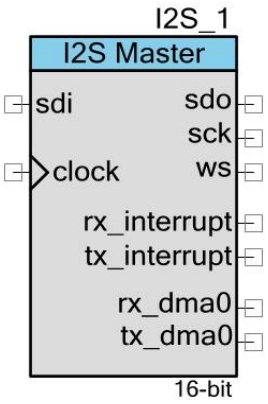
串行数字音频总线 (I2S)

2.40

特性



- 仅用于主控
- 每次采样 8 到 32 数据位
- 16、32、48 或 64 位字选择周期
- 数据速率高达 96 kHz，64 位字选择周期：6.144 MHz
- Tx 和 Rx FIFO 中断
- DMA 支持
- 独立的左声道和右声道 FIFO 或交叉立体声 FIFO
- 独立启用 Rx 和 Tx

概述

串行数字音频总线 (I2S) 是用于将数字音频组件连接在一起的串行总线接口标准。此规范来自于 Philips® Semiconductor（I2S 总线规范；1986 年 2 月，修订时间为 1996 年 6 月 5 日）。

I2S 组件仅在主控模式下运行。它还可在两个方向上运行，作为发射器 (Tx) 和接收器 (Rx)。Tx 和 Rx 的数据是独立的字节流。字节流首先包含最高有效字节，并且第一个字的第 7 位 中存放最高有效位。用于每次采样的字节数（左/右声道的采样）是保持样品所需的最少字节数。

何时使用 I2S

组件为立体声音频数据提供串行总线接口。此接口是音频 ADC 和 DAC 组件最常用的接口。

输入/输出连接

本节介绍 I2S 组件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。

sdi — 输入*

串行数据输入。如果为 **Direction**（方向）参数选择了 Rx 选项，则显示此信息。

如果此信号连接到输入引脚，则应禁用此引脚的“Input Synchronized”（同步输入）选择。此信号应已同步到 SCK，所以，用输入引脚同步器延迟信号会导致信号移入下一个时钟周期中。

时钟 — 输入

提供的时钟频率必须是输出串行时钟 (SCK) 所需时钟频率的两倍。例如，要产生 64 位字选择周期的 48 kHz 音频，时钟频率应为：

2 × 48 kHz × 64 = 6.144 MHz

sdo — 输出*

串行数据输出。如果为 **Direction**（方向）参数选择了 Tx 选项，则显示此信息。

Sck — 输出

输出串行时钟。

ws — 输出

字选择输出指示要传输的通道。

rx_interrupt — 输出*

Rx 方向中断。如果为 **Direction**（方向）参数选择了 Rx 选项，则显示此信息。

tx_interrupt — 输出*

Tx 方向中断。如果为 **Direction**（方向）参数选择了 Tx 选项，则显示此信息。

rx_DMA0 — 输出*

Rx 方向 DMA 请求 FIFO 0（左侧或交错）。如果选择了 **DMA Request**（DMA 请求）参数下的 **Rx DMA**，则显示此信息。

rx_DMA1 — 输出*

Rx 方向 DMA 请求 FIFO 1（右侧）。如果为 Rx 选择了 **DMA Request**（DMA 请求）参数下的 **Rx DMA** 和 **Data Interleaving**（数据交错）参数下的 **Separated L/R**（单独的 L/R），则显示此信息。

tx_DMA0 — 输出*

Tx 方向 DMA 请求 FIFO 0（左侧或交错）。如果选择了 **DMA Request**（DMA 请求）参数下的 **Tx DMA**，则显示此信息。

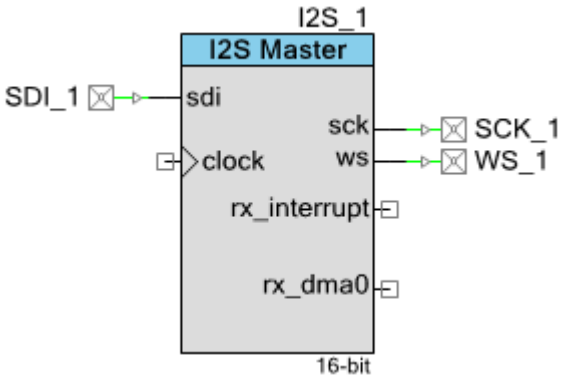
tx_DMA1 — 输出*

Tx 方向 DMA 请求 FIFO 1（右侧）。如果为 Tx 选择了 **DMA Request**（DMA 请求）参数下的 **Tx DMA** 和 **Data Interleaving**（数据交错）参数下的 **Separated L/R**（单独的 L/R），则显示此信息。

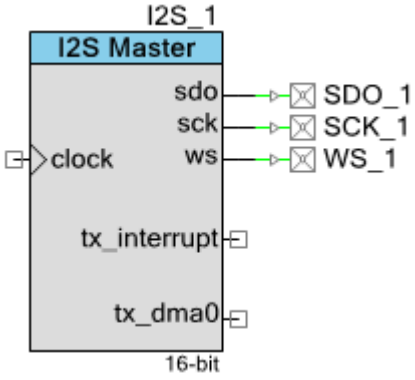
原理图宏信息

默认情况下，PSoC Creator 组件目录为 I2S 组件提供了三个原理图宏实现。这些宏包含已连接到数字引脚组件的 I2S 组件。取消选择了 SDI 引脚上的“Input Synchronized”（输入同步）选项，并关闭了所有引脚的 API 生成。原理图宏使用为仅 Rx、仅 Tx，以及 Rx 和 Tx 方向配置的 I2S 组件，如下图中所示。

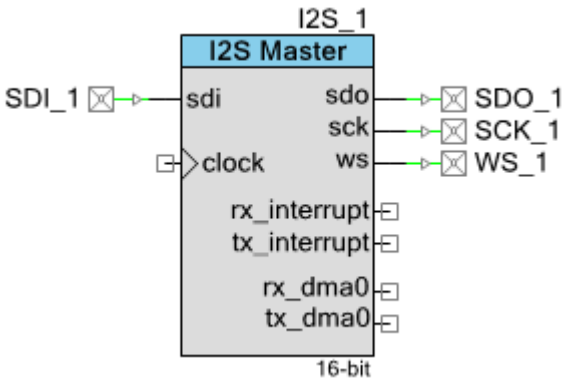
仅 I2S Rx



仅 I2S Tx



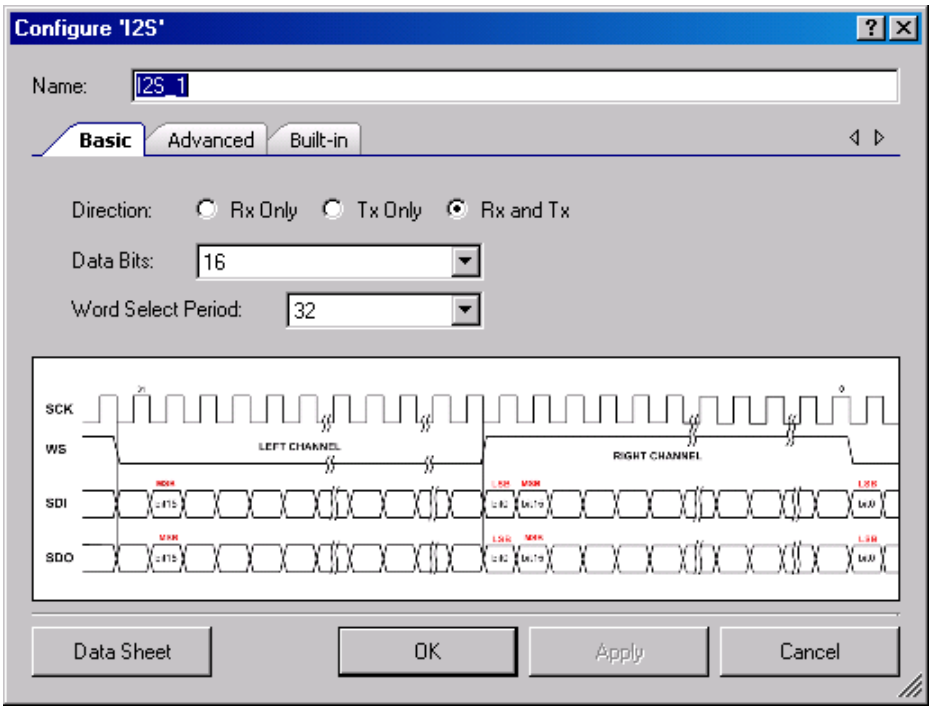
I2S Rx 和 Tx



组件参数

将 I2C 组件拖放到您的设计上，双击它以打开 **Configure**（配置）对话框。该对话框有两个可用于设置 I2S 组件的选项卡。

图 1. “Basic”（基本）选项卡



方向

确定组件运行的方向。此值可设为 **Rx Only**（仅 Rx）、**Tx Only**（仅 Tx）或 **Rx and Tx**（Rx 和 Tx）（默认值）。

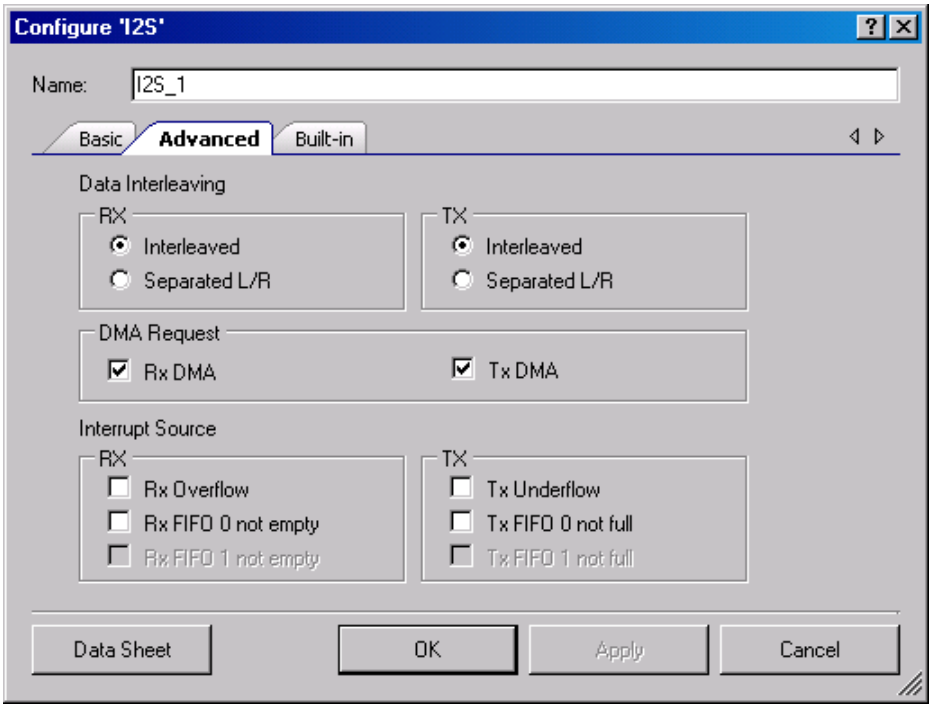
Data Bits（数据位）

确定为各个采样配置的数据位数（编译的硬件）。该值可以设置为 8 到 32。默认设置为 **16**。

Word Select Period（字选择周期）

定义左右声道的完整样品的周期。该值可设置为 **16**、**32**（默认）、**48** 或 **64**。

图 2. Advanced（高级）选项卡



Data Interleaving（数据交错）

用于选择数据为 **Interleaved**（交错）（默认值）还是 **Separate L/R**（单独 L/R）。可单独选择 Rx 和 Tx。

DMA Request（DMA 请求）

用于启用和禁用组件的 DMA 请求信号。可单独选择 Rx 和 Tx。这些选项默认为**启用**。

Interrupt Source（中断源）

用于配置 Rx 和 Tx 方向的中断源。Rx 和 Tx 方向具有单独的中断输出。各个方向的多个源共同进行“或”运算。此软件可随时重新配置这些源；这些参数只定义了初始配置。设置包括：

- Rx：
  - **Rx Overflow（Rx 溢出）**
  - **Rx FIFO 0 not empty**（Rx FIFO 0 非空）（左侧或交错）
  - **Rx FIFO 1 not empty**（Rx FIFO 1 非空）（右侧）— 如果不是交错，则仅有一个选项
- Tx：
  - **Tx Underflow（Tx 下溢）**
  - **Tx FIFO 0 not full**（Tx FIFO 0 未满）
  - **Tx FIFO 1 not full**（Tx FIFO 1 未满）（右侧）ORed 如果不是交错，则仅有一个选项

时钟选择

此组件中没有内部时钟。您必须附加时钟源。提供的时钟频率必须是输出串行时钟 (SCK) 所需时钟频率的两倍。

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“I2S_1”分配给指定设计中组件的第一个实例。可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。出于可读性考虑，下表中使用的实例名称为“I2S”。

函数	说明
I2S_Start()	启动 I2S 接口
I2S_Stop()	禁用 I2S 接口
I2S_EnableTx()	启用 I2S 接口的 Tx 方向
I2S_DisableTx()	禁用 I2S 接口的 Tx 方向
I2S_EnableRx()	启用 I2S 接口的 Rx 方向
I2S_DisableRx()	禁用 I2S 接口的 Rx 方向
I2S_SetRxInterruptMode()	设置 I2S Rx 方向中断的中断源

函数	说明
I2S_SetTxInterruptMode()	设置 I2S Tx 方向中断的中断源
I2S_ReadRxStatus()	返回 I2S Rx 状态寄存器中的状态
I2S_ReadTxStatus()	返回 I2S Tx 状态寄存器中的状态
I2S_ReadByte()	返回 Rx FIFO 中的一个字节
I2S_WriteByte()	将一个字节写入 Tx FIFO
I2S_ClearRxFIFO()	清除 Rx FIFO
I2S_ClearTxFIFO()	清除 Tx FIFO
I2S_Sleep()	保存配置和禁用 I2S 接口
I2S_Wakeup()	恢复配置和启用 I2S 接口
I2S_Init()	启用 I2S 接口
I2S_Enable()	初始化或恢复默认 I2S 配置
I2S_SaveConfig()	保存 I2S 接口配置
I2S_RestoreConfig()	恢复 I2S 接口配置

全局变量

变量	说明
I2S_initVar	表示是否已初始化 I2S。变量将初始化为 0，并在第一次调用 I2S_Start() 时设置为 1。这使得组件无需在第一次调用 I2S_Start() 子程序后重新初始化便可重新启动。 如果需要重新对组件进行初始化，在调用 I2S_Start() 之前先调用 I2S_Init()。或可调用 I2S_Init() 和 I2S_Enable() 函数重新对 I2S 进行初始化。

void I2S_Start(void)

说明： 启动 I2S 接口。根据需要，此函数启用活动模式电源模板位或门控时钟。启动生成 sck 和 ws 输出。Tx 和 Rx 方向保持禁用状态。

参数： 无

返回值： 无

副作用： 无

void I2S_Stop(void)

说明： 禁用 I2S 接口。根据需要，禁用活动模式电源模板位或门控时钟。sck 和 ws 输出设为 0。禁用 Tx 和 Rx 方向，并清除其 FIFO。

参数： 无

返回值： 无

副作用： 无

void I2S_EnableTx(void)

说明： 启用 I2S 接口的 Tx 方向。在下一个字选择下降沿开始传输。

参数： 无

返回值： 无

副作用： 无

void I2S_DisableTx(void)

说明： 禁用 I2S 接口的 Tx 方向。数据传输停止，在下一个字选择下降沿传输常量 0 值。

参数： 无

返回值： 无

副作用： 无

void I2S_EnableRx(void)

说明： 启用 I2S 接口的 Rx 方向。在下一个字选择下降沿开始接收数据。

参数： 无

返回值： 无

副作用： 无

void I2S_DisableRx(void)

说明： 禁用 I2S 接口的 Rx 方向。在下一个字选择下降沿，不再将数据接收信息发送到接收 FIFO。

参数： 无

返回值： 无

副作用： 无

void I2S_SetRxInterruptMode(uint8 interruptSource)

说明： 设置 I2S Rx 方向中断的中断源。多个源可能均是“或”运算。

参数： uint 8：包含选定中断源的常量的字节

I2S Rx 中断源	值	类型
RX_FIFO_OVERFLOW	0x01	读取时清除
RX_FIFO_0_NOT_EMPTY	0x02	透明
RX_FIFO_1_NOT_EMPTY	0x04	透明

返回值： 无

副作用： 如果将读取后清除位作为中断生成的源，则 rx_interrupt 输出将保持设置，直至读取 I2S Rx 状态寄存器。

void I2S_SetTxInterruptMode(uint8 interruptSource)

说明： 设置 I2S Tx 方向中断的中断源。多个源可能均是“或”运算。

参数： uint 8：包含选定中断源的常量的字节

I2S Tx 中断源	值	类型
TX_FIFO_UNDERFLOW	0x01	读取时清除
TX_FIFO_0_NOT_FULL	0x02	透明
TX_FIFO_1_NOT_FULL	0x04	透明

返回值： 无

副作用： 如果将读取后清除位作为中断生成的源，则 tx_interrupt 输出将保持设置，直至读取 I2S Tx 状态寄存器。

uint8 I2S_ReadRxStatus(void)

说明： 返回 I2S Rx 状态寄存器的状态。

参数： 无

返回值： uint 8：I2S Rx 状态寄存器的状态

I2S RX 状态掩码	值	类型
RX_FIFO_OVERFLOW	0x01	读取时清除
RX_FIFO_0_NOT_EMPTY	0x02	透明
RX_FIFO_1_NOT_EMPTY	0x04	透明

副作用： 清除处于清除读取状态的 I2S Rx 状态寄存器位。

uint8 I2S_ReadTxStatus(void)

说明： 返回 I2S Tx 状态寄存器的状态。

参数： 无

返回值： uint 8：I2S Tx 状态寄存器的状态

I2S Tx 状态掩码	值	类型
TX_FIFO_UNDERFLOW	0x01	读取时清除
TX_FIFO_0_NOT_FULL	0x02	透明
TX_FIFO_1_NOT_FULL	0x04	透明

副作用： 清除处于清除读取状态的 I2S Rx 状态寄存器位。

uint8 I2S_ReadByte(uint8 wordSelect)

说明： 返回 Rx FIFO 中的单字节。在进行此调用之前检查 Rx 状态，以确认 Rx FIFO 非空。

参数： uint 8：指示从左 (0) 或右 (1) 通道读取。在交错模式中，忽略此参数。

返回值： uint 8：包含已接收数据的字节

副作用： 无

void I2S_WriteByte(uint8 wrData, uint8 wordSelect)

说明： 将单字节写入 Tx FIFO。在进行此调用之前检查 Tx 状态，以确认 Tx FIFO 未满。

参数： uint8 wrData：包含要传输数据的字节

uint8 wordSelect：指示写入到左 (0) 或右 (1) 通道。在交错模式中，忽略此参数

返回值： 无

副作用： 无

void I2S_ClearRxFIFO(void)

说明： 清除 Rx FIFO。FIFO 中的任何数据都将丢失。仅在禁用了 Rx 方向时才可调用此函数。

参数： 无

返回值： 无

副作用： 无

void I2S_ClearTxFIFO(void)

说明： 清除 Tx FIFO。FIFO 中的任何数据都将丢失。仅在禁用了 Tx 方向时才可调用此函数。

参数： 无

返回值： 无

副作用： 无

void I2S_Sleep(void)

说明： 这是准备组件睡眠的首选子程序。I2S_Sleep() 子程序保存当前组件的状态。然后，它调用 I2S_Stop() 函数，并调用 I2S_SaveConfig() 以保存硬件配置。根据需要，禁用活动模式电源模板位或门控时钟。sck 和 ws 输出设为 0。禁用 Tx 和 Rx 方向。

在调用 CyPmSleep() 或 CyPmHibernate() 函数之前调用 I2S_Sleep() 函数。有关电源管理函数的更多信息，请参考 PSoC Creator *System Reference Guide*（《系统参考指南》）。

参数： 无

返回值： 无

副作用： 无

void I2S_Wakeup(void)

说明： 恢复 I2S 配置和非保留寄存器值。根据需要，启用活动模式电源模板位或门控时钟。启动生成 sck 和 ws 输出。根据 Rx 和 Tx 方向在睡眠之前的状态，启用 Rx 和/或 Tx 方向。

参数： 无

返回值： 无

副作用： 调用 I2S_Wakeup() 函数前未调用 I2S_Sleep() 或 I2S_SaveConfig() 函数可能会产生意外行为。

void I2S_Init(void)

说明： 初始化或恢复随定制器提供的默认 I2S 配置，该配置定义组件的中断源。

参数： 无

返回值： 无

副作用： 仅恢复中断生成的掩码寄存器。它不从 FIFO 中清除数据，也不复位组件硬件状态机制。

void I2S_Enable(void)

说明： 激活硬件并开始执行组件操作。无需调用 I2S_Enable()，因为 I2S_Start() 子程序会调用该函数，这是开始组件操作的首选方法。

参数： 无

返回值： 无

副作用： 无

void I2S_SaveConfig(void)

说明： 此函数会保存组件配置和非保留寄存器。它还保存 Configure（配置）对话框中定义的或通过相应 API 修改的当前组件参数值。该函数由 I2S_Sleep() 函数调用。

参数： 无

返回值： 无

副作用： 无

void I2S_RestoreConfig(void)

说明： 此函数会恢复组件配置和非保留寄存器。它还将组件参数值恢复到调用 I2S_Sleep() 函数之前的值。此子程序由 I2S_Wakeup() 调用，用于在组件退出睡眠状态时恢复组件。

参数： 无

返回值： 无

副作用： 必须仅在 I2S_SaveConfig() 子程序之后才可调用。否则，将用组件的初始配置覆盖组件配置。

MISRA 合规性

本节介绍了本组件与 MISRA-C:2004 的合规和偏差情况。定义了两种类型的偏差：

- 项目偏差 - 适用于所有 PSoC Creator 组件的偏差
- 特定偏差 - 仅适用于此组件的偏差

本节提供了有关组件特定偏差的信息。*系统参考指南*的“MISRA 合规性”章节中介绍项目偏差以及有关 MISRA 合规性验证环境的信息。

I2S 组件没有任何特定偏差。

固件源代码示例

PSoC Creator 在“查找示例项目”对话框中提供了大量包括原理图和代码的例子项目。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

左/右和 Rx/Tx 配置

左右通道的配置（Rx 和 Tx 方向、位数和字选择周期）是相同的。如果应用程序必须有不同的 Rx 和 Tx 配置，则使用两个单向组件实例。

数据流格式

Tx 和 Rx 的数据是独立的字节流。字节流首先包含最高有效字节，并且第一个字的第 7 位 中存放最高有效位。用于各次采样的字节数（用于左/右声道）是保持采样所需的最少字节数。任何未使用的位在 Tx 上将被忽略，在 Rx 上为 0。

一个方向的数据流可以是单字节流，或是双字节流。在一个字节流的情况下，左通道和右通道与采样互相交错，左通道后面紧接右通道。对于双字节流，左右通道字节流使用单独的 FIFO。

DMA

I2S 接口是连续接口，需要不中断的数据流。对于大部分应用，这需要使用 DMA 传输，以避免 Tx 方向的下溢或 Rx 方向的溢出。

I2S 在各个方向最多可驱动两个 DMA 组件。使用 DMA 向导按如下所示配置 DMA 操作：

DMA 向导中 DMA 源/目标的名称	方向	DMA 请求信号	DMA 请求类型	说明
I2S_RX_FIFO_0_PTR	源	rx_dma0	电平	接收左通道或交错通道的 FIFO
I2S_RX_FIFO_1_PTR	源	rx_dma1	电平	接收右通道的 FIFO
I2S_TX_FIFO_0_PTR	目标	tx_dma0	电平	传输左通道或交错通道的 FIFO
I2S_TX_FIFO_1_PTR	目标	tx_dma1	电平	传输右通道的 FIFO

在所有情况下，DMA 请求信号上的高电平信号表示可以传输一个附加字节。

启用系统

Rx 和 Tx 方向是单独启用的。未启用时，Tx 方向传输所有 0 值，而 Rx 方向忽略所有接收数据。启用状态的进入和退出发生在字选择边界时，从而始终发送或接收左/右样品对。

错误处理

组件的两种错误情况发生条件：1）发送 FIFO 为空，且发生后续读取（发送下溢）；2）接收 FIFO 已满，且发生后续写入（接收溢出）。

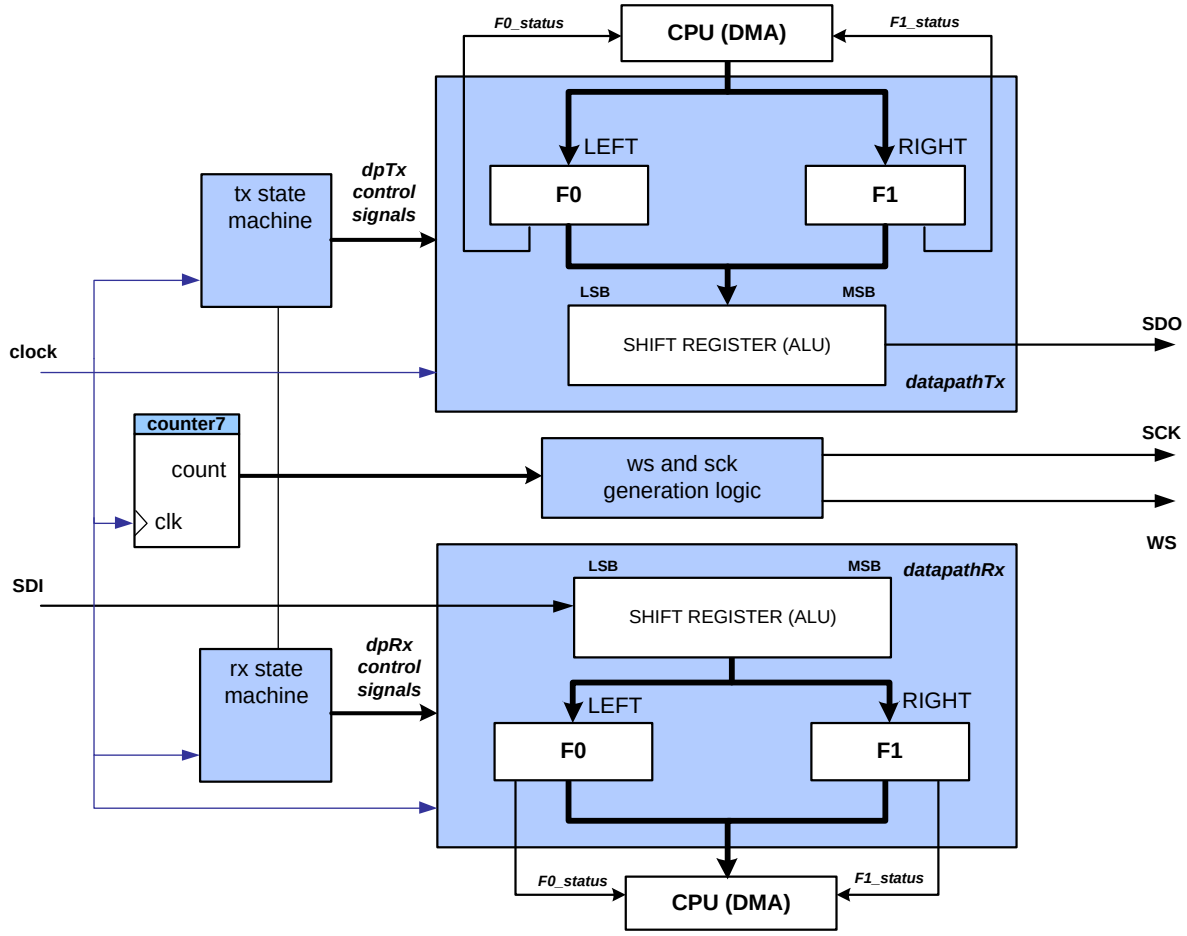
启用了传输时，如果发送 FIFO 变为空，且数据不可用于传输（发送下溢），组件会强制进行 0 的常量传输。再次启动传输前，必须禁用传输，并不应清除 FIFO，必须缓冲传输数据，然后重新启用传输。CPU 可使用发送状态位 I2S_TX_FIFO_UNDERFLOW 监控此下溢情况。此外，还可以针对此错误条件配置中断。

启用接收之后，如果接收 FIFO 变为满，且收到其他数据（接收溢出），组件会停止捕获数据。在此开始接收之前，必须禁用接收，再清除 FIFO，然后重新启用接收。CPU 可使用接收状态位 I2S_RX_FIFO_OVERFLOW 监控此溢出情况。此外，还可以针对此错误条件配置中断。

框图和配置

I2S 作为一组已配置 UDB 进行实施。在以下框图中显示该实现。

图 3. I2S 实现



寄存器

I2S_CONTROL_REG

位	7	6	5	4	3	2	1	0
值	保留					使能	rxenable（rx 启用）	Txenable

- 启用：启用/禁用 I2S 组件
- rxenable（rx 启用）、txenable（tx 启用）：分别启用/禁用 Rx 和 Tx 方向

I2S_TX_STATUS_REG

位	7	6	5	4	3	2	1	0
值	保留					F1_not_full	F0_not_full	下溢出

- F1_not_full：如果设置了此参数，Tx FIFO 1 未满
- F0_not_full：如果设置了此参数，Tx FIFO 0 未满
- underflow（下溢）：如果设置了此参数，Tx FIFO 下溢事件已发生

可用 I2S_ReadTxStatus() API 函数读取寄存器值。

I2S_RX_STATUS_REG

位	7	6	5	4	3	2	1	0
值	保留					F1_not_empty	F0_not_empty	上溢出

- F1_not_empty：如果设置了此参数，Rx FIFO 1 非空
- F0_not_empty：如果设置了此参数，Rx FIFO 0 非空
- overflow（溢出）：如果设置了此参数，Rx FIFO 溢出事件已发生

可用 I2S_ReadRxStatus() API 函数读取寄存器值。

注意：Tx 状态寄存器的位 0（下溢）和 Rx 状态寄存器的位 0（溢出）均配置为读取后清除位。在此模式下，在状态寄存器时钟的每个周期完成后，会对输入状态进行采样。当输入为高电平时，设置寄存器位，并保持设置，而无需考虑输入的后续状态。在后续读取时，通过 CPU 清除寄存器位。

资源

I2S 组件放置在整个 UDB 阵列中。该组件利用以下资源。

配置	资源类型					
	数据路径单元	宏单元	状态单元	控制单元	DMA 通道	中断
仅 Rx	1	7	1	2	–	–
仅 Tx	1	8	1	2	–	–
Rx 和 Tx	2	13	2	4	–	–

API 存储器使用

组件使用情况显著不同，取决于编译器、设备、使用 API 的数量以及组件配置。下表提供指定组件配置中可用的 API 的存储器使用。

此次测量使用的编译器配置是针对代码大小优化的 Release（发布）模式 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
仅 Rx	220	3	384	5	392	5
仅 Tx	220	3	384	5	392	5
Rx 和 Tx	326	3	570	5	592	5

直流和交流电气特性

除非另有说明，否则这些规范的适用条件是 -40 °C ≤ T_A ≤ 85 °C 且 T_J ≤ 100 °C。除非另有说明，否则这些规范的适用范围为 1.71 V 到 5.5 V。

直流特性

参数	说明	最小值	典型值[1]	最大值	单位[2]
$I_{DD(RX)}$	组件电流消耗（仅 Rx）				
	空闲状态电流[3]	–	15	–	µA/MHz
	工作电流[4]	–	21	–	µA/MHz
$I_{DD(TX)}$	组件电流消耗（仅 Tx）				
	空闲状态电流[3]	–	15	–	µA/MHz
	工作电流[4]	–	20	–	µA/MHz

1. 未包括设备 IO 和时钟分配的电流。这些值是在 25 °C 时的值。
2. 电流消耗根组件的输入时钟相关。
3. 启用组件后，即使不传输/接收数据，组件也消耗电流。
4. 启用组件并传输/接收数据时，组件会消耗电流。

参数	说明	最小值	典型值[1]	最大值	单位[2]
$I_{DD(RX_TX)}$	组件电流消耗（Rx 和 Tx）				
	空闲状态电流[3]	–	16	–	µA/MHz
	工作电流[4]	–	26	–	µA/MHz

交流特性

参数	说明	最小值	典型值	最大值	单位
f_{SCK}	串行时钟（输出）频率	–	–	6.144	MHz
f_{CLOCK}[5]	组件时钟频率	–	2 × f_{SCK}	–	MHz
t_{SCKH}	SCK 高电平时间	–	0.5	–	1/f_{SCLK}
t_{SCKL}	SCK 低电平时间	–	0.5	–	1/f_{SCLK}
t_{SCK_WS}	延迟时间、SCK 下降沿到 WS 有效的时间	–20	–	20	ns
t_{SCK_SDO}	延迟时间、SCK 下降沿到 SDO 有效的时间	–20	–	20	ns
t_{S_SDI}[6]	SDI 设置时间	25	–	–	ns

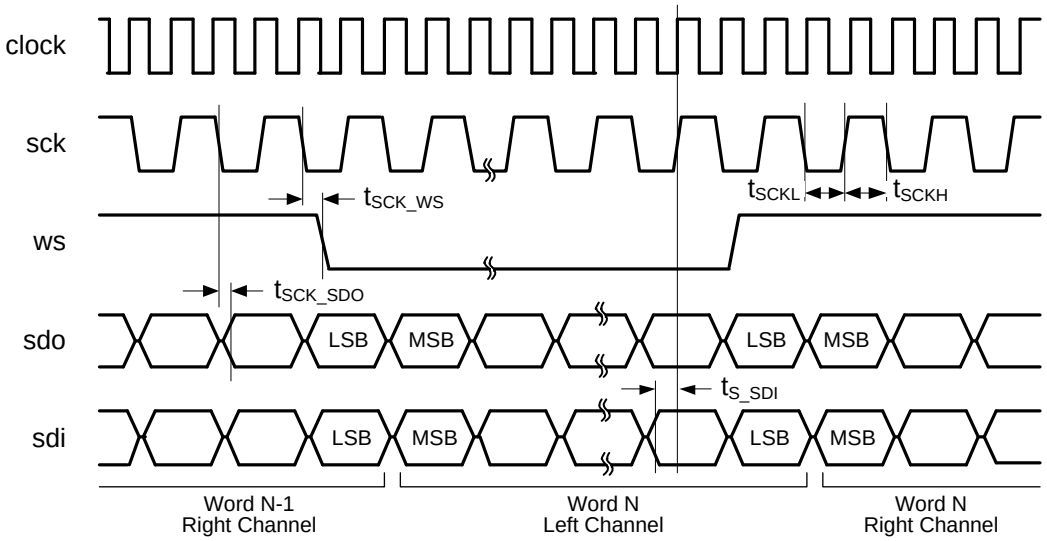
常见音频采样频率的组件时钟频率

采样频率	组件时钟频率 (f_{CLOCK}) MHz			
	t_{WS} = 16 位	t_{WS} = 32 位	t_{WS} = 48 位	t_{WS} = 64 位
8 kHz	0.2560	0.5120	0.7680	1.0240
16 kHz	0.5120	1.0240	1.5360	2.0480
32 kHz	1.0240	2.0480	3.0720	4.0960
44.1 kHz	1.4112	2.8224	4.2336	5.6448
48 kHz	1.5360	3.0720	4.6080	6.1440
88.2 kHz	2.8224	5.6448	8.4672	11.2896
96 kHz	3.0720	6.1440	9.2160	12.2880
192 kHz	6.1440	12.2880	无	无

5. 组件的最大组件时钟频率派生自 t_{CLK_SCK} 与 SCK 输出和 SDI 输入的路由路径延迟的组合（本文中稍后将描述）。典型值提供了此组件的最大安全工作频率。可以在更高的时钟频率运行此组件，在该频率将需要使用 STA 结果验证时序要求

6. V_{DDIO} 电源电压范围为 3.0 V 至 5.5 V。对于 PSoC 5 该值可能有所不同，因此，必须使用 STA 结果进行验证。

图 4. 数据转换时序图

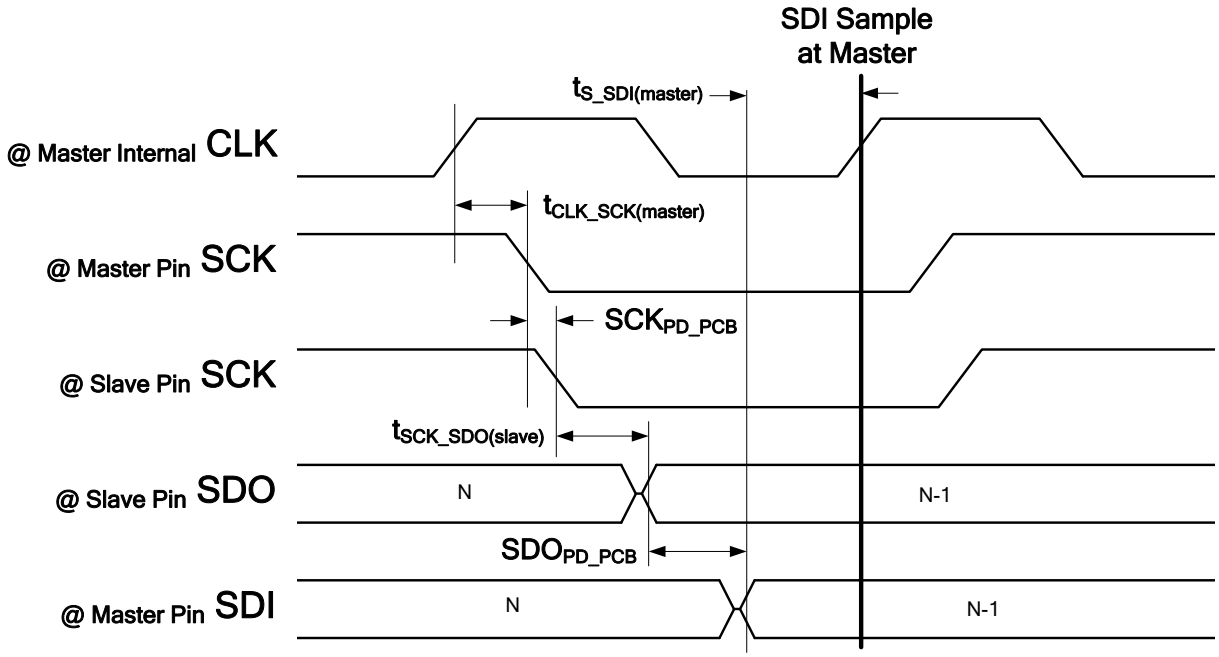


注意：所有组件内部逻辑完全从输入 2 倍时钟运行。这导致对于此内部时钟，需要参考一些时序限制，例如 t_{S_SDI} 和 t_{CLK_SCK}（本文中稍后将描述）。

如何将 STA 结果用于特性数据

f_{SCK} STA 中不直接提供 SCK 的最大频率（或最大比特率）。不过，STA 结果中提供的数据指示了某些内部逻辑时序限制。要计算最大比特率，必须考虑若干因素。需要板卡布局和从组件通信组件规范，才能完全了解最大值。此参数的限制因素是主控引脚的 SCK 下降沿到从组件的往复路径延迟，以及从组件的 SDO 输出返回到主控的 SDI 输入的路径延迟。在此情况下，组件必须使用下列等式满足主控的 SDI 设置时间。

图 5. 计算最大 fSCK 频率



在此情况下，请使用下列等式计算 f_{SCK} 频率：

$$f_{SCK} < 1 \div [2 \times [t_{RT_PD} + t_{CLK_SCK(master)} + t_{S_SDI(master)}]]$$

其中：

$$t_{RT_PD} = [SCK_{PD_PCB} + t_{SCK_SDO(slave)} + SDO_{PD_PCB}]$$

并且，

SCK_{PD_PCB} 是主控组件引脚到从组件引脚的 SCK 的 PCB 路径延迟。

$t_{SCK_SDO(slave)}$ 必须源自从组件数据表

$t_{CLK_SCK(master)}$ 是主控组件的内部 CLK 到 SCK 引脚的路径延迟。这是在 STA 结果时钟输出部分中提供的，如下所示：

- Clock To Output Section

- CLK

Source	Destination	Delay (ns)
\I2S:BitCounter\/count_0	SCK(0)_PAD	24.484
Net_5/q	SDO(0)_PAD	23.808
Net_4/q	WS(0)_PAD	22.958

$t_{S_SDI(master)}$ 是主控组件的 SDI 引脚到内部逻辑的路径延迟。这是在 STA 结果输出时钟部分中提供的，如下所示：

- Input To Clock Section

- CLK

Source	Destination	Delay (ns)
SDI(0)_PAD	\I2S:Rx:dpRx:u0\/route_si	19.977

提供 SCK 的最大频率的最后等式如下；最大比特率为：

$$f_{SCK}\,(Max.) = 1 \div [2 \times [t_{CLK_SCK(master)} + SCK_{PD_PCB} + t_{SCK_SDO(slave)} + SDO_{PD_PCB} + t_{S_SDI(master)}]]$$

f_{CLOCK} 在时序结果的时钟总览里，名称为外部时钟(此例中的CLK)一栏中提供了最大组件时钟频率。下面是 STA 报告中的内部时钟限制示例：

+ Clock Summary Section

Clock	Type	Nominal Frequency (MHz)	Required Frequency (MHz)	Maximum Frequency (MHz)	Violation
BUS_CLK	Sync	24.000	24.000	N/A	
CLK	Sync	6.000	6.000	65.398	
ClockBlock/clk_bus	Async	24.000	24.000	N/A	
ClockBlock/dclk_0	Async	6.000	6.000	N/A	
ILO	Async	0.001	0.001	N/A	
IMO	Async	3.000	3.000	N/A	
MASTER_CLK	Sync	24.000	24.000	N/A	
PLL_OUT	Async	24.000	24.000	N/A	

t_{SCKH} I2S 组件生成 50% 占空比 SCK

t_{SCKL} I2S 组件生成 50% 占空比 SCK

t_{SCK_WS} SCK 下降沿与 WS 有效之间的延迟。可将此值作为 WS 引脚和 SCK 引脚的时钟输出时间之间的差异进行计算。从 STA 报告中提取的数据示例如下：

- Clock To Output Section

- CLK

Source	Destination	Delay (ns)
\I2S:BitCounter\/count_0	SCK(0)_PAD	24.484
Net_5/q	SDO(0)_PAD	23.808
Net_4/q	WS(0)_PAD	22.958

t_{SCK_SDO} SCK 下降沿与 WS 有效之间的延迟。可将此值作为 SDO 引脚和 SCK 引脚的时钟输出时间之间的差异进行计算。这些值是在 STA 结果时钟输出时间中提供的，如下所示：

- Clock To Output Section

- CLK

Source	Destination	Delay (ns)
\I2S:BitCounter\/count_0	SCK(0)_PAD	24.484
Net_5/q	SDO(0)_PAD	23.808
Net_4/q	WS(0)_PAD	22.958

t_{S_SDI} SDI 设置时间是主控组件的 SDI 引脚到内部逻辑的路径延迟。这是在 STA 结果输入时钟部分中提供的，如下所示：

- Input To Clock Section

- CLK

Source	Destination	Delay (ns)
SDI(0)_PAD	\I2S:Rx:dpRx:u0\/route_si	19.977

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
2.40.a	更新了数据表，添加了 PSoC 4 的内存使用情况。	
2.40	已添加 MISRA 合规性章节。	此组件没有任何特定偏差。
2.30	向.cyre 文件中包括的所有组件 API 添加了 CYREENTRANT 关键词。	并非所有 API 都是真正可重入的。组件 API 源文件中的注释指出了适用的函数。 需要此更改为采用安全方式使用并且不是可重入函数消除编译器警告：通过标志或关键节防止并发调用。
	添加了 PSoC 5LP 支持	
	向数据表中添加了直流特性章节。	
2.20	已添加组件的内部 FIFO 错误检测。	在出现溢出/下溢错误时，提高组件的错误处理功能。
2.10	将 FIFO 模块状态信号重新采样到 DP 时钟中。	允许组件针对所有 PSoC 3 和 PSoC 5 芯片使用相同的时序结果。
	向数据表中添加了特性数据	
	对数据表进行了少量编辑和更新	
2.0	已更改此组件的硬件实现，在时钟输入上要求 2 倍频率信号。将通过输入时钟除以 2 产生 SCK 输出信号。	提高对 SCK、WS、SDO 和 SDI 信号之间的时序关系的控制。
	更新 I2S_Start() 函数以匹配实施中的更改。未更改功能。	需要对初始化过程进行更改以简化实施。
	添加了睡眠模式 API。	用于支持低功耗模式。
	已将状态位 tx_not_full 和 rx_not_emty 从读取清除模式更改为透明模式。	FIFO 中任何“满”或“空”状态的状态位都需为透明，以仅表示 FIFO 的当前实时状态。

版本	更改说明	更改/影响原因
	向组件中添加了 DMA 功能文件。	此文件允许 PSoC Creator 中的 DMA 向导工具支持 I2S。
	已更改 I2S_Stop() API，以在禁用组件之后清除 rx 和 tx FIFO。	将 Tx 和 Rx FIFO 状态复位为初始值。防止重新启用组件后出现意外操作。
	添加了 Keil 功能重新进入支持。	添加此功能，以便使客户能够指定各个生成函数可重入。

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。