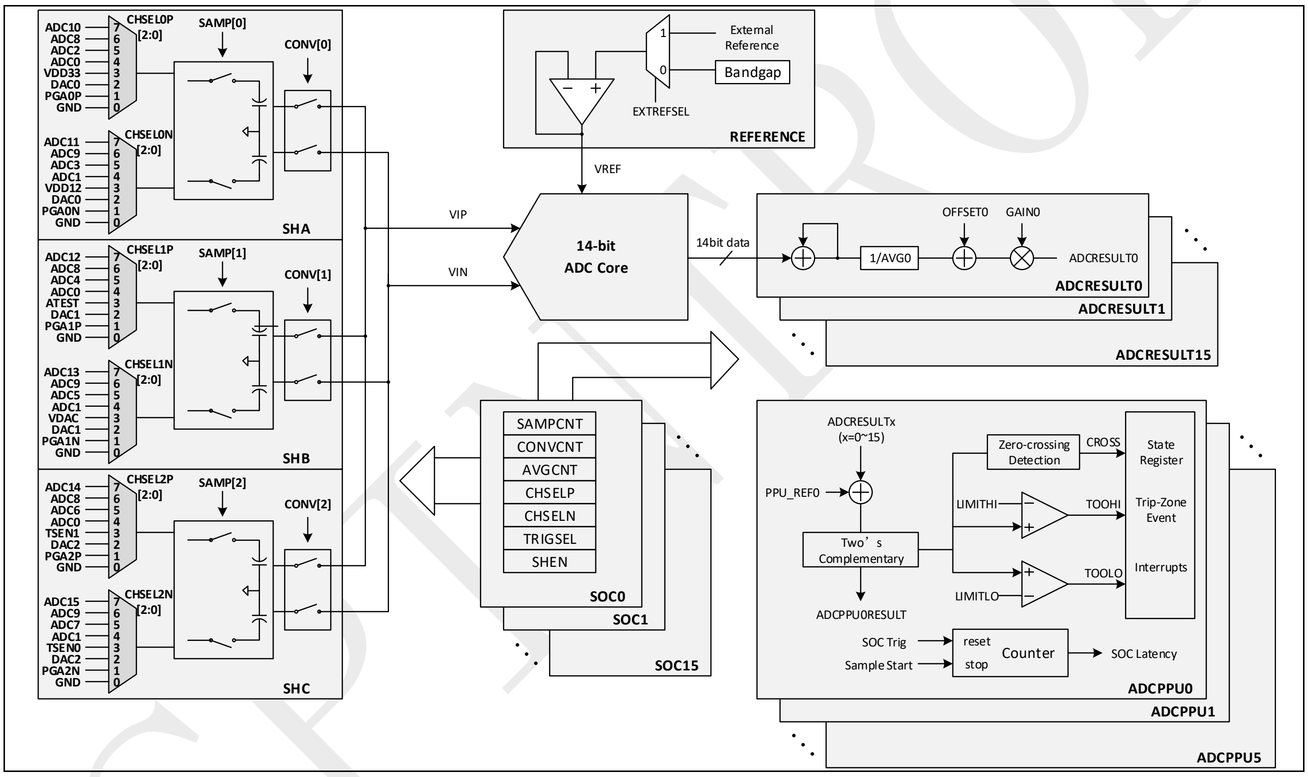
SPC1168 ADC 使用指南

概述

在电力电子系统中，通常使用 ADC 进行电压采样，将模拟电压转换为数字电压。SPC1168 中有一个 14 位 SAR ADC，其结构框图如图 1 所示。

图 1：ADC 结构框图



目录

图片列表

版本历史

版本	日期	作者	状态	变更
A/0	2023-06-15	CanChai	Outdated	首次发布。
C/0	2024-04-22	Jiali Zhou	Released	修改排版格式。

术语或缩写

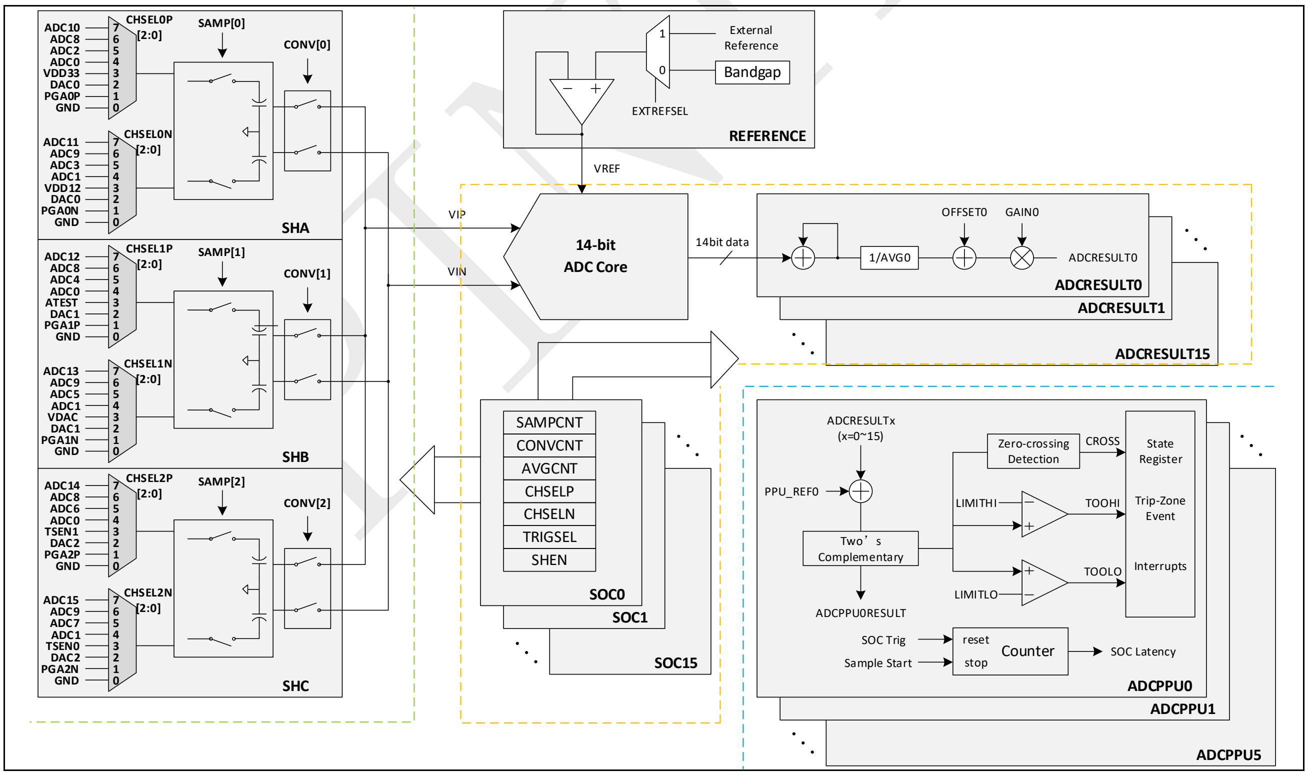
术语或缩写	描述
共模电压	差分电路两个输入端电压的平均值，也称共模信号。
SAR ADC	逐次逼近型 ADC

1 整体功能解释

在实际使用 ADC 过程中，可以按照功能区块来帮助理解怎样使用 ADC，ADC 大体上可以按照图 1 中的不同颜色虚线分成 3 个功能区块，在使用过程中按照如下步骤进行配置：

- 首先绿色虚线框内只需要使能采样及转换开关，这个步骤可以等其它配置都设置完成之后再进行；
- 然后设置黄色虚线框部分，在这部分中，有 16 路独立的采样转换配置（SOC，Set-of-Convert）可以预先设置，每个 SOC 均可根据不同的目的而配置不同的参数，例如：采样时间（具体需要设置多长的采样时间，请参考《ADC 建立时间计算方法使用指南》），ADC core 转换时间，是否需要多次采样且取平均值，ADC 输入的正负端，触发源选择等；在设置完这些信息之后，对应 SOC 的设置产生的 ADC 转换结果将存储在对应的 ADCRESULTSx（x=0,1，…，15）中。
- 最后，如果有需求，可以继续设置蓝色虚线框部分，此部分即为 ADC PPU 单元。通常工程使用中，需要知道 ADC 转换的结果相对某个参考值而言是过高还是过低，这部分的工作就可以交给 PPU 单元处理，以减少 CPU 的计算压力。除此之外，PPU 单元还可以检测出从发起 ADC 转换请求至开始 ADC 转换时的时间计数。

图 1-1：ADC 功能分区框图



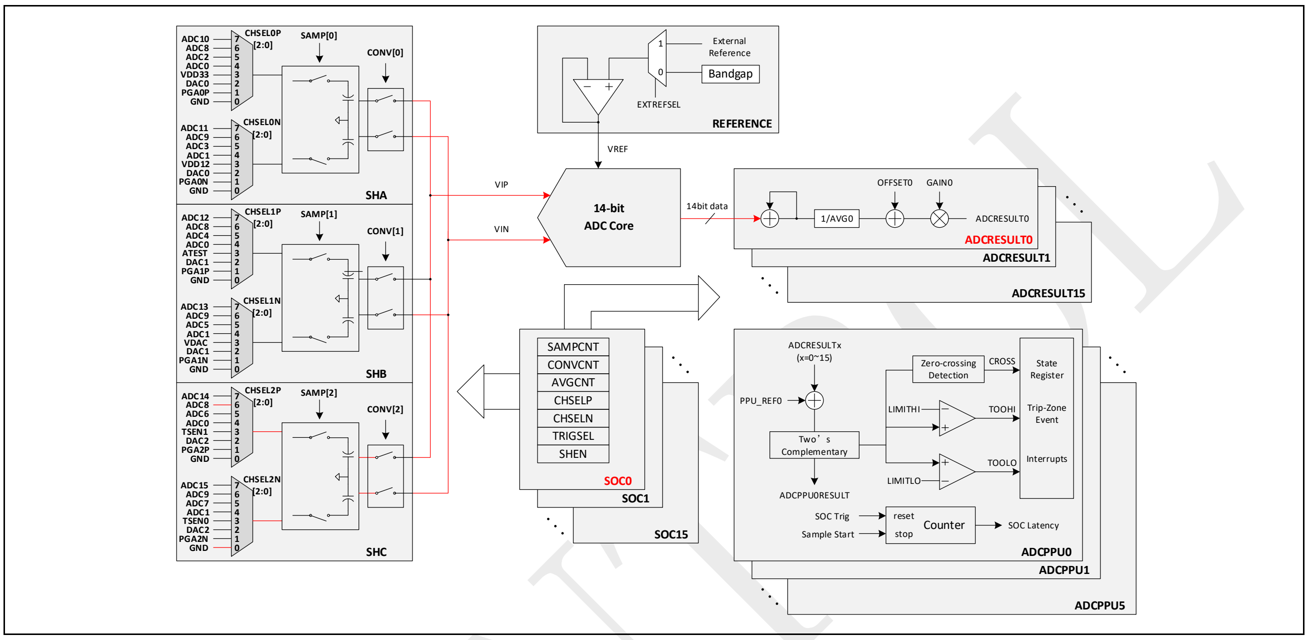
注意： 需要注意的一点是，SPC1168 的 ADC 正端或者负端输入的绝对电压必须保持大于等于零，否则 ADC 工作不正常。

2 ADC 示例

2.1 ADC 单端采样

利用 ADC 进行单端（也即有一端接 GND）采样，如图 2-1 所示。

图 2-1：ADC 单端采样信号流



实现图示中描述的功能的代码步骤如下：

在 ADC_Init 函数接口中填入相关参数，即可完成 ADC 时钟等的初始化，ADC 通道选择为 SOC0，正端输入选择为 GPIO_8，负端输入选择为 GND，触发源，采样时间，转换时间的相关设置。需要注意的是，函数接口内会将对应的 GPIO 设置成为模拟 IO。

若想单独设置其它参数，可调用其它接口进行设置，例如，在以下代码中，调用 ADC_SetAverageCnt 进行对目标电压采样 32 次，且结果取均值的设置。

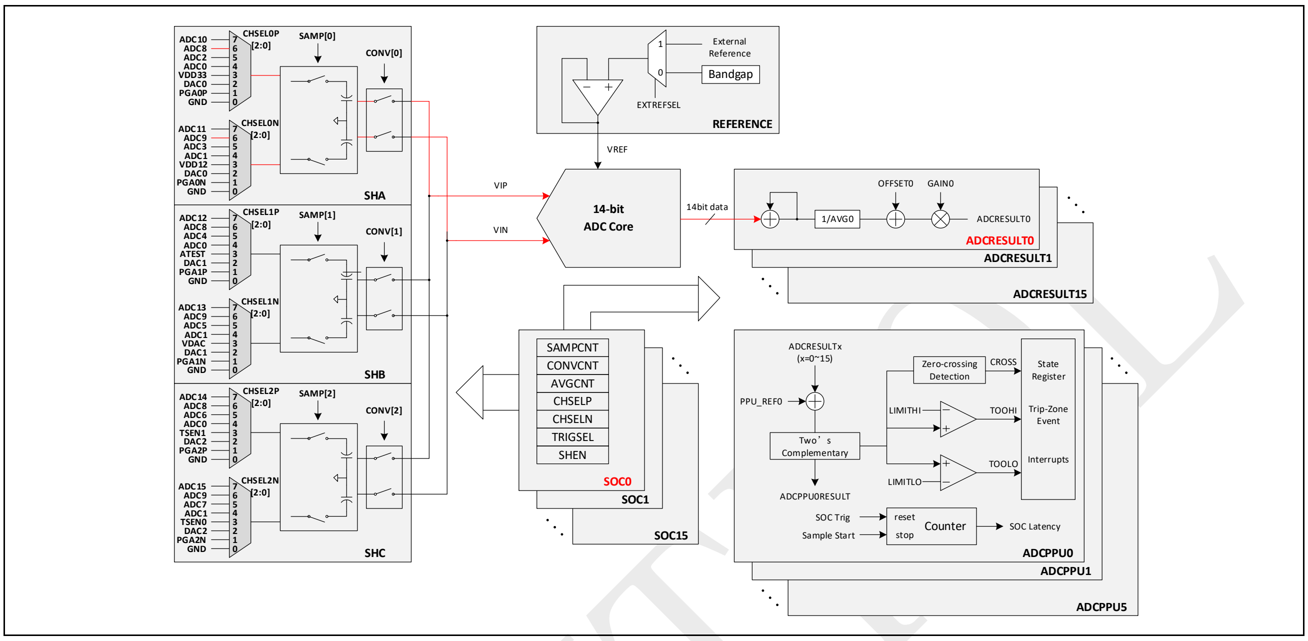
Example Code

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

int32_t          i32VSP;

int main()
{
    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);

    /*ADC Init*/
    GPIO_SetPinChannel(GPIO_8, GPIO8_ADC8);
     ADC_Init(ADC_SOC_0, ADC_SHC_P_ANA_IN8, ADC_SHC_N_GND, SHC,
ADCTRIG_Software);
     ADC_SetAverageCnt(ADCSOCCTL0_BIT_AVGCNT_AVG_32);

    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));

        /* Result gotten from 'ADC_GetResult()' can be a negative value or a
positive value */
        i32VSP = ADC_GetResult(ADC_SOC_0);
        printf("ADC SOC0 Voltage = %2fV(Raw data is %d)\n",
(double)ValueToVoltage(i32VSP), i32VSP);

        /* Result gotten from 'ADC_GetTrimResult1()' is a positive value */
        i32VSP = ADC_GetTrimResult1(ADC_SOC_0);
        printf("ADC SOC0 Voltage = %2fV(Trim data is %d)\n",
(double)ValueToVoltage(i32VSP), i32VSP);

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

        Delay_Ms(500);
    }
}
```

2.2 ADC 双端采样

利用 ADC 进行双端采样，如图 2-2 所示。

图 2-2：ADC 双端采样信号流



实现图示中描述的功能的代码步骤如下：

在 ADC_Init 函数接口中填入相关参数，即可完成 ADC 时钟等的初始化，选择 SOC0，正端输入选择 GPIO_8，负端输入选择 GPIO_9，采样保持器为采样保持 C，最后设置触发源、采样时间、转换时间。在此需要说明的是，函数接口内会将对应的 GPIO 设置成为模拟 IO。

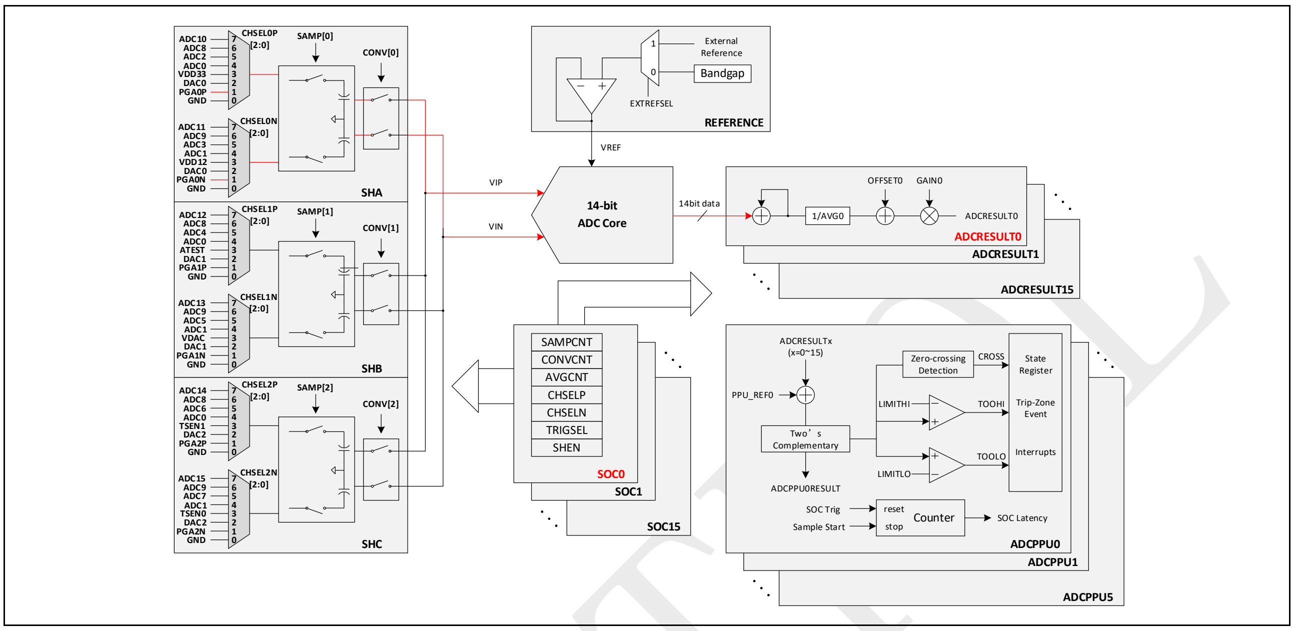
Example Code

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

int32_t             i16VSP;


/*******************************************************************************
*******************************************
*
* @brief     In this case, we use two GPIO(8~9) as the input of the ADC, then
getting result from the trim register.
*
*******************************************************************************
******************************************/


int main()
{
    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);

    /* ADC Init */
    GPIO_SetPinChannel(GPIO_8, GPIO8_ADC8);
    GPIO_SetPinChannel(GPIO_9, GPIO9_ADC9);
    ADC_Init(ADC_SOC_0, ADC_SHA_P_ANA_IN8, ADC_SHA_N_ANA_IN9, SHA,
ADCTRIG_Software);

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

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

        /* Get result */
        i16VSP = ADC_GetTrimResult2(ADC_SOC_0);

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

        printf("ADC SOC9 Voltage = %2fV(Trim data is %d)\n",
(double)ValueToVoltage(i16VSP), i16VSP);

        Delay_Ms(500);
    }
}
```

2.3 ADC PGA 采样

利用 ADC 对 PGA 的输出进行采样，如图 2-3 所示。

图 2-3：ADC 对 PGA 输出的采样信号流



实现图示中描述的功能的代码步骤如下：

设置 GPIO_8 和 GPIO_9 为模拟 IO，并调用 PGA_DifferentialInit 接口，将 PGA0 的正端输入设置为模拟 IO_8（也即 GPIO_8），负端设置为模拟 IO_9（也即 GPIO_9），且将放大倍数设置为 4 倍。

在 ADC_Init 函数接口中填入相关参数，即可完成 ADC 时钟等的初始化，选择 SOC0，正端输入选择 PGA0 的正端输出，负端输入选择 PGA0 的负端输出，采样保持器为采样保持 A，最后设置触发源、采样时间、转换时间。

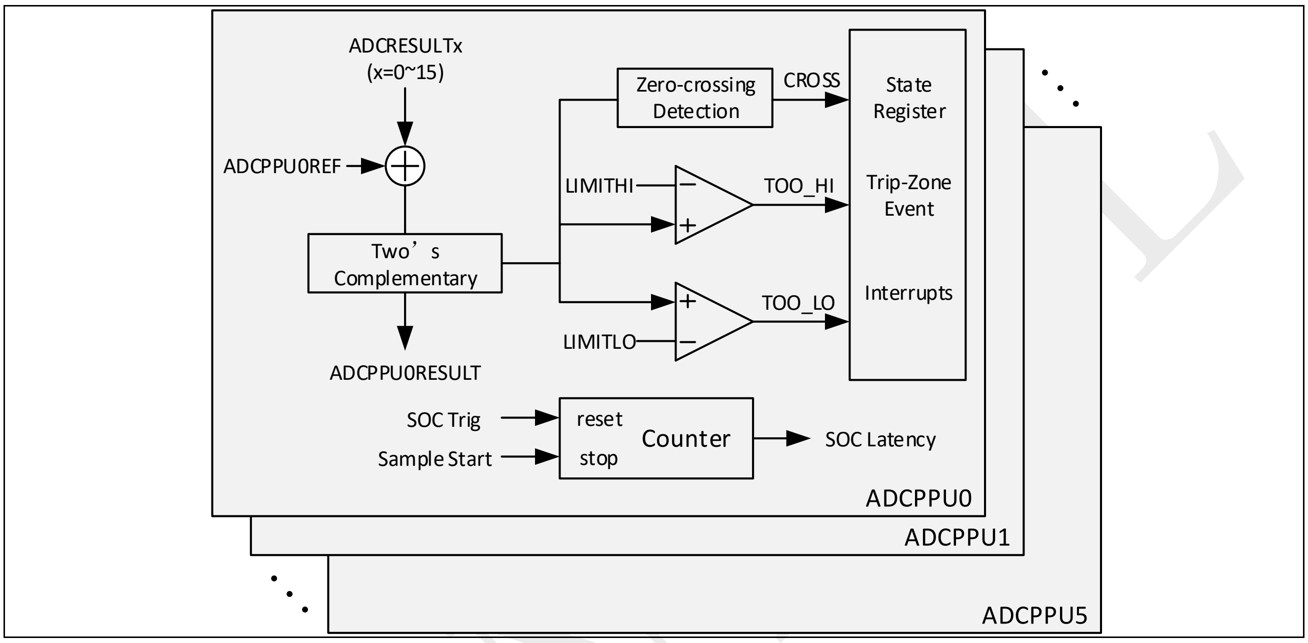
Example Code

```
/* 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();

    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);

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

2.4 ADC 后处理单元

如手册第一章节所述，通常工程使用中，需要知道 ADC 转换的结果相对某个参考值而言是过高还是过低，这部分的工作就可以交给 PPU 单元处理，以减少 CPU 的计算压力。除此之外，PPU 单元还可以检测出从发起 ADC 转换请求至开始 ADC 转换时的时间计数。PPU 单元的框图如图 2-4 所示，每个 PPU 单元均可独立选择一路 SOC 的结果进行侦测。

图 2-4：PPU 结构框图



以下示例代码将举例说明 PPU 单元的使用方法：

- 调用 ADC_Init 初始化 ADC，并进行相关设置；
- 调用 ADC_PPUInit 函数，选择 PPU 的输入，设置比较极性，设置参考电压值，随后使能 PPU；
- 打开 PPU 过压、欠压、过零中断；
- 设置 PPU 过压以及欠压阈值；
- 在示例代码中，当加在 GPIO_8 电压超过设定阈值时，将会产生对应中断。

Example Code

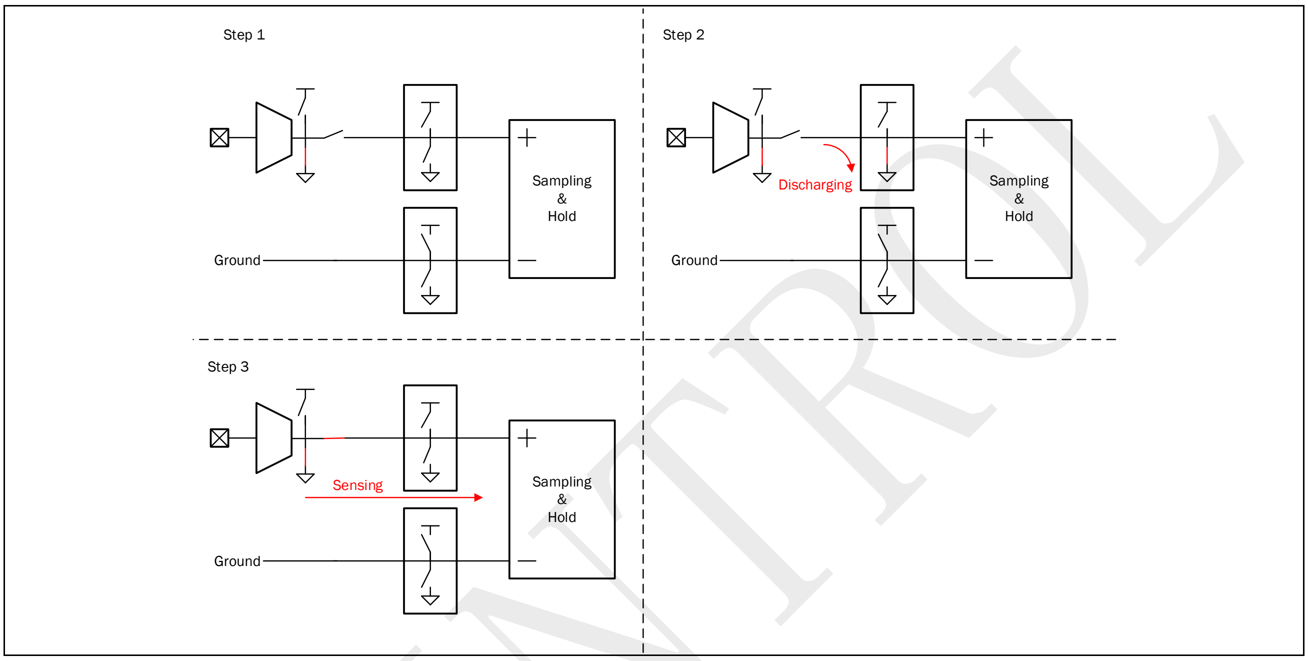
```
/* As description below, ADC result convert to voltage can calculate as the
follow */
#define           ValueToVoltage(x)           ((x * 3657) / 8192) /1000

int32_t           i32PPUResult;

int main()
{
    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);

    /* ADC Init */
    GPIO_SetPinChannel(GPIO_8, GPIO8_ADC8);
    ADC_Init(ADC_SOC_0, ADC_SHC_P_ANA_IN8, ADC_SHC_N_GND, SHC,
ADCTRIG_Software);

    /* Init ADC PPU */
    ADC_PPUInit(ADC_PPU_0, 0, 100, ADC_PPU_CODE_MINUS_REF);

    /* Enable zero-crossing interrupt of ADC PPU0 */
    ADC_EnablePPUInt(ADC_PPU_0, ADC_PPU_CROSS_ZERO);
    /* Enable too high interrupt of ADC PPU0 */
    ADC_EnablePPUInt(ADC_PPU_0, ADC_PPU_TOO_HIGH);
    /* Enable too low interrupt of ADC PPU0 */
    ADC_EnablePPUInt(ADC_PPU_0, ADC_PPU_TOO_LOW);

    /* Set PPU0 high threshold is 1500 */
    ADC_SetPPUTooHighThreshold(ADC_PPU_0, 1500);
    /* Set PPU0 low threshold is 1000 */
    ADC_SetPPUTooLowThreshold(ADC_PPU_0, 1000);

    /* Clear SOC0 INT flag */
    ADC_ClearInt0();

    /* Clear PPU Unit INT */
    ADC_ClearPPUGlobalInt(ADC_PPU_0);
    ADC_ClearPPUInt(ADC_PPU_0, ADC_PPU_TOO_LOW);
    ADC_ClearPPUInt(ADC_PPU_0, ADC_PPU_TOO_HIGH);
    ADC_ClearPPUInt(ADC_PPU_0, ADC_PPU_CROSS_ZERO);

    NVIC_EnableIRQ(ADCPPU0_IRQn);

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

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

    while (1)
    {

    }
}

void ADCPPU0_IRQHandler(void)
{
    if (ADC_GetPPUIntFlag(ADC_PPU_0, ADC_PPU_CROSS_ZERO) != 0)
    {
        printf("Zero-cross INT had happened\n");
    }

    if (ADC_GetPPUIntFlag(ADC_PPU_0, ADC_PPU_TOO_HIGH) != 0)
    {
        printf("ADC channel voltage too-high INT had happened\n");
    }

    if (ADC_GetPPUIntFlag(ADC_PPU_0, ADC_PPU_TOO_LOW) != 0)
    {
        printf("ADC channel voltage too-low INT had happened\n");
    }

    /* Result gotten from 'ADC_GetResult()' can be a negative value or a
positive value */
    i32PPUResult = ADC_GetResult(ADC_SOC_0);
    printf("ADC SOC0 Voltage = %2fV(data is %d)\n",
(double)ValueToVoltage(i32PPUResult), i32PPUResult);

    /* Result gotten from 'ADC_GetTrimResult1()' is a positive value */
    i32PPUResult = ADC_GetTrimResult1(ADC_SOC_0);
    printf("ADC SOC0 Voltage = %2fV(Trim data is %d)\n",
(double)ValueToVoltage(i32PPUResult), i32PPUResult);

    i32PPUResult = ADC_GetPPUResult(ADC_PPU_0);
    printf("SOC0 Comparison data : %d\n", i32PPUResult);

    i32PPUResult = ADC_GetSOCDelay(ADC_PPU_0);
    printf("The period between the trigger signal and the start of converting :
%d\n", i32PPUResult);

    /* Clear the PPU0 INTs and global INT of PPU0 */
    ADC_ClearPPUInt(ADC_PPU_0, ADC_PPU_CROSS_ZERO);
    ADC_ClearPPUInt(ADC_PPU_0, ADC_PPU_TOO_HIGH);
    ADC_ClearPPUInt(ADC_PPU_0, ADC_PPU_TOO_LOW);
    ADC_ClearPPUGlobalInt(ADC_PPU_0);

    /* Continues to use software to trigger the ADC to work */
    ADC_SoftwareTrigger(ADC_SOC_0);

    Delay_Ms(2000);
}
```

2.5 ADC 开路检测

如果用于模拟输入的引脚未与外部传感器正常连接，电压将无法正确测量，可能导致某些应用程序的错误。ADC 单元提供了 ADC 输入浮动检测电路，用以帮助检测此类错误。

可以使用如下两个步骤检测 ADC 单元输入是否浮空。如图 2-5 所示，展示了步骤 1，如图 2-6 所示，展示了步骤 2。

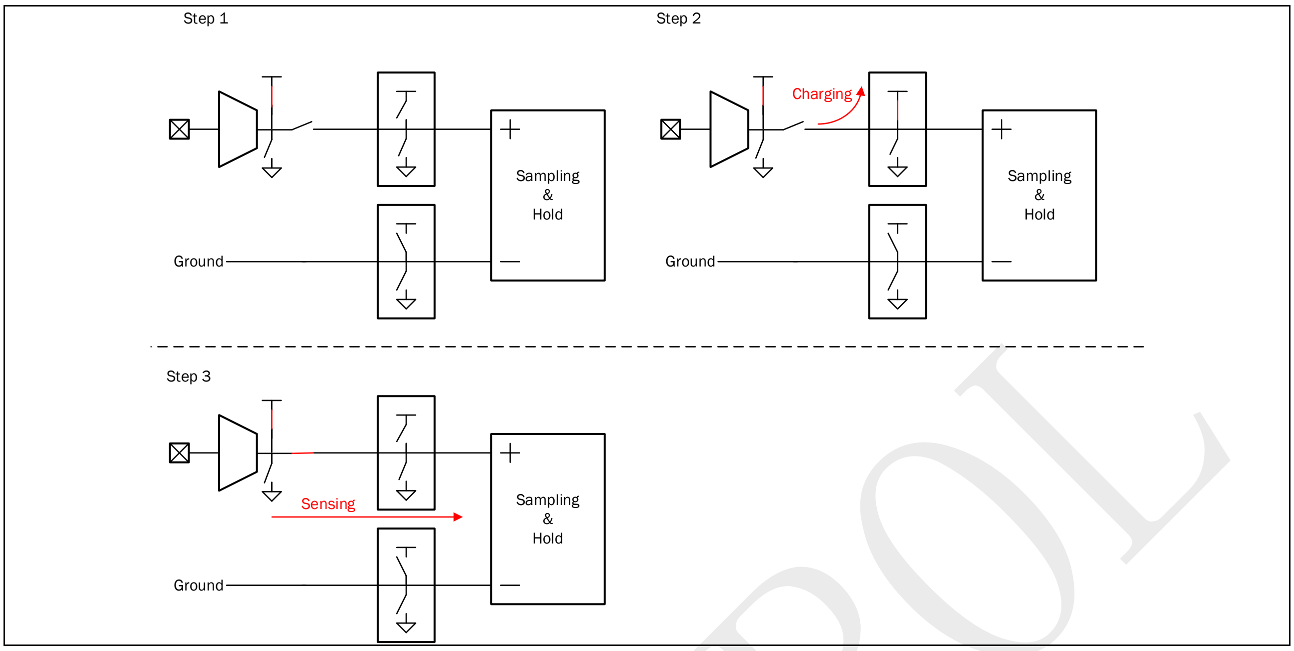
图 2-5：检测 ADC 输入端引脚是否浮空（正端强制接地）



步骤 1 的具体操作如下：

- Step1：将 PIN 脚切成 GPIO 功能，设置为输入，并使能 GPIO 下拉，释放 IO 上残存的电量；
- Step2：将 ADC 采样保持的正端强制接地，释放通路上的残存电量；
- Step3：将 ADC 采样保持的正端恢复断开状态，将 PIN 脚切成 ADC 功能，降低 ADC 模块的时钟频率，设置较长采样时间，以期获得更为准确的测试数据，随后开始测量，获得电压数值 Result1；

图 2-6：检测 ADC 输入端引脚是否浮空（正端强制接高）



步骤 2 的具体描述如下：

- Step1：将 PIN 脚切成 GPIO 功能，设置为输入，并使能 GPIO 上拉，给 IO 的通路进行预充电；
- Step2：将 ADC 采样保持的正端强制拉高，对采样保持器通路进行预充电；
- Step3：将 ADC 采样保持的正端恢复断开状态，将 PIN 脚切成 ADC 功能，降低 ADC 模块的时钟频率，设置较长采样时间，以期获得更为准确的测试数据，随后开始测量，获得电压数值 Result2；

如果 Result1 在 0V 左右，且 Result2 在 3.3V 左右，则表明采样器的输入未链接外部输入，即正输入端处于漂浮状态。类似的步骤也可以用来检测采样器的负输入引脚是否悬空。

Example Code

```
#define         RESULTTOVOLTAGE(x)          (double)(((double)((3657 * x) /
8192)) / 1000)

int32_t         i32lresult, i32hresult;


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);

    printf("======= Bgin to try to tied the ADC8 low and then sample the
voltage ======\n");
    /* Turn off the two switch */
    ADC_SetSOCSH(ADC_SOC_0, SH_DISABLE);

    /* Set GPIO8 as GPIO mode */
    GPIO_SetPinChannel(GPIO_8, GPIO8_GPIO8);

    /*
     * Enable GPIO input, otherwise, the input will be always zero and the
result of ADC sampler
     * will be always zero.
     */
    PINMUX->GPIO8.bit.IE = 1;

    /*
     * Set the GPIO8 as pull down model to discharge, keep in mind that the pull
up and pull down
     * is not exclusive.
     */
    GPIO_DisablePullUp(GPIO_8, GPIO8_GPIO8);
    GPIO_EnablePullDown(GPIO_8, GPIO8_GPIO8);

    /* Force the SHC positive pin low */
    ADC->ADCCTL.bit.TIECP = ADCCTL_BIT_TIECP_FORCE_LOW;

    /* Delay for a short time to let the discharge over */
    Delay_Ms(100);

    ADC->ADCCTL.bit.TIECP = ADCCTL_BIT_TIECP_DISABLE;

    /*
     * ADC Init, config the GPIO8 as the positive channel and the negative is
GND.
     * Then On two switch.
     */
    GPIO_SetPinChannel(GPIO_8, GPIO8_ADC8);
    ADC_Init(ADC_SOC_0, ADC_SHC_P_ANA_IN8, ADC_SHC_N_GND, SHC,
ADCTRIG_Software);


    /* Div ADC clk, the more lower clock the more accurate the sample result */
    CLOCK->ADCCLKCTL.bit.DIV = 0xFF;

    /*
     * set ADC sample time (as the same as the ADC clock), the bigger value of
     * SAMPCNT/CONVCNT, the more accurate the sample result
     */
    ADC->ADCSOCCTL[0].bit.SAMPCNT = 255;   /* SAMPCNT is 8-bit length */
    ADC->ADCSOCCTL[0].bit.CONVCNT = 1;     /* CONVCNT is 7-bit length */

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

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

    /* Get the result */
    i32lresult = ADC_GetResult(ADC_SOC_0);
    printf("Voltage is (%d) : %0.3fV\n", i32lresult,
RESULTTOVOLTAGE(i32lresult));

    printf("======= Bgin to try to tied the ADC8 high and then sample the
voltage ======\n");
    /* Turn off the two switch */
    ADC_SetSOCSH(ADC_SOC_0, SH_DISABLE);

    /* Set GPIO8 as GPIO mode */
    GPIO_SetPinChannel(GPIO_8, GPIO8_GPIO8);

    /*
     * Set the GPIO8 as pull up model to pre-charge, keep in mind that the pull
up and pull down
     * is not exclusive.
     */
    GPIO_EnablePullUp(GPIO_8, GPIO8_GPIO8);
    GPIO_DisablePullDown(GPIO_8, GPIO8_GPIO8);

    /* Force the SHC positive pin high */
    ADC->ADCCTL.bit.TIECP = ADCCTL_BIT_TIECP_FORCE_HIGH;

    /* Delay for a short time to let the pre-charge over */
    Delay_Ms(100);
    ADC->ADCCTL.bit.TIECP = ADCCTL_BIT_TIECP_DISABLE;

    /*
     * ADC Init, config the GPIO8 as the positive channel and the negative is
GND.
     * Then On two switch.
     */
    GPIO_SetPinChannel(GPIO_8, GPIO8_ADC8);
    ADC_Init(ADC_SOC_0, ADC_SHC_P_ANA_IN8, ADC_SHC_N_GND, SHC,
ADCTRIG_Software);

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

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

    /* Get the ADC SOC result */
    i32hresult = ADC_GetResult(ADC_SOC_0);
    printf("Voltage is (%d) : %0.3fV\n", i32hresult,
RESULTTOVOLTAGE(i32hresult));

    /* Check the result */
    if (i32lresult < 300 && i32hresult > 6500)
    {
        printf("GPIO8 is disconnected\n");
    }
    else
    {
        printf("GPIO8 is connected\n");
    }

    while (1)
    {

    }
}
```

2.6 ADC 短路检测

当来自传感器的模拟信号异常短路时，我们的 ADC 具有短路检测功能，这样可以给 MCU 提供失效安全处理的机会。

输入短路检测功能可以用 ADCCTL 寄存器设置。检测可以分为下面几步：

- ADC 测量输入 PIN 脚电压，设为结果 DATA1。
- 将 PIN 脚切成 GPIO 功能，设置为输入，给输入 PIN 接弱下拉 GND，ADC 测量输入 PIN 脚电压，设为结果 DATA2。
- 将 PIN 脚切成 GPIO 功能，设置为输入，给输入 PIN 接弱上拉 VDD，ADC 测量输入 PIN 脚电压，设为结果 DATA3。
- 如果 DATA1~= DATA2~=DATA3，那么输入 PIN 脚短路。

注意： 只有当驱动源驱动能力很强的时候，才能测量输入 PIN 脚短路。假设 ADC 的分辨率为 1mV，ADC 内部的弱上拉或者弱下拉电流为 8uA，那么最大允许的驱动源阻抗为 1mV/8uA=125 Ω。如果 ADC 输入驱动源能力较弱，它的输出阻抗大于这个值，那么就不能采用上述方法检测输入短路。

Example Code

```
#define          RESULTTOVOLTAGE(x)      (double)(((double)((3657 * x) /
8192)) / 1000)

int32_t          VALUE1, VALUE2, VALUE3;

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);

    printf("=======              Bgin to measure GPIO8 voltage
======\n");
    /*
     * ADC Init, config the GPIO8 as the positive channel and the negative is
GND.
     * Then On two switch.
     */
    GPIO_SetPinChannel(GPIO_8, GPIO8_ADC8);
    ADC_Init(ADC_SOC_0, ADC_SHC_P_ANA_IN8, ADC_SHC_N_GND, SHC,
ADCTRIG_Software);

    /* div ADC clk, the more lower clock the more accurate the sample result*/
    CLOCK->ADCCLKCTL.bit.DIV = 0xFF;

    /*
     * set ADC sample time (as the same as the ADC clock), the bigger value of
     * SAMPCNT/CONVCNT, the more accurate the sample result
     */
    ADC->ADCSOCCTL[0].bit.SAMPCNT = 255;   /* SAMPCNT is 8-bit length */
    ADC->ADCSOCCTL[0].bit.CONVCNT = 1;     /* CONVCNT is 7-bit length */

    /* Software trigger */
    ADC_SoftwareTrigger(ADC_SOC_0);

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

    /*Get the result*/
    VALUE1 = ADC_GetResult(ADC_SOC_0);
    printf("Voltage is (%d) : %0.3fV\n", VALUE1, RESULTTOVOLTAGE(VALUE1));

    printf("======= Bgin to try to tied the ADC8 to GND and then sample the
voltage ======\n");
    /* Set GPIO8 as GPIO mode */
    GPIO_SetPinChannel(GPIO_8, GPIO8_GPIO8);

    /*
     * Enable GPIO input, otherwise, the input will be always zero and the
result of ADC sampler
     * will be always zero.
     */
    PINMUX->GPIO8.bit.IE = 1;

    /*
     * Set the GPIO8 as pull down model to discharge, keep in mind that the pull
up and pull down
     * is not exclusive.
     */
    GPIO_DisablePullUp(GPIO_8, GPIO8_GPIO8);
    GPIO_EnablePullDown(GPIO_8, GPIO8_GPIO8);

    /*Delay for a short time to let the discharge over*/
    Delay_Ms(100);

    /*
     * ADC Init, config the GPIO8 as the positive channel and the negative is
GND.
     * Then On two switch.
     */
    GPIO_SetPinChannel(GPIO_8, GPIO8_ADC8);
    ADC_Init(ADC_SOC_0, ADC_SHC_P_ANA_IN8, ADC_SHC_N_GND, SHC,
ADCTRIG_Software);

    /* div ADC clk, the more lower clock the more accurate the sample result*/
    CLOCK->ADCCLKCTL.bit.DIV = 0xFF;

    /*
     * set ADC sample time (as the same as the ADC clock), the bigger value of
     * SAMPCNT/CONVCNT, the more accurate the sample result
     */
    ADC->ADCSOCCTL[0].bit.SAMPCNT = 255;   /* SAMPCNT is 8-bit length */
    ADC->ADCSOCCTL[0].bit.CONVCNT = 1;     /* CONVCNT is 7-bit length */

    /* Use software to trigger ADC SOC0 to sample and convert */
    ADC_SoftwareTrigger(ADC_SOC_0);

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

    /* Get the result */
    VALUE2 = ADC_GetResult(ADC_SOC_0);
    printf("Voltage is (%d) : %0.3fV\n", VALUE2, RESULTTOVOLTAGE(VALUE2));

    printf("======= Bgin to try to tied the ADC8 to AVDD and then sample the
voltage ======\n");
    /* Set GPIO8 as GPIO mode */
    GPIO_SetPinChannel(GPIO_8, GPIO8_GPIO8);

    /*
     * Set the GPIO8 as pull up model to pre-charge, keep in mind that the pull
up and pull down
     * is not exclusive.
     */
    GPIO_EnablePullUp(GPIO_8, GPIO8_GPIO8);
    GPIO_DisablePullDown(GPIO_8, GPIO8_GPIO8);

    /* Delay for a short time to let the pre-charge over */
    Delay_Ms(100);

    /*
     * ADC Init, config the GPIO8 as the positive channel and the negative is
GND.
     * Then On two switch.
     */
    GPIO_SetPinChannel(GPIO_8, GPIO8_ADC8);
    ADC_Init(ADC_SOC_0, ADC_SHC_P_ANA_IN8, ADC_SHC_N_GND, SHC,
ADCTRIG_Software);

    /* Use software to trigger ADC SOC0 to sample and convert */
    ADC_SoftwareTrigger(ADC_SOC_0);

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

    /* Get the result */
    VALUE3 = ADC_GetResult(ADC_SOC_0);
    printf("Voltage is (%d) : %0.3fV\n", VALUE3, RESULTTOVOLTAGE(VALUE3));

    /* Check the result */
    if ((ABS(VALUE1 - VALUE2) < 200) && (ABS(VALUE1 - VALUE3) < 200))
    {
        printf("GPIO8 is short\n");
    }
    else
    {
        printf("GPIO8 is not short\n");
    }

    while (1)
    {

    }
}
```

3 常见问题 QA

3.1 ADC 转换结果为负数

如果 ADC 正端电压高于负端，则 ADC 转换结果会有负的码值，此时负码值，并非表示 ADC 输入端的某个端口电压为负电压，应该称之为相对负电压或者差分负电压。如手册第一章所述，SPC1168 的 ADC 端口的输入的绝对电压不允许为负电压，否则 ADC 将无法正常工作。以 S/H A 为例：

$$\text{Voltage ADC} = \text{AIP} - \text{AIN}$$

其中，$0 < \text{AIP} < +3.657$，$0 < \text{AIN} < +3.657$

则在 ADC 端电压范围为：

$$-3.657 < \text{Voltage ADC} < +3.657$$

此时转换出的码值量程为[-8192, 8191]。