA0P	1	1	PGA0N	0	1	DAC1	DAC0
0	PGA0N	1	0	PGA0P	0	0	DAC0	DAC1
Comparator1 MUX								
INSEL	**INPUT**	**PHCOMPEN**	**REFSEL**	**REFP/N**	**PHCOMPEN**	**REFSEL**	**REFP**	**REFN**
3	ADC8	1	3	ADC10	0	3	DAC3	DAC2
2	ADC0	1	2	ADC2	0	2	DAC2	DAC3
1	PGA1P	1	1	PGA1N	0	1	DAC1	DAC0
0	PGA1N	1	0	PGA1P	0	0	DAC0	DAC1
Comparator2 MUX								
INSEL	**INPUT**	**PHCOMPEN**	**REFSEL**	**REFP/N**	**PHCOMPEN**	**REFSEL**	**REFP**	**REFN**
3	ADC8	1	3	ADC11	0	3	DAC3	DAC2
2	ADC0	1	2	ADC3	0	2	DAC2	DAC3
1	PGA2P	1	1	PGA2N	0	1	DAC1	DAC0
0	PGA2N	1	0	PGA2P	0	0	DAC0	DAC1
Comparator3 MUX								
INSEL	**INPUT**	**PHCOMPEN**	**REFSEL**	**REFP/N**	**PHCOMPEN**	**REFSEL**	**REFP**	**REFN**
3	ADC12	1	3	ADC13	0	3	DAC3	DAC2
2	ADC8	1	2	ADC9	0	2	DAC2	DAC3
1	ADC4	1	1	ADC5	0	1	DAC1	DAC0
0	ADC0	1	0	ADC1	0	0	DAC0	DAC1
Comparator4 MUX								
INSEL	**INPUT**	**PHCOMPEN**	**REFSEL**	**REFP/N**	**PHCOMPEN**	**REFSEL**	**REFP**	**REFN**
3	ADC14	1	3	ADC15	0	3	DAC3	DAC2
2	ADC10	1	2	ADC11	0	2	DAC2	DAC3
1	ADC6	1	1	ADC7	0	1	DAC1	DAC0
0	ADC2	1	0	ADC3	0	0	DAC0	DAC1

3.4 COMP 应用于过电流防护实例

■ **范例码展示**

COMP 可选择 PGA 输出作为电流防护使用，以下为一个使用实例，以 PGA0 搭配 COMP0 为例，并且使 PWM 停止波形输出。

Example Code

```
void COMPandPGA_InitExample3_4_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 */
  GPIO_SetPinAsAnalog(GPIO_1); /* ADC1 */
  /* Init PGA0 */
  PGA_DifferentialInit(PGA0, PGA0_CH_P_ADC0, PGA0_CH_N_ADC1,
PGA_SCALE_8X);
  /* Init ADC in 2 channel mode */
  ADC_EasyInit2(ADC_SOC_0,ADCx_PGA0P,ADCx_PGA0N,ADCTRIG_Software);
  /* Init COMP0 High and Low */
  COMP_Init(COMP_0_HI,
            COMP0_FROM_PGA0P_OUT,  /* From PGA0P output */
            2450              /* DAC HI compare voltage mV */,
```

```
            300                 /* Output deglitch time(ns)  */);
  COMP_Init(COMP_0_LO,
            COMP0_FROM_PGA0N_OUT,  /* From PGA0N output */
            850                 /* DAC LO compare voltage mV */,
            300                 /* Output deglitch time(ns)  */);
/* When COMP0H or COMP0L equals 1, it can turn off 3 PWM pairs(6 PWM CH)
immediately */
/* Affected by results monitored from all three phases */
  PWM_EnableDCAHTripEvent(PWM1, DC_TRIP_COMP0H |
                              DC_TRIP_COMP0L );
  PWM_EnableDCAHTripEvent(PWM2, DC_TRIP_COMP0H |
                              DC_TRIP_COMP0L );
  PWM_EnableDCAHTripEvent(PWM3, DC_TRIP_COMP0H |
                              DC_TRIP_COMP0L );

  PWM_SetRawDCAEVT0(PWM1, DCH_HIGH_DCL_HIGH);
  PWM_SetRawDCAEVT0(PWM2, DCH_HIGH_DCL_HIGH);
  PWM_SetRawDCAEVT0(PWM3, DCH_HIGH_DCL_HIGH);

  PWM_SetOneShotTripEvent(PWM1, TRIP_EVENT_DCAEVT,
TRIP_OUTPUT_LATCH);
  PWM_SetOneShotTripEvent(PWM2, TRIP_EVENT_DCAEVT,
TRIP_OUTPUT_LATCH);
  PWM_SetOneShotTripEvent(PWM3, TRIP_EVENT_DCAEVT,
TRIP_OUTPUT_LATCH);

  /* Set the filtered event as the final DCAEVT0, and trigger TZ sub-model
when DCH is high */
  PWM_SetDCAEVT0(PWM1, DCEVT_FILTERED);
  PWM_SetDCAEVT0(PWM2, DCEVT_FILTERED);
  PWM_SetDCAEVT0(PWM3, DCEVT_FILTERED);

  /* Set output to tri-state upon DCAEVT0 trip event */
  PWM_SetCHAOutputWhenTrip(PWM1, TZU_TRIP_DO_NOTHING |
                              TZD_TRIP_DO_NOTHING |
                              DCEVT0U_TRIP_AS_TRI_STATE |
                              DCEVT0D_TRIP_AS_TRI_STATE |
                              DCEVT1U_TRIP_DO_NOTHING |
                              DCEVT1D_TRIP_DO_NOTHING);

  PWM_SetCHBOutputWhenTrip(PWM1, TZU_TRIP_DO_NOTHING |
                              TZD_TRIP_DO_NOTHING |
                              DCEVT0U_TRIP_AS_TRI_STATE |
                              DCEVT0D_TRIP_AS_TRI_STATE |
```

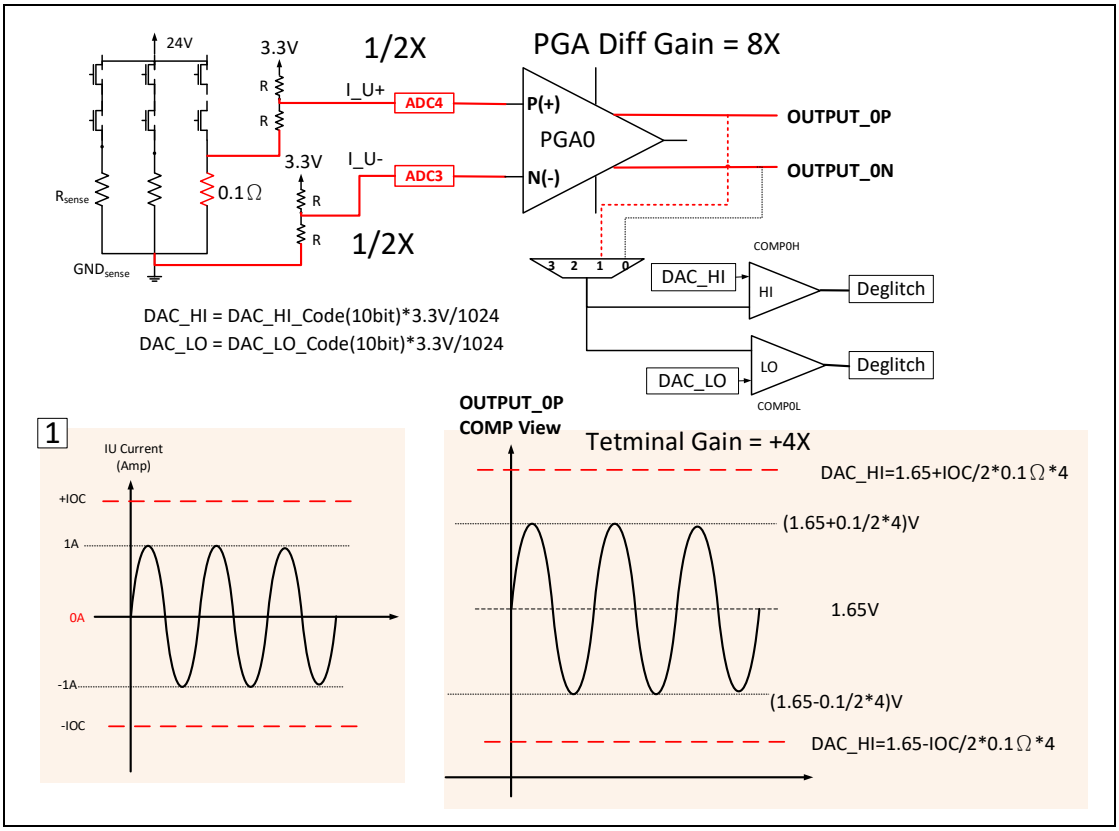
```
                                DCEVT1U_TRIP_DO_NOTHING |
                                DCEVT1D_TRIP_DO_NOTHING);
  PWM_SetCHAOutputWhenTrip(PWM2, TZU_TRIP_DO_NOTHING |
                                TZD_TRIP_DO_NOTHING |
                                DCEVT0U_TRIP_AS_TRI_STATE |
                                DCEVT0D_TRIP_AS_TRI_STATE |
                                DCEVT1U_TRIP_DO_NOTHING |
                                DCEVT1D_TRIP_DO_NOTHING);

  PWM_SetCHBOutputWhenTrip(PWM2, TZU_TRIP_DO_NOTHING |
                                TZD_TRIP_DO_NOTHING |
                                DCEVT0U_TRIP_AS_TRI_STATE |
                                DCEVT0D_TRIP_AS_TRI_STATE |
                                DCEVT1U_TRIP_DO_NOTHING |
                                DCEVT1D_TRIP_DO_NOTHING);
  PWM_SetCHAOutputWhenTrip(PWM3, TZU_TRIP_DO_NOTHING |
                                TZD_TRIP_DO_NOTHING |
                                DCEVT0U_TRIP_AS_TRI_STATE |
                                DCEVT0D_TRIP_AS_TRI_STATE |
                                DCEVT1U_TRIP_DO_NOTHING |
                                DCEVT1D_TRIP_DO_NOTHING);

  PWM_SetCHBOutputWhenTrip(PWM3, TZU_TRIP_DO_NOTHING |
                                TZD_TRIP_DO_NOTHING |
                                DCEVT0U_TRIP_AS_TRI_STATE |
                                DCEVT0D_TRIP_AS_TRI_STATE |
                                DCEVT1U_TRIP_DO_NOTHING |
                                DCEVT1D_TRIP_DO_NOTHING);
}
```

SPC11X8/SPD11X8 的 COMP 特点在于任何一个 COMP 讯号，均可触发任一 PWM 关断功能，上例为当 COMP0H 或是 COMP0L 产生讯号后，可立即马上关断三相六臂 PWM 讯号。以上代码同时设定了 COMP 与 PGA，ADC，COMP 的 DAC_HI 阀值为 2.45V，DAC_LOW 为 0.850V，设计原因请参考以下讲解。

■ 图形化说明

请注意 COMP 只能挑选一个 PGA 通道进行比较，而 PGA 若是选择差动增益，实际上单端增益只有差动增益的一半，因此设定 COMP 的比较阀值（DAC_{HI}与DAC_{LO}）时必须特别小心。如下图所示，PGA 设定差动增益为 8X，但是单端对地的增益只有 4X。

图 3-2:　COMP 范例（IOC=2A）



上图的解释如下：

（1）　假若输入是 1A 摆幅，采样电阻 0.1 欧姆，想设定的过电流防护 IOC 为 2A。

（2）　COMP0 选择 PGA0 的正端输出作为比较器的输入。

注意三组 COMP，可以共享相同的DAC_{HI}与DAC_{LO}。

（3）　假若 PGA 差动增益选择为 8X，但这是输出的正端对负端之增益，事实上单端对地的增益只有 4X，而 COMP 看到的增益亦只有 4X。

（4）　COMP 看到的电压摆幅只有 4 倍，因此必须设定 DAC 的范围如上图所示。

$$DAC_{HI} = 1.65 + 2A * \frac{0.1}{2} * 4 = 2.05V$$

$$DAC_{LO} = 1.65 - 2A * \frac{0.1}{2} * 4 = 1.25V$$

（5）　根据以下公式，可以决定DAC_{HI}与DAC_{LO}的设定数值

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

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

根据（4）的结果，可以计算出：

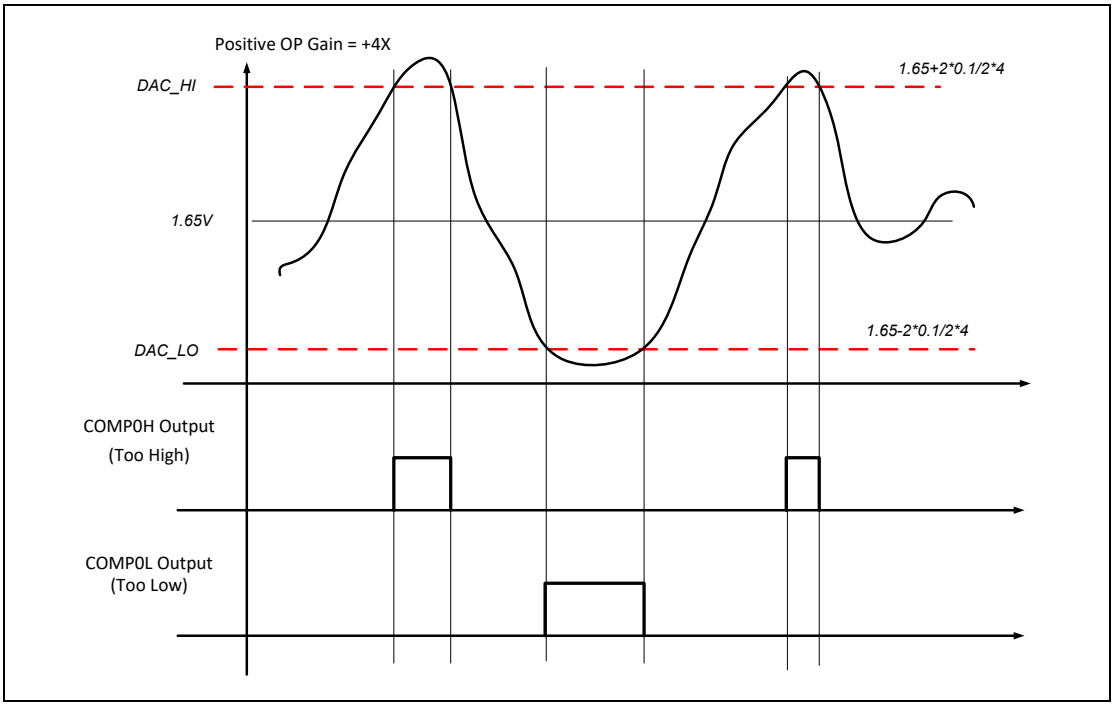
DAC_HI_CODE（寄存器数值）须设定为 636，DAC_LO_CODE（寄存器数值）须设定为 387。为了使用方便，Spintrol 提供之 COMP_Init（）函数可直接输入 mV 值进行设定。

（6）　COMP 的输出需要通过数字滤波器，可避免 PWM 开关之噪声误触 COMP。为了使用方便，Spintrol 提供之 COMP_Init（）函数可直接输入此滤波器之宽度，单位为 ns，此例设定的滤波时间长度为 300ns，可滤除 300ns 长度以下的噪声。详细配置请见下一章节之 API 介绍。

■ **COMP 输出演示**

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。

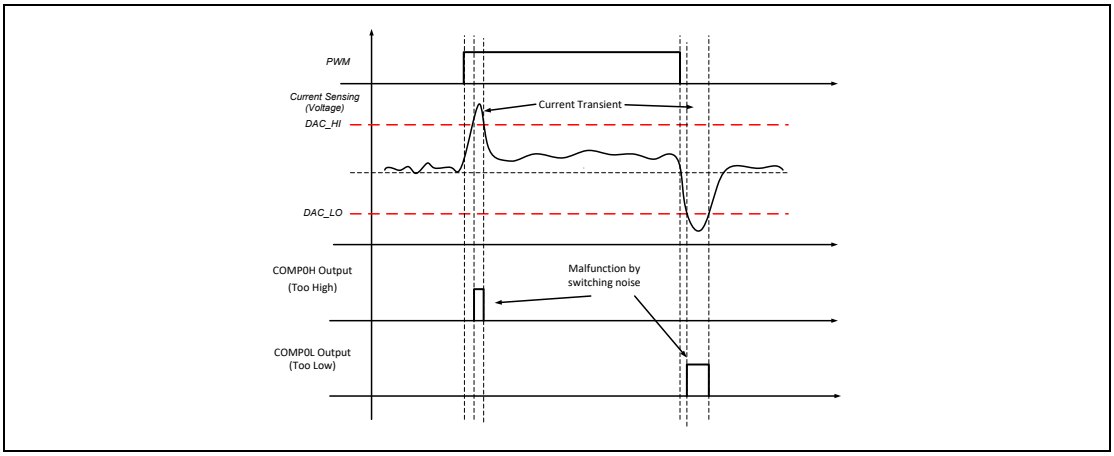
图 3-3: COMP 输出演示



■ **COMP 输出 Deglitch（Debounce）功能**

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），此干扰与瞬时是正常反应，如图 3-4 所示。

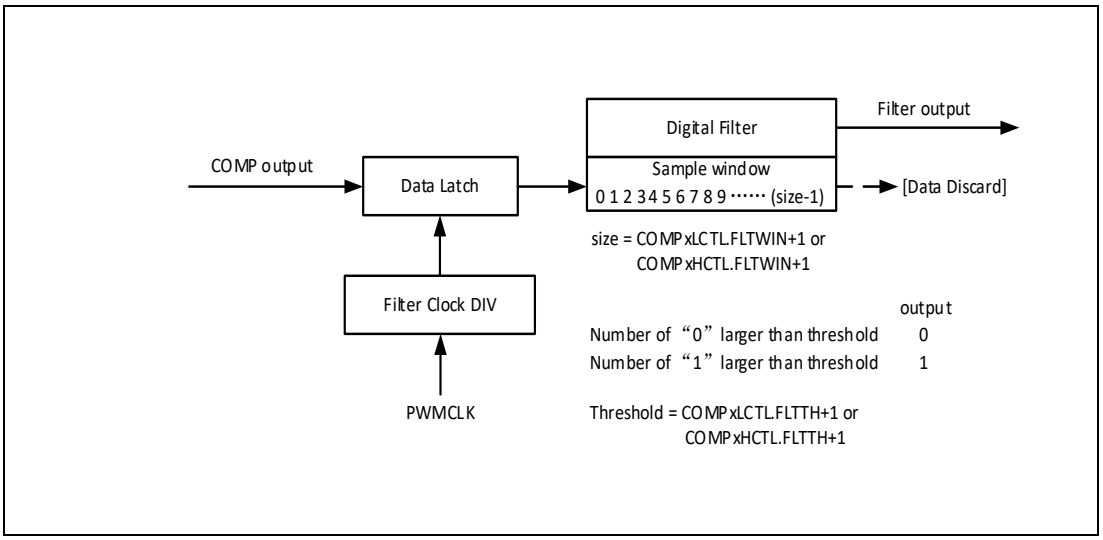
图 3-4: PWM 切换噪声



但是假若 COMP 的输出没有数字滤波电路（Deglitch）功能，此瞬时响应会误触发 COMP 的输出，并且将 PWM 关断，造成不必要的停机，因此在设计上，提供了输出数字滤波器。其基本原理如下图所示，COMP 的输出首先送到锁存器中，时钟来源于 PWMCLK，并且最大可以分频到 1024 倍。然后设置采样窗口的大小和阈值的大小，范围可以从 0~31 中选择，并且要保证阈值要大于

采样窗口的一半，如果采样窗口内采到的 COMP 输出 0 的个数大于阈值，则数字滤波器输出为 0，如果采样窗口内的 COMP 输出 1 的个数大于阈值，则数字滤波器输出为 1，如果采用窗口内的 COMP 输出 0 和 1 个数都没有超过阈值，则数字滤波器输出不变。

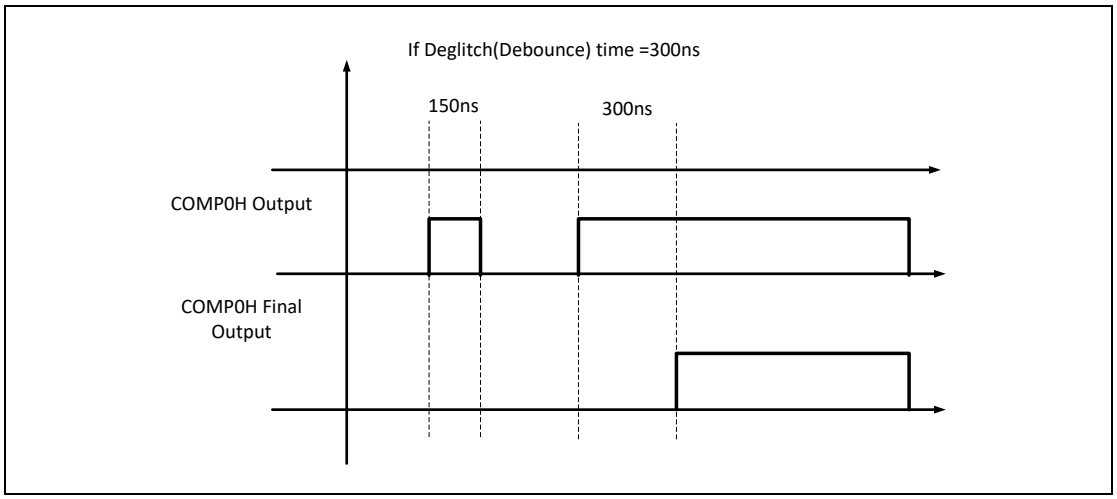
图 3-5:　数字滤波器模拟框图



重要 Note：

请执行 COMP 初始化前，务必执行 SPC11X8/SPD11X8 SDK 中的系统初始化与 Clock 的初始化函数，此两个初始化函数会更新 Clock 信息，如此才能作正确的配置相关信息。加入数字滤波器之后，COMP 输出如下所示。

图 3-6:　带数字滤波器的 COMP 输出



注意 SPC11X8/SPD11X8 有十个 COMP，而这些 COMP 的输出滤波电路可以分别设置滤波长度。

3.5 COMP 初始化 API 使用介绍

COMP 初始化之 API 一共有四个输入参数可以设定，如下面范例所示。

Example Code

```
void COMP_Init(COMP_ComparatorSelEnum eComp, COMP_CHSelEnum eCH,
               uint32_t u32DACVoltageMV, uint32_t u32DeglicthTimeNs);

COMP_Init(COMP_0_HI,
          COMP0_FROM_PGA0P_OUT,  /* From PGA0P input              */
          2450                   /* DAC HI/LO compare voltage mV */,
          300                    /* Output deglitch time(ns)     */
          );
```

COMP_Init（）函数的参数介绍如下表所示：

表 3-4: COMP_Init 函数介绍

COMP_Init	
Parameter	**Description**
eCOMP	It can be COMP_0_LO ~ COMP_4_LO COMP_0_HI ~ COMP_4_HI 可选择配置以上十个 COMP
eCH	It can be COMPy_FROM_PGAxN_OUT COMPy_FROM_PGAxP_OUT COMPy_FROM_ADCx 选择该 COMP 之输入通道
u32DACVoltageMV	0～3300（mV）
u32DeglicthTimeNs	输出数字滤波器之时间长度（5 bit），此滤波器来源为分频后的 PWMCLK if PWMCLK=100MHZ，10 分频 Its range from (0 ~ 31)*（1/100MHz * 10） = 3100 ns

请注意以下两点：

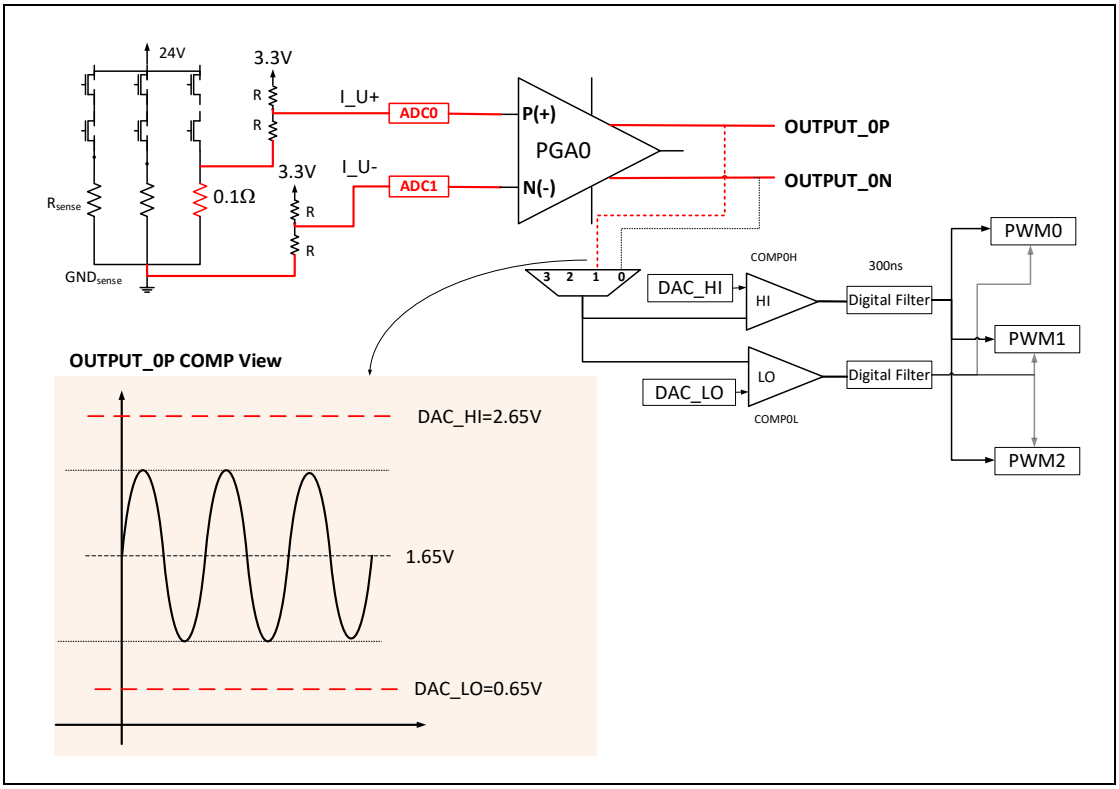
- 所有 Too High comparator 仅共享一组 DAC，并非每一个都有独立的 DAC。同理 Too Low comparator 亦同，因此用 API 设定时，最后设定值将覆盖之前设定的 DAC 数值；
- 所有的 COMP 输出数字滤波时间宽度可以分别设置。

3.6 COMP 停止（Trip）PWM 波形功能介绍

SPC11X8/SPD11X8 的 COMP 专为相电流防护做特殊设计。当 PGA 作为三相电流采集的时候，SPC11X8/SPD11X8 的 COMP 提供了每一相独立的电流防护，较一般的总电流防护更加安全，以下代码以 U 相电流为例，当 PGA0 输出超过 3V 时，COMP0H 触动 PWM 输出停止，当 PGA0 的电流小于 1V，由 COMP0L 触发 PWM 停止。Spintrol 的 COMP 触发 PWM 停止可达成任一个 COMP 同时停止所有 PWM 输出波形功能，详见 PWM trip zone 章节。

举例来说，COMP0H 与 COMP0L，只要任一种状况发生，均可以同时将三相 PWM 停止，同理其他四个 COMP 亦具有同样功能，三相电流一共有六个 COMP，只要其中一个发生触发，均可同时将 PWM 停止，对于相电流防护来说是很重要的一个功能。

图 3-7: COMP 停止 PWM 波形原理图



相比于大部分 IC 需要复杂的设定，Spintrol 亦提供了简单的 API 设定 COMP 触发 PWM trip 功能。当 eCOMP 代表的比较器发生输出由 Low 转 High 的瞬间，关闭 PWMx 的 CHA 与 CHB 输出。可重复呼叫设定多种组合，请见范例码。

以下为范例码：

Example Code

```
void MotorPWM_InitTripZoneAction(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);

/* Note: Please initial PGA pin before comparator initial */
COMP_Init(COMP_0_HI,
        COMP0_FROM_PGA0P_OUT,     /* From PGA0P output        */
             2650          /* DAC HI compare voltage mV */,
             300           /* Output deglitch time(ns)  */);
COMP_Init(COMP_0_LO,
        COMP0_FROM_PGA0N_OUT,     /* From PGA0N output        */
             650           /* DAC LO compare voltage mV */,
             300           /* Output deglitch time(ns)  */);
/* Step 3: PWM U,V,W will be triped when comparator0 output high.
   This code is prepared for total current protection*/
/* Affected by results monitored from all three phases */
PWM_EnableDCAHTripEvent(PWM_U, DC_TRIP_COMP0H |
                        DC_TRIP_COMP0L );
PWM_EnableDCAHTripEvent(PWM_V, DC_TRIP_COMP0H |
                        DC_TRIP_COMP0L );
PWM_EnableDCAHTripEvent(PWM_W, DC_TRIP_COMP0H |
                        DC_TRIP_COMP0L );

PWM_SetRawDCAEVT0(PWM_U, DCH_HIGH_DCL_HIGH);
PWM_SetRawDCAEVT0(PWM_V, DCH_HIGH_DCL_HIGH);
PWM_SetRawDCAEVT0(PWM_W, DCH_HIGH_DCL_HIGH);

PWM_SetOneShotTripEvent(PWM_U, TRIP_EVENT_DCAEVT,
```

```
TRIP_OUTPUT_LATCH);
  PWM_SetOneShotTripEvent(PWM_V, TRIP_EVENT_DCAEVT,
TRIP_OUTPUT_LATCH);
  PWM_SetOneShotTripEvent(PWM_W, TRIP_EVENT_DCAEVT,
TRIP_OUTPUT_LATCH);

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

  /* Set output to tri-state upon DCAEVT0 trip event */
  PWM_SetCHAOutputWhenTrip(PWM_U, TZU_TRIP_DO_NOTHING |
                           TZD_TRIP_DO_NOTHING |
                           DCEVT0U_TRIP_AS_TRI_STATE |
                           DCEVT0D_TRIP_AS_TRI_STATE |
                           DCEVT1U_TRIP_DO_NOTHING |
                           DCEVT1D_TRIP_DO_NOTHING);

  PWM_SetCHBOutputWhenTrip(PWM_U, TZU_TRIP_DO_NOTHING |
                           TZD_TRIP_DO_NOTHING |
                           DCEVT0U_TRIP_AS_TRI_STATE |
                           DCEVT0D_TRIP_AS_TRI_STATE |
                           DCEVT1U_TRIP_DO_NOTHING |
                           DCEVT1D_TRIP_DO_NOTHING);
  PWM_SetCHAOutputWhenTrip(PWM_V, TZU_TRIP_DO_NOTHING |
                           TZD_TRIP_DO_NOTHING |
                           DCEVT0U_TRIP_AS_TRI_STATE |
                           DCEVT0D_TRIP_AS_TRI_STATE |
                           DCEVT1U_TRIP_DO_NOTHING |
                           DCEVT1D_TRIP_DO_NOTHING);

  PWM_SetCHBOutputWhenTrip(PWM_V, TZU_TRIP_DO_NOTHING |
                           TZD_TRIP_DO_NOTHING |
                           DCEVT0U_TRIP_AS_TRI_STATE |
                           DCEVT0D_TRIP_AS_TRI_STATE |
                           DCEVT1U_TRIP_DO_NOTHING |
                           DCEVT1D_TRIP_DO_NOTHING);
  PWM_SetCHAOutputWhenTrip(PWM_W, TZU_TRIP_DO_NOTHING |
                           TZD_TRIP_DO_NOTHING |
                           DCEVT0U_TRIP_AS_TRI_STATE |
                           DCEVT0D_TRIP_AS_TRI_STATE |
                           DCEVT1U_TRIP_DO_NOTHING |
```

```
                              DCEVT1D_TRIP_DO_NOTHING);

  PWM_SetCHBOutputWhenTrip(PWM_W, TZU_TRIP_DO_NOTHING |
                              TZD_TRIP_DO_NOTHING |
                              DCEVT0U_TRIP_AS_TRI_STATE |
                              DCEVT0D_TRIP_AS_TRI_STATE |
                              DCEVT1U_TRIP_DO_NOTHING |
                              DCEVT1D_TRIP_DO_NOTHING);
  /* Step 5: Set PWM reaction after one shot or cycle by cycle event */
  PWM_SetCHAOutputWhenTrip(PWM_U,TRIP_AT_LOW);
  PWM_SetCHBOutputWhenTrip(PWM_U,TRIP_AT_LOW);

  PWM_SetCHAOutputWhenTrip(PWM_V,TRIP_AT_LOW);
  PWM_SetCHBOutputWhenTrip(PWM_V,TRIP_AT_LOW);

  PWM_SetCHAOutputWhenTrip(PWM_W,TRIP_AT_LOW);
  PWM_SetCHBOutputWhenTrip(PWM_W,TRIP_AT_LOW);
  /* Step 6: Enable one shot interrupt in this application */
  PWM_EnableOneShotTripInt(PWM_U);
  PWM_EnableOneShotTripInt(PWM_V);
  PWM_EnableOneShotTripInt(PWM_W);
}
```

4 修订记录

表 4-1: 文档修订记录

日期	版本	修改内容
2019-07-25	1	初始版本
2021-11-13	2	1. 更新图 3-1。 2. 更新表 3-3。