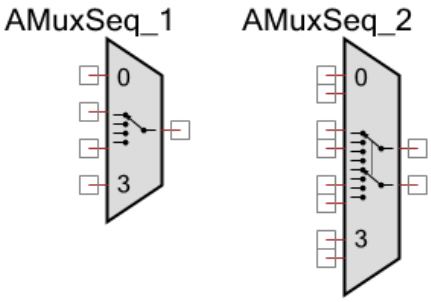
模拟定序复用器 (AMuxSeq)

1.70

特性



- 单端或差分连接
- 可在单个 AMux 的 1 和 64 连接之间调整，以及差分 AMux 的 1 和 32 连接之间调整
- 软件控制
- 连接可以为引脚或内部来源
- 无同时连接
- 双向（无源）

概述

模拟定序复用器 (AMuxSeq) 组件用于通过按连接序列顺序断开和闭合连接的方式一次将一个模拟信号连接到不同的模拟信号。AMuxSeq 主要用于时分复用。

何时使用 AMuxSeq

任何时候需要将多个模拟信号复用到一个源或目标时都可使用 AMuxSeq 组件。由于 AMuxSeq 是无源的，它可用于复用输入或输出信号。

AMuxSeq 拥有的 API 比 AMux 的 API 要更简单、更快捷。当不需要多个同时连接且始终以同一顺序访问信号时，使用 AMuxSeq，而不是 AMux。

输入/输出连接

本节介绍 AMuxSeq 的各种输入和输出连接。I/O 列表中的星号 (*) 表示，在 I/O 说明中列出的情况下，该 I/O 可能不可见。

0-63 – 模拟

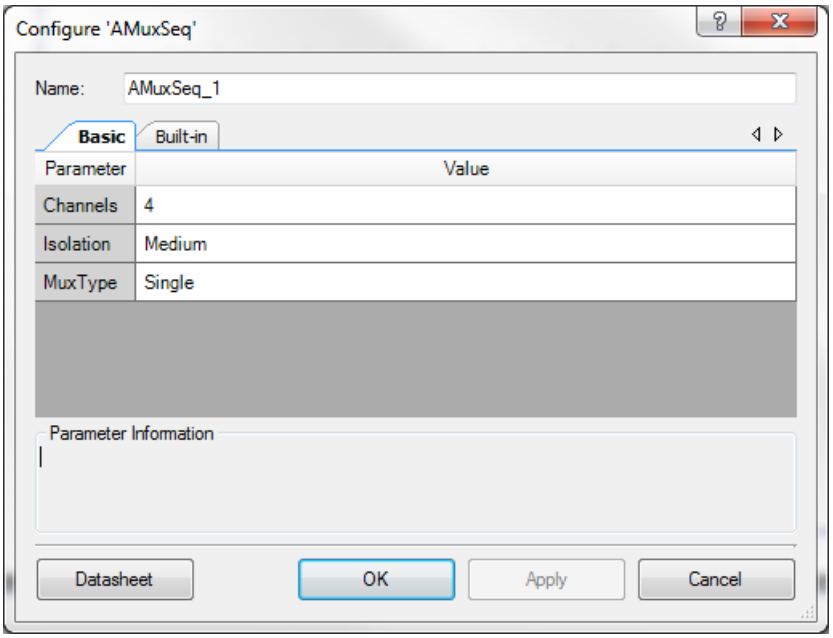
AMuxSeq 可在单个 AMux 的 1 和 64 模拟连接之间切换，以及差分 Amux 的 1 和 32 模拟连接之间切换。当 **MuxType** 参数设置为 **Differential**（差分）时，显示成对连接。

common（公共端） – 模拟

“公共端”信号采用公共端连接；它没有标签。使用 AMuxSeq_Next() 函数选择的可交换连接信号将连接到此端子。当 **MuxType** 参数设置为 **Differential**（差分）时，显示成对信号。

组件参数

将 AMuxSeq 组件拖动到设计中，双击它以打开 **Configure**（配置）对话框。



AMuxSeq 提供下列参数。

通道

此参数用于根据 **MuxType** 选择输入或成对的输入的数量。对单个 AMux 而言，1 和 64 之间的任何值均为有效值，对差分 AMux 而言，1 和 32 之间的任何值均为有效值。

MuxType

此参数用于选择单端输入（**Single**）与差分输入 **Differential**。**当输入信号都是相对于同一参考信号（如 V_{SSA}）时，使用 Single**。当两个或更多信号可能相对于不同的参考源时，选择 **Differential**（差分）选项。“差分”模式最常用于提供差分输入的 ADC。

隔离

此参数用于选择如下一种隔离模式：

- **最低 –**使用单个外部开关 此模式能够确保切换时间最快。
- **中等（默认） –**尝试使用双重切换，即外部开关和仅独特的内部开关。如果没有可用的独特内部开关，则将单独使用外部开关。双重切换方式可提升隔离效果，但是也会增加切换时间。
- **最高 –**使用双重切换方式，即外部开关和潜在的共享内部开关。内部开关不需要具有独特性。参考计数允许共享内部开关。不使用独特的内部开关时，切换时间将进一步受影响。

下图显示了 Amux 的三种可能切换实现（两个开关均未连接、下方开关连接和两个开关均连接）：

图 1. 单切换，无内部开关

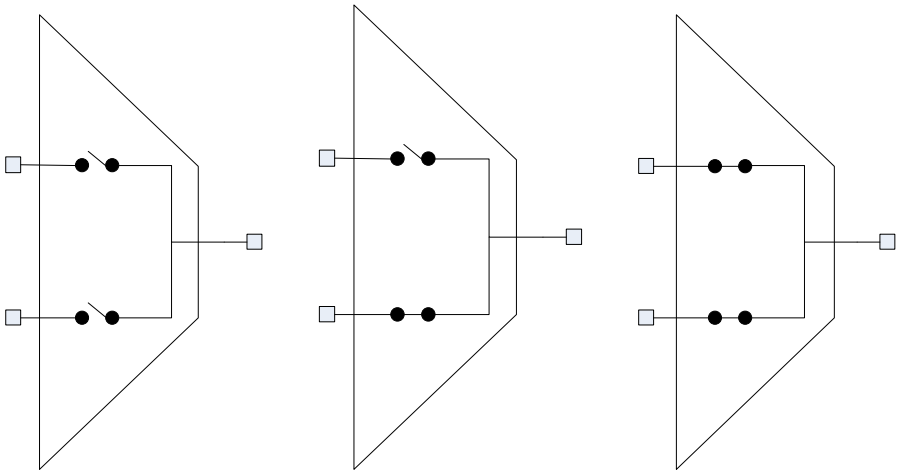


图 2. 双重切换，独特的内部开关

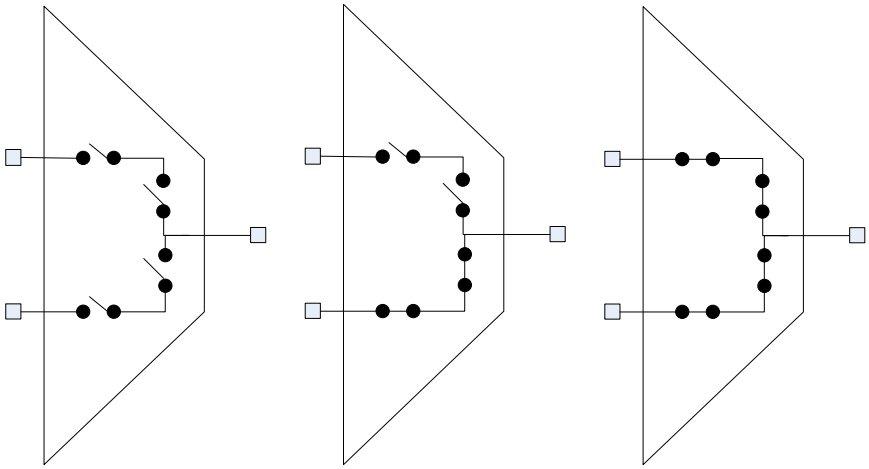
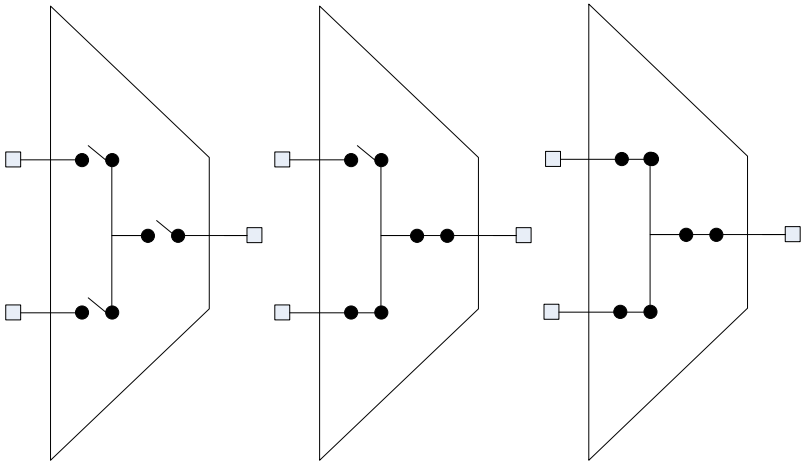


图 3. 动态切换，共享的内部开关



应用程序编程接口

应用程序编程接口 (API) 子程序允许您使用软件配置组件。下表列出了每个函数的接口，并进行了说明。以下各节将更详细地介绍每个函数。

默认情况下，PSoC Creator 将实例名称“AMuxSeq_1”分配给提供的设计中的第一个组件实例。您可以将该实例重命名为符合标识符语法规则的任意唯一值。实例名称会成为每个全局函数名称、变量和常量符号的前缀。为增加可读性，下表中使用了实例名称“AMuxSeq”。

函数	说明
AMuxSeq_Init()	断开所有通道的连接
AMuxSeq_Start()	断开所有通道的连接
AMuxSeq_Stop()	断开所有通道的连接
AMuxSeq_Next()	断开之前通道的连接并按照顺序连接到下一个通道。
AMuxSeq_DisconnectAll()	断开所有通道的连接
AMuxSeq_GetChannel()	返回当前连接的通道。如果没有连接通道，返回 –1。

void AMuxSeq_Init(void)

说明： 断开所有通道的连接。下次调用 AMuxSeq_Next() 时，选择第一个通道。

参数： 无

返回值： 无

副作用： 所有寄存器将复位为其初始值。

void AMuxSeq_Start(void)

说明： 断开所有通道的连接。下次调用 AMuxSeq_Next() 时，选择第一个通道。

参数： 无

返回值： 无

副作用： 无

void AMuxSeq_Stop(void)

说明： 断开所有通道的连接。下次调用 AMuxSeq_Next() 时，选择第一个通道。

参数： 无

返回值： 无

副作用： 无

void AMuxSeq_Next(void)

说明： 断开之前通道的连接并按照顺序连接到下一个通道。第一次调用 AMuxSeq_Next() 时或者在 AMuxSeq_Init(), AMuxSeq_Start()、AMuxSeq_Enable()、AMuxSeq_Stop() 或 AMuxSeq_DisconnectAll() 之后调用此函数时，它将连接通道 0。

参数： 无

返回值： 无

副作用： 无

void AMuxSeq_DisconnectAll(void)

说明： 此函数用于断开所有通道的连接。下次调用 AMuxSeq_Next() 时，选择第一个通道。

参数： 无

返回值： 无

副作用： 无

int8 AMuxSeq_GetChannel(void)

说明： 返回当前连接的通道。如果没有连接通道，返回 –1。

参数： 无

返回值： 当前通道或 –1。

副作用： 无

固件源代码示例

PSoC Creator 在“查找示例项目”对话框中提供了很多包括原理图和代码示例的示例项目。要获取组件特定的示例，请打开组件目录中的对话框或原理图中的组件实例。要获取通用的示例，请打开 Start Page（开始页）或 **File**（文件）菜单中的对话框。根据需要，使用对话框中的 **Filter Options**（筛选选项）可缩小可选项目的列表。

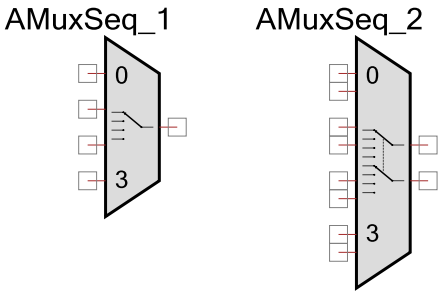
有关更多信息，请参见 PSoC Creator 帮助中的“Find Example Project（查找示例项目）”主题。

功能描述

AMuxSeq 由固件而不是硬件控制。一次只能有一个信号连接到公共端信号。

下面显示了 AMuxSeq 配置为单端和差分的流程。

单端差分



性能

连续模拟复用器由软件控制，因此切换性能取决于提供的 API 的执行时间。性能取决于设计中复用器的精确配置，但是对输入的数量无影响，因为每次调用都是断开一个输入并连接另一个输入。表 1 旨在提供有关切换性能的指南。

所有的性能测量都是在 CPU 频率为 48 MHz 的情况下测得的。性能测量与 CPU 频率几乎呈现线性关系。编译器优化被配置为 PSoC Creator 捆绑的编译器所能提供的最高级别。对于 PSoC 3，Keil 编译器设置为优化级别 5（针对大小或速度）。对于 PSoC 4 和 PSoC 5LP，GNU 编译器设置为针对大小或速度的优化。

表 1. 性能

函数	优化	PSoC 3 (µs)	PSoC 4 (µs)	PSoC 5LP (µs)
Next	大小	8.8	1.8	1.4
	速度	2.4	1.7	1.4

资源

AMuxSeq 使用单独开关将模块和引脚连接到模拟总线。

API 存储器使用

根据编译器、组件、所用 API 数量和组件配置的不同，组件内存使用会出现较大变化。下表提供指定组件配置中可用的 API 的存储器使用。

已利用释放模式中配置的相关编译器进行了测量，大小采用了优化设定。有关特定的设计，可分析编译器生成的映射文件以确定内存使用情况。

配置	PSoC 3 (Keil_PK51)		PSoC 4 (GCC)		PSoC 5LP (GCC)	
	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节
单	24	1	48	1	40	1
差分	38	1	72	1	68	1

MISRA 合规性

尚未根据 MISRA-C:2004 编码准则合规性，验证组件源代码。

直流和交流电气特性

AMuxSeq 将在所有有效供电电压下运行。

组件更改

本节介绍组件与以前版本相比的主要更改。

版本	更改说明	更改/影响原因
1.70a	添加了 PSoC 4 资源使用和性能信息。	
1.70	放宽了 AMux 输入连接的范围。“单个”AMux 的输入范围是 1-64，“差分”AMux 的输入范围是 1-32。	以解决客户的需求。
	添加了 MISRA 合规性章节。尚未根据 MISRA-C:2004 编码准则合规性，验证此组件。	
1.60	添加了隔离参数。更新了屏幕截图。	允许您选择隔离模式，以控制切换时间。
1.50.c	在数据手册中添加了“性能”一节	
1.50.b	对数据手册进行了少量编辑和更新	
1.50.a	对数据手册进行了少量编辑和更新	
1.50	添加了 AMuxSeq_Init() 函数。	为了符合公司标准，并提供 API 以便无需启动组件即可初始化或恢复组件。
1.20.a	向组件中添加了信息，以说明它与芯片修订版的兼容性。	如果组件在不兼容的芯片上使用，该工具将报告错误或警告。如果发生此情况，请更新到支持您的目标组件的修订版。

版本	更改说明	更改/影响原因
1.20	更新了符号图片。	更新的目的是为了符合公司标准并说明排序。
	添加了 AMuxSeq_GetChanne()l API 函数。	用于获取当前连接的通道。
	为没有参数的函数添加了缺少的‘void’。 将 AMux 通道变量的类型从无符号整数更改为了有符号整数，因为 -1 用于说明没有选择通道。	这些更改消除了使用 MDK 和 RVDS 编译器进行编译期间显示的有关过期声明的警告。

赛普拉斯半导体公司，2013-2016 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约由赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件并不附随有一份许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方属人性质的、非独家且不可转让的如下许可（无再许可权）（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码形式的向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和（2）在被软件（由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。

在适用法律允许的限度内，赛普拉斯未对本文件或任何软件作出任何明示或暗示的担保，包括但不限于关于适销性和特定用途的默示保证。赛普拉斯保留更改本文件的权利，届时将不另行通知。在适用法律允许的限度内，赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为供参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何主张、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿赛普拉斯因赛普拉斯产品任何非预期用途产生或相关的所有索赔、费用、损失和其他责任，包括因人身伤害或死亡引起的主张，并使之免受损失。

赛普拉斯、赛普拉斯徽标、Spansion、Spansion徽标，及上述项目的组合，及PSoC、CapSense、EZ-USB、F-RAM和Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产。