SPC2188 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	2023-11-20	X.He	Released	首次发布。

术语或缩写

术语或缩写	描述

1 DAC 实例

1.1 DAC 转换并输出到引脚

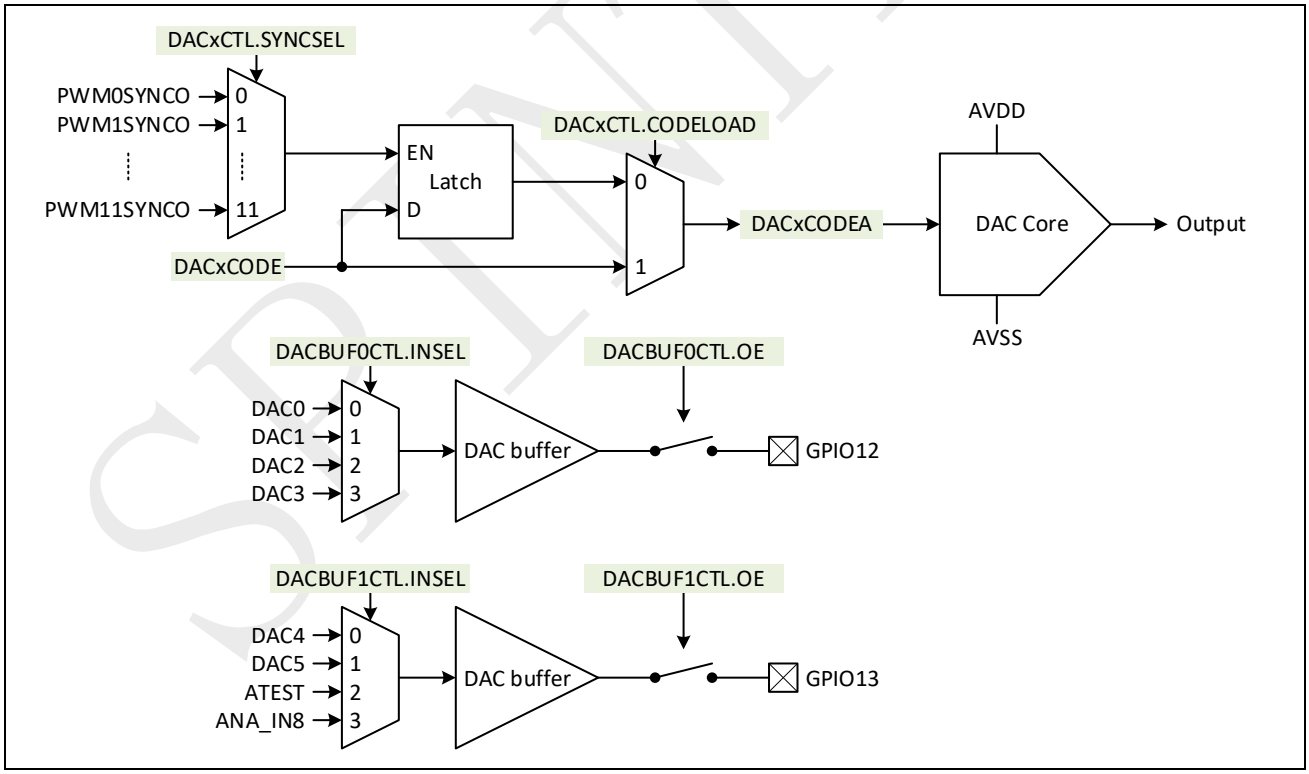
DAC 最常规的用法，是将 DACXCODEA 直接送入 DAC，从而产生模拟电压，如图 1-1 所示。将 DACOCODE 的数值经 10-bit DAC 转换为模拟信号，该模拟信号既可直接传给 COMP，亦可经过 DAC buffer 直接输出到 GPIO12 或者 GPIO13 引脚上。

$$V_{out} = VDD3 * \frac{CODE}{1024}$$

通常 VDD3 == 3.3V。

- 当 DACxCTL.CODELOAD = 0 时，开启影子模式。此时写入 DACxCODE 寄存器的码值并不会立即生效，而是要等到由 DACxCTL.SYNCSEL 所选定的 PWMySYNCO 事件发生时才会更新 DACxCODEA。
- 当 DACxCTL.CODELOAD = 1 时，开启即时模式。此时写入 DACxCODE 寄存器的码值立即同步更新到 DACxCODEA。

图 1-1：DAC 示意图



以下一个例子通过 DAC 在 GPIO12 输出 1.65V 电压：

Example Code

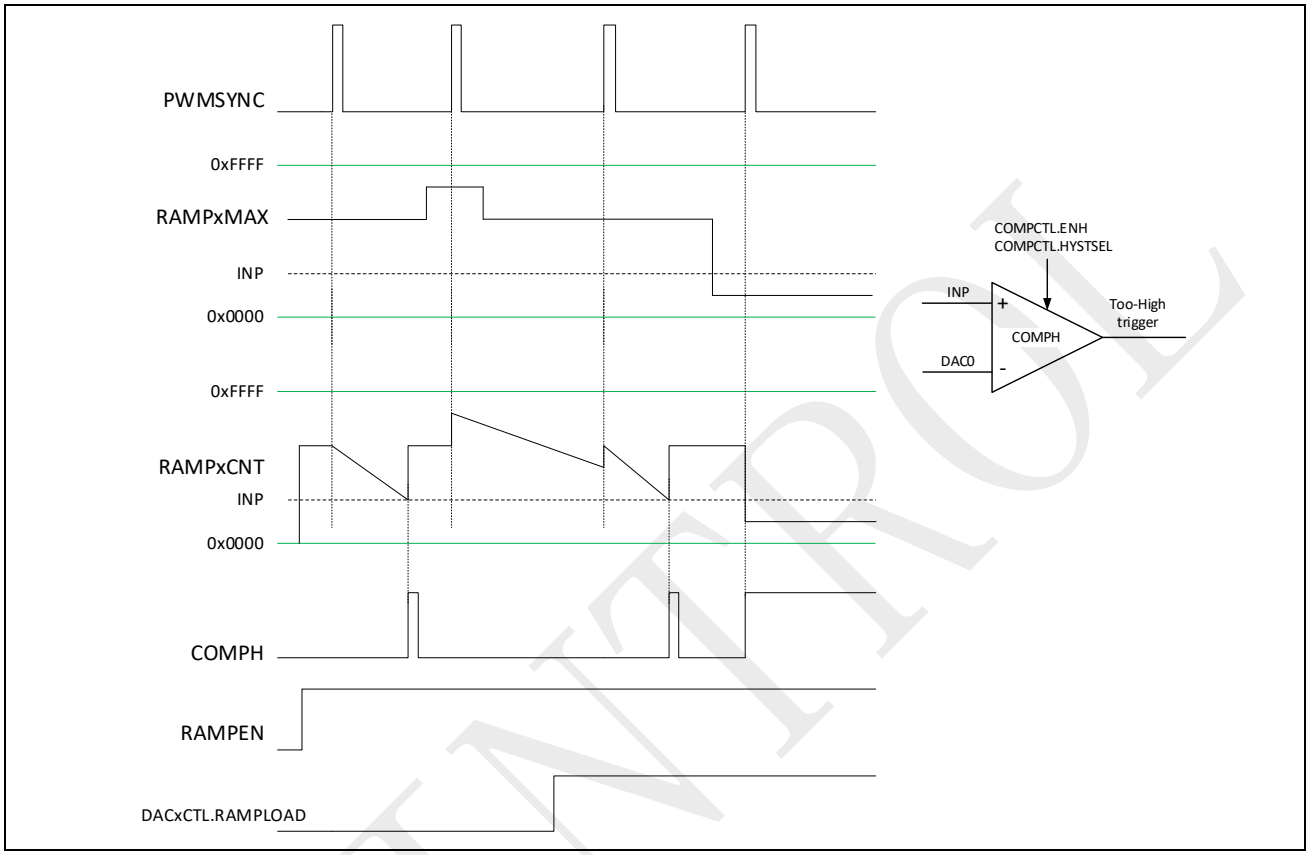
```
#include "spc2188.h"
#include <stdio.h>


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


int32_t                     i32VSP;                             /* Result Value */

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);

    printf("Enter the test\n");

    /* Initial ADC and set DAC buffer as the negative input of the ADC CH0 */
    ADC_Init(ADC1, ADC_CH0, ADC1_SH0_P_GND, ADC1_SH0_N_DACBUF0,
ADC_SOC_TRIGGER_FROM_SOFTWARE);

    /* enable DAC0 */
    COMP_EnableDAC(DAC0);

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

    /* set DAC0 output to 1.65V  */
    COMP_SetDACCode(DAC0, 512);

    /* DAC Buffer Initial, only DACBUF0_FROM_DAC0 ~ DACBUF0_FROM_DAC3 connect to
DACBUF0*/
    COMP_SetDACBuffer0Input(DACBUF0_FROM_DAC0);
    COMP_EnableDACBuffer(DACBUF0);

    /* Enable DAC Buffer Output To GPIO */
    COMP_EnableDACBufferOutputToGPIO(DACBUF0);


    /* Enable ADC1_IRQn */
    NVIC_EnableIRQ(ADC1CH0_IRQn);

    while(1)
    {
        /* Use software to trigger ADC CH0 to work */
        ADC_ForceChannelSOC(ADC1,ADC_CH0);

        Delay_Ms(500);
    }
}
```

```
void ADC1CH0_IRQHandler(void)
{
    /* Result gotten from 'ADC_GetChannelResult()' can be a negative value or a
positive value */
    i32VSP = ADC_GetChannelResult(ADC1,ADC_CH0);
    printf("ADC data is %d\n",i32VSP);
    printf("voltage = %d mV\n", ValueToVoltage(i32VSP));

    /* Clear SOC INT */
    ADC_ClearChannelInt(ADC1, ADC_CH0);
}
```

1.2 斜坡发生器

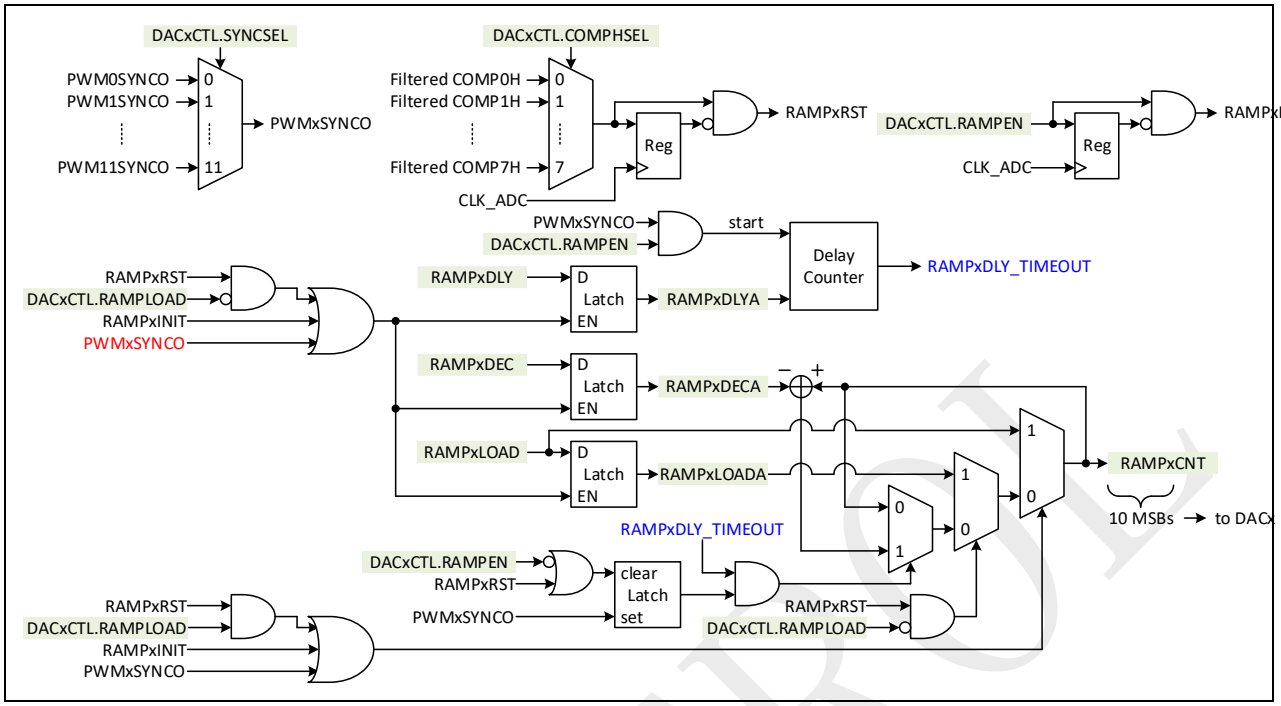
PWM, DAC,COMPH 可以共同构成斜坡发生器，产生斜坡电压 RAMPxCNT，如图 1-2 所示。

图 1-2：斜坡发生器



- 通过DACxCTL.RAMPEN使能斜坡发生器后,触发初始化加载影子寄存器RAMPxDLY、RAMPxDEC、RAMPxLOAD 到对应的有效值寄存器 RAMPxDLYA、RAMPxDECA、RAMPxLOADA；
- RAMPxCNT 计数器维持初值不变；
- 在 PWMxSYNCO 事件触发后，延时计数器从其初值 RAMPxDLYA 开始递减计数，直到超时；
- 延时结束后，RAMPxCNT 计数器以 RAMPxDECA 为步长递减计数；
- 由于 RAMPxCNT 的高 10 位用作 DACx 的码值，因此其输出的电压递减，即 COMPyH 的参考电压递减；
- 随着 RAMPxCNT 的减小，COMPyH 输出在某个时候会变高，这使得 RAMPxCNT 回到初值，在新的 PWMxSYNCO 事件到来后重复延时等待和递减计数的过程；

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

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

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);

    /* enable DAC0 */
    COMP_EnableDAC(DAC0);

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

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

    /* DAC Buffer Initial, only DACBUF0_FROM_DAC0 ~ DACBUF0_FROM_DAC3 connect to
DACBUF0*/
    COMP_SetDACBuffer0Input(DACBUF0_FROM_DAC0);
```

```
    COMP_EnableDACBuffer(DACBUF0);

    /* Enable DAC Buffer Output To GPIO */
    COMP_EnableDACBufferOutputToGPIO(DACBUF0);

    CLOCK_EnableModule(PWM_MODULE);

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

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

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

    /* 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_Enable(TIMER1);

    COMP_SetDACRampDelay(DAC0, 200);
    COMP_SetDACRampDecrement(DAC0, 1);
    COMP_EnableDACRamp(DAC0);

    /* Set PGAP as input to comparator, set Too High threshold is 3000mV, filter
window is 200ns */
    COMP_Init(COMP0_H, COMP0_FROM_ANA_IN6, COMP0_REF_DAC0_DAC1, 3000, 200);

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

    COMP_SetDACRampReloadValue(DAC0, (COMP->DAC0CODE << 6U));

    while (1)
    {

    }
}


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

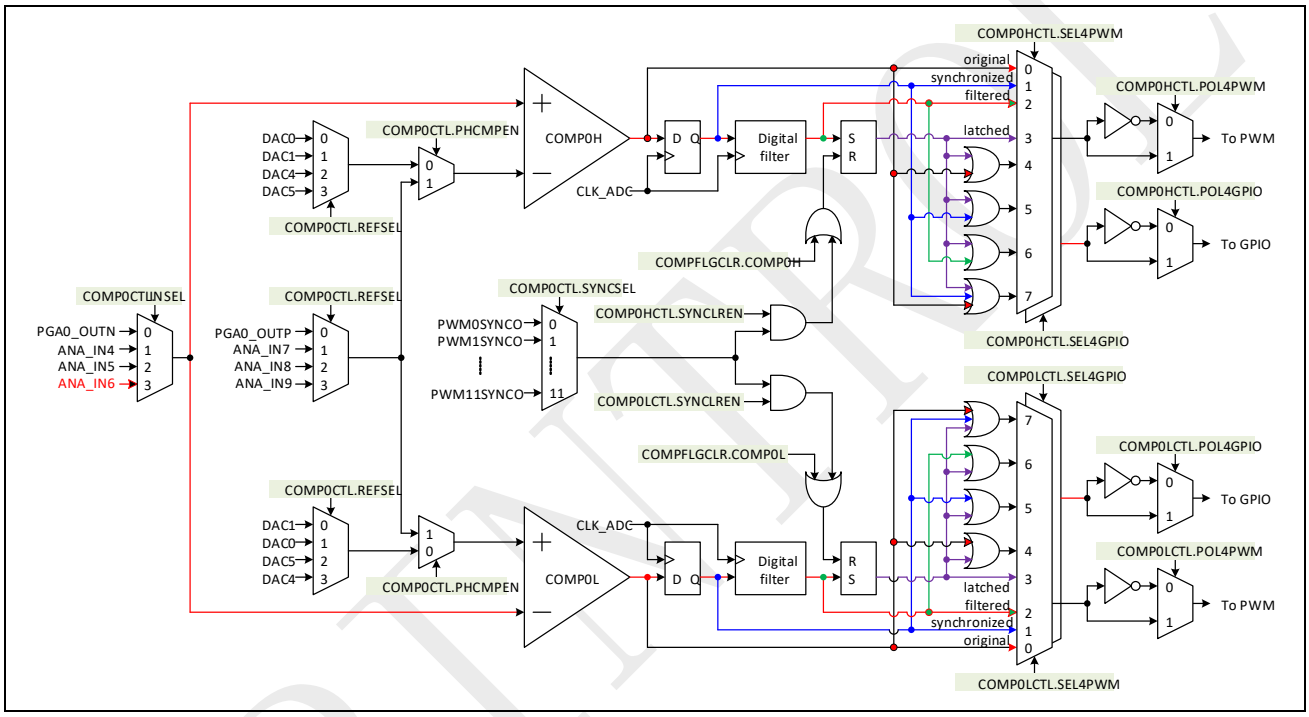
2 COMP 实例

2.1 COMP 示意图

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。

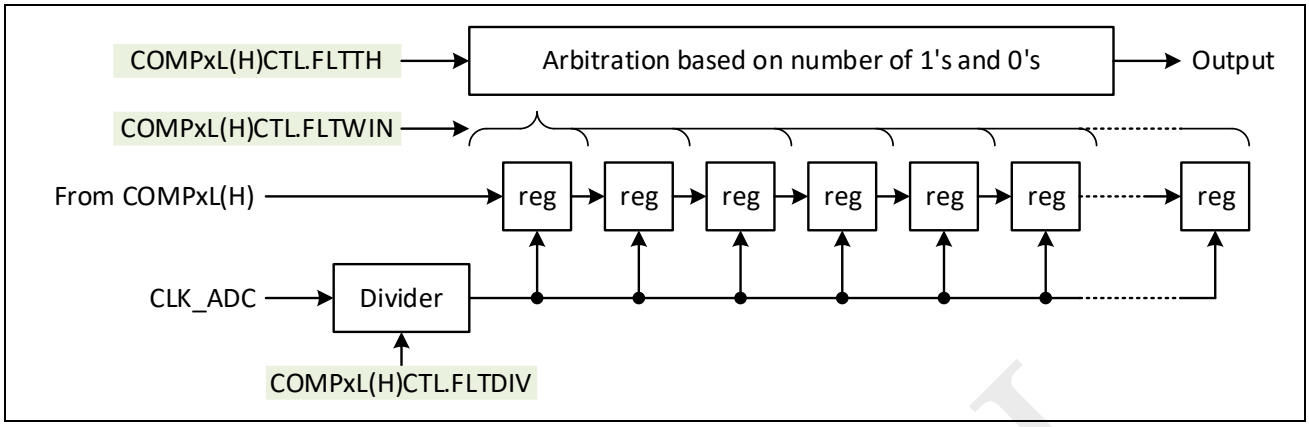
COMP 输入源有 GPIO，PGA，参考电压为 DAC。其输出事件作用于 PWM 模块（产生对应的同步，封锁，TZ 中断，ADC 采样，PWM 中断等事件），其详细信号流图可参考《PWM 使用指南》。

图 2-1：COMP 示意图



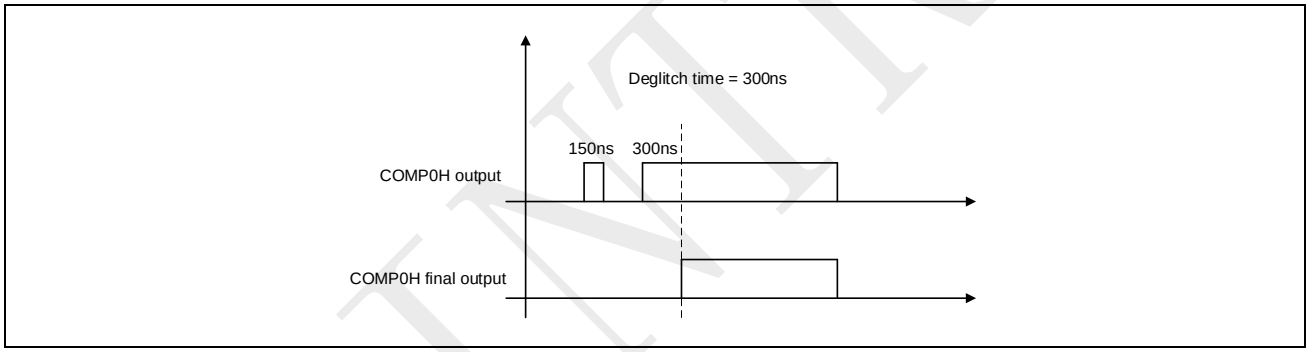
此瞬时响应会误触发 COMP 的输出，并且将 PWM 关断，造成不必要的停机，因此在设计上，提供了输出数字滤波器。其基本原理如下图 2-2 所示，COMP 的输出首先送到锁存器中，时钟来源于 ADCCLK，并且最大可以分频到 1024 倍。然后设置采样窗口的大小和阈值的大小，范围可以从 0~31 中选择，并且要保证阈值要大于采样窗口的一半，如果采样窗口内采到的 COMP 输出 0 的个数大于阈值，则数字滤波器输出为 0，如果采样窗口内的 COMP 输出 1 的个数大于阈值，则数字滤波器输出为 1，如果采用窗口内的 COMP 输出 0 和 1 个数都没有超过阈值，则数字滤波器输出不变。

图 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，才能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 GPIO6 和 DAC0, DAC1 的电压。

Example Code

```
#include <stdio.h>

#include "spc2188.h"

#define             SampleRgisterNVol      1200                              /*
Reference voltage : mV */
#define             VolOffsetmV           100                              /*
Reference offset voltage : mV*/
#define             COMP_FilterWIN         300                               /* Ns
*/

int main(void)
{

    CLOCK_InitWithRCO(240000000U);

    Delay_Init();

    /*
```

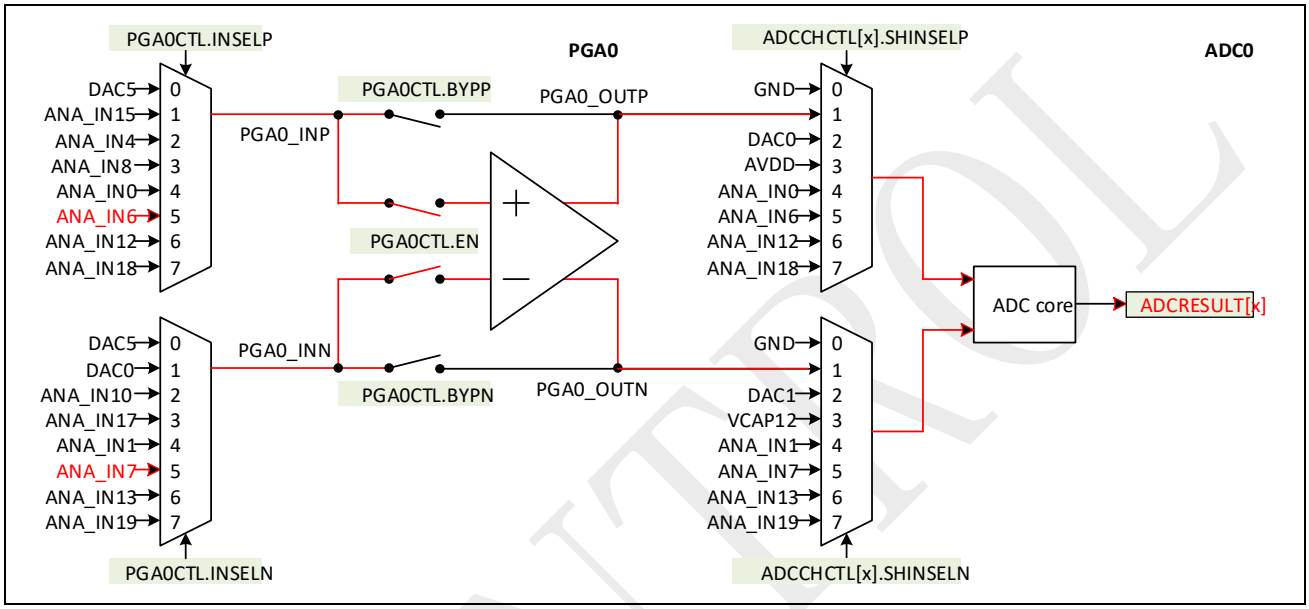
```
    * Initial the UART0
    */
    PIN_SetChannel(PIN_GPIO62, PIN_GPIO62_UART0_TXD);
    PIN_SetChannel(PIN_GPIO63, PIN_GPIO63_UART0_RXD);
    UART_Init(UART0, 38400);

    /*
    * set the GPIO as ADC.
    */
    PIN_SetChannel(PIN_GPIO6,PIN_GPIO6_ANA_IN6);

    /*
    * 1. Initialize comparator with 300ns filtering window.
    * 2. Set DAC voltage, COMP_H negative port is (SampleRgisterNVol -
VolOffsetmV)mV,
    *  COMP_L positive port is (SampleRgisterNVol + VolOffsetmV)mV
    */
    COMP_Init(COMP0_H, COMP0_FROM_ANA_IN6, COMP0_REF_DAC0_DAC1,
SampleRgisterNVol + VolOffsetmV, COMP_FilterWIN);
    COMP_Init(COMP0_L, COMP0_FROM_ANA_IN6, COMP0_REF_DAC0_DAC1,
SampleRgisterNVol - VolOffsetmV, COMP_FilterWIN);

    while(1)
    {
        /* Get the comparator status */
        if(COMP_GetFilterOutputFlag(COMP0_H) != 0)
        {
            printf("The detect voltage is higher than %d mV\n",SampleRgisterNVol
+ VolOffsetmV);
        }
        else if(COMP_GetFilterOutputFlag(COMP0_L) != 0)
        {
            printf("The detect voltage is lower than %d mV\n",SampleRgisterNVol -
VolOffsetmV);
        }

        /* Clear latched filter status */
        COMP_ClearFilterOutputFlag(COMP0_L|COMP0_H);

        Delay_Ms(500);
    }
}
```

3 PGA 实例

3.1 PGA 单端放大送入 ADC 采样

使用 ADC 对 PGA 单端放大后电压采样，其连接图如图 3-1 所示。

图 3-1：ADC PGA 单端采样



以下例子演示使用 ADC 对 PGA 的 positive 和 negative 端放大后的电压进行采样，将 PGA 放大后的电压送给 ADC0 CH0 和 CH1 进行采样，默认采样时间，转换时间均设定为 140ns。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。

Example Code

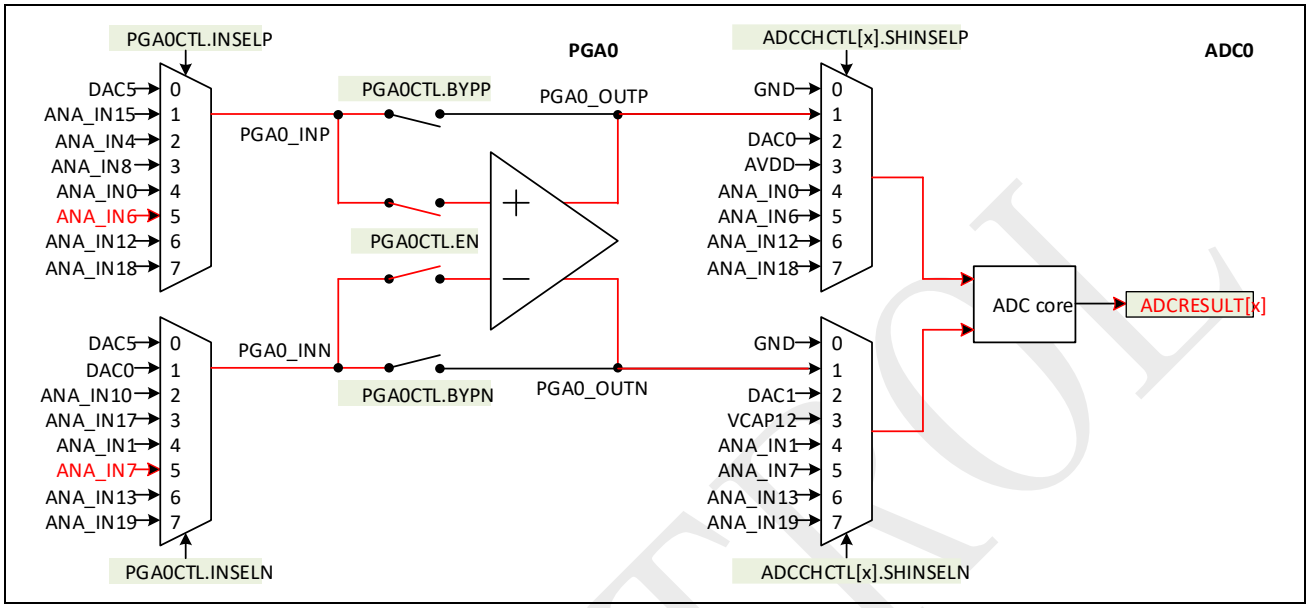
```
#include "spc2188.h"
#include <stdio.h>


/* 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  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);
    }
}
```

3.2 PGA 差分放大送入 ADC 采样

使用 ADC 对 PGA 差分放大后电压采样，其连接图如图 3-2 所示。

图 3-2：ADC PGA 差分采样



以下例子演示使用 ADC 对 PGA 差分放大后电压进行采样，将 PGA 放大后电压送给 ADC CH0 进行采样，默认采样时间，转换时间均设定为 140ns。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。

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);


    /* 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);
    }
}
```