SPC1068 ADC 使用指南

概述

SPC1068 集成了高速差分 ADC，可抗拒同模噪声，最快转换时间（conversion time）为 125 ns，采取信道（Channel）与转换器（SOC）分离的设计，使用弹性高。

注意： 本文档主要以 SPC1068 为例进行介绍。

SPINTROL

目录

SPINTROL

©2024 旋智电子科技（上海）有限公司

图片列表

SPINTROL

表格列表

SPINTROL

©2024 旋智电子科技（上海）有限公司

版本历史

版本	日期	作者	状态	变更
C/0	2024-02-26	周佳莉	Released	首次发布。

SPINTROL

术语或缩写

术语或缩写	描述
ADC	Analog to Digital Converter，模数转换器

SPINTROL

©2024 旋智电子科技（上海）有限公司

1 ADC

1.1 ADC 单元概述

SPC1068 集成了高速差分 ADC，可抗拒同模噪声，最快转换时间（conversion time）为 125 ns，采取信道（Channel）与转换器（SOC）分离的设计，使用弹性高。

ADC 的基本特点如下：

- Truly differential ADC，可抗拒共模噪声
- 精度 12 位，内建 2 路采样保持单元
- 16 个转换器可设定独立采样时间与转换时间
- 转换时间最快可达 125ns
- 讯号采样时间可根据外部电路调整得最佳效果（预设 125ns，自行调整采样时间）
- 输入范围： 0 V to 3.658 V
  输入可为 0~VDDXA（3.3V）
- 最高 16 路 ADC 输入
- 可配置为同时采样模式
- 支持内部 PGA 输出直接采样

在本章节中，将介绍

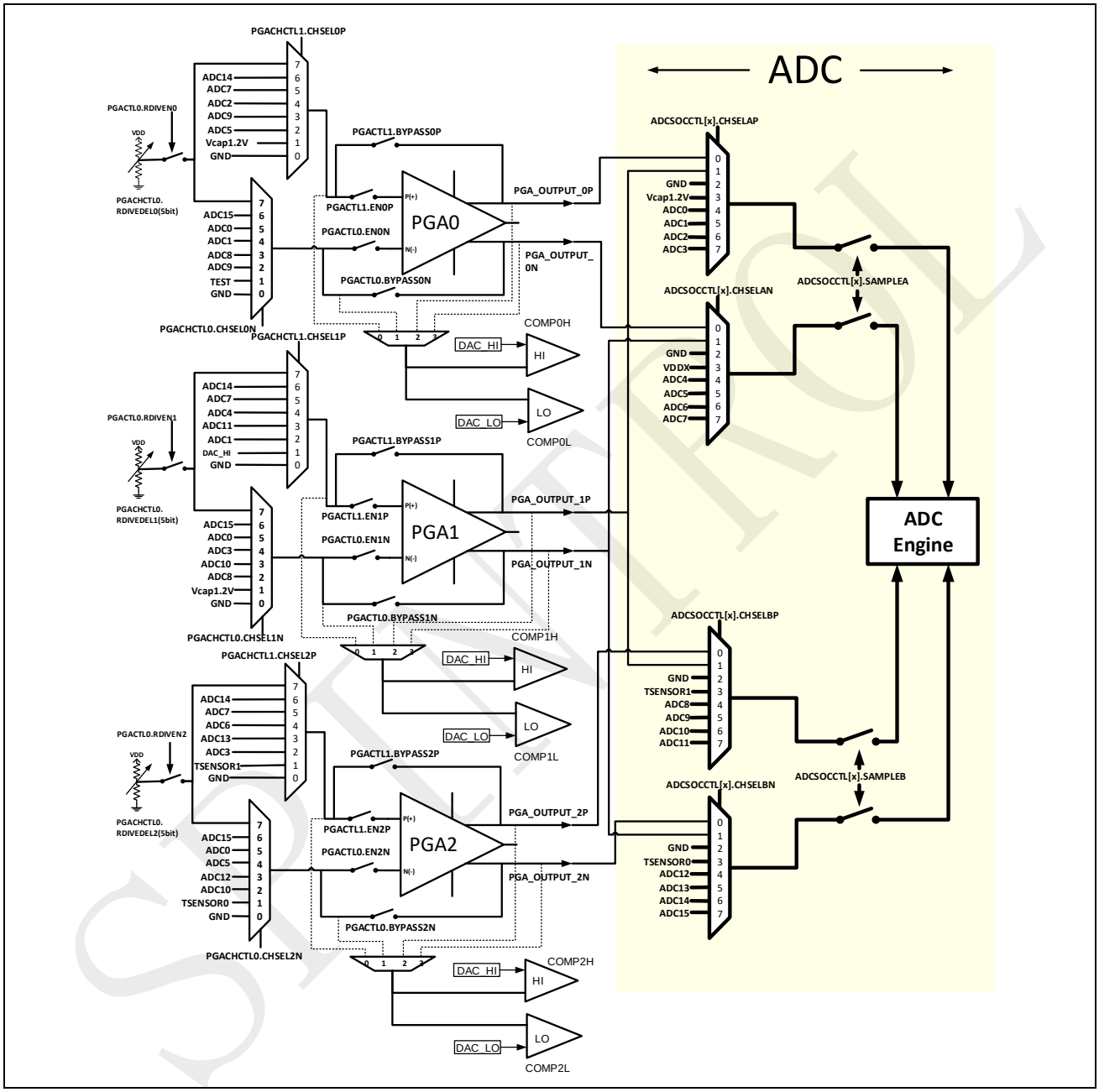
- SPC1068 模拟架构与 ADC 架构
- 差分型 ADC 转换数值范围
- 常见 Q&A
- 单端 ADC 初始化范例：寄存器等级配置
- 单端 ADC 使用范例：使用 ADC_EasyInit1()快速配置
- 双端（差分）ADC 使用范例：使用 ADC_EasyInit2()快速配置
- ADC 直接采样内部 PGA 放大讯号
- 同步采样模式范例与注意事项
- PWM 触发 ADC 采样范例
- 电机驱动建议管脚配置（3 shunt 与 1 shunt 电流采样配置建议）
- 常用 SDK API 参考表

SPINTROL

1.2 SPC1068 模拟架构与 ADC 架构

下图为 SPC1068 仿真电路构架，ADC 为下图套色部分。

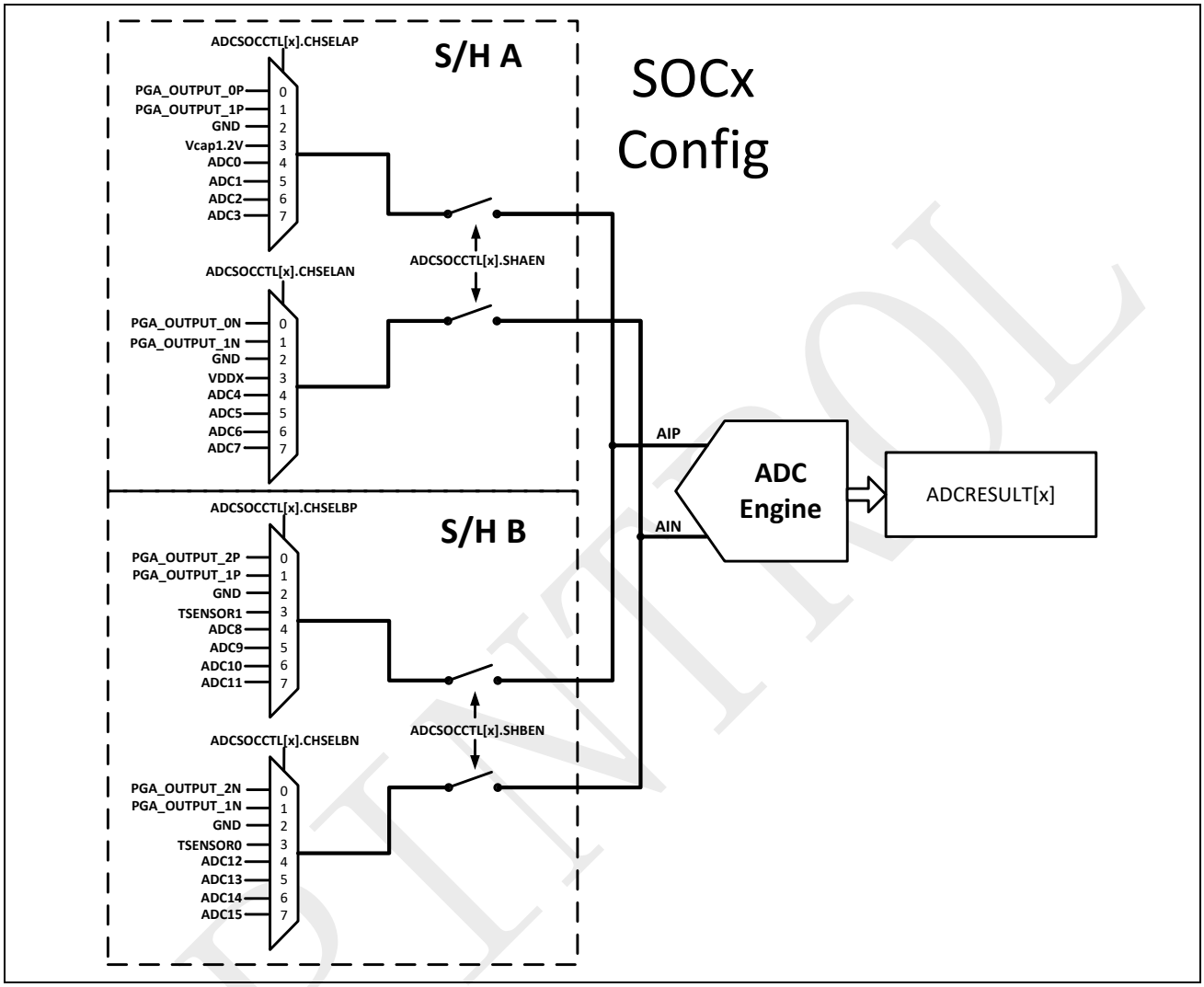
图 1-1：ADC 与模拟信号架构



©2024 旋智电子科技（上海）有限公司

下图是 ADC 的结构简图，从这个图里，可以清晰的看出，ADC 中包含 2 个采样保持单元，采样保持单元可以与外部引脚相连接，把外部引脚上的信号传送到内部的 ADC 转换器中；而内部的 ADC 转换器则负责把采样保持单元传送过来的信号转换成为数字信号。

图 1-2：ADC 结构简图



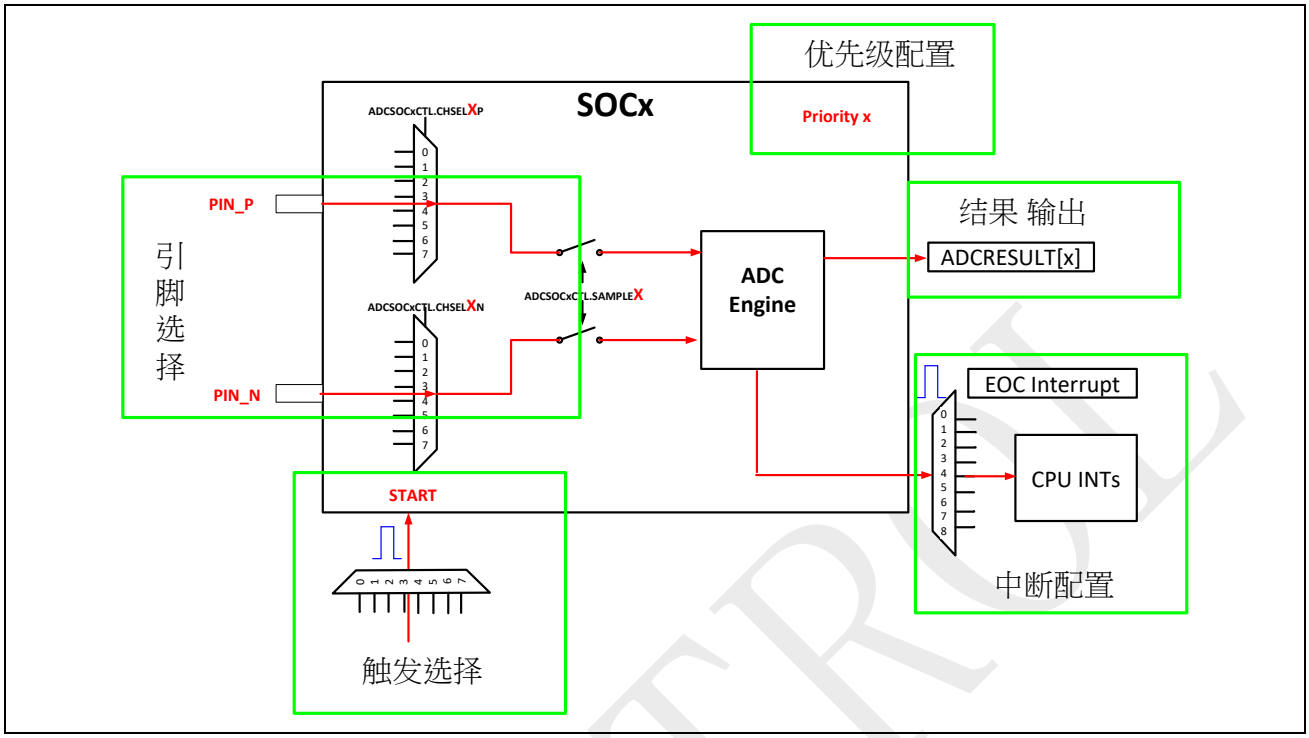
ADC 的每次转换可以通过一个 SOC 来配置，可以认为 SOC 是 ADC 配置的一个基础单元。SOC 做如下配置：

- 信道信号选择：选择哪个引脚/内部讯号的信号被转换
- 触发选择：确定 ADC 什么时间点被转换
- 中断信号致能：确定 SOC 转换完成后（EOC），是否产生中断
- 优先级配置：配置每个 SOC 的优先级（若无配置，则以 SOC 编号较小者优先级高）
- 采样时间与转换时间配置：每个 SOC 可以有不同的采样时间与转换时间设定

SPINTROL

下面的框图从总体上说明了 SOC 结构。

图 1-3：SOC 结构



SPC1068 的 ADC 是一个差分（differential）转换器，每一个 SOC 有两个差分 Sample and Hold，分别为 S/H A 与 S/H B，而每一个 Sampler 内又有正端（P）与负端讯号（N）的区别，注意 S/H A 与 S/H B 若同时 enable，则自动转为同步采样模式，详见同步采样范例，用户可根据上图挑选想要的通道进行转换，其中一端接上 GND 即是常见的单端输入 ADC。

ADC 作为 SP1068 内部较为复杂的外设，配置起来比较复杂。以下章节会通过几个例子把 ADC 的使用简化成几个模式，用户的使用方法如果较为简单，可以直接移植例子中使用的方法。

©2024 旋智电子科技（上海）有限公司

1.3 常用 API 列表

另外，在 SP1068 的驱动库中已经包含了抽象化的 API 函数，可以简化 ADC 的配置过程，常用 API 如下所列。

表 1-1：常用 ADC API 函数列表

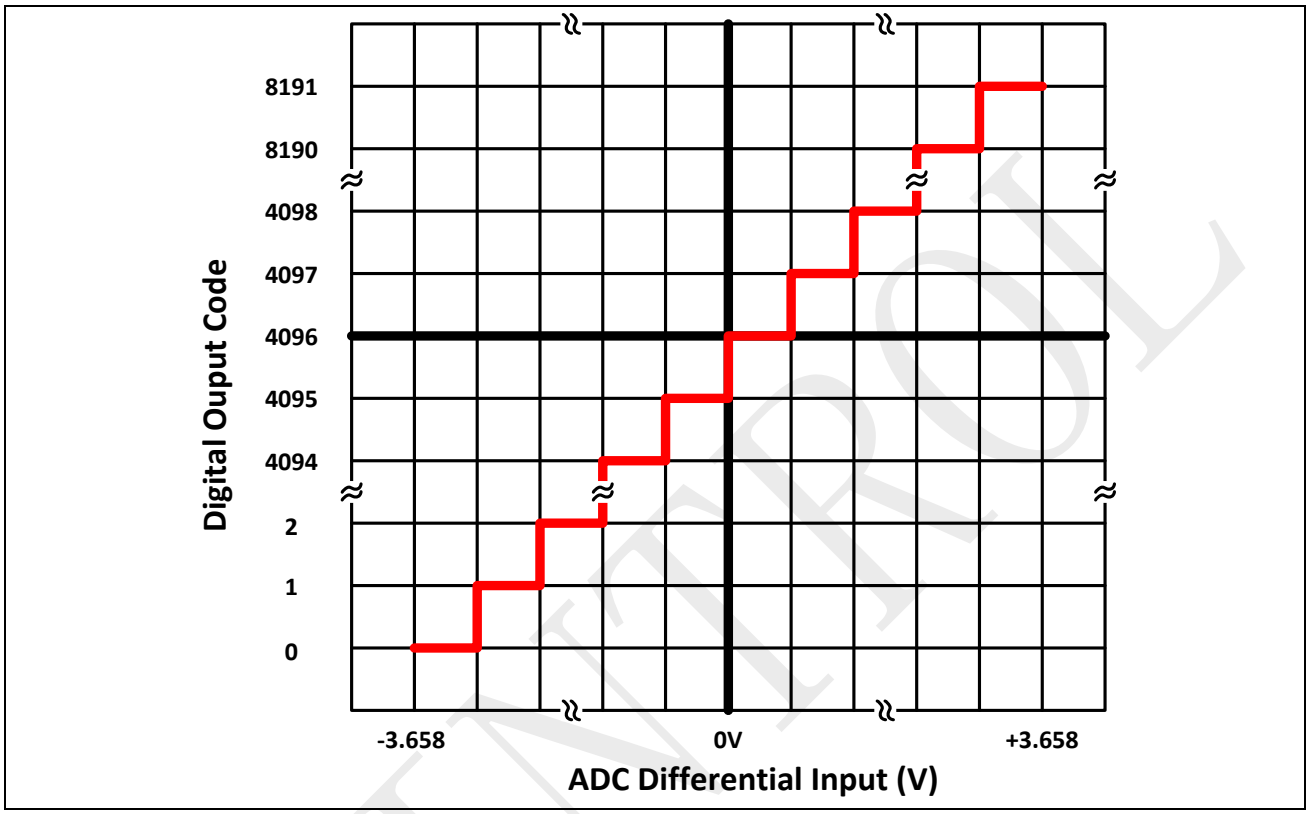
函数名称	描述
ADC_EasyInit1 （ SOC ，Channel，TrigSrc）	快速初始化单端 ADC 通道，默认 4MSPS 转换率，自动初始化管脚
ADC_GetTrimResult1（SOC）	取得单端信号转换数值（已较准） 数值范围 0～4095 （int16）
ADC_EasyInit2（SOC，CH1，CH2，TrigSrc）	快速初始化双端（差分）ADC 通道，默认 4M SPS 转换率，自动初始化管脚
ADC_GetTrimResult2（SOC）	取得单端信号转换数值（已较准） 数值范围-4095～4095（int16）
ADC_GetRawResult（SOC）	取得 ADC 原始信号转换数值（未较准） 数值范围 0~8191（uint16）
ADC_PowerUP（void）	启动 ADC 模块，加载原厂较准数值
ADC_EnableInt（SOC）	使能 SOC 相对之中断信号
ADC_ClearInt（SOC）	清除 SOC 相对之中断信号
ADC_GetIntFlag（SOC）	读取 SOC 相对之中断信号
ADC_SetSampleAndConvTime （ SOC ，u32SampleTimeNs ，u32ConvTimeNs）	设定 SOC 转换讯号信号时之采样时间（Sampling Time）与转换时间（Conversion Time），单位为纳秒 Ns
ADC_SelectPinSingleEnd（SOC，Channel，TrigSrc）	选择 SOC 与单端通道
ADC_SelectPinDifferetial（SOC，CH1，CH2，TrigSrc）	选择 SOC 与双端通道

SPINTROL

1.4 差分型 ADC 转换数值范围

差分型 ADC 转换后的数值编码范围与单端 ADC 略有不同，此乃容易造成混淆之处，本章节就常见的几个问题做出解答。SPC1068 的 ADC 输出是一个 13 bit（0~8191）的数值范围，对应的模拟范围是-3.658V~+3.658V，因此在 0~3.658V 内的精度则是 12 bit。

图 1-4：ADC 输入和数值输出关系



SPINTROL

©2024 旋智电子科技（上海）有限公司

1.5 常见问题 Q&A

以下就常见的问题进行解答。

Q1：ADC 为什么转换出来会有负电压？

A1：这边的负电压，是指 ADC 看到的电压差值为负电压，并不代表可以采到真正的负电压。SPC1068 的 ADC 拥有差动讯号输入的能力，在 ADC 看到的负电压为相对负电压，或称之为差分负电压，不是绝对的负电压，以 S/H A 为例：

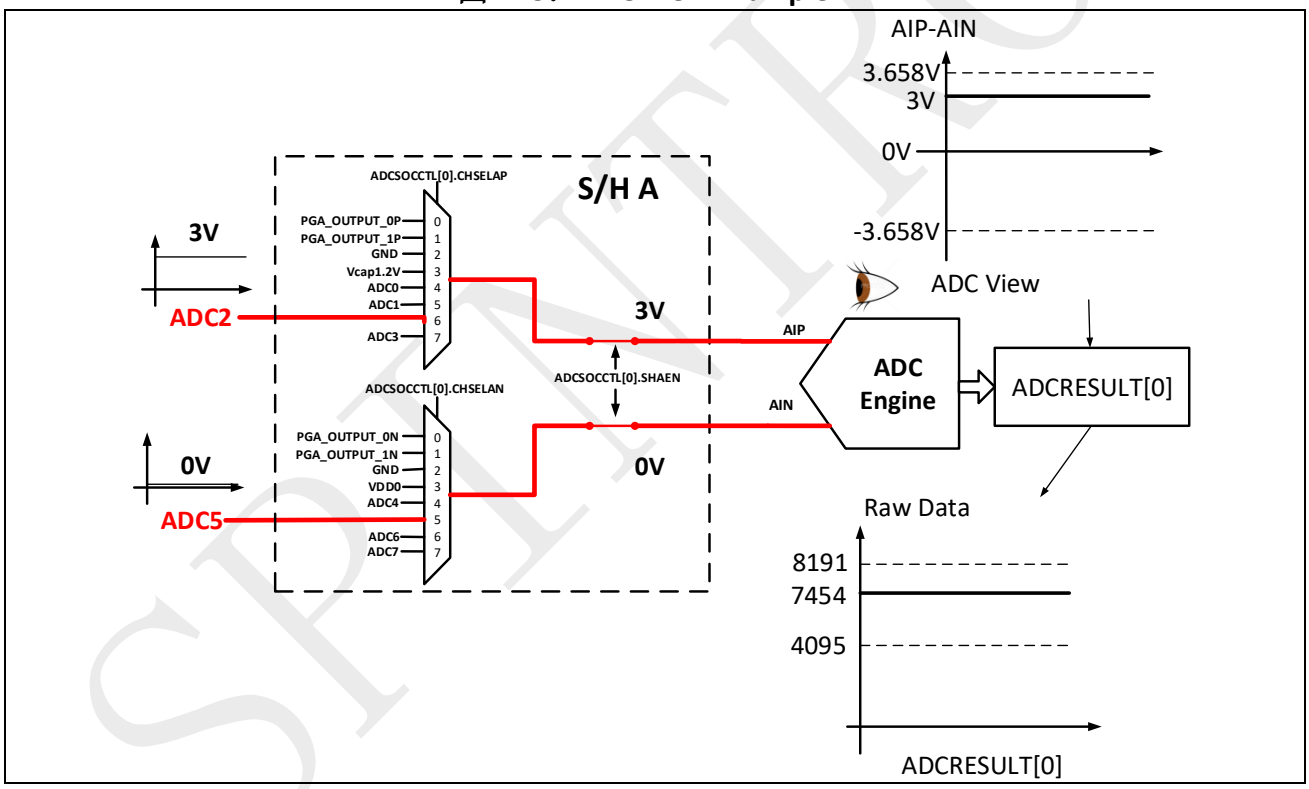
Voltage ADC = AIP - AIN

其中，0 < AIP < +3.65，0 < AIN < +3.65

在 ADC 端所视电压范围为

-3.65 < Voltage ADC < +3.65

图 1-5：ADC View Example 1



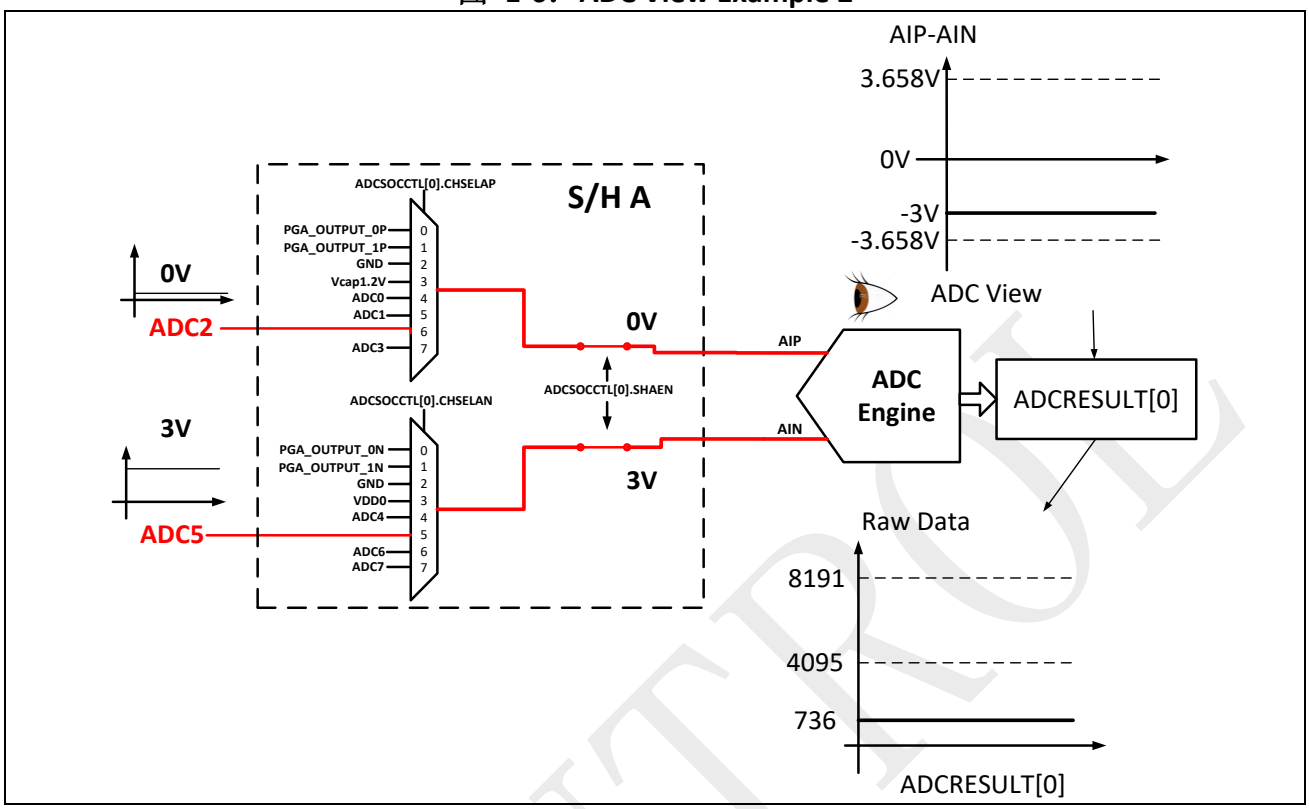
以**错误!未找到引用源。**为例，当 ADC2 为 3V，ADC5 为 0V 时，AIP 为 3V，AIN 为 0V，因此根据转换公式

$$ADCRESULT = 4095 + \frac{AIP - AIN}{3.658} \times 4096$$

以 ADC 视角来说，看到的电压是正的，转换结果为 7454。

SPINTROL

图 1-6：ADC View Example 2



上图是另一个例子，当 ADC2 为 0V，ADC5 为 3V，AIP 此时为 0V，AIN 为 3V，根据转换公式

$$ADCRESULT = 4095 + \frac{AIP - AIN}{3.658} \times 4096$$

此时以 ADC 的视角来说，看到的电压是负的，转换结果为 736。

注意： AIP 与 AIN 两者的电压都必须是正电压，亦即实际接到芯片脚位上的信号不可为负电压，SPC1068 无法读取低于 0 电压的实际模拟讯号，但以 ADC 来看，电压变化可以为-3.658~3.658 伏特，对应的数值变化为 0~8192。

Q2：如果实际的管脚给予一个负电压，ADC 是否能转换出来呢？

A2：这与 IC 设计工艺有关，如果这个负电压很小，一般是大于-0.7V，还是有可能转换出来的，但**不建议操作在负电压**。因为 IC 的管脚可视为一个二极管，一般的正电压不会使他导通，但是负电压小于此二极馆的导通电压，就会使二极管导通，造成大电流，电压也会箝在-0.7V 左右，有可能造成 IC 损坏。SPC1068 不保证输入负电压后 IC 可正常运行，建议设计人员尽量避免此操作区。

©2024 旋智电子科技（上海）有限公司

Q3：为什么设定为单端输入，输入的也是正电压，但是 ADC 采样到的结果却是负电压？

A3：这是因为单端输入其实本质上仍为双端输入，GND 设定在采保器（S/H A 或 S/H B）的正端输入（AP），信号则是由采保器的负端进来，因此在 ADC 观点，看到的是负电压，但实际上他是一个正电压的讯号，用户可自行在软件中修正极性。

Q4：为什么输入 3.3V，转换出来的编码却不是 8191？

A4：这是因为 ADC **参考电压为 3.658V，因此实际上满幅值对应的是 3.658V**

Q5：能否将满幅值（8191）时对应的电压改为 3.3V？

A5：可以的，需要在外部参考电压口输入大约 1.083V 左右的参考电压，详细计算公式请参见 SPC1068 Technical Reference Manual，同时也会需要一个 ADC 管脚，建议使用电压基准芯片达到最佳噪声比。

Q6：为什么有时候转换出来的编码会有误差？该如何处理？

A6：因为 IC 制造工艺的关系，每一片芯片多少都有些误差，SPC1068 在出厂前均为每一片芯片做过严格的校准程序，并将校准参数写入 Flash 的 OTP （一次性烧写区块） 中，此参数在 MCU 开机时会读入系统层级的 Global 变量中，供作校准使用，建议使用 Spintrol 提供的 API：ADC_GetTrimResult1（适合单端讯号使用）与 ADC_GetTrimResult2（适合双端/差分讯号使用），这两个函数会根据校准参数，计算出精确的转换结果。

注意：	在 Main 的开头务必执行 Sys_Init()函数，才会将校正参数导入，否则将采不到正确数值。如果使用 ADC_GetTrimResult2，则需要注意信道配置与数值极性问题。

SPINTROL

1.6 单端 ADC 初始化与采样范例：使用 ADC_EasyInit1()进行快速配置

差分 ADC 可配置成常见的单端 ADC 使用，建议采用 SDK 内提供之 EasyInit1()进行设定，可省去复杂的寄存器配置过程，仅须一条指令即可快速配置 ADC 信道与触发来源。

使用者只需输入 GPIO 脚位的编号（或是内部讯号之编号），并且指定此脚位给负责转换的 SOC，设定好触发来源后，即可马上使用。

1.6.1 ADC_EasyInit1 使用介绍

透过 ADC_EasyInit1()与 ADC_GetTrimResult1()的搭配，使用者可以简单地延续过去使用单端 ADC 的经验，快速配置 ADC 与读取信道信号。

ADC_EasyInit1（eSOC，u8PinSel，TrigSrc）函数一共有三个输入必须选择，以下就其选择与执行内容作介绍。

表 1-2：ADC_EasyInit1()函数介绍

<table>
<tr><th colspan="3">ADC_EasyInit1（eSOC，u8PinSel，TrigSrc）增加 Sampling time configure</th></tr>
<tr><th>Parameter</th><th colspan="2">Description</th></tr>
<tr><td>eSOC</td><td colspan="2">It can be
ADC_SOC0
ADC_SOC1
…
ADC_SOC15</td></tr>
<tr><td rowspan="2">u8PinSel</td><td colspan="2">可以输入以下两种定义</td></tr>
<tr><td>GPIO 编号</td><td>定义讯号源（ADC_PinSelEnum in adc.h）</td></tr>
<tr><td></td><td>GPIO_1 （ADC0）
GPIO_2 （ADC1）
GPIO_3 （ADC2）
GPIO_4 （ADC3）
GPIO_5 （ADC4）
GPIO_6 （ADC5）
GPIO_7 （ADC6）
GPIO_8 （ADC7）
GPIO_9 （ADC8）
GPIO_10 （ADC9）
GPIO_11 （ADC10）
GPIO_12 （ADC11）
GPIO_13 （ADC12）
GPIO_14 （ADC13）
GPIO_15 （ADC14）
GPIO_16 （ADC15）</td><td>ADC0_GPIO1
ADC1_GPIO2
ADC2_GPIO3
ADC3_GPIO4
ADC4_GPIO5
ADC5_GPIO6
ADC6_GPIO7
ADC7_GPIO8
ADC8_GPIO9
ADC9_GPIO10
ADC10_GPIO11
ADC11_GPIO12
ADC12_GPIO13
ADC13_GPIO14
ADC14_GPIO15
ADC15_GPIO16
---------以下为内部讯号源--------
ADCx_PGA0P
ADCx_PGA0N
ADCx_PGA1P
ADCx_PGA1N
ADCx_PGA2P
ADCx_PGA2N</td></tr>
</table>

©2024 旋智电子科技（上海）有限公司

		ADCx_TSENSOR_L ADCx_TSENSOR_H ADCx_VDD12 内部 1.2V ADCx_VDDA 内部 3.3V
TrigSrc	可选择以下作为输入（ADC_TriggerSourceEnum in adc.h） ADCTRIG_Software 软件触发 ADCTRIG_Timer0 Timer0 Overflow 触发 ADCTRIG_Timer1 ADCTRIG_Timer2 ADCTRIG_XINT2 外部 IO 触发 ADCTRIG_PWM0A PWM0 SOCA 事件触发 ADCTRIG_PWM0B PWM0 SOCB 事件触发 ADCTRIG_PWM1A ADCTRIG_PWM1B ADCTRIG_PWM2A ADCTRIG_PWM2B ADCTRIG_PWM3A PWMx SOCA 事件触发 ADCTRIG_PWM3B PWMx SOCB 事件触发 ADCTRIG_PWM4A ADCTRIG_PWM4B ADCTRIG_PWM5A ADCTRIG_PWM5B ADCTRIG_PWM6A ADCTRIG_PWM6B	

ADC_EasyInit 1()有以下几个特点：

- 自动使能 ADC 模块
- 若讯号源来自外部，可输入引脚 GPIO 编号，自动初始化为 AD 引脚
- 若讯号源来自内部，可使用 ADC 自定义讯号源表来配置
- 自动配置 S/H A 或是 S/H B
- 自动选择 GND 端入口
- 自动根据 ADC 模块频率配置转换速率为 4M SPS
- 其中采样（Sample）时间为 125ns，转换时间（Conversion）为 125ns。
- 配置 interrupt 讯号（但不使能 M3 中断服务程序）

注意：

- ADC_EasyInit1 仅适用于单端讯号
- 请搭配 ADC_GetTrimResult1 使用
- **默认 Sample 时间为 125ns，假若外部电路对 ADC pin 脚充电速度较慢，请自行调整 Sample 时间**。
- 限定 PGA1 的输入使用 SH/B，若需弹性配置，请自行操作寄存器。
- 当 GPIO 编号超过 16 时，比如 GPIO17，此时 GPIO17 非模拟口，此设置并无作用。
- 如果您要选择其他内部讯号，比如内部 PGA 输出，或是内部 1.2V 作为讯号源

SPINTROL

或是内部温度传感器，请使用“定义讯号源”
- 转换完毕后，建议搭配 ADC_GetTrimResult1（SOCx）函数，可将数值经过校准并且调整至 0~4095 的范围
- 虽然始能 ADC 中断讯号，但只要不开启 M3 系统层级的中断致能，即使产生中断讯号，也不会进入中断服务程序，建议用户打开此中断讯号，可作为 ADC 转换完成的标志位。

1.6.2 使用范例

下例为使用 SOC0 采取 ADC2 的范例码，在此例中，触发来源为软件触发。最后则是使用软件触发并且读取数值的范例。

Example 1.6-1

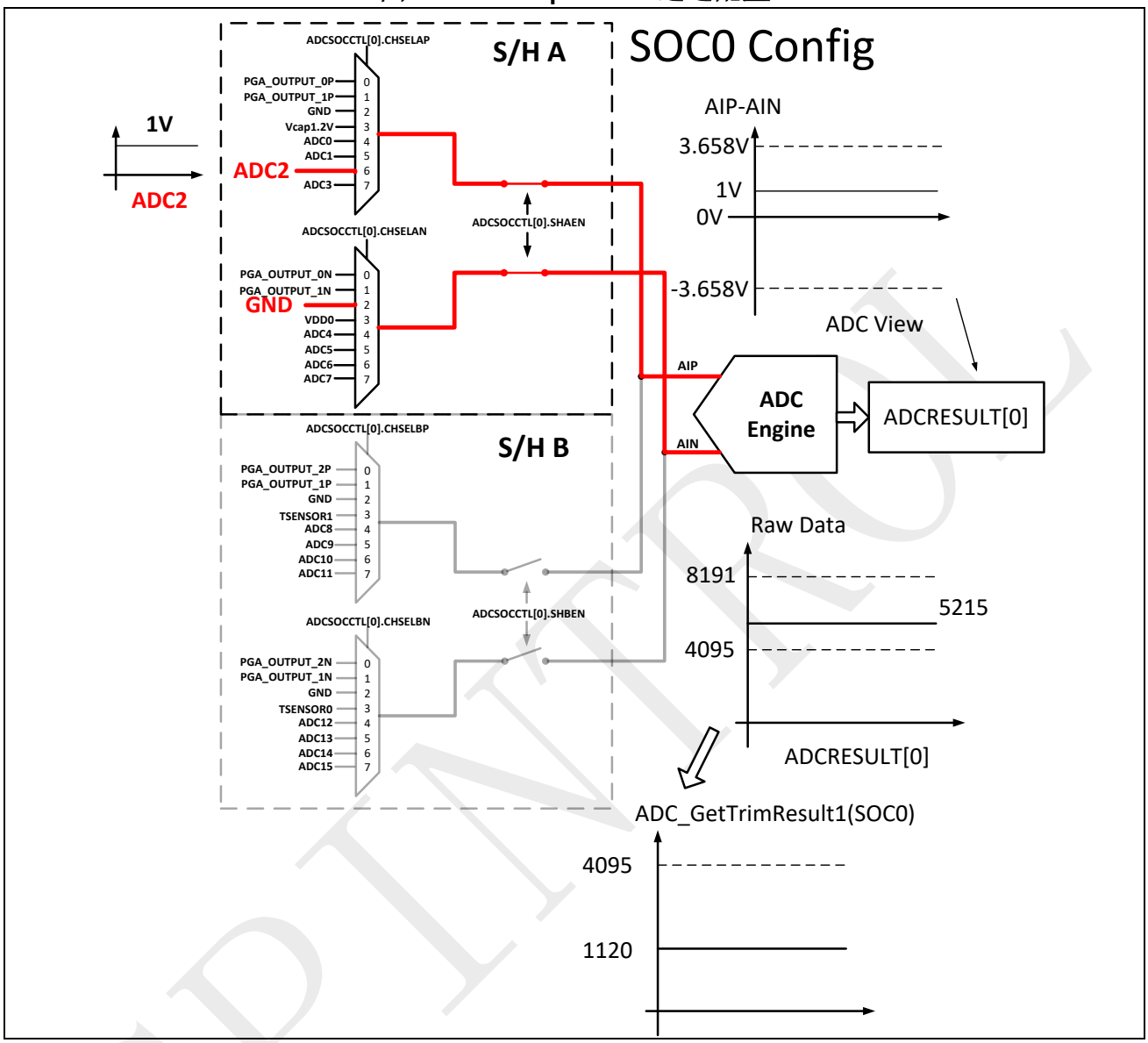
```
void main()
{
  int16_t i16VSP;
  ADC_EasyInit1(ADC_SOC_0,      GPIO_3,   ADCTRIG_Software);
  /*         The above is the same as :   */
  ADC_EasyInit1(ADC_SOC_0,   ADC2_GPIO3,   ADCTRIG_Software);

  /* Optional : Adjust by circuit connected with ADC pin */
  ADC_SetSampleAndConvTime(ADC_SOC_0,150 /*ns*/ ,150);


  /* Software trigger and get trim result */
  ADC_SoftwareTrigger(ADC_SOC_0);

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

  /* Get result (already trim and rescale from 0~4095) */
  i16VSP = ADC_GetTrimResult1(ADC_SOC_0);

  /* Clear interrupt flag */
  ADC_ClearInt(ADC_SOC_0);
}
```

上例透过 ADC_EasyInit1 初始化后，即可立刻使用 ADC 功能，默认的采样时间（Sample time）与转换时间（Conversion Time）均为 150ns，因此转换一笔时间需要 300ns，假若 ADC 使用的电路需要较长的充电时间，可利用 ADC_SetSampleAndConvTime 重新设定该 SOC 的采样时间。

©2024 旋智电子科技（上海）有限公司

上述代码的通道设定如下图所示：

图 1-7：Example 1.6-1 通道配置



当输入是 1V 时，ADC 转换端看到的电压为 AIP=1V，AIN=0V，根据以下公式，可得转换结果为 5215。

$$ADCRESULT = 4095 + \frac{AIP - AIN}{3.658} \times 4096$$

当使用 ADC_GetTrimResult1()后，所得到的数值将自动转换为 0~4095 的间的数值，方便使用者延续过去 12 bit ADC 的使用经验。

SPINTROL

1.6.3 单端讯号数据采集 API 介绍

SPC1068 的 driver 中，对于单端 ADC 配置，提供了两种的采值方式，推荐采用第一种方式，细节如下。

1. ADC_GetTrimResult1（SOC x）--去除极性且校准（建议）

```
/*****************************************************************************//**
 * @brief  For Single End Mode Only  (One terminal is GND)
 *         Get Trim result(code) from ADC result register
 *         The trim coefficient locates at flash and load in SysInfo when start up
 *
 * @param[in]  u8SOC:  Select SOC to be used
 *
 * @return  ADC result in 0~4095
 *            Code            Real Voltage in ADC view
 *            0               0 V
 *            4095            3.65 V
 *****************************************************************************/
#define ADC_GetTrimResult1(u8SOC)           (abs(((((int16_t)ADC->ADCRESULT[(u8SOC)].bit.VAL+\
                                              SysInfo.i16SHAOffset)*(int32_t)(SysInfo.i16SHAGain)
```

若为单端输入讯号，建议均采用 ADC_TrimResult1 的方式进行采值，使用者可以视为一般常见的单端 ADC 采值，并可采到精准的电压讯号，此函数已经将采到的数值进行校准，并且重新计算为对 GND 的电压，范围是 0～4095。

2. 只采 Raw Data

```
/*****************************************************************************//**
 * @brief  Get raw result(code) from ADC result register
 *
 * @param[in]  u8SOC:  Select SOC to be used
 *
 * @return  ADC result in 0~8191
 *            Code            Relative voltage in ADC view (ADC_AP-ADC_AN or ADC_BP-ADC_BN)
 *            0               -3.65 V
 *            4095            0 V
 *            8191            3.65 V
 * @detail  SPC1068 is a truly differential ADC, the code is the transformation of voltage
 *          Sampler Positive - Sampler Negative
 *          The negative voltage in ADC is "relative" but not "absolute" negative voltage.
 *          It can not be real absolute negative voltage in pin pad!!
 *****************************************************************************/
#define ADC_GetRawResult(u8SOC)    (ADC->ADCRESULT[(u8SOC)].bit.VAL)
```

第二种为直接读取 ADC 转换结果，未经过校准，8191 对应的是 3.65V，0 对应的是-3.65V，4095 则是对应到 0V，用户必须根据通道的极性自行计算数值范围。

©2024 旋智电子科技（上海）有限公司

1.6.4 单端 ADC 初始化与采样范例 2：当讯号由负端输入

差分 ADC 可配置成常见的单端 ADC 使用，但须注意信道配置极性与转换结果的关系。为了与上一章节做比较，本章节采取信号由 S/H 的负端输入的讯号为您解释。

下例为使用 SOC0 采取 ADC14 的范例码，在此例中，触发来源为软件触发。最后则是使用软件触发并且读取数值的范例。依照 ADC 通道硬件架构，ADC14 必须由 S/H B 的负端进入 ADC。

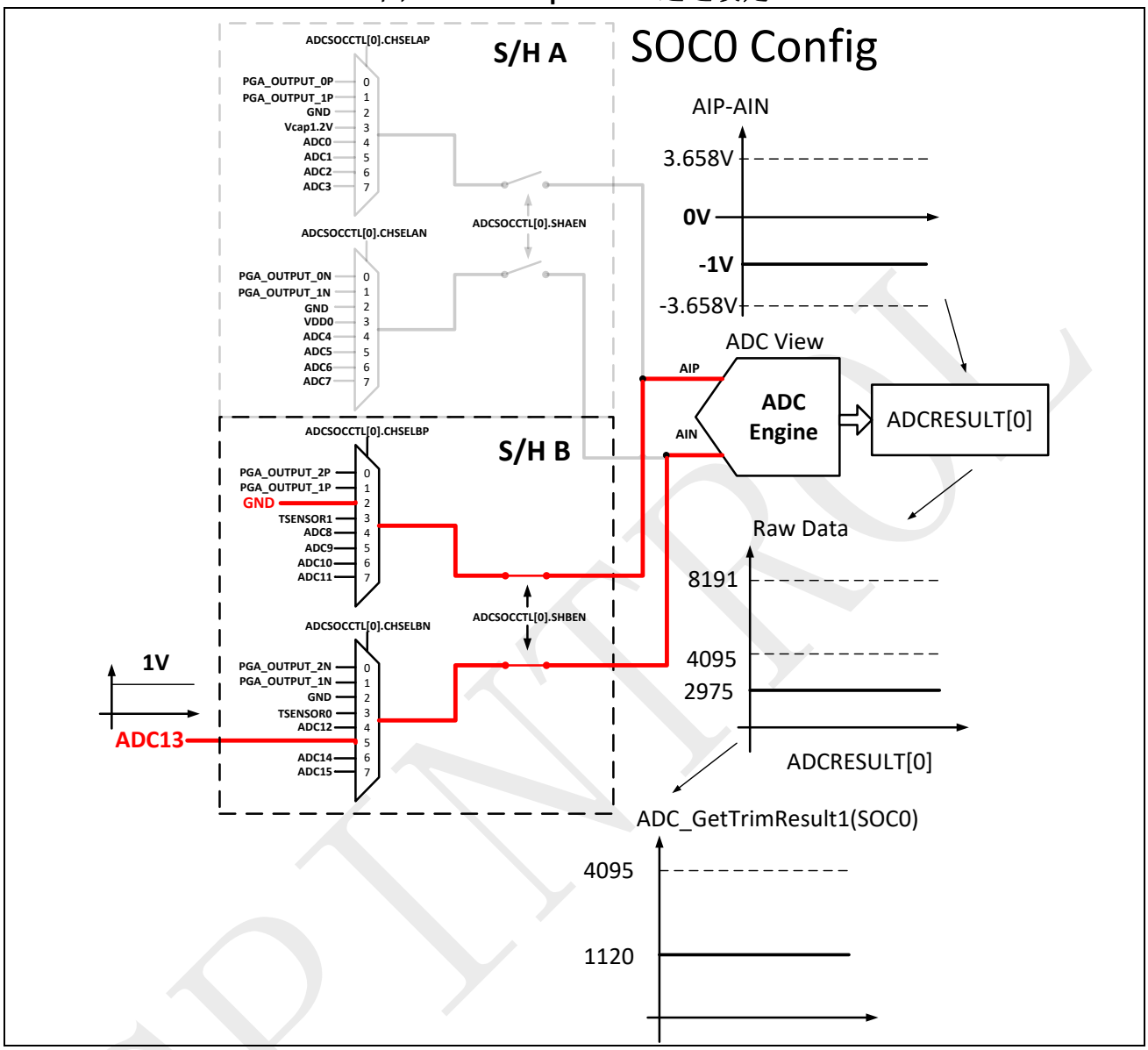
Example 1.6-2

```
void main()
{
  int16_t i16VSP;
  ADC_EasyInit1(ADC_SOC_0,      GPIO_14,   ADCTRIG_Software);
  /*         The above is the same as :   */
  ADC_EasyInit1(ADC_SOC_0,   ADC13_GPIO14,   ADCTRIG_Software);

  /* Optional : Adjust by circuit connected with ADC pin */
  ADC_SetSampleTime(ADC_SOC_0,150 /*ns*/ ,150);

  /* Software trigger and get trim result */
  ADC_SoftwareTrigger(ADC_SOC_0);

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

  /* Get result (already trim and rescale from 0~4095) */
  i16VSP = ADC_GetTrimResult1(ADC_SOC_0);

  /* Clear interrupt flag */
  ADC_ClearInt(ADC_SOC_0);
}
```

SPIN

以上代码通道配置如下图所示：

图 1-8：Example 1.6-2 通道设定



因为输入讯号是由 S/H B 的负端进入，当输入是 1V 时，ADC 转换端看到的电压为 AIP=0V，AIN=1V，所以就 ADC 的角度来说，看到的是-1V，根据以下公式，可得转换结果为 2975。

$$ADCRESULT = 4095 + \frac{AIP - AIN}{3.658} \times 4096$$

ADC_GetTrimResult1()函数会自动将码转回 0~4095 的区间，方便使用者使用。

透过 ADC_EasyInit1()与 ADC_GetTrimResult1()的搭配，使用者可以简单地延续过去使用单端 ADC 的经验，快速配置 ADC 与读取信道信号。

©2024 旋智电子科技（上海）有限公司

1.7 Appendix：单端 ADC 初始化与采样范例—直接寄存器配置

以下介绍最基本的 ADC 初始化寄存器操作方式。SPC1068 的 ADC 信道设计弹性，但是相对复杂，建议使用者参考上一章节，直接使用 SDK 内的 ADC 初始化 API 进行初始化，可大幅减少开发时间。

若是用户只需简单初始化几个 ADC 管脚，不需要用到复杂功能，可以使用寄存器等级的配置，**在 Code size 的降低上仍有效益**。

一个正常的单端 ADC 初始化步骤如下：

Step 1：使能 ADC 模块

Step 2：将 GPIO 脚位选择为模拟功能（例：GPIO3-ADC2）

Step 3：设定 SOCx 转换通道为该 GPIO 管脚（例： SOC0 采集 ADC2）

Step 4：选择 SOCx 触发源（例：PWM1A 的 SOC 事件触发 SOC0 转换）

Step 5：设定转换时间

Step 6：Enable 中断讯号（但不使能中断服务程序）

以上六个步骤可见 Example 1.7-1，以初始化 ADC2（GPIO3）管脚为例，建议搭配图表进行 ADC 信道配置，用户也可以善用 Goto definition 功能查询其他配置。

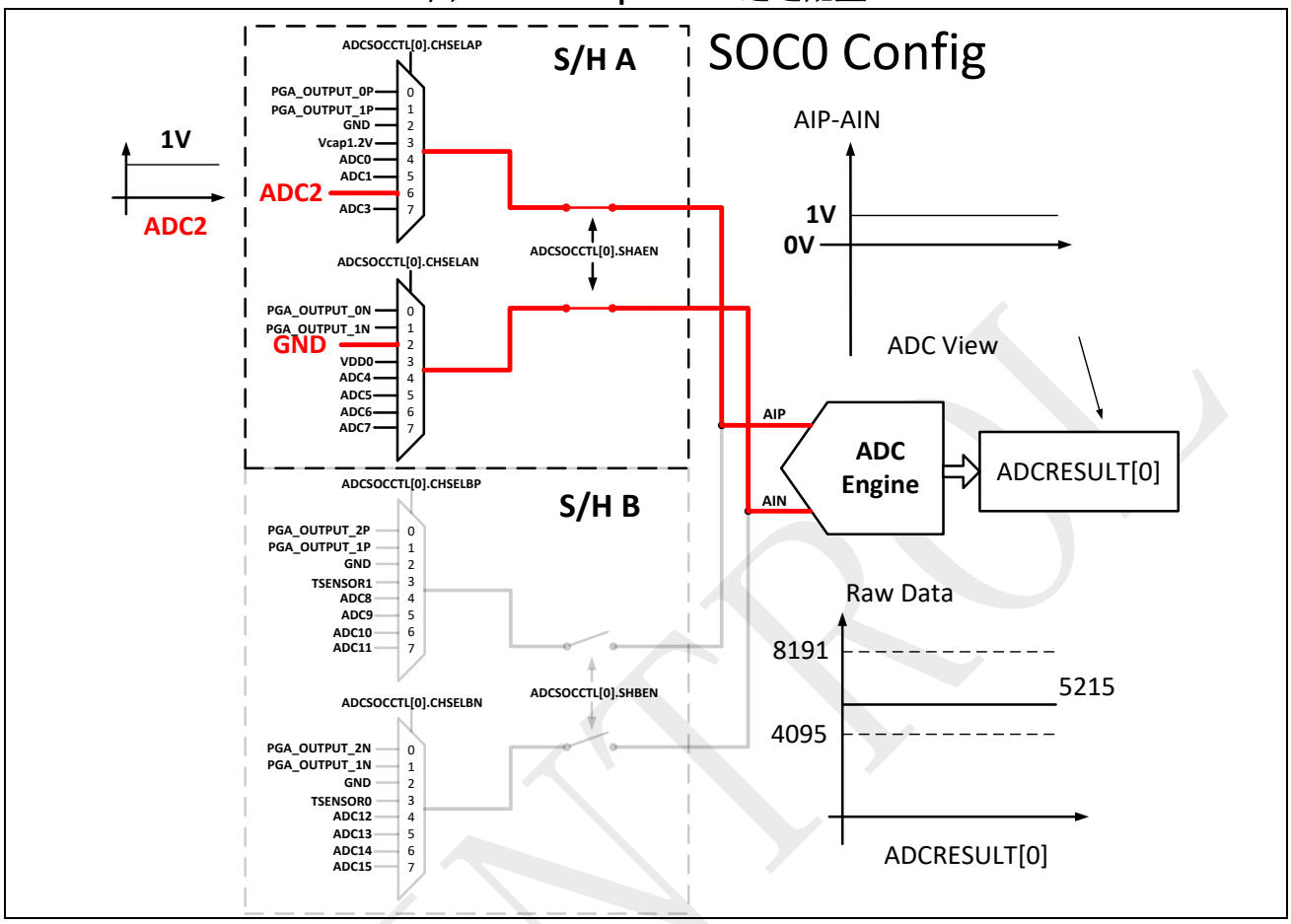
其中建议将中断讯号打开，此讯号可作为 ADC 转换结束的标志位，若有需要进入中断，使用者可根据需求决定是否使能 M3 的中断服务程序，请见最后的 Optional 代码。

Example 1.7-1

```
void MotorADC_RegisterInitExample(void)
{
  /* Step1 : Enable ADC module */
  ADC_PowerUP();

  /* Step2 : Pin Mux configuration */
  GPIO_SetPinAsAnalog(GPIO_3); /* Select GPIO3(ADC2) as analog */

  /* Step3~Step5 : ADC channel configuration */
  ADC->ADCSOCCTL[ADC_SOC_0].all  /* SOC0 is selected as Sampler */
  = ADCSOC0CTL_ALL_CHSELAP_ADC2  /* S/H A Positive = ADC2 */
    | ADCSOC0CTL_ALL_CHSELAN_GND  /* S/H A Negative = GND */
    | ADCSOC0CTL_ALL_SHAEN_ENABLE  /* Enable S/H A */
    | ADCSOCxCTL_ALL_TRIGSEL_PWM1A /* PWM1A SOC event will trigger
SOC0 */
    | ADCSOC0CTL_ALL_SAMPCNT_(5)    /* Sampling time = (5+2)*20ns =
140ns if ADC = 50MHz */
    | ADCSOC0CTL_ALL_CONVCNT_(7);  /* Conversion time = 140ns*/

  /* Step 6 : Enable interrupt signal but do NOT enable ADC interrupt
service in M3 */
   ADC_EnableInt(ADC_SOC_0);

  /* Optional: Enable ADC interrupt service routine in Cortex-M3 */
  /* NVIC_EnableIRQ(ADC0_IRQn);*/
}
```

以上的配置，反映在信道上的配置如下图所示：

图 1-9：Example 1.7-1 通道配置



当输入是 1V 时，根据以下公式，可得转换结果为 5215。

$$ADCRESULT = 4095 + \frac{AIP - AIN}{3.658} \times 4096$$

SPC1068 的 SDK 中提供了方便使用的 API，协助您轻松获取单端信号于 0~4095 尺度范围内的数值，方便与一般 MCU 进行接轨，请见下一章节的范例。

注意： 当 ADC 的 clock=50MHz 时，有关讯号采样时间与转换时间建议如下：

- Sampling Time 在上例中为 140 ns，此时间为讯号建立在 S/H 电路所需的时间，与 ADC 脚位接触的外部电路有关，常见的使用范围在 125ns~800ns。
- Conversion Time 为 140ns，在 SPC1068 的设计中，此数值一定得大于 125ns。转换一笔讯号的时间为上述两者相加，因此在这个范例中，SOC0 转换一笔讯号所需时间为 280ns。
- 一般使用上建议可将两者均设定为 125ns，转换一笔数据所需时间为 250ns。

©2024 旋智电子科技（上海）有限公司

1.8 双端（差分）ADC 初始化与采样范例：使用 ADC_EasyInit2()进行快速配置

SPC1068 的差分 ADC 脚位配置有其限制，请参考整体架构图。为了方便使用，Spintrol 亦提供了简单的初始化配置 ADC_EasyInit2 配置双端讯号。

使用者只需输入两组 GPIO 脚位的编号（或是内部讯号的编号），并且指定此脚位给负责转换的 SOC，设定好触发来源后，即可马上使用。

1.8.1 ADC_EasyInit2 使用介绍

ADC_EasyInit2()的使用与 ADC_EasyInit1()几乎一致，差别只在于 ADC_EasyInit2()必须配置两个讯号来源 u8PinSel_1，u8PinSel_2。

ADC_EasyInit2（eSOC，u8PinSel_1，u8PinSel_2，TrigSrc）

其中 eSOC，TrigSrc，u8PinSel_1，u8PinSel_2 的选项均与 ADC_EasyInit1()相同，容易让人混淆的处在于，ADC_EasyInit2()中，1 与 2 的讯号来源是不分正负的，初始化代码会自动判断 1 与 2 在实际硬件上的位置，并且使能对应的寄存器，真正的讯号正负极性，请参考上一章节的通道选择图，用户可根据通道极性自行在软件中修正。

表 1-3：ADC_EasyInit2()函数介绍

ADC_EasyInit2（eSOC，u8PinSel_1，TrigSrc）		
Parameter	**Description**	
eSOC	It can be ADC_SOC0 ADC_SOC1 ADC_SOC15	
u8PinSel_1	It can be	
	GPIO 编号	定义讯号源（ADC_PinSelEnum in adc.h）
	GPIO_1 （ADC0） GPIO_2 （ADC1） GPIO_3 （ADC2） GPIO_4 （ADC3） GPIO_5 （ADC4） GPIO_6 （ADC5） GPIO_7 （ADC6） GPIO_8 （ADC7） GPIO_9 （ADC8） GPIO_10 （ADC9） GPIO_11 （ADC10） GPIO_12 （ADC11） GPIO_13 （ADC12） GPIO_14 （ADC13） GPIO_15 （ADC14） GPIO_16 （ADC15）	ADC0_GPIO1 ADC1_GPIO2 ADC2_GPIO3 ADC3_GPIO4 ADC4_GPIO5 ADC5_GPIO6 ADC6_GPIO7 ADC7_GPIO8 ADC8_GPIO9 ADC9_GPIO10 ADC10_GPIO11 ADC11_GPIO12 ADC12_GPIO13 ADC13_GPIO14 ADC14_GPIO15 ADC15_GPIO16 ---------以下为内部讯号源-------- ADCx_PGA0P ADCx_PGA0N

		ADCx_PGA1P ADCx_PGA1N ADCx_PGA2P ADCx_PGA2N ADCx_TSENSOR_L ADCx_TSENSOR_H ADCx_VDD12 内部 1.2V ADCx_VDDA 内部 3.3V
u8PinSel_2	同上	
TrigSrc	可选择以下作为输入（ADC_TriggerSourceEnum in adc.h） ADCTRIG_Software 软件触发 ADCTRIG_Timer0 Timer0 Overflow 触发 ADCTRIG_Timer1 ADCTRIG_Timer2 ADCTRIG_XINT2 外部 IO 触发 ADCTRIG_PWM0A PWM0 SOCA 事件触发 ADCTRIG_PWM0B PWM0 SOCB 事件触发 ADCTRIG_PWM1A ADCTRIG_PWM1B ADCTRIG_PWM2A ADCTRIG_PWM2B ADCTRIG_PWM3A PWMx SOCA 事件触发 ADCTRIG_PWM3B PWMx SOCB 事件触发 ADCTRIG_PWM4A ADCTRIG_PWM4B ADCTRIG_PWM5A ADCTRIG_PWM5B ADCTRIG_PWM6A ADCTRIG_PWM6B	

ADC_EasyInit2()有以下几个特点：

- 自动致能 ADC 模块
- 必须输入两个讯号来源编号，若输入错误，回传 0，调适阶段将打印出错误讯息
- 输入两个讯号来源可对调传入参数的位置，

  ADC_EasyInit2（ADC_SOC_0，GPIO_6，GPIO_3，ADCTRIG_Software）；

  ADC_EasyInit2（ADC_SOC_0，GPIO_3，GPIO_6，ADCTRIG_Software）；

  两者的结果都是一样的。输入时没有正负端的区分，原因为正负端的讯号已经被硬件架构决定。
- 两个输入讯号的正负极性，请参考 ADC 架构图
- 若讯号源来自外部，可输入引脚 GPIO 编号，自动初始化为 AD 引脚
- 若讯号源来自内部，可使用 ADC 定义讯号源表来配置
- 自动配置 S/H A 或者 S/H B

©2024 旋智电子科技（上海）有限公司

- 自动根据 ADC 模块频率配置转换速率为 4M SPS
- 用户可根据外部电路配置适当的 Sampling time
- 配置 interrupt 讯号（但不使能 M3 中断服务程序）

注意：	- 建议搭配 ADC_GetTrimResult2()进行采值，输出范围自动调整为-4095~4095 - 请注意通道选择与极性，适当在软件中调整数值极性

使用上建议采用如下步骤：

Step 1：

在调适阶段，为了让用户方便了解配置是否正确，请在 spc1068.h 中将以下辅助配置的功能打开。此功能建议确定初始化没问题的后可关起，节省 Code size。

```
/**
 *  @brief  Conditional branch selection for hardware intialization debug
 *
 *  @note   If SPC1068_DRIVER_DEBUG 0
 *          There is no parameter check in initial function and suitable for code size reduction
 *          If SPC1068_DRIVER_DEBUG 1
 *          Parameter check in initial function but code size increases.
 *          Spintrol suggests that user open this function when develop but turn off it after then.
 */
#define SPC1068_DRIVER_DEBUG 1
```

如下图，若设定错误，会打印出错误信息，代码停止在无穷循环中。

```
#if SPC1068_DRIVER_DEBUG
if((ABGroup_1 != ABGroup_2) || (PNChannel_1 == PNChannel_2))
{
  printf("Error in ADC_SelectPinDifferetial\n");
  while(1){};
}

printf("CHA Enable = %d\n",ADC->ADCSOCCTL[u8SOC].bit.SHAEN);
printf("CHB Enable = %d\n",ADC->ADCSOCCTL[u8SOC].bit.SHBEN);
printf("CH SEL AN  = %d\n",ADC->ADCSOCCTL[u8SOC].bit.CHSELAN);
printf("CH SEL AP  = %d\n",ADC->ADCSOCCTL[u8SOC].bit.CHSELAP);
printf("CH SEL BN  = %d\n",ADC->ADCSOCCTL[u8SOC].bit.CHSELBN);
printf("CH SEL BP  = %d\n",ADC->ADCSOCCTL[u8SOC].bit.CHSELBP);


#endif
```

Step 2： 呼叫 ADC_EasyInit2

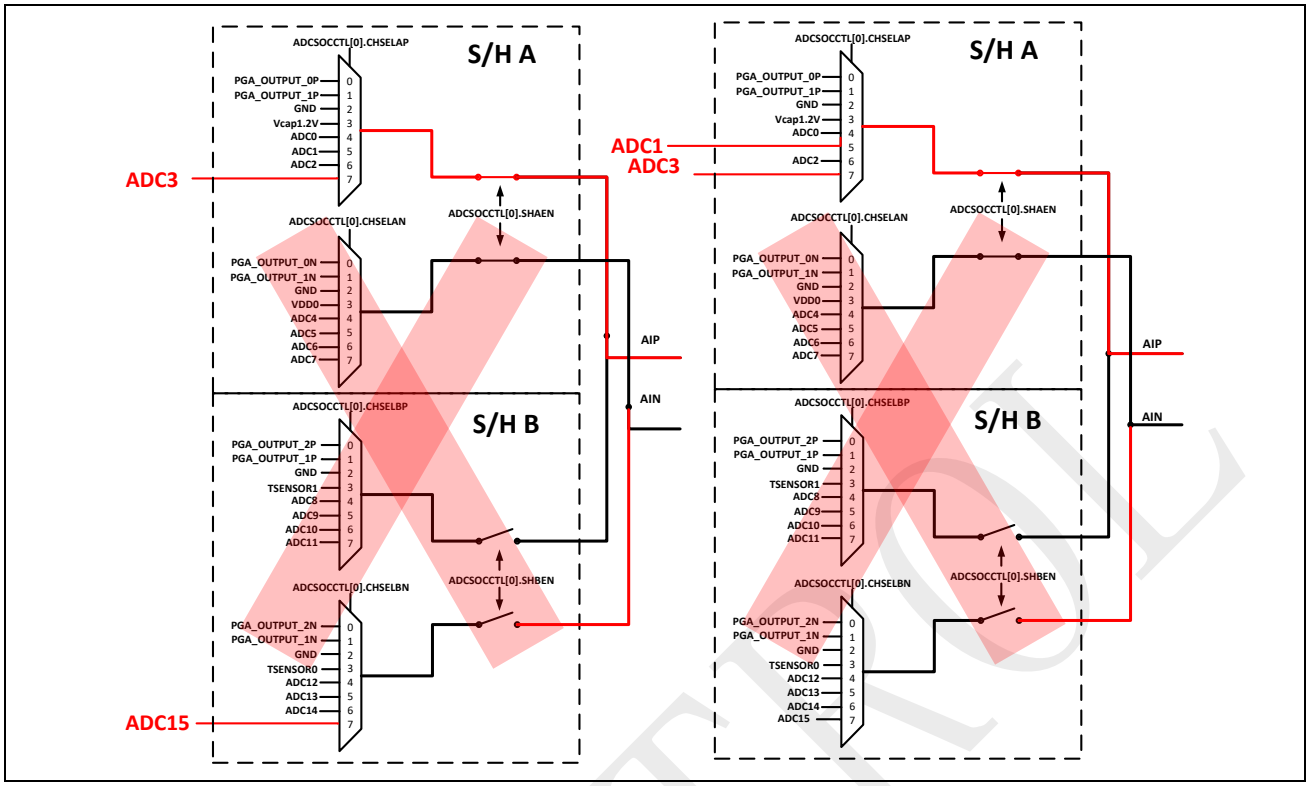
ADC_EasyInit2（ADC_SOC_3，GPIO_5，GPIO_4，ADCTRIG_Software）；

此函数的第二与第三个 parameter 放置双端的脚位，注意此两个参数并无方向性，因为正端与负端的通道已经被硬件架构所决定。

Step 3： 调适阶段，观察 UART 打印讯息是否有错

配置双端讯号请参照 ADC 信道表，只有 S/H A 中分类为 P 的通道可以与 S/H A 中分类为 N 的信道群配置为双端差分模式，P 类的信道无法与 P 类的信道配置为差分，不同的 S/H 的讯号也无法配置为差分，因此一不小心很可能会有错误的情况。

图 1-10：错误的差分通道配置示例



如上图所示，将 ADC3（GPIO4）与 ADC15（GPIO16）就无法配置为双端差分讯号，输入信道若是 ADC3 与 ADC1 一样会产生错误。

Step 4： 双端讯号采值

SPC1068 的 driver 中，对于双端（差分）ADC 配置，提供了两种的采值方式，推荐采用第一种方式，细节如下。

1. ADC_GetTrimResult2（SOC x）--校准后数据位移至-4096~4096（建议）

```
/*****************************************************************************//**
 * @brief  For Differential Mode
 *         Get Trim result(code) from ADC result register
 *         The trim coefficient locates at flash and load in SysInfo when start up
 *
 * @param[in]  u8SOC:  Select SOC to be used
 *
 * @return  ADC result in 0~4095
 *            Code            Real Voltage in ADC view
 *            -4095           -3.65V
 *            0               0 V
 *            4095            3.65 V
 *****************************************************************************/
#define ADC_GetTrimResult2(u8SOC)           ((((((int16_t)ADC->ADCRESULT[(u8SOC)].bit.VAL+\
                                              SysInfo.i16SHAOffset)*(int32_t)(SysInfo.i16SHAGain)
```

建议均采用 ADC_TrimResult2 的方式进行采值，此函数已经将采到的数值进行校准，范围是-4095～4095。但须注意的是，用户必须对讯号的极性，对输出的数值进行调整。

©2024 旋智电子科技（上海）有限公司

2. 只采 Raw Data

```
/*******************************************************************************//**
 * @brief  Get raw result(code) from ADC result register
 *
 * @param[in]  u8SOC:  Select SOC to be used
 *
 * @return  ADC result in 0~8191
 *            Code            Relative voltage in ADC view (ADC_AP-ADC_AN or ADC_BP-ADC_BN)
 *            0               -3.65 V
 *            4095            0 V
 *            8191            3.65 V
 * @detail  SPC1068 is a truly differential ADC, the code is the transformation of voltage
 *          Sampler Positive - Sampler Negative
 *          The negative voltage in ADC is "relative" but not "absolute" negative voltage.
 *          It can not be real absolute negative voltage in pin pad!!
 ******************************************************************************/
#define ADC_GetRawResult(u8SOC)    (ADC->ADCRESULT[(u8SOC)].bit.VAL)
```

第二种为直接读取 ADC 转换结果，未经过校准，8191 对应的是 3.65V，0 对应的是-3.65V，4095 则是对应到 0V，用户必须根据通道的极性自行计算数值范围。

建议搭配 ADC_GetTrimResult2()进行采值，输出范围自动调整为-4095~4095。

SPINTROL

1.8.2 初始化范例

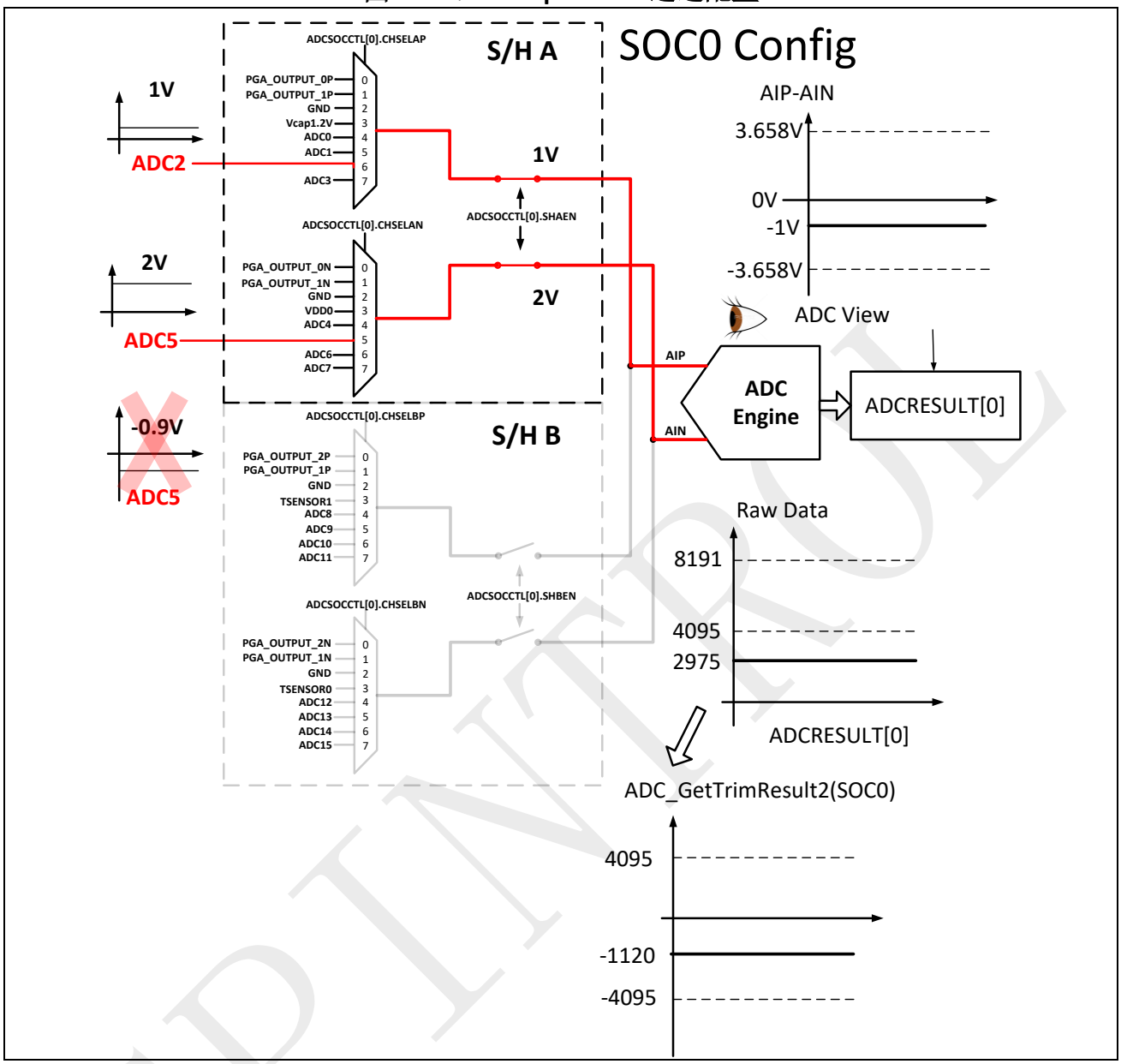
透过 ADC_EasyInit2()与 ADC_GetTrimResult2()的搭配，用户可以快速配置 ADC 与读取信道信号。下例为使用 SOC0 采取 ADC2（GPIO3）与 ADC5（GPIO6）两端差分讯号的范例码，与 ADC_EasyInit1 不同的处只在于必须设定两个差分讯号来源，在此例中，触发来源为软件触发。最后则是使用软件触发并且读取数值的范例。

Example 1.8-1

```
void main()
{
  int16_t i16VSP;
  ADC_EasyInit2(ADC_SOC_0, GPIO_3, GPIO_6, ADCTRIG_Software);
  /*         The above is the same as :   */
  ADC_EasyInit2(ADC_SOC_0, GPIO6, GPIO_3, ADCTRIG_Software);

  /* Optional : Adjust by circuit connected with ADC pin */
  ADC_SetSampleAndConvTime(ADC_SOC_0,250 /*ns*/ );

  /* Software trigger and get trim result */
  ADC_SoftwareTrigger(ADC_SOC_0);

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

  /* Get result (already trim and rescale from 0~4095) */
  i16VSP = ADC_GetTrimResult2(ADC_SOC_0);

  /* Clear interrupt flag */
  ADC_ClearInt(ADC_SOC_0);
}
```


©2024 旋智电子科技（上海）有限公司

图 1-11：Example 1.8-1 通道配置



因为输入讯号是由 S/H A 的两端进入，当输入 ADC2 是 1V，ADC5 是 2V，ADC 转换端看到的电压为 AIP=1V，AIN=2V，所以就 ADC 的角度来说，看到的是-1V，根据以下公式，可得转换结果为 2975。

$$ADCRESULT = 4095 + \frac{AIP - AIN}{3.658} \times 4096$$

ADC_GetTrimResult2()函数会自动更正，并且将数值拉到-4095~4095 的间，以此例来说，会得到-1120 的数值。

注意： ADC2 或是 ADC5 上的讯号不能为负电压。

1.9 ADC 采样内部 PGA 放大讯号范例

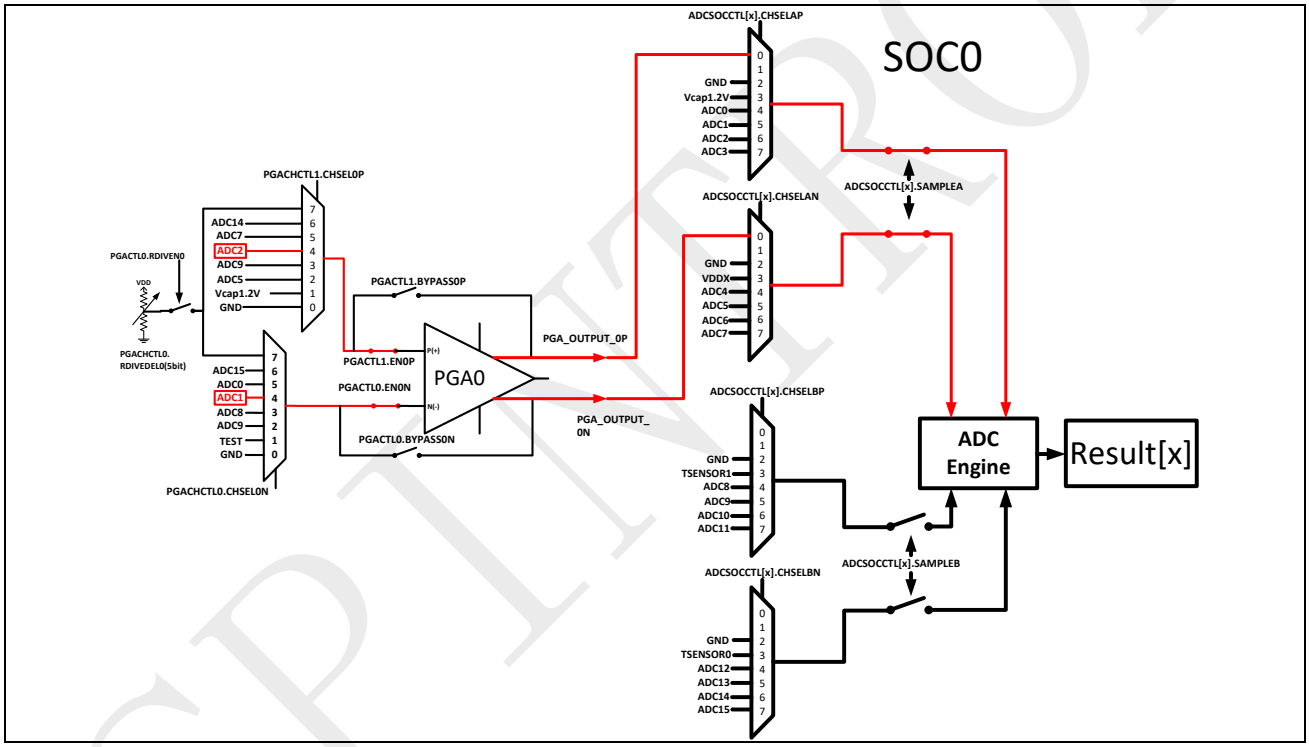
下例为 ADC 采样内部 PGA 放大后的讯号，以下假定 PGA 已经设定完成，并且设定为差动放大。

Example 1.9-1

```
void MotorADC_InitWithPGAExample(void)
{
  ADC_EasyInit2(ADC_SOC_0 ,ADCx_PGA0P ,ADCx_PGA0N ,ADCTRIG_PWM1A);
}
```

以上的范例如下图所示，此时 SOC0 将被 PWM1A 的触发信号所触发。

图 1-12：ADC 采样 PGA0 放大信号的差分通道配置



©2024 旋智电子科技（上海）有限公司

1.10 SDK 初始化 ADC 通道与采值常用 API 归纳

SPC10168 的 ADC 为真正的差分型 ADC，可有效消除共模噪声，亦可配置为传统的单端输入型 ADC，以下为 API 使用时的建议

1．当作单端输入使用时

- 使用 ADC_EasyInit1()进行 ADC 配置
- 使用 ADC_GetTrimResult1()进行数值的采取，数值范围为 0~4095。（此时输出是有号数）对应的电压数值为 0~3.658V

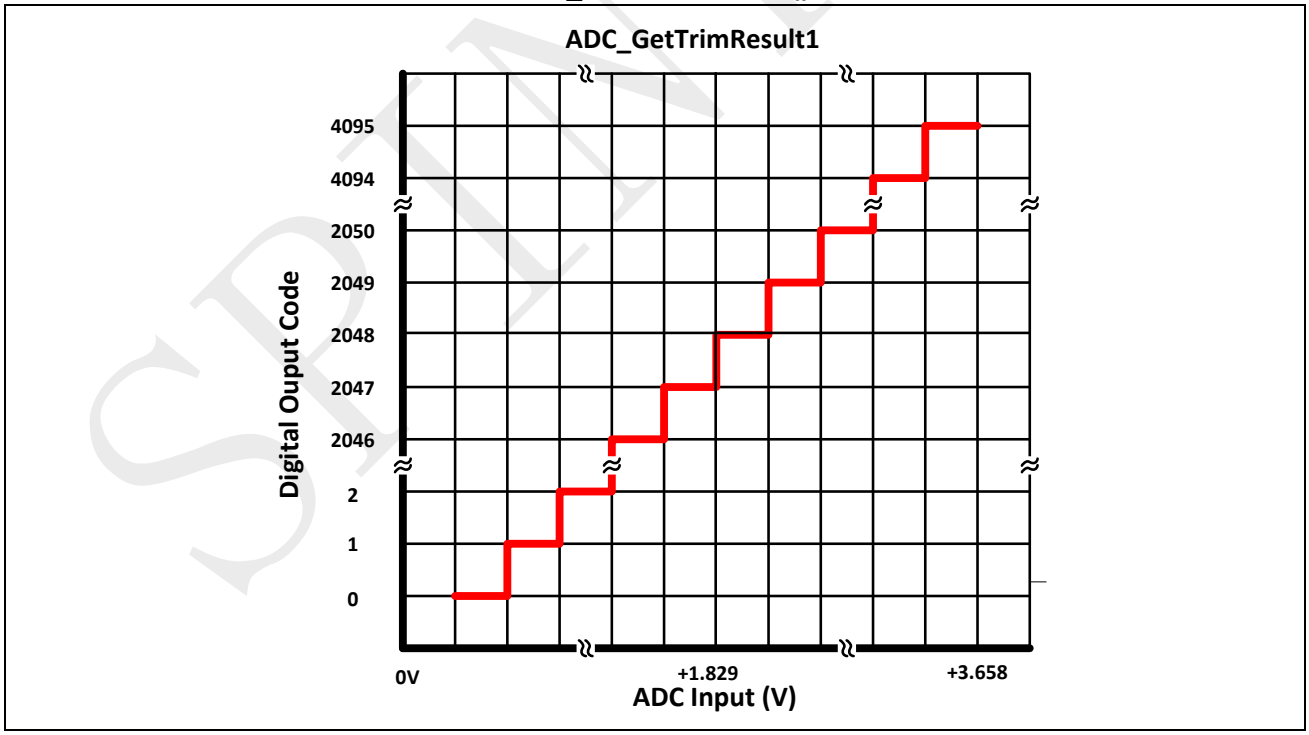
2．当作双端差分输入使用时

- 使用 ADC_EasyInit2()进行 ADC 配置
- 使用 ADC_GetTrimResult2()进行数值的采取，数值范围为-4095~4095。（此时输出是有号数）ADC 端点看到的相对电压范围为-3.658V~+3.658V

1.10.1 ADC_GetTrimResult1 数值范围分析

ADC_GetTrimResult1()的数值范围如下图所示。

图 1-13：ADC_GetTrimResult1()数值范围

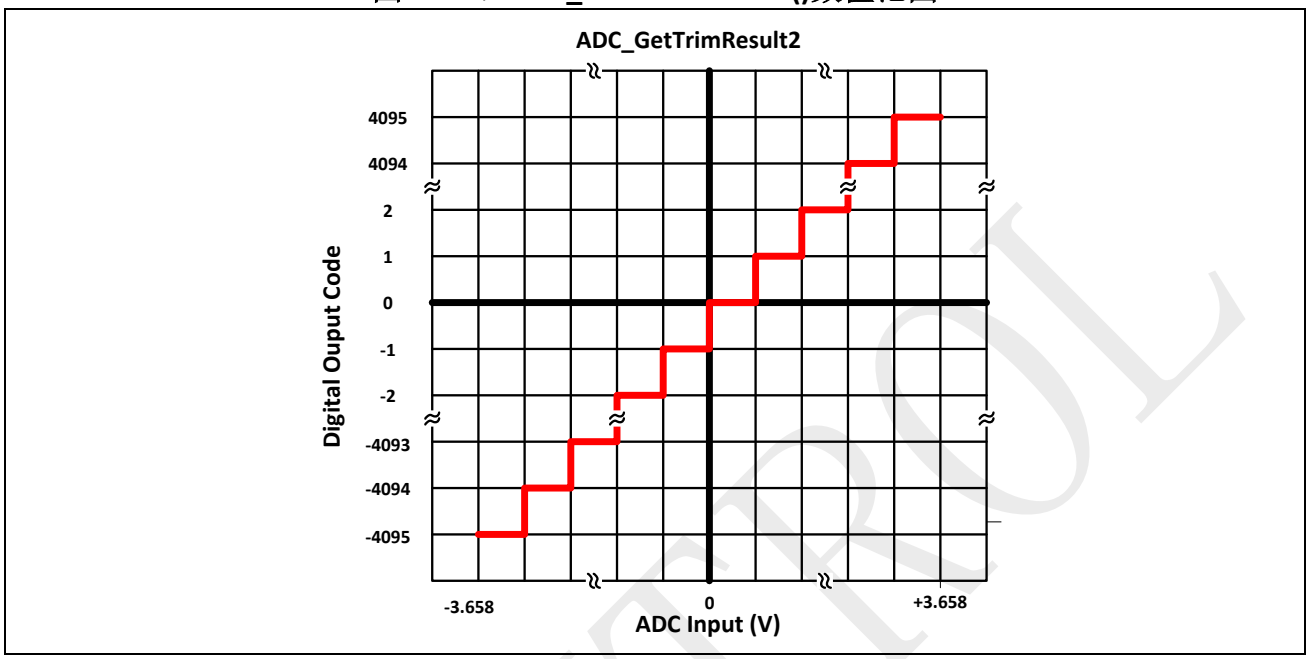


SPINTROL

1.10.2 ADC_GetTrimResult2 数值范围分析

ADC_GetTrimResult2()的数值范围如下图所示。

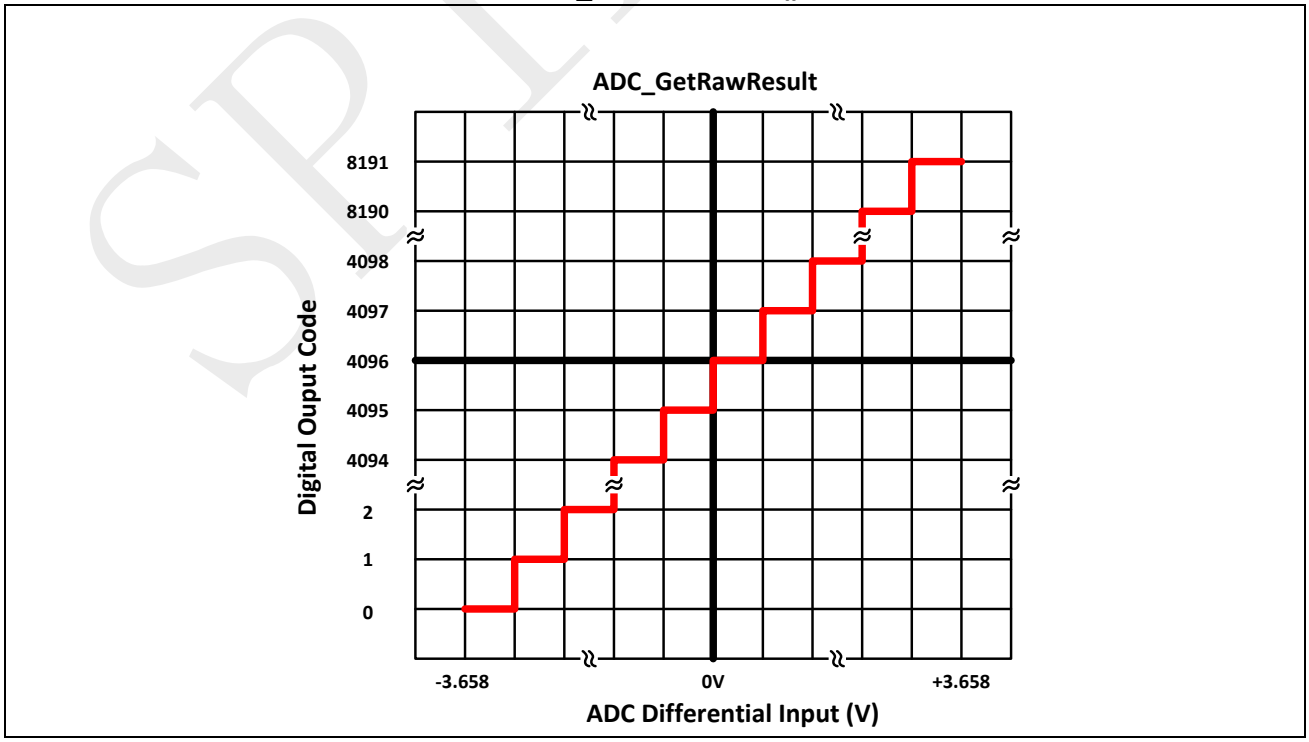
图 1-14：ADC_GetTrimResult2()数值范围



1.10.3 ADC_GetRawResult 数值范围分析

ADC_GetRawResult()的数值范围如下图所示。

图 1-15：ADC_GetRawResult()数值范围



©2024 旋智电子科技（上海）有限公司

1.11 ADC 同时采样模式配置

SPC1068 的 ADC 支持同时采样，当触发信号发生时，两个 Sample and hold 电路（S/H A 与 S/H B）将同时对信号进行采样，此功能适合电机电流采样等应用。

下图为同时采样 ADC1（GPIO2）与 ADC11（GPIO12）的同时采样配置。设定同时采样时，请注意以下事项：

- 请在偶数的 SOC 内配置信道
- 以下为例，即为 SOC0
- S/H A 与 S/H B 必须同时 Enable
- 转换完的后：偶数的 SOC 将通道 A 的转换结果存放在偶数的 Result 中。
- 偶数+1 的 SOC 将通道 B 的转换结果存放在（偶数+1）的 Result 中。
- 以下为例，转换后，ADC1 的转换结果将存在 ADCRESULT[0]中
- ADC11 的转换结果将存在 ADCRESULT[1]中
- 使用同时采样时，请勿使能奇数通道的 S/H。

以下为例，请勿使能 SOC1 的 S/H A and S/H B。

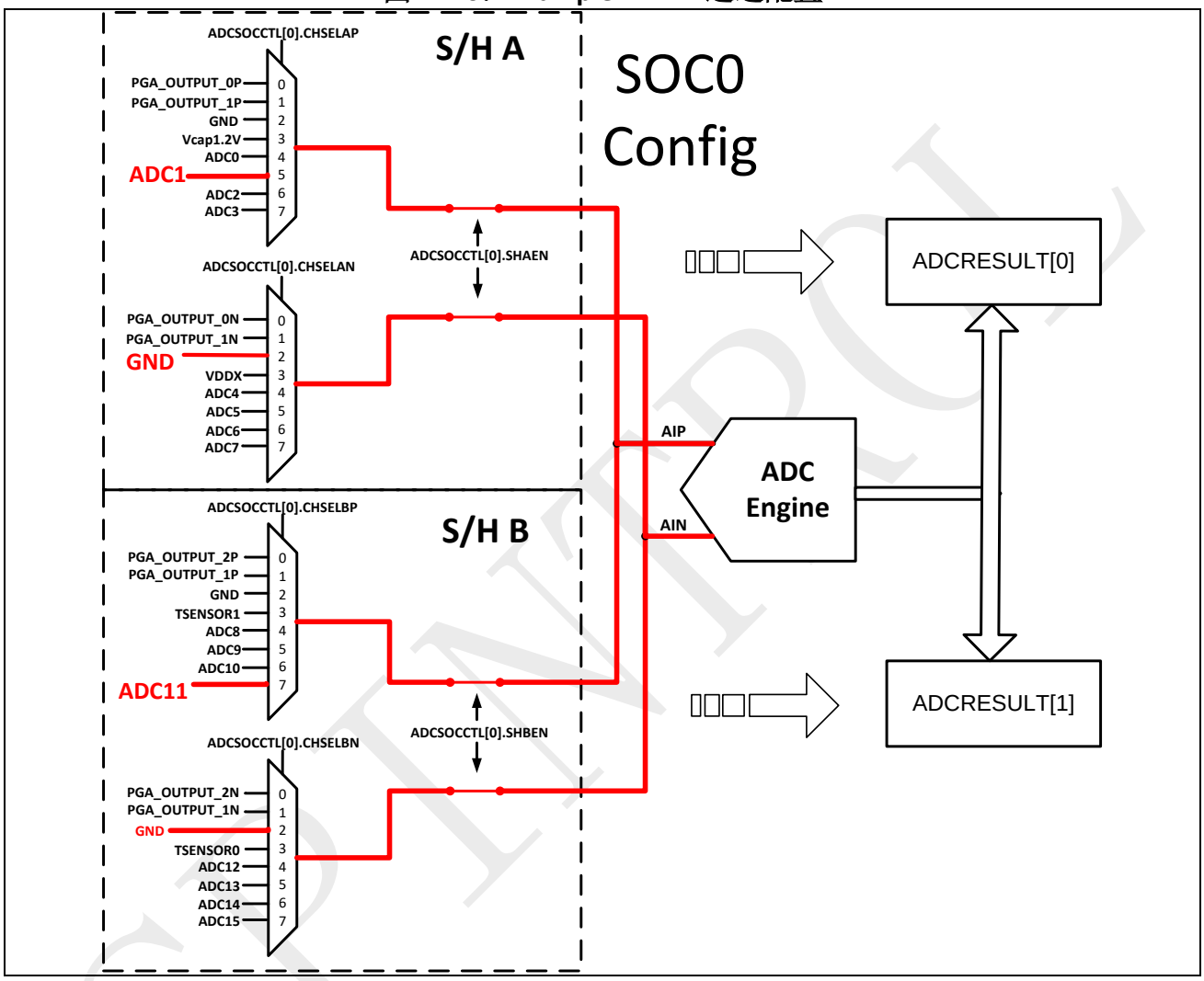
Example 1.11-1

```
void MotorADC_RegisterInitExample(void)
{
  /* Step1 : Enable ADC module */
  ADC_PowerUP();

  /* Step2 : Pin Mux configuration   */
  GPIO_SetPinAsAnalog(GPIO_1); /* Select GPIO2(ADC1) as analog */
  GPIO_SetPinAsAnalog(GPIO_12); /* Select GPIO12(ADC11) as analog
*/

  /* Step3~Step5 : ADC channel configuration  */
  ADC->ADCSOCCTL[ADC_SOC_0].all /* SOC0 is selected as Sampler */
  = ADCSOC0CTL_ALL_CHSELAP_ADC1 /* S/H A Positive = ADC1  */
   |ADCSOC0CTL_ALL_CHSELAN_GND  /* S/H A Negative = GND    */
   |ADCSOC0CTL_ALL_SHAEN_ENABLE /* Enable S/H A            */
   |ADCSOC0CTL_ALL_CHSELBP_ADC11 /* S/H B Positive = ADC11 */
   |ADCSOC0CTL_ALL_CHSELBN_GND   /* S/H B Negative = GND    */
   |ADCSOC0CTL_ALL_SHBEN_ENABLE /* Enable S/H A           */
   |ADCSOCxCTL_ALL_TRIGSEL_PWM1A  /* PWM1A SOC event will trigger
SOC0     */
   |ADCSOC0CTL_ALL_SAMPCNT_(5)    /* Sampling time = (5+2)*20ns =
                                      140ns if ADC = 50MHz
                                      */
   |ADCSOC0CTL_ALL_CONVCNT_(7); /* Conversion time = 140ns */

  /* Step 6 :Enable interrupt signal but do NOT enable ADC interrupt
service in M3 */
  ADC_EnableInt(ADC_SOC_0);

  /* Optional: Enable ADC interrupt service routine in Cortex-M3*/
```

```
  /* NVIC_EnableIRQ(ADC0_IRQn);*/
}
```

以上范例码的通道配置见下图。

图 1-16：Example 1.11-1 通道配置



©2024 旋智电子科技（上海）有限公司

1.12 PWM 触发 ADC 采样范例

下面以一个驱动 U 相开关的 PWM 触发三相电流采样为例。

Example 1.12-1

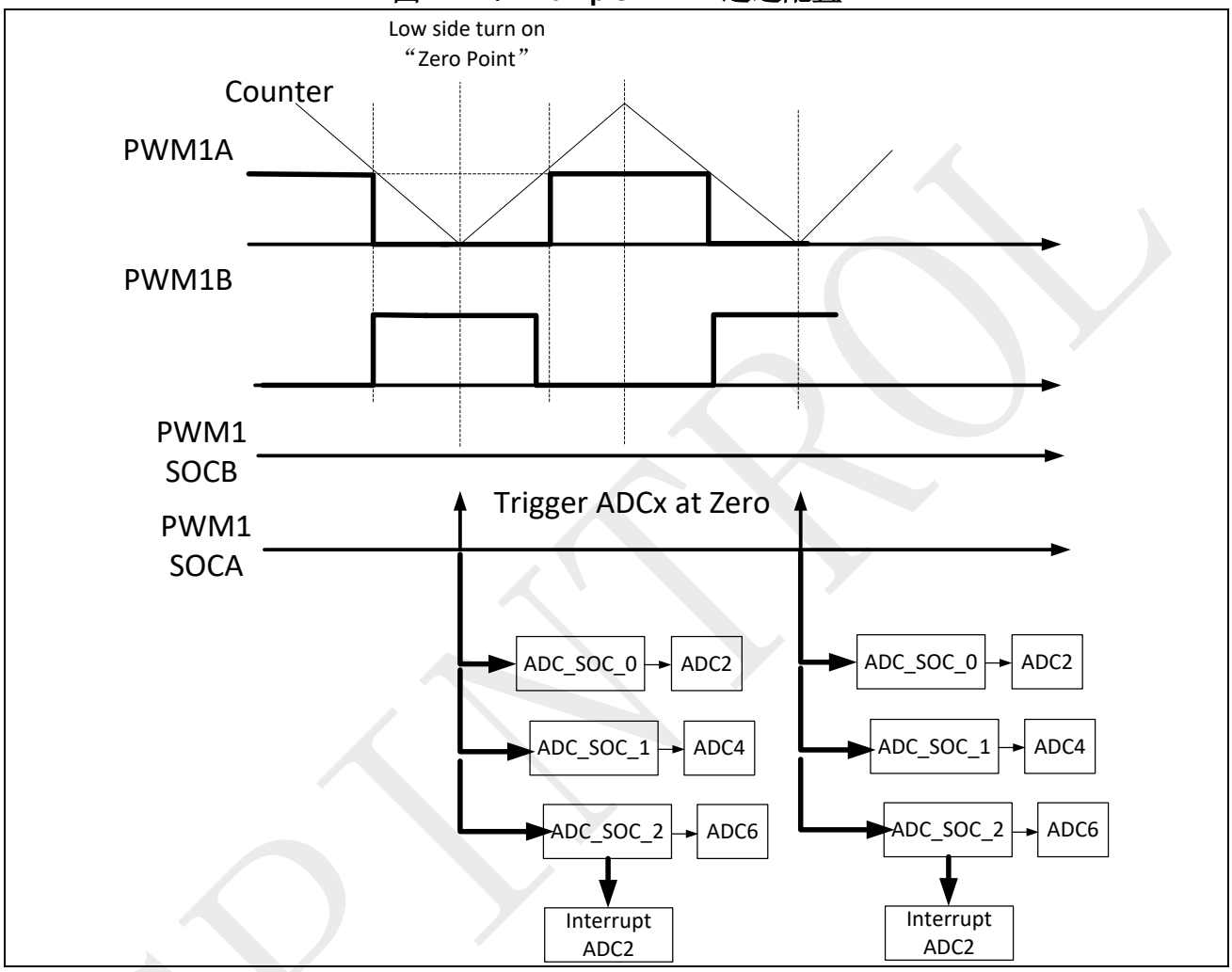
```
void MotorPWM_ExampleTrigADC()
{
  Sys_Init();
  CLOCK_InitWithRCO(CLOCK_HCLK_150MHZ);

  /*-------------------------------------------------------
      Step 1: Initial Basic Complementary PWM
  -------------------------------------------------------*/
  PWM_ComplementaryPairChannelInit(PWM1  ,  15000/*  PWM  frequency
(Hz) */, 1000 /* Deadtime(ns) */);
  PWM_ComplementaryPairChannelInit(PWM2  ,  15000/*  PWM  frequency
(Hz) */, 1000 /* Deadtime(ns) */);
  PWM_ComplementaryPairChannelInit(PWM3  ,  15000/*  PWM  frequency
(Hz) */, 1000 /* Deadtime(ns) */);

  /*-------------------------------------------------------
      Step 2: PWM Trig Select
  -------------------------------------------------------*/
  PWM1 ->ETSEL.all  |= ETSEL_ALL_SOCAEN_ENABLE // Enable ADC trigger
signal(SOCA)
                    |ETSEL_ALL_SOCASEL_TBCTR_EQU_ZERO  ;  //  while
counter(TBCTR) reaches zero
                                                    //(PWM low size
center where mosfet turn on)
  PWM1 ->ETPS.bit.SOCAPRD = ETPS_BIT_SOCAPRD_GEN_SOCA_ON_1ST_EVENT;

  /*-------------------------------------------------------
      Step 3: ADC initial
  -------------------------------------------------------*/
  ADC_EasyInit1(ADC_SOC_0   /*   ADC_SOC_0   */,GPIO_3   /*   ADC2
*/,ADCTRIG_PWM1A);
  ADC_EasyInit1(ADC_SOC_1   /*   ADC_SOC_1   */,GPIO_5   /*   ADC4
*/,ADCTRIG_PWM1A);
  ADC_EasyInit1(ADC_SOC_2   /*   ADC_SOC_2   */,GPIO_7   /*   ADC6
*/,ADCTRIG_PWM1A);

  /*-------------------------------------------------------
      Step 4. Interrupt service routine configuration
  -------------------------------------------------------*/
  NVIC_SetPriority(ADC2_IRQn,1);  /* 0 is highest priority and the
4 is lowest priority */
  NVIC_EnableIRQ(ADC2_IRQn);
}
```

以上先使用基本的初始化互补对，PWM1～PWM3 依次控制 UVW 三相，设定 PWM1 的 Counter 过零时触发 ADC，一次同时触发 SOC0～SOC2 分别负责转换三相电流，当 SOC2 转换完

的后，进入中断服务程序（SOC2 对应的中断服务程序为 ADC2_IRQ），进行电流采值。其中设定 PWM1 由 SOCA 送出触发讯号，ADC 相对的设定为被 PWM1A 所触发。

以上的配置如下图所示：

图 1-17：Example 1.12-1 通道配置



当 SOC0～SOC2 同时被触发时，根据优先层级，SOC0 会优先被转换，最后是 SOC2，上图设定为每一个 PWM 周期均送出 SOC 触发讯号。PWM 尚可从 CMPA，CMPB 点，也能从 Counter 的周期点进行触发。

©2024 旋智电子科技（上海）有限公司

1.13 电机驱动 ADC 建议管脚配置

SPC 1068 为无感 FOC 电机控制作了专门的优化，可以适用于以下各种方式的电机控制。

1. 双电机三电阻采样 FOC 控制

在这种方案里，第一台电机的电流讯号，可由 ADC2，ADC4，ADC6 引入 SPC1068，若有差分需求，可将差分讯号引入 ADC1，ADC3 与 ADC5，此配置亦可兼容于内部 PGA，详见 PGA 使用指南，由于 SPC 1068 的三个内部 PGA 都已经分配给第一个电机，所以对于第二个电机，需要增加外置运放。

表 1-4：双电机三电阻采样 FOC 控制管脚配置

管脚名称	管脚功能	备注
GPIO0		
TRSTn		
RESETn		
ADC0	OVS	采集母线电压
ADC1	IU-（Optional）	可输入内部 PGA
ADC2	IU+	
ADC3	IV-（Optional）	可输入内部 PGA
ADC4	IV+	
ADC5	IW-（Optional）	可输入内部 PGA
ADC6	IW+	
VDDA		
VSSA		
ADC7	VSP	调速输入
ADC8		
ADC9	VSP2	调速输入
ADC10	IU2	电机 2 U 相电流反馈
ADC11	IV2	电机 2 V 相电流反馈
ADC12	IW2	电机 2 W 相电流反馈
ADC13	DIR2	方向控制
ADC14	Fault2	模块告警
ADC15		

©2024 旋智电子科技（上海）有限公司

2. 双电机单电阻采样无感 FOC 控制

单电阻采样情况下，一般推荐使用 SPC1068 的内部 PGA 进行电流采样，这样组合很灵活。在实际选择上，可以在双电机单电阻采样无感 FOC 的两个电机管脚分配中选择一种就好。

表 1-5：双电机单电阻采样无感 FOC 控制管脚配置

管脚名称	管脚功能	备注
GPIO0		
TRSTn		
RESETn		
ADC0	OVS	采集母线电压
ADC1	IDC_1-（Optional）	可输入内部 PGA
ADC2	IDC_1+	
ADC3		
ADC4		
ADC5		
ADC6		
VDDA		
VSSA		
ADC7	VSP	调速输入
ADC8		
ADC9	VSP2	调速输入
ADC10	IDC_2-（Optional）	
ADC11	IDC_2+	
ADC12		
ADC13	DIR2	方向控制
ADC14	Fault2	模块告警
ADC15		

©2024 旋智电子科技（上海）有限公司