SPC1168 DAC COMP PGA 使用指南

概述

在电力电子系统中，DAC，COMP，PGA 有时会组合起来，采用 PGA，DAC 作为 COMP 的输入，从而检测过流事件，触发 PWM 封锁保护。

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A/0	2023-06-11	HangSu	Outdated	首次发布。
C/0	2024-04-22	Jiali Zhou	Released	修改排版格式。

术语或缩写

术语或缩写	描述
/	/

1 DAC 实例

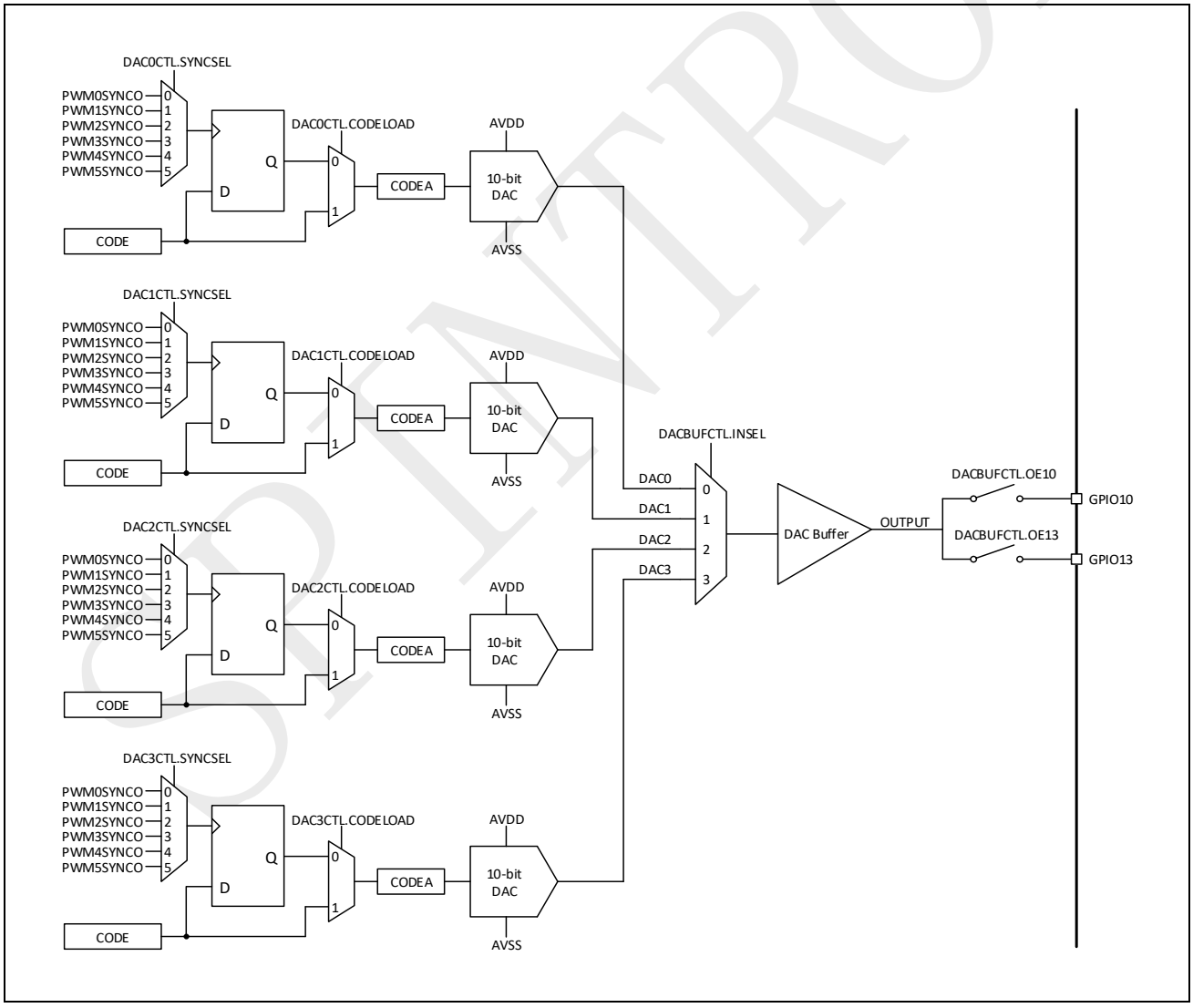
1.1 DAC 示意图

SPC1168 包含 4 个 10 比特 DAC 和一个可以驱动容性负载的缓冲器（Buffer）。DAC 输出可以配置成直流电压和交流波形，如三角波、方波等。DAC 的输出可由下面表达式计算：

$$\mathrm{VOUT} = \frac{AVDD}{2^{10}} * CODE \quad (AVDD\text{典型值为 } 3.3V)$$

DAC 最常规的用法，是将 DACXCODEA 直接送入 DAC，从而产生模拟电压，如图 1-1 所示。将 DAC0CODE 的数值经 10-bit DAC 转换为模拟信号，该模拟信号既可直接传给 COMP，亦可经过 DAC buffer 直接输出到 GPIO10 或 GPIO13。

图 1-1：DAC 示意图



以下一个例子通过 DAC 产生 1.65V 电压，并送入 ADC 测量。

Example Code

```
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);

    /* Enable COMP model */
    CLOCK_EnableModule(COMP_MODULE);

    /* Enable DAC buffer */
    COMP_DACBufferInit(DAC0, FALSE, FALSE);

    /* Set DAC0 output to 1.65v */
    COMP_SetDACValue10Bit(DAC0, DACTest165V);

    /* Set DAC0 as Direct mode(DAC code is immediately update) */
    COMP_SetDACCodeLoadTiming(DAC0, DAC0CTL_BIT_CODELOAD_DIRECT_MODE);

    /* Enable DAC0 */
    COMP_EnableDAC(DAC0);

    /* Init ADC */
    ADC_Init(ADC_SOC_0, ADC_SHB_P_GND, ADC_SHB_N_DAC_BUF, SHB,
ADCTRIG_Software);
```

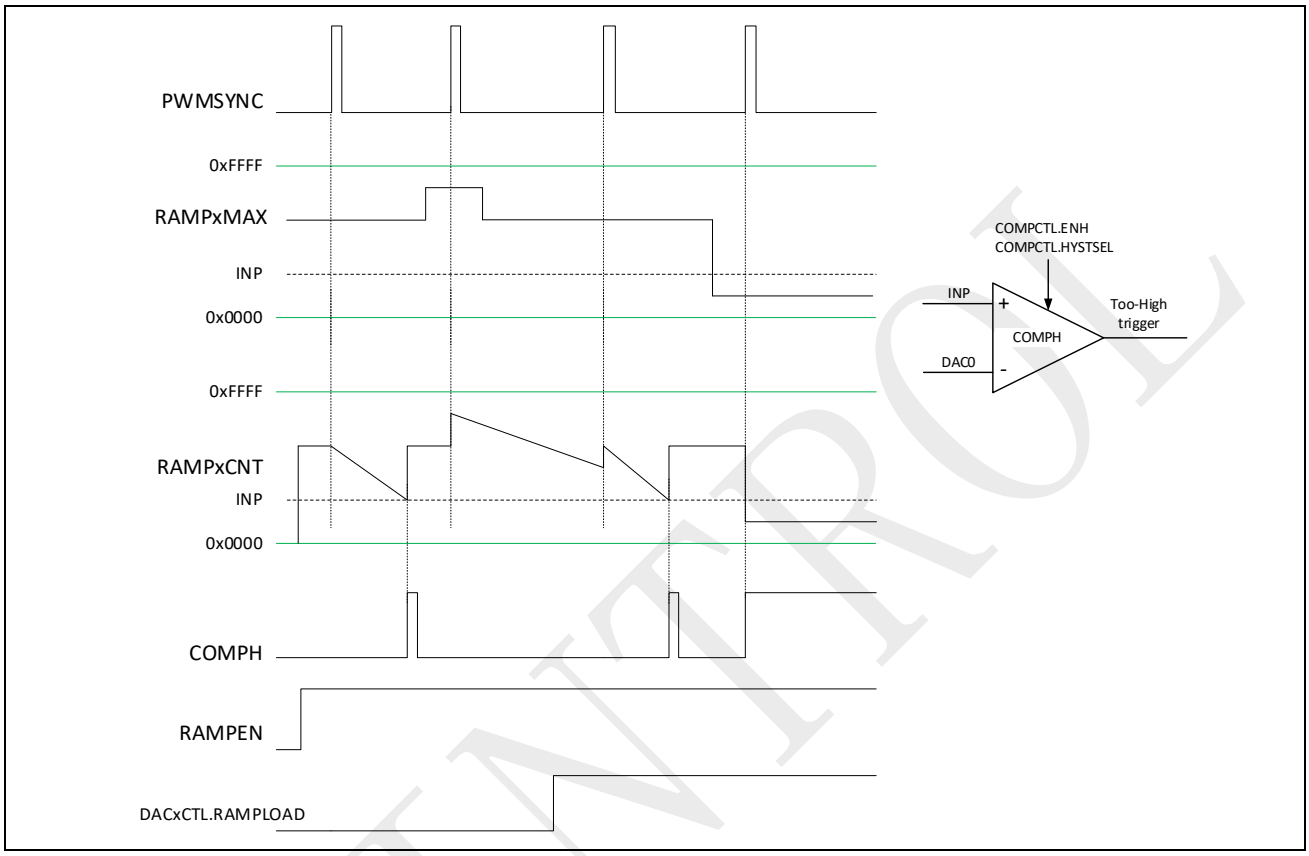
Example Code

```
    /* Enable the global INT for the ADC */
    NVIC_EnableIRQ(ADC0_IRQn);

    /* Set ADC as the software trigger model */
    ADC_SoftwareTrigger(ADC_SOC_0);

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

    while (1)
    {

    }
}

void ADC0_IRQHandler(void)
{
    i32VSP = ADC_GetResult(ADC_SOC_0);
    printf("ADC SOC0 Voltage = %2fV(Raw data is %d)\n",
(double)ValueToVoltage(i32VSP), i32VSP);

    /* If the result is in error range, the test PASS */
    if ((abs(i32VSP) > DACTestLTH) && (abs(i32VSP) < DACTestHTH))
    {
        printf("The DAC test PASS\n");
    }
    else
    {
        printf("The DAC test FAIL, raw data is %d\n", i32VSP);
    }

    ADC_ClearInt(ADC_SOC_0);

    Delay_Ms(500);

    /* Use software to trigger ADC SOC0 to work */
    ADC_SoftwareTrigger(ADC_SOC_0);
}
```

1.2 斜坡发生器

PWM, DAC,COMPH 可以共同构成斜坡发生器，产生斜坡电压 RAMPxCNT，如图 1-2 所示。

图 1-2：斜坡发生器

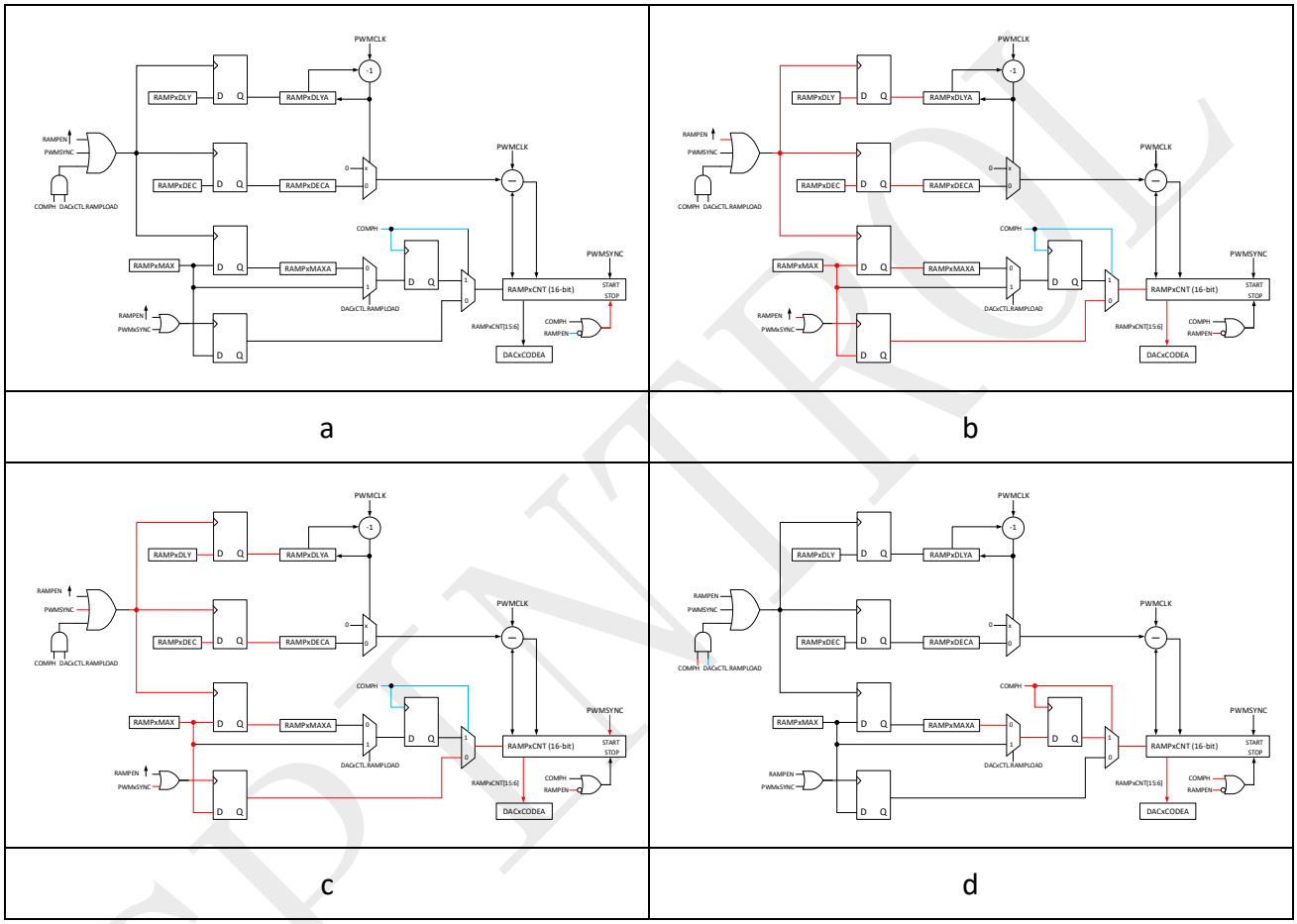


- 因为 DAC0 只能接到 COMPH 的负端，为了保证默认状态无 COMPH 事件，必须保证默认状态 DAC0CODEA(RAMP0CNT[15:6])电压 > INP；
- 初始条件 COMPH 是禁止的（通过软件设定），因此虽然 INP > DAC0CODEA(RAMP0CNT[15:6])电压，但是没有对应的 COMPH 事件；同时 RAMPEN 为低，停止 RAMPxCNT 的递减计数，如图 1-3a；
- 下一时刻，RAMPEN 产生上升沿脉冲，由于此时 COMPH 仍是禁止，没有 COMPH 事件，从而控制下路连通，RAMPxMAX 通过下路触发器直接加载到 RAMPxCNT，同时上路的各寄存器初始值均加载到对应的活跃寄存器；当 RAMPxCNT 加载完成后，其[15:6]位构成 DACxCODEA，在其模拟电压稳定后，使能 COMPH，此时 DAC0CODEA(RAMP0CNT[15:6])电压>INP，没有 COMPH 事件，COMPH 事件维持低电平图 1-3b；
- 下一时刻，PWMSYNCO 产生上升沿脉冲，由于没有 COMPH 事件，从而控制下路连通；RAMPxMAX 通过下路触发器直接加载到 RAMPxCNT，同时上路的各寄存器初始值均加载到对应的活跃寄存器，同时 PWMSYNCO 也会使能 RAMPxCNT 的递减计数图 1-3c；
- 当 DACxCODEA 电压递减到<INP 时，COMPH 事件触发，由于 RAMPLOAD 为 0，RAMPxMAXA 通过触发器直接加载到 RAMPxCNT，COMPH 事件消失，同时 COMPH 关闭 RAMPxCNT 的递减计数图 1-3d；

- 下一时刻，PWMSYNCO 产生上升沿脉冲，由于没有 COMPH 事件，从而控制下路连通；RAMPxMAX 通过下路触发器直接加载到 RAMPxCNT，同时上路的各寄存器初始值均加载到对应的活跃寄存器，同时 PWMSYNCO 也会使能 RAMPxCNT 的递减计数图 1-3c；

- 下一时刻，PWMSYNCO 产生上升沿脉冲，由于没有 COMPH 事件，从而控制下路连通；RAMPxMAX 通过下路触发器直接加载到 RAMPxCNT，同时上路的各寄存器初始值均加载到对应的活跃寄存器，同时 PWMSYNCO 也会使能 RAMPxCNT 的递减计数图 1-3c；

图 1-3：根据 PWM 同步输出信号递减电平



示例演示了通过定时器产生 SYNCI 信号并通过 SYNCO 触发 DAC 递减计数，从而产生斜坡电压。

实验时，COMPH 对应正端电压必须大于 0V，否则负端电压无法降落到正端电压之下。

Example Code

```
int main()
{
    FLASH_WALLOW();

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

Example Code

```
    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);

    /* enable DAC0 */
    COMP_EnableDAC(DAC0);

    /* Set DAC0 as Direct mode(DAC code is immediately update) */
    COMP_SetDACCodeLoadTiming(DAC0, DAC0CTL_BIT_CODELOAD_DIRECT_MODE);

    /* set DAC0 output to 3000mV  */
    COMP_SetDACVoltage(DAC0, 3000);

    /* DAC Buffer Init */
    COMP_DACBufferInit(DAC0, FALSE, TRUE);

    /* Select the PWM synchronous output signal for DAC */
    COMP_SetDACSyncEvent(DAC0, SEL_PWM0);

    CLOCK_EnableModule(PWM_MODULE);

    /* Connect the input sync signal with output */
    PWM_SetSyncOutEvent(PWM0, SYNC_SYNCI_AND_FRCSYNC);

    /* Enable PWM SYNC by the signal coming from TIMER1 */
    PWM_EnableSyncFromTIMER1(INC_PWM0 );

    printf("PWMCFG->TMR1SYNCIEN is %x\n", PWMCFG->TMR1SYNCEN.all);

    /* Init TIMER1 */
    TIMER_Init(TIMER1, 1);

    /* Set TIMER1 generate SYNC to PWM when count down to 0 */
    TIMER_EnablePWMSync(TIMER1);

    /* Open Global INT for TIMER1 */
    NVIC_EnableIRQ(TIMER1_IRQn);

    /* Enable Timer */
    TIMER_Run(TIMER1);
```

Example Code

```
    COMP->RAMP0DLY.all = 200;
    COMP->RAMP0DEC.all = 1;
    COMP->DAC0CTL.bit.RAMPEN = ENABLE;

    /* Enable Comparator clock */
    CLOCK_EnableModule(COMP_MODULE);

    /* Enable the ADC Bandgap */
    ADC_EnableBandgap();

    /* Enable comparator */
    COMP_Enable(COMP_0_HI);

    /* Set Input channel */
    COMP_SelectChannel(COMP_0_HI, COMP0_FROM_ADC0);

    COMP->COMP0CTL.bit.REFSEL = 0;

    /* Set comparator output = Filtered */
    COMP_SetOutputType(COMP_0_HI, COMP_OUTPUT_FILTERED);

    /* Set Filter window - Threshld value = 0.6 * Window Size */
    COMP_SetFilterWindowTimeNs(COMP_0_HI, 200, 200 * 3 / 5);

    /* Clear latched filter status */
    COMP_ClearAllFilterOutputStatus();

    printf("COMP->DAC0CODE is %d\n", COMP->DAC0CODE.all);

    COMP->RAMP0MAX.all = (COMP->DAC0CODE.all << 6U);

    printf("COMP->RAMP0MAX is %d\n", COMP->RAMP0MAX.all);

    while (1)
    {

    }
}

void TIMER1_IRQHandler()
{
    /* Clear the INT */
    TIMER_ClearInt(TIMER1);
}
```

2 COMP 实例

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
- 剩余两组 COMP 作为扩展，支持更多应用
- 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

2.1 COMP 示意图

COMP0 的模拟架构如图 2-1 所示，其余 COMP1~COMP4 架构和 COMP0 相同，其输入端选择见表 2-1 所示。

图 2-1：COMP0 示意图

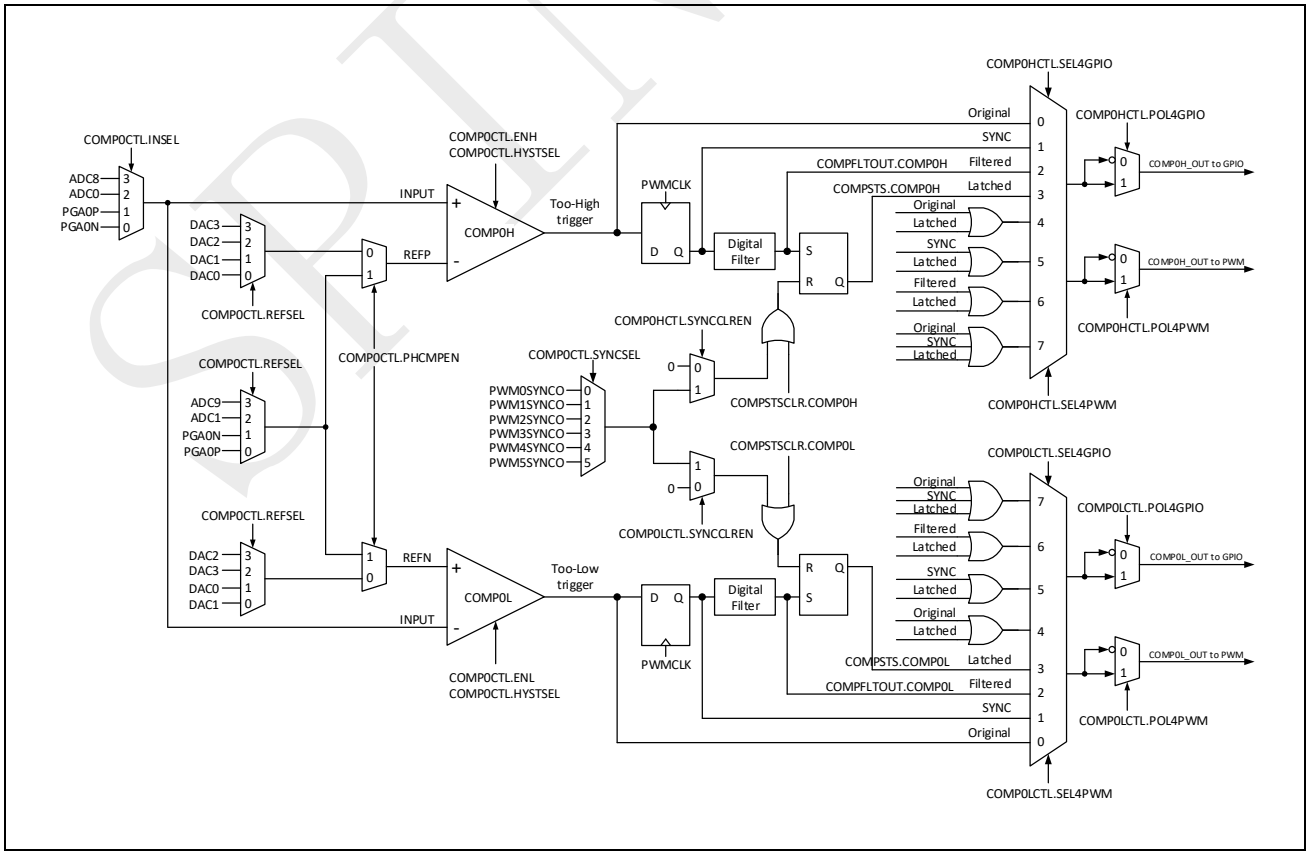


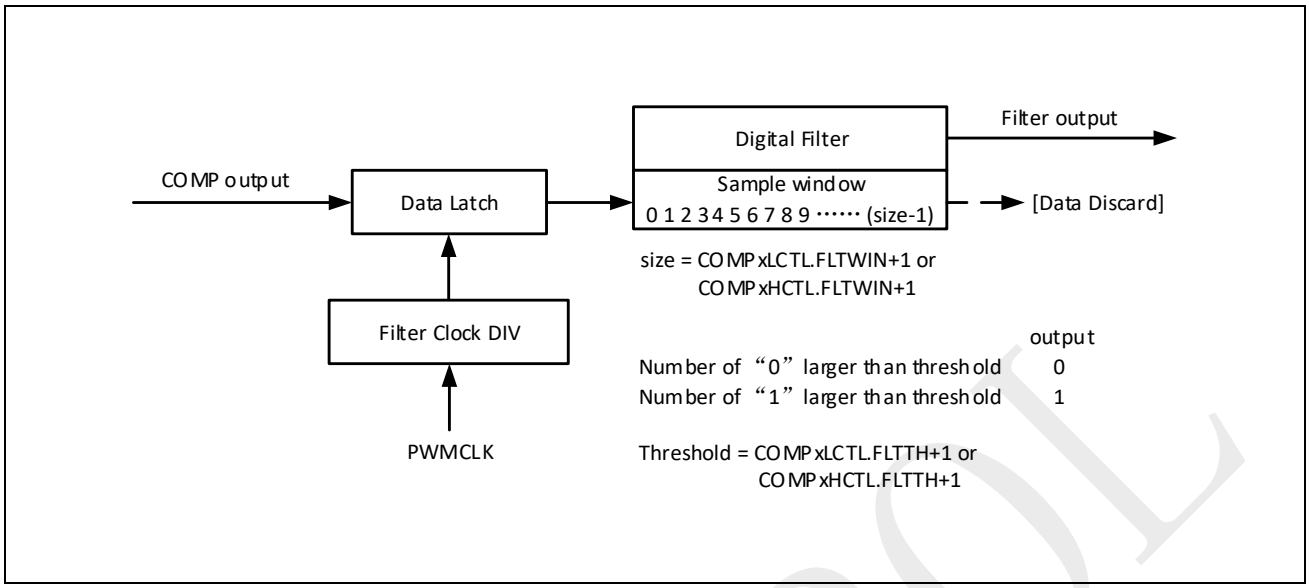
表 2-1：Comparator 0~4 输入选择

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

COMP 最常规的用法，是将输入电压与 DAC 比较，从而产生 COMPH 或 COMPL 事件，如图 2-1 所示。比较器的输出可以被送到数字滤波器的采样窗口，采样时钟来自 PWMCLK，可由 COMPxLCTL.FLTDIV/COMPxHCTL.FLTDIV 来设置分频大小，窗口大小由 COMPxLCTL.FLTWIN/COMPxHCTL.FLTWIN 设置，最大为 32。通过 COMPxLCTL.FLTTH/ COMPxHCTL.FLTTH 设置阈值的大小，数字滤波器可以计算采样窗口内 0 和 1 发生的次数，如果 0 发生的次数大于之前设置的的阈值大小，则数字滤波器的输出会变为 0，同理，如果 1 发生的次数大于阈值，数字滤波器的输出会变为 1，如图 2-2 所示。

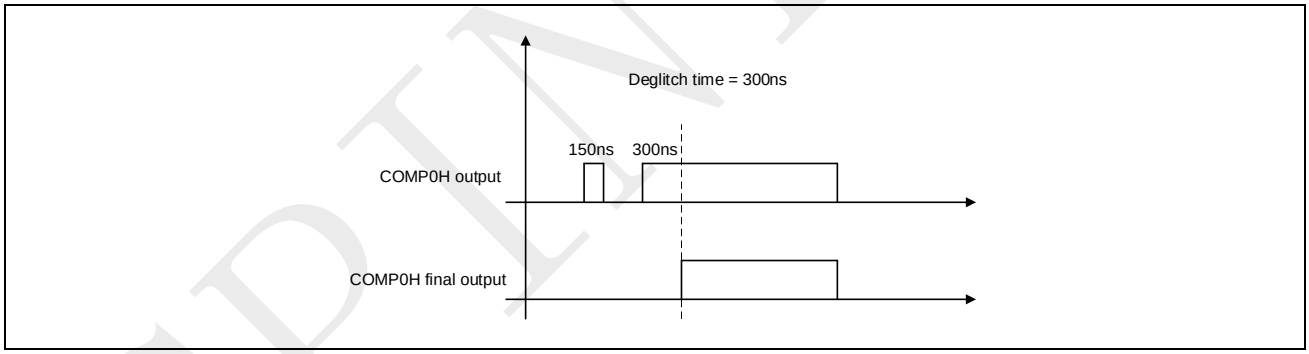
为了使滤波器工作正常，需要保证设置的阈值 FLTTH 要大于 FLTWIN/2。

图 2-2：数字滤波器框图



请执行 COMP 初始化前，务必执行 Clock 的初始化函数，如此才能作正确的配置相关信息。加入数字滤波器之后，COMP 输出如图 2-3 所示，假设设定阀值是 300ns，不足 300ns 的信号被过滤。

图 2-3：数字滤波器模拟框图



以下以一个例子通过 COMP 比较 GPIO8 和 DAC2, DAC3 的电压，并将 COMP 信号输出到 GPIO0，GPIO1 的操作如下。

Example Code

```
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();
```

Example Code

```
    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);

    /*
     * Configure Comparator.
     *
     * 1.Set low detection.
     *
     * 2.Set ADC8 as the input.
     *
     * 3.Set DAC as 1000mV for low voltage detecting reference voltage.
     *
     * 4.Set the DigitaL Filter as 25ns, in order to filter the noise.
     */
    COMP_Init(COMP_0_LO, COMP0_FROM_ADC8, 1000, 25);

    /* Set high detection for ADC8 and reference value is 2000mV */
    COMP_Init(COMP_0_HI, COMP0_FROM_ADC8, 2000, 25);

    COMP->COMP0HCTL.bit.SEL4GPIO = COMP0HCTL_BIT_SEL4GPIO_FILTRED;
    COMP->COMP0HCTL.bit.POL4GPIO = COMP0HCTL_BIT_POL4GPIO_ACTIVE_HIGH;
    GPIO_SetPinChannel(GPIO_0, GPIO0_COMP0H);

    COMP->COMP0LCTL.bit.SEL4GPIO = COMP0LCTL_BIT_SEL4GPIO_FILTRED;
    COMP->COMP0LCTL.bit.POL4GPIO = COMP0LCTL_BIT_POL4GPIO_ACTIVE_HIGH;
    GPIO_SetPinChannel(GPIO_1, GPIO1_COMP0L);
    while (1)
    {
        /* Get the comparator status */
        u32status = COMP_GetFilterOutputStatus(COMP_0_HI);
        if (u32status & 0x1)
        {
            COMP_ClearAllFilterOutputStatus();
            printf("The target voltage is higher than high-boundary of
comparator\n");
        }

        u32status = 0;
        u32status = COMP_GetFilterOutputStatus(COMP_0_LO);
        if (u32status & 0x1)
        {
            COMP_ClearAllFilterOutputStatus();
            printf("The target voltage is lower than low-boundary of
comparator\n");
```

Example Code

```
        }

        Delay_Ms(500);
    }
}
```

3 PGA 实例

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_Init（）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。

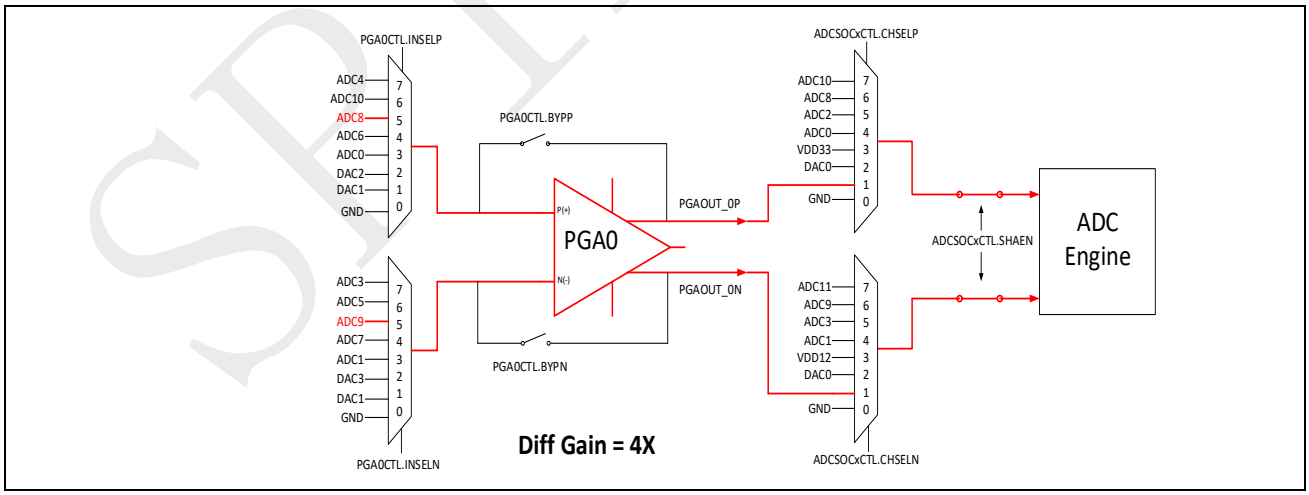
PGA 的基本特点如下：

- 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。

3.1 PGA 送入 ADC 采样

使用 ADC 对 PGA 采样，其链接图如图 3-1 所示。

图 3-1：ADC PGA 采样



以下例子演示使用 ADC 对 PGA 电压进行采样，将 PGA 电压送给 ADC CH0 进行采样，默认采样时间，转换时间均设定为 150ns。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。

在实际的使用中 PGA 输入电压必须符合表 3-1。

表 3-1：给定增益下 PGA 输入输出范围

PGAxCTL.GAINP/ PGAxCTL.GAINN	$Gain_{PGA}$	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

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.
     *
     * 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.
```

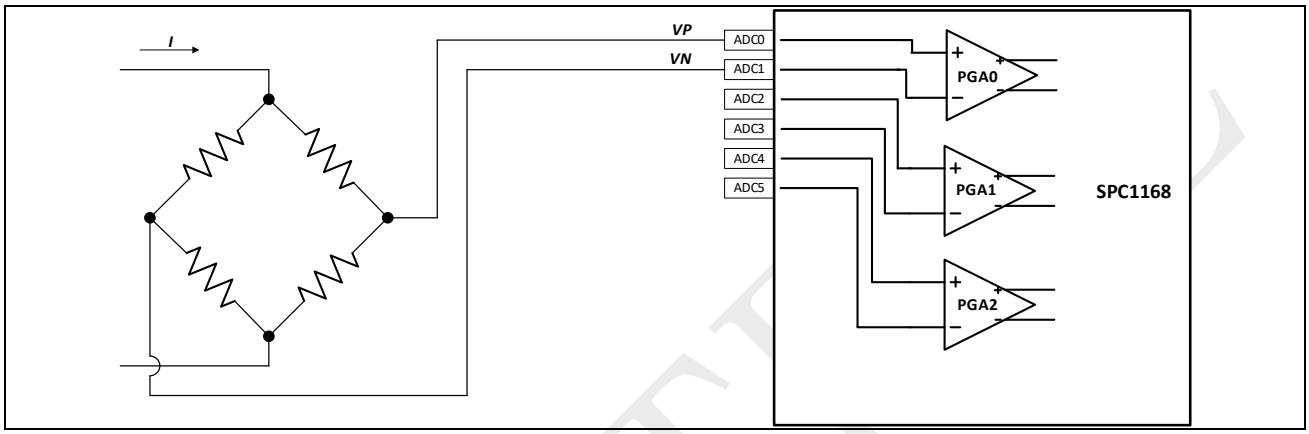
Example Code

```
     *
     * 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);
    }
}
```

3.2 Sensor 模式下的 PGA 配置

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 3-2：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
   *
   * 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);
}
```

3.3 将一个 PGA 拆分成两个单端 PGA 配置

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$OUT_P = IN_P * Gain_P, OUT_N = IN_N * Gain_N$.

范例代码如下：

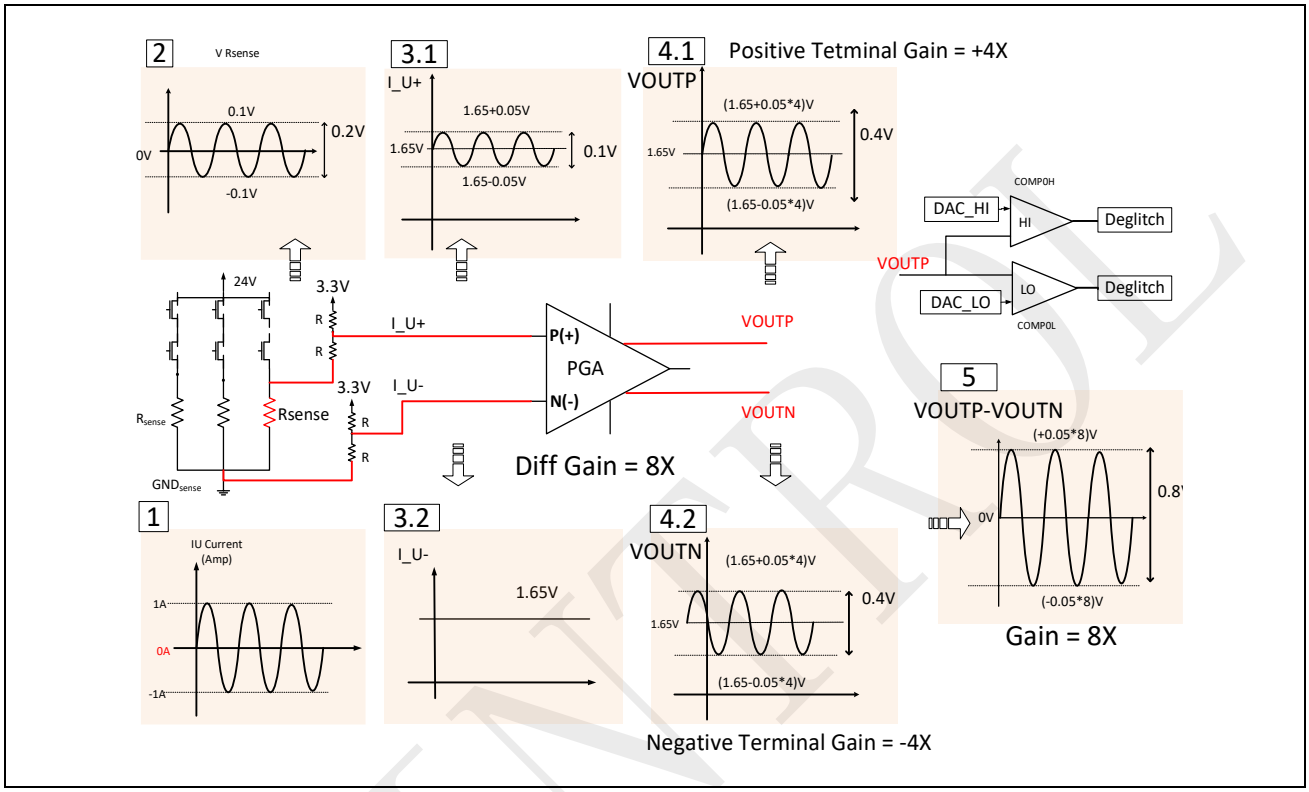
Example Code

```
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);
}
```

3.4 使用 PGA 进行过电流保护

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 3-3 所示。

图 3-3：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；

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

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

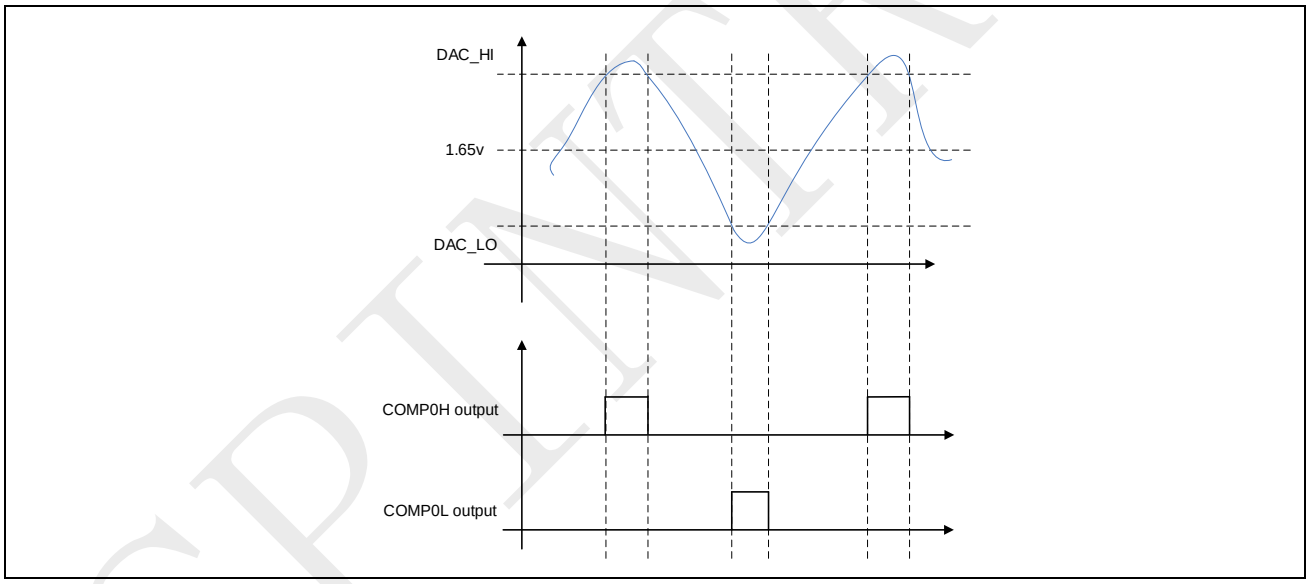
- 根据以下公式，可以决定$\mathrm{DAC_{HI}}$与$\mathrm{DAC_{LO}}$的设定数值；

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

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

- 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 3-4：COMP 输出演示

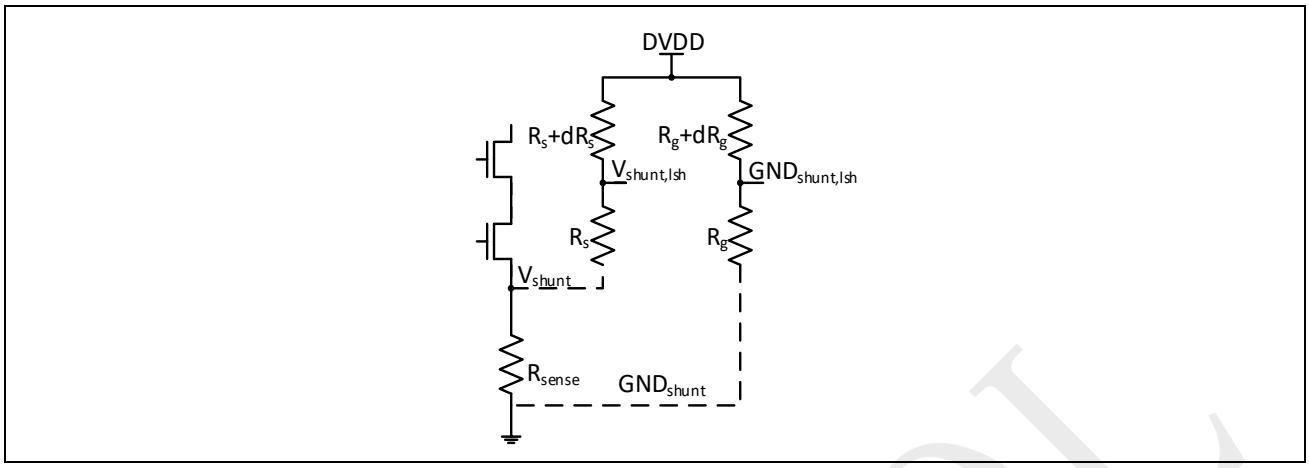


PWM 控制 H 桥电路，在电机控制系统或是电源控制系统是非常常见的技术，但是在 PWM 开关的瞬间，往往会对电流回授电路有较大的干扰，或是开关的瞬间常有较大的电流瞬时（Current Transient）。解决方法参见章节 2 中 COMP 滤波器。

3.4.1 电平移位器校准

图 3-3 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 3-5 所示。

图 3-5：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);
```

Example Code

```
    /* 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);
    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);
}
```

Example Code

```
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.
     *
     * 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);
```

Example Code

```
    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);


    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);
    }
```

Example Code

```
}

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);
}
```

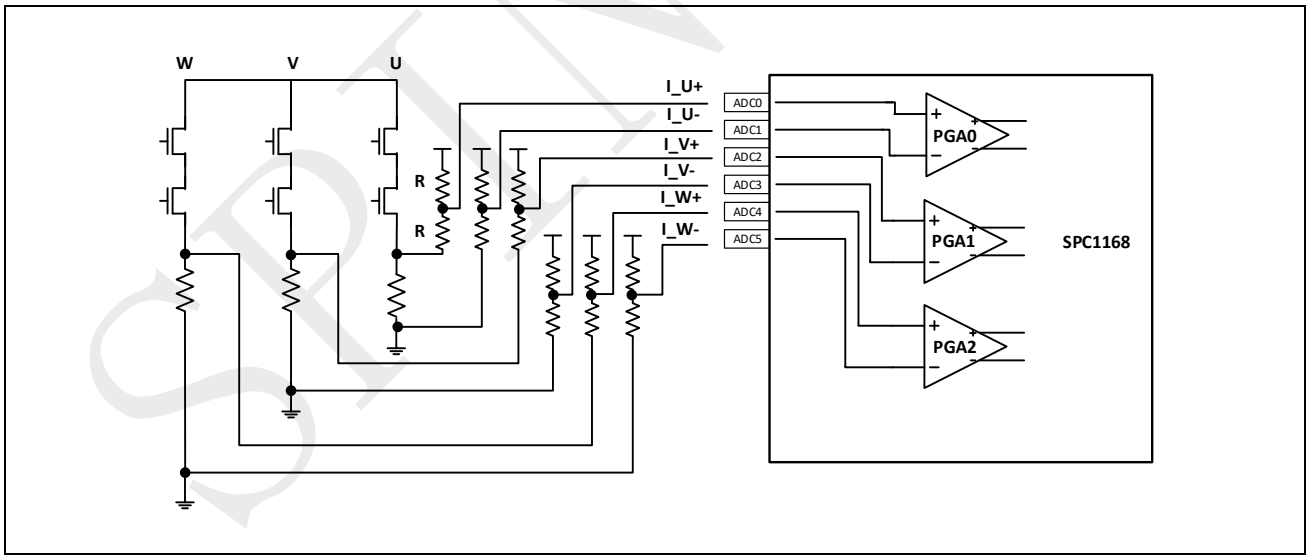
3.4.2 MCU 管脚建议排法

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 3-6：3 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。

Example Code

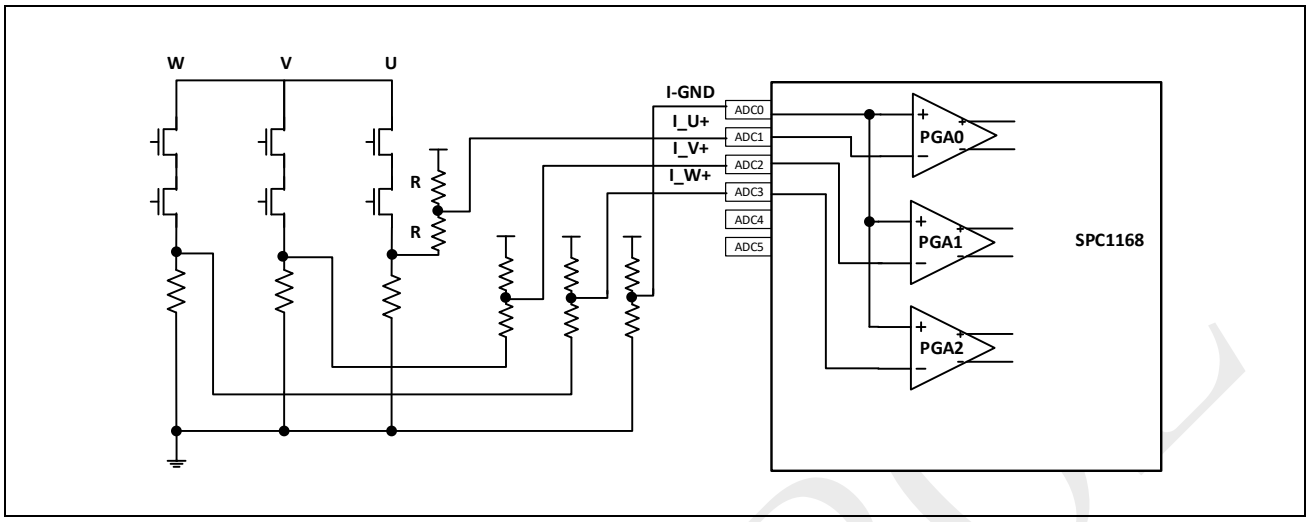
```
void PGA_InitExample2_6_1(void)
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  /* Disable flash write access after flash operation had done */
```

Example Code

```
  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 3-7：3 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。

Example Code

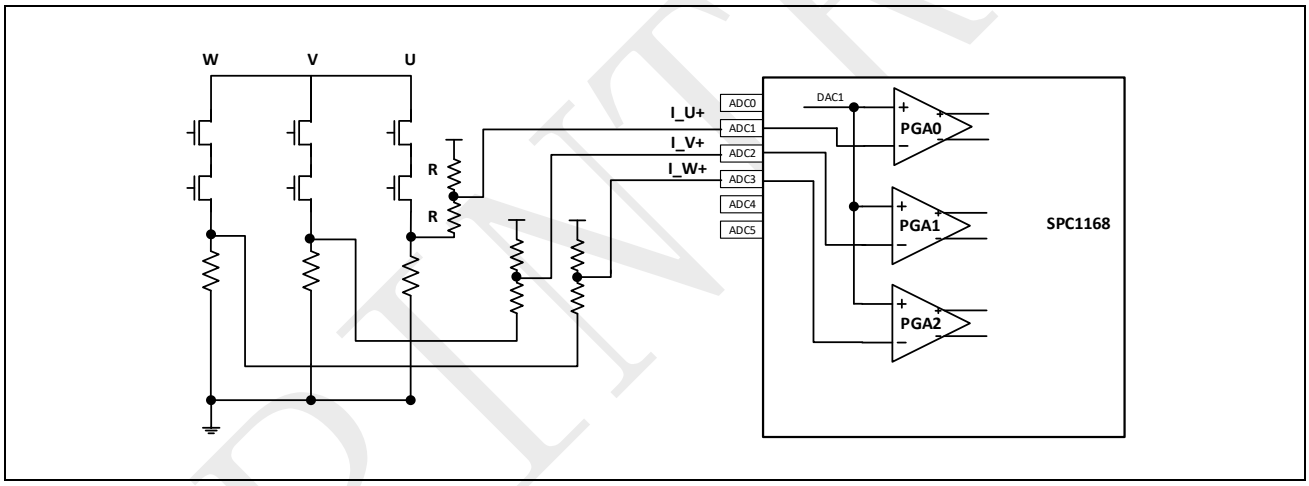
```
void PGA_InitExample2_6_2(void)
{
  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  /* Disable flash write access after flash operation had done */
  FLASH_WDIS();

  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,
```

Example Code

```
            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 3-8：3 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。

Example Code

```
void PGA_InitExample2_6_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
   *
```

Example Code

```
   * 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);
}
```