应用报告
ZHCA510A–July 2009–Revised January 2013

使用 Stellaris® 微控制器模数转换器

Eric Ocasio *Stellaris® Microcontrollers*

摘要

Stellaris 微控制器配备有使用创新型、基于序列采样架构的模数转换器，此转换器十分灵活且易于使用。 这份应用报告对 ADC 的采样架构进行了说明。 由于程序设计人员可通过 Stellaris 系列驱动程序库或直接写入器件控制寄存器来配置 Stellaris 微控制器，所有本文档对这两种方法都进行了说明。 本文档中的信息是为了补充器件专用数据表内的 ADC 一章，并认为本文档的读者已经对 ADC 的运行有了基本的了解。

内容

图片列表

图表列表

1 采样序列发生器

大多数在 8，16 和 32 位微控制器 (MCU) 内执行的模数转换器在模拟输入或通道改变时需要处理器的干预来配置每个转换。 Stellaris MCU 所使用的基于序列的架构使得软件能够使用一系列单一配置写入来启用多达四个独立的采样序列（包括所有模拟输入通道）。

SPMA028 — http://www-s.ti.com/sc/techlit/SPMA028

ADC 模块总共有四个采样序列发生器，可实现一个单触发事件的单个模拟源、四个模拟源（有两个 4 节拍序列发生器）或八模拟源的采样（如图 1中所示）。 每个采样序列发生器有其自身的配置寄存器组，这使得它完全独立于其它序列发生器。 一个采样序列发生器内的所有步进是可配置的，这使得软件能够选择模拟输入通道（包括温度传感器）、单端或差分模式采样和是否在步进完成后生产一个中断。 采样序列还具有可配置的优先级来处理同一触发源或触发器同时触发多个序列的情况。

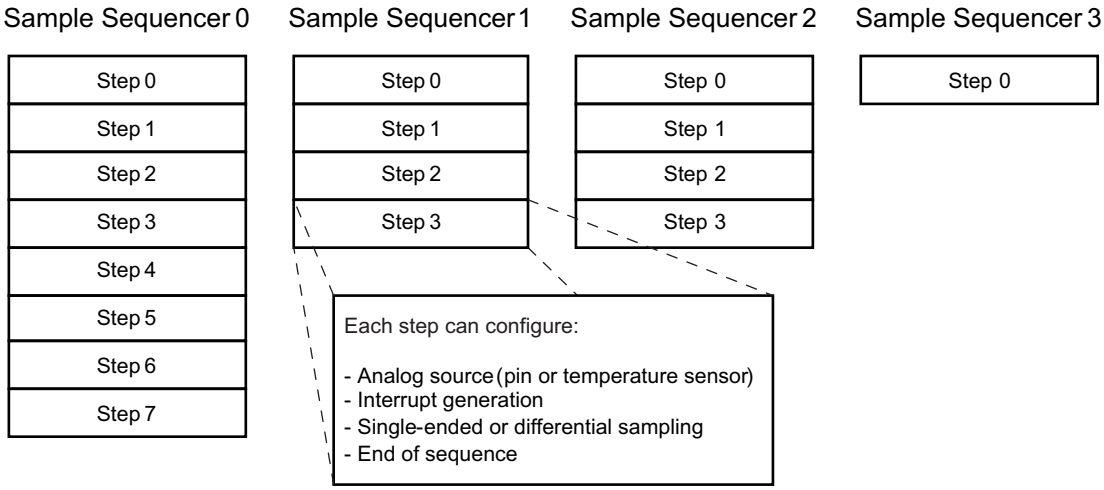

图 **1.** 采样序列发生器结构

一个采样序列发生器可由多个源触发，其中包括处理器、定时器、模拟比较器、脉宽调制 (PWM) 单元，或者通用输入/输出 (GPIO)。 对于多个序列具有同一个触发源或被同时触发的情况，ADC 根据已配置的序列优先级来判断执行顺序。 当一个采样序列被触发时，根据器件的不同，它开始使用已设定的采样速率 - 每秒 250K，500K 或 1M 采样 (SPS) 进行采样 - 并且重复序列的步进。 采样持续到步进的终止 (END) 位被设定，这个位的设定表示序列结束。 可针对序列中的任一步进设定终止位，这意味着无需一个指定的采样序列来收集其最大的样本数量。 当采样序列完成时，转换结果被存储在采样序列 FIFO 中并可由处理器进行检索。

2 模块配置示例

为了演示配置 ADC 所需的步进，请考虑图 2中显示的示例。 除了片载温度传感器之外，总共监控三个传感器。 由于使用三个模拟输入，这个示例假定特定的 Stellaris 器件具有至少三个模拟输入。 每个采样序列有其自身的 FIFO，FIFO 槽的数量等于序列发生器的大小。

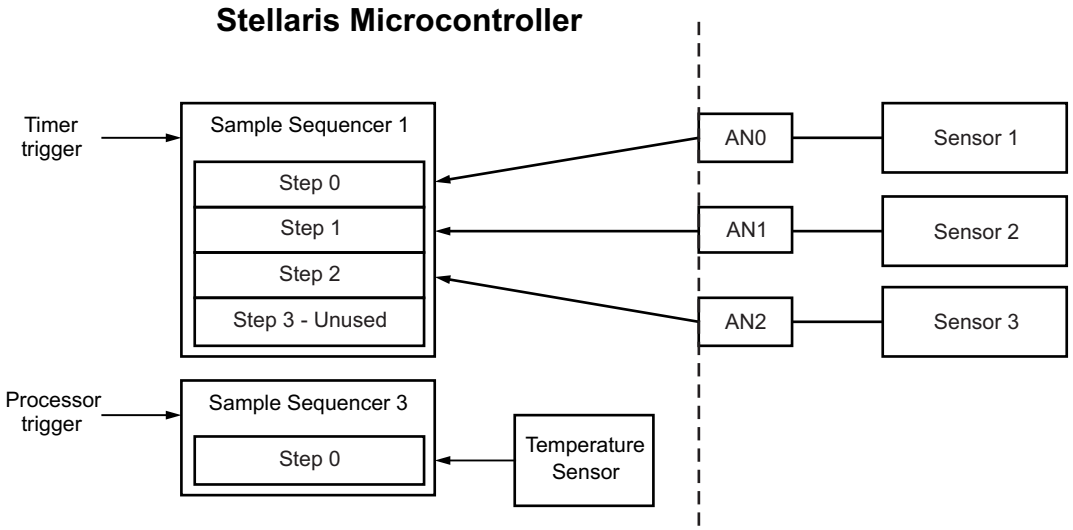

图 2. 示例系统配置

请注意，模拟输入是如何被映射到采样序列发生器内单个步进的。 要监控三个传感器输入；因此，四个步进序列发生器中的一个（在这个情况下为采样序列发生器 1）被使用。 使用采样序列 3 来采样温度传感器，这是因为它只需一个步进并且有一个独立的触发源。 如果温度传感器被配置成具有一个定时器触发器，它应该被放置在序列 1 的未使用的步进中。

注: 下面部分中的所有代码示例显示了直接寄存器写入和读取以及到 Stellaris 外设驱动程序库的 API 调用。 如果尝试重新生成直接寄存器访问示例，适当的 IC 头文件，例如用于 LM3S811 部件的*lm3s811.h*，必须被包括在内。 这些头文件，每个 Stellaris 系列的产品成员一个，可在已安装的软件树中的`StellarisWare/inc`目录内找到。 为了使用 Stellaris 外设驱动程序库而非直接寄存器访问，需要头文件*hw_types.h*，*hw_memmap.h*和*adc.h*。 这些头文件可在 StellarisWare 和 StellarisWare/DriverLib 目录中找到。

2.1 模块初始化

在复位之外，所有外设时钟被禁用以减少功耗并且必须针对每个外设模块被启用。 启用一个外设时钟是一项简单的任务，此任务需要写入到系统控制模块内运行模式时钟选通控制寄存器 (RCGCx) 中的一个。 为了启用到 ADC 的时钟，将一个 '1' 写入到运行模式时钟选通控制 0 寄存器 (RCGC0)（地址 0x400FE100）的位 16（ADC 位）。

表 1. 启用 ADC 时钟

使用直接寄存器写入

`//`
`// Enable the clock to the ADC module`
`//`
`// System Control RCGC0 register`
`//`
`SYSCTL_RCGC0_R |= SYSCTL_RCGC0_ADC;`

使用 **DriverLib**

`//`
`// Enable the clock to the ADC module`
`//`
`SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC);`

程序设计人员必须配置的另外一个 ADC 方面是采样速率。 复位之后的缺省采样速率为 125KSPS。 根据器件的不同，通过写入 RCGC0 寄存器的位 8-11，ADC 采样速率可被设定为 250KSPS，500KSPS 或 1MSPS。表 2显示了位域的有效数据范围和可用的采样速率。 您不能将采样速率设定成高于器件的最大速率。 如果软件尝试将 ADC 的采样速率配置成器件不支持的速率，器件保持在缺省值或者最近一次被设定的值。表 2显示了如何将 ADC 的采样速率配置为 500KSPS。

表 2. 设定 ADC 采样速率

使用直接寄存器写入

`//`
`// Configure the ADC to sample at 500KSps`
`//`
`// System Control RCGC0 register`
`//`
`SYSCTL_RCGC0_R |= SYSCTL_RCGC0_ADCSPD500K;`

使用 **DriverLib**

`//`
`// Configure the ADC to sample at 500Ksps`
`//`
`SysCtlADCSpeedSet(SYSCTL_SET0_ADCSPEED_500KSPS)`

值	采样速率		值	采样速率
0x0	每秒 125K 次采样		0x2	每秒 500K 次采样
0x1	每秒 250K 次采样		0x3	每秒 1M 次采样

2.2 采样序列配置

在改变 ADC 内的配置参数之前，将目标采样序列禁用是一个好的做法。 禁用采样序列可安全修改配置参数，而不会发生意外触发。 为了禁用一个或多个序列，将 ADC 激活采样序列发生器 (ADCACTSS) 寄存器的相应位设定为 '0'。 对于这个示例，序列 1 和 3 应该被禁用。

表 **3.** 禁用采样序列

使用直接寄存器写入

```
//
// Disable sample sequences 1 and 3
//
// ADC Active Sample Sequencer register
//
ADC_ACTSS_R &= ~(ADC_ACTSS_ASEN1 | ADC_ACTSS_ASEN3);
```

使用 **DriverLib**

```
//
// Disable sample sequences 1 and 3
//
ADCSequenceDisable(ADC_BASE, 1);
ADCSequenceDisable(ADC_BASE, 3);
```

序列被禁用时，现在可安全加载新的配置参数。 首先配置采样序列的优先级。 在多个 ADC 触发器被同时激活的情况下，或者多个序列被同一个源触发时，ADC 控制逻辑必须决定哪个采样序列先运行。 复位后，根据采样序列的编号对它们进行优先级排序，这意味着序列 0 具有最高的优先级，而序列 3 具有最低的优先级（寄存器位域范围从 0-3，0 的优先级最高，而 3 的优先级最低）。 这个示例没有特别的优先级要求，所有序列 3 被配置为具有最高的优先级。

除了优先级，必须配置 ADC 触发源。 ADC 提供广泛的触发源，其中包括处理器，模拟比较器（如果可用的话），GPIO，PWM 和定时器，但是这个示例要求序列 1 具有一个定时器触发器，而序列 3 需要具有一个处理器触发器。

表 **4.** 配置序列优先级和触发器

使用直接寄存器写入

```
//
// Configure sequence priority: order (highest to lowest)= 3, 1, 0, 2
//
// ADC Sample Sequencer Priority register
//
ADC_SSPRI_R = (ADC_SSPRI_SS3_1ST | ADC_SSPRI_SS1_2ND |
               ADC_SSPRI_SS0_3RD | ADC_SSPRI_SS2_4TH);
//
// Set up sequence triggers: sequence 1 = timer (encoding 0x5),
// sequence 3 = Processor (encoding 0x0)
//
// ADC Event Multiplexer Select register
//
ADC_EMUX_R = (ADC_EMUX_EM1_TIMER | ADC_EMUX_EM3_PROCESSOR);
```

使用 **DriverLib**

```
//
// Configure sample sequence 1: timer trigger, priority = 1
//
ADCSequenceConfigure(ADC_BASE, 1, ADC_TRIGGER_TIMER, 1);
//
// Configure sample sequence 3: processor trigger, priority = 0
//
ADCSequenceConfigure(ADC_BASE, 3, ADC_TRIGGER_PROCESSOR, 0);
```

配置过程的下一步是建立序列步进。 有两个寄存器负责采样序列的独立步进：ADC 采样序列输入 复用器选择 (ADCSSMUX) 和 ADC 采样序列控制 (ADCSSCTL) 寄存器。

ADCSSMUX 寄存器允许软件为序列中的每一步进选择模拟输入源，而一个步进能够选择模拟输入中的任何一个。 如果一个序列步进正在对温度传感器进行采样，硬件将忽略 ADCSSMUX 寄存器内相应的值。

诸如采样模式（单端或差分）、温度传感器采样、中断和序列末尾的控制信息被存储在 ADCSSCTL 寄存器内。 采样序列中的每个步进有其自身的 4 位半字节，这使得软件能够设定任一之前提到的配置选项。 序列中的第一个步进占用寄存器内的最低有效半字节，以此类推。 软件负责设定一个序列最后一个步进的MM位。 如果未设定序列的MM位，会发生无法预计的运行状态。

图 3显示了配置半字节的布局。 每个域旁边的*n*与步进编号有关；对于步进 3，*n*为 3。

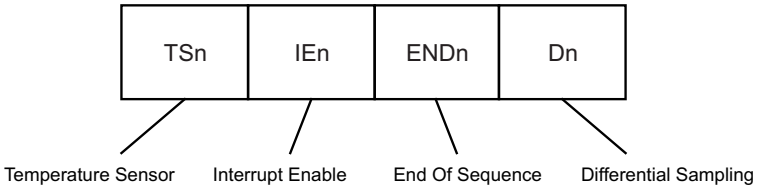

图 3. 序列步进配置半字节

表 5. 配置序列步进

使用直接寄存器写入

```
//
// Configure sample sequence 1 input multiplexer:
//
// - Step 0: Analog Input 0
// - Step 1: Analog Input 1
// - Step 2: Analog Input 2
//
// ADC Sample Sequence Input Multiplexer Select 1 register
//
ADC_SSMUX1_R = ((0 << ADC_SSMUX1_MUX0_S) |
                (1 << ADC_SSMUX1_MUX1_S) |
                (2 << ADC_SSMUX1_MUX2_S));
//
// Configure sample sequence 1 control
//
// - Step 0: Single-ended, No temp sensor, No interrupt
// - Step 1: Single-ended, No temp sensor, No interrupt
// - Step 2: Single-ended, No temp sensor, Interrupt, End of sequence
//
// ADC Sample Sequence Control 1 register
//
ADC_SSCTL1_R = (ADC_SSCTL1_END2 | ADC_SSCTL1_IE2);
//
// Configure sample sequence 3 input multiplexer - not necessary since
// sequence 3 samples the temperature sensor
//
//
// Configure sample sequence 3 control
//
// - Step 1: Single-ended, temp sensor, no interrupt, end of sequence
//
// ADC Sample Sequence Control 1 register
//
ADC_SSCTL3_R = (ADC_SSCTL3_TS0 | ADC_SSCTL3_END0);
```

使用 **DriverLib**

`//`
`// Configure sample sequence 3 steps 0, 1 and 2`
`//`
`ADCSequenceStepConfigure(ADC_BASE, 1, 0, ADC_CTL_CH0);`
`ADCSequenceStepConfigure(ADC_BASE, 1, 1, ADC_CTL_CH1);`
`ADCSequenceStepConfigure(ADC_BASE, 1, 2, ADC_CTL_CH2 | ADC_CTL_IE | ADC_CTL_END)`
`//`
`// Configure sample sequence 3 step 0`
`//`
`ADCSequenceStepConfigure(ADC_BASE, 3, 0, ADC_CTL_TS | ADC_CTL_END);`

2.3 使用 ADC 中断

Stellaris 基于序列的架构提供大量的中断编程灵活性。 一个采样序列的所有步进能够触发一个中断，使得软件能够在采样序列的任一点上设定标识符。

为了启用中断，软件必须为要求中断的采样序列设定屏蔽 (MASK) 位。 例如，如果采样序列 1 被配置成触发一个中断，ADC 中断屏蔽 (ADCIM) 寄存器的 MASK1 位被设定为 '1'。

表 **6.** 启用采样序列发生器中断

使用直接寄存器写入

`//`
`// Enable the interrupt for sample sequence 1`
`//`
`// ADC Interrupt Mask register`
`//`
`ADC_IM_R = ADC_IM_MASK1;`

使用 **DriverLib**

`//`
`// Enable the interrupt for sample sequence 1`
`//`
`ADCIntEnable(ADC_BASE, 1);`

即使在中断屏蔽位被设定时，要使一个中断发生，软件必须为采样序列的一个或多个步进设定步进中断使能位IE。 步进中断使能位是 ADCSSCTL 寄存器内步进配置半字节的一部分。 在这里的示例配置中，针对步进 2 的IE位被设定（请见表 5）。

2.4 数据检索

每个采样序列有其自己的 FIFO，其深度等于特定序列所支持的步进的数量（也就是说，采样序列 0 具有一个 8 条目 FIFO，这是因为这个序列有 8 个步进）。 从 FIFO 中检索数据是十分简单的；对 ADC 采样序列 FIFO*n*(ADCSSFIFOn) 寄存器的读取将从 FIFO 返回一个值。

FIFO 读取期间被返回的数据是一个 32 位值，转换结果存储在这个值较低的 10 位中。 在 ADC 上溢状态 (ADCOSTAT) 和 ADC 下溢状态 (ADCUSTAT) 寄存器中提供 FIFO 上溢和下溢标志，并在 ADC 采样序列 FIFO 寄存器中提供 FIFO 空、满、头和尾指针信息。

表 7. 从 FIFO 中检索数据

使用直接寄存器读取

`//`
`// Retrieve data from sample sequence 1 FIFO`
`//`
`// ADC Sample Sequence 1 FIFO register`
`//`
`ulSensor0Data = ADC_SSFIFO1_R;`
`ulSensor1Data = ADC_SSFIFO1_R;`
`ulSensor2Data = ADC_SSFIFO1_R;`
`//`
`// Retrieve data from sample sequence 3 FIFO`
`//`
`// ADC Sample Sequence 3 FIFO register`
`//`
`ulTempSensorData = ADC_SSFIFO3_R;`

使用 **DriverLib**

`//`
`// Retrieve data from sample sequence 1 FIFO`
`//`
`ADCSequenceDataGet(ADC_BASE, 1, &ulSeq1DataBuffer);`
`//`
`// Retrieve data from sample sequence 3 FIFO`
`//`
`ADCSequenceDataGet(ADC_BASE, 3,&ulSeq3DataBuffer);`

3 差分采样

除了传统的单端采样，ADC 模块还支持对两个模拟输入通道进行差分采样。 为了启用差分采样，软件必须在一个步进配置半字节中设定 D 位（请见图 3）。

当一个序列步进被配置用于差分采样时，它在 ADCSSMUX 寄存器内相应的值必须被设定为四个差分信号对（编号 0-3）中的一个。 差分信号对 0 对模拟输入端 0 和 1 进行采样；差分信号对 1 对模拟输入端 2 和 3 进行采样，依此类推（请见表 8）。 ADC 不支持其它差分信号配对，诸如模拟输入端 0 与模拟输入端 3。所支持的差分信号对的数量取决于特定 Stellaris 微控制器器件上模拟输入的数量。

表 8. 差分采样

差分信号对	模拟输入
0	0 和 1
1	2 和 3
2	4 和 5
3	6 和 7

差分模式下采样电压是奇数通道与偶数通道电压的差值：

因此，ΔV（差分电压）= V_0（偶数通道）– V_1（奇数通道）。
如果 ΔV = 0，则转换结果 = 0x1FF
如果 ΔV > 0，则转换结果 > 0x1FF（取值范围 0x1FF–0x3FF）
如果 ΔV < 0，则转换结果 < 0x1FF（取值范围 0–0x1FF）

差分信号对为模拟输入指定极性：偶数输入总是为正，奇数输入总是为负。 为了出现一个有效的转换结果，负输入必须在正输入的 ±1.5V 范围之内。 如果一个模拟输入大于 3V 或少于 0V（模拟输入的有效范围），那么输入电压会被削减，这意味着它出现在 ADC 上的电压分别为 3V 或 0V。

图 4显示了一个负极中心输入电压为 1.5V 的示例。在该配置中，差分范围从 -1.5V 到 1.5V。图 5显示了负极中心输入电压为 -0.75V 的示例，这意味着由于输入电压少于 0V，正极输入上的输入电压饱和之前的 -0.75 差分电压。显示了负极中心输入电压为 2.25V 的示例，在这里由于输入电压将大于 3V，正通道上的输入电压饱和之前的 0.75V 差分电压。

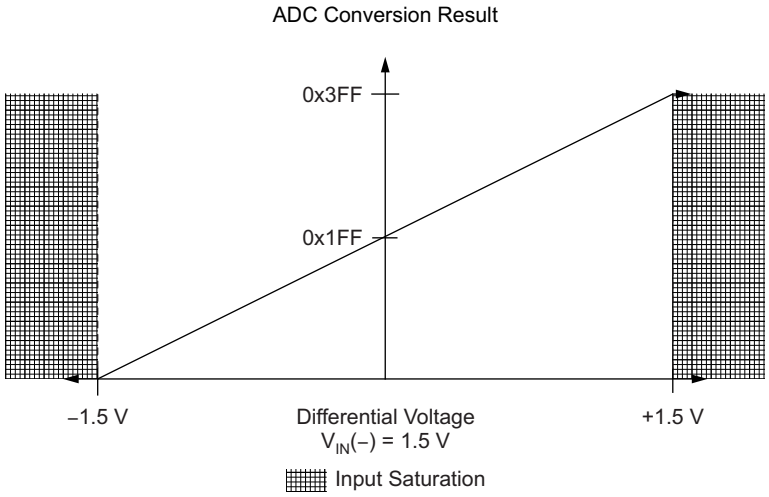

图 **4.** 差分采样范围，**V_{IN}(-) = 1.5V**

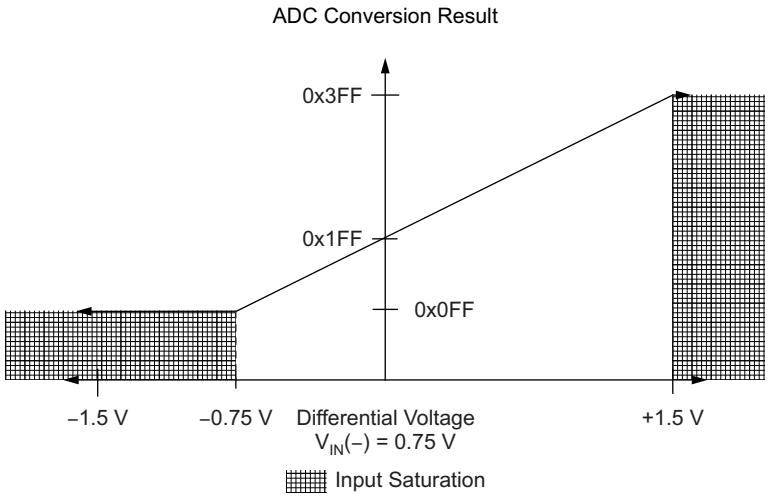

图 **5.** 差分采样范围：**V_{IN}(-) = 0.75V**

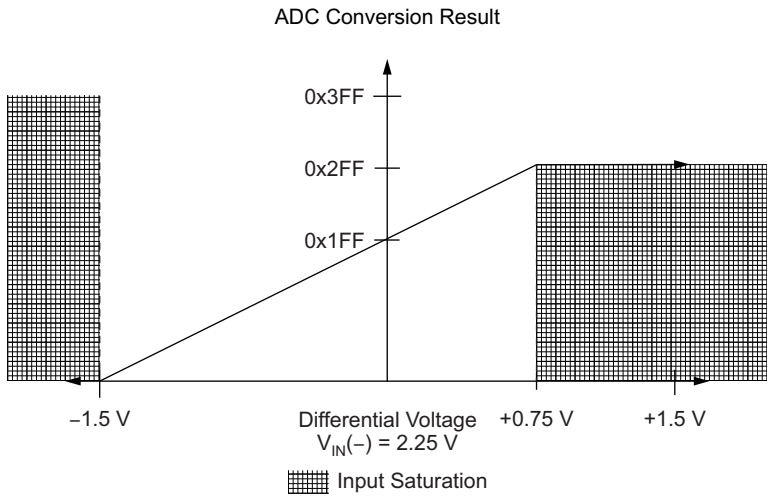

图 **6.** 差分采样范围：**V_{IN}(-) = 2.25V**

表 **9.** 启用差分采样

使用直接寄存器写入

`//`
`// Configure sequence 3, step 0 for differential sampling`
`//`
`// ADC Sample Sequence Control 3 register`
`//`
`ADC_SSCTL3_R = (ADC_SSCTL3_D0 | ADC_SSCTL3_END0);`
`//`
`// Configure sample sequencer 3 input multiplexer to sample differential`
`// pair 1`
`//`
`// ADC Sample Sequence Input Multiplexer Select 3 register`
`//`
`ADC_SSMUX3_R = (1 << ADC_SSMUX3_MUX0_S);`

使用 **DriverLib**

`//`
`// Configure sequence 3, step 0 to sample differential pair 1`
`//`
`ADCSequenceStepConfigure(ADC_BASE, 3, 0, ADC_CTL_CH1 | ADC_CTL_D | ADC_CTL_END);`

4 硬件取平均值电路

某些应用要求的精度超出了 ADC 的标准技术规格。 为了满足这一需要，ADC 模块包含一个内置的硬件取平均值电路，此电路可对输入源进行 64 倍的过采样。

当硬件取平均值电路被启用时，对所有由 ADC 采集的原始数据进行同样倍数的过采样；取平均值电路不能在采样序列中被逐步启用或禁用。 信号路径细节请见图 7。

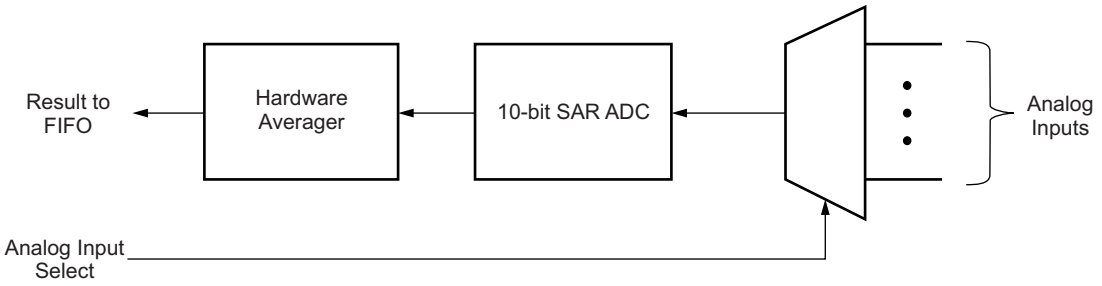

图 **7.** 硬件取平均值电路

在使用硬件取平均值电路之前还需要将带宽的影响考虑在内。 不论为取平均值电路选择的过采样因子是多少，都会将总体 ADC 吞吐量减少同样的数量级。 例如，8 倍过采样将一个 500KSPS ADC 模块的吞吐量减少至 62.5KSPS，这是因为 ADC 在将一个单一转换结果返回到 FIFO 之前要采集并将 8 个样本取平均值。

为了启用硬件取平均值电路，软件将一个介于 1 和 6 之间的值写入 ADC 采样取平均值控制寄存器 (ADCSAC) 的AVG域。 应用到输入的过采样数量等于 2^{AVG}。 当AVG位为 '0' 时（缺省配置），输入上不采用取平均值运算。

表 **10.** 启用 **8** 倍硬件取平均值

使用直接寄存器写入

`//`
`// Enable 8x hardware averaging`
`//`
`// ADC Sample Averaging Control register`
`//`
`ADC_SAC_R = ADC_SAC_AVG_8X;`

使用 **DriverLib**

`//`
`// Enable 8x hardware averaging`
`//`
`ADCHardwareOversampleConfigure(ADC_BASE, 8);`

5 结论

虽然与很多竞争对手生产的 ADC 模块在架构上有所不同，Stellaris ADC 在不增加处理器开销的同时为用户提供更多的灵活性、控制功能和特性。 通过将简单编程接口和所包含的驱动程序库组合在一起，很多用户将会发现他们能够很容易地将 Stellaris ADC 集成到他们的设计中去。

6 参考

以下相关文档和软件可从Stellaris 网址为www.ti.com/stellaris：

- Stellaris 微控制器数据表（产品选择工具中提供单独的器件文档）。
- Stellaris 微控制器 ROM 用户指南（产品选择工具中提供单独的器件文档）。
- StellarisWare 驱动程序库。 可从www.ti.com/tool/sw-drl内下载。
- StellarisWare 驱动程序库用户手册，版号 SW-DRL-UG（文献编号SPMU019）。

重要声明

德州仪器(TI) 及其下属子公司有权根据 JESD46 最新标准, 对所提供的产品和服务进行更正、修改、增强、改进或其它更改， 并有权根据 JESD48 最新标准中止提供任何产品和服务。客户在下订单前应获取最新的相关信息, 并验证这些信息是否完整且是最新的。所有产品的销售都遵循在订单确认时所提供的TI 销售条款与条件。

TI 保证其所销售的组件的性能符合产品销售时 TI 半导体产品销售条件与条款的适用规范。仅在 TI 保证的范围内，且 TI 认为 有必要时才会使用测试或其它质量控制技术。除非适用法律做出了硬性规定，否则没有必要对每种组件的所有参数进行测试。

TI 对应用帮助或客户产品设计不承担任何义务。客户应对其使用 TI 组件的产品和应用自行负责。为尽量减小与客户产品和应 用相关的风险，客户应提供充分的设计与操作安全措施。

TI 不对任何 TI 专利权、版权、屏蔽作品权或其它与使用了 TI 组件或服务的组合设备、机器或流程相关的 TI 知识产权中授予 的直接或隐含权限作出任何保证或解释。TI 所发布的与第三方产品或服务有关的信息，不能构成从 TI 获得使用这些产品或服 务的许可、授权、或认可。使用此类信息可能需要获得第三方的专利权或其它知识产权方面的许可，或是 TI 的专利权或其它 知识产权方面的许可。

对于 TI 的产品手册或数据表中 TI 信息的重要部分，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况 下才允许进行复制。TI 对此类篡改过的文件不承担任何责任或义务。复制第三方的信息可能需要服从额外的限制条件。

在转售 TI 组件或服务时，如果对该组件或服务参数的陈述与 TI 标明的参数相比存在差异或虚假成分，则会失去相关 TI 组件 或服务的所有明示或暗示授权，且这是不正当的、欺诈性商业行为。TI 对任何此类虚假陈述均不承担任何责任或义务。

客户认可并同意，尽管任何应用相关信息或支持仍可能由 TI 提供，但他们将独力负责满足与其产品及在其应用中使用 TI 产品 相关的所有法律、法规和安全相关要求。客户声明并同意，他们具备制定与实施安全措施所需的全部专业技术和知识，可预见 故障的危险后果、监测故障及其后果、降低有可能造成人身伤害的故障的发生机率并采取适当的补救措施。客户将全额赔偿因 在此类安全关键应用中使用任何 TI 组件而对 TI 及其代理造成的任何损失。

在某些场合中，为了推进安全相关应用有可能对 TI 组件进行特别的促销。TI 的目标是利用此类组件帮助客户设计和创立其特 有的可满足适用的功能安全性标准和要求的终端产品解决方案。尽管如此，此类组件仍然服从这些条款。

TI 组件未获得用于 FDA Class III（或类似的生命攸关医疗设备）的授权许可，除非各方授权官员已经达成了专门管控此类使 用的特别协议。

只有那些 TI 特别注明属于军用等级或“增强型塑料”的 TI 组件才是设计或专门用于军事/航空应用或环境的。购买者认可并同 意，对并非指定面向军事或航空航天用途的 TI 组件进行军事或航空航天方面的应用，其风险由客户单独承担，并且由客户独 力负责满足与此类使用相关的所有法律和法规要求。

TI 已明确指定符合 ISO/TS16949 要求的产品，这些产品主要用于汽车。在任何情况下，因使用非指定产品而无法达到 ISO/TS16949 要求，TI不承担任何责任。

	产品		应用
数字音频	www.ti.com.cn/audio	通信与电信	www.ti.com.cn/telecom
放大器和线性器件	www.ti.com.cn/amplifiers	计算机及周边	www.ti.com.cn/computer
数据转换器	www.ti.com.cn/dataconverters	消费电子	www.ti.com/consumer-apps
DLP® 产品	www.dlp.com	能源	www.ti.com/energy
DSP - 数字信号处理器	www.ti.com.cn/dsp	工业应用	www.ti.com.cn/industrial
时钟和计时器	www.ti.com.cn/clockandtimers	医疗电子	www.ti.com.cn/medical
接口	www.ti.com.cn/interface	安防应用	www.ti.com.cn/security
逻辑	www.ti.com.cn/logic	汽车电子	www.ti.com.cn/automotive
电源管理	www.ti.com.cn/power	视频和影像	www.ti.com.cn/video
微控制器 (MCU)	www.ti.com.cn/microcontrollers		
RFID 系统	www.ti.com.cn/rfidsys		
OMAP应用处理器	www.ti.com/omap		
无线连通性	www.ti.com.cn/wirelessconnectivity	德州仪器在线技术支持社区	www.deyisupport.com

邮寄地址： 上海市浦东新区世纪大道 1568 号，中建大厦 32 楼 邮政编码： 200122