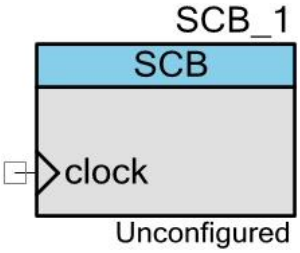
PSoC 4 串行通信模块 (SCB)

1.0



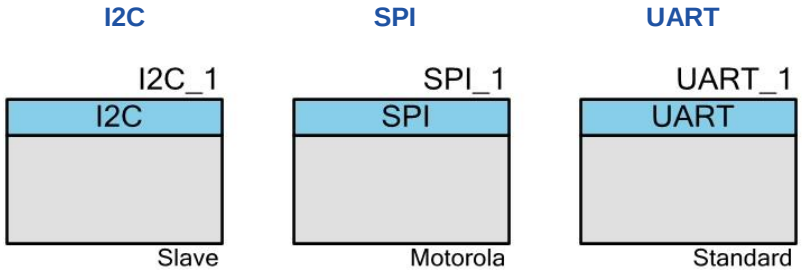
特性

- 预配置组件：
  - 工业标准 NXP® I²C 总线接口
  - 标准 SPI 主设备与从设备功能，采用 Motorola、Texas Instruments 和 National Semiconductor 的 Microwire 协议
  - 标准 UART TX 与 RX 功能，带智能卡读卡器与 IrDA 协议
  - 针对 SPI 与 I²C 协议的低功率（深度睡眠）运行模式（使用外部时钟）
- 运行时间自定义功能
- I²C 引导加载程序功能

概述

PSoC 4 SCB 组件为一种多功能硬件模块，能实现以下组件。在 PSoC Creator 组件目录中，各个组件作为预配置示意宏，该目录标记为 “SCB 模式”

点击以下链接之一，跳转到相应的章节：



在组件目录中还有未配置的 SCB 组件条目。

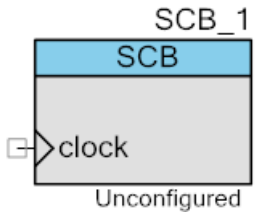
何时使用 SCB 组件

SCB 能在预配置模式中使用：I2C、SPI 和 UART。另外，SCB 可在创建时不予以配置，在运行时可采用应用程序编程接口 (API) 的设置值配置为任意模式。所有配置均可在运行时进行设定。

预配置模式是一种典型应用实例。最简便的方法是按需要的运行模式配置 SCB。未配置方法可用于创建设计，该设计可用于多种应用中；当创建 PSoC Creator 硬件设计时，设计中 SCB 的具体用途是未知的。

未配置的 SCB

SCB 可在运行时进行配置，以便在任何模式下运行，从未配置模式中选择任何模式。



输入/输出连接

本部分描述了 SCB 组件用的各种各样的输入/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。

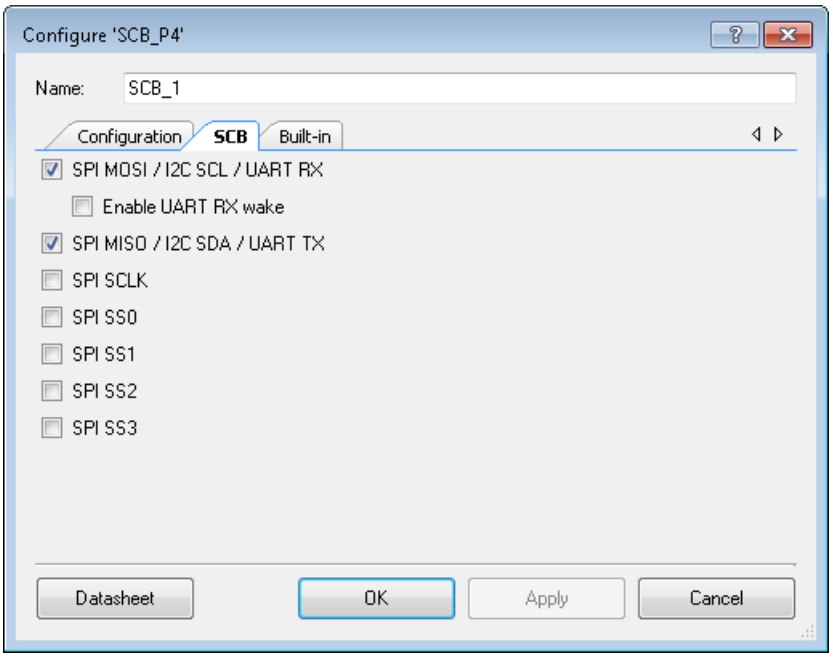
时钟－输入

时钟负责运行此模块。终端通常处于未配置模式中。为其它模式提供了该选项，以便使用与终端相连的内时钟或外时钟。

组件内部埋有特定接口的引脚，因为这些引脚使用的是专用连接，而且不可作为通用信号。欲获得更多信息，请见芯片《技术参考手册》 (TRM) 的“ I/O 系统”部分。

未配置的 SCB 参数

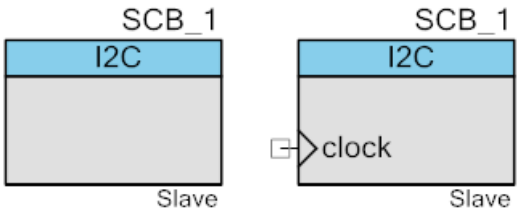
将 SCB 组件拖放到您的设计上，双击它可打开 **配置** 对话框。



SCB 标签选项可以选择引脚，该引脚将被配置接口所使用。接口类型与引脚名称列于复选框内，以启用符号上的终端。

Enable UART RX wake 向 RX 引脚添加中断，以实现 UART 唤醒功能。 当放置了该 RX 引脚时，该选项限制对来自端口的任何其它引脚中断进行处理。

I2C



I²C 总线是 Philips 基于行业标准开发的两线制接口。主设备会启动 I²C 总线上的所有通信，并为所有从装置提供时钟。在一个单板或小系统上连接多个设备时，I²C 是一种理想解决方案。

此组件支持 I²C 从设备、主设备、多主设备与多主从设备配置。

此组件支持速度达 1000 kbps 的标准时钟。它与 I²C 标准、快速和超快速模式的设备兼容，装置见 NXP I²C-总线规范[1]。此组件与其他第三方从设备与主设备相兼容。

输入/输出连接

本部分描述了 SCB 组件各种各样的输入/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。

时钟 – 输入*

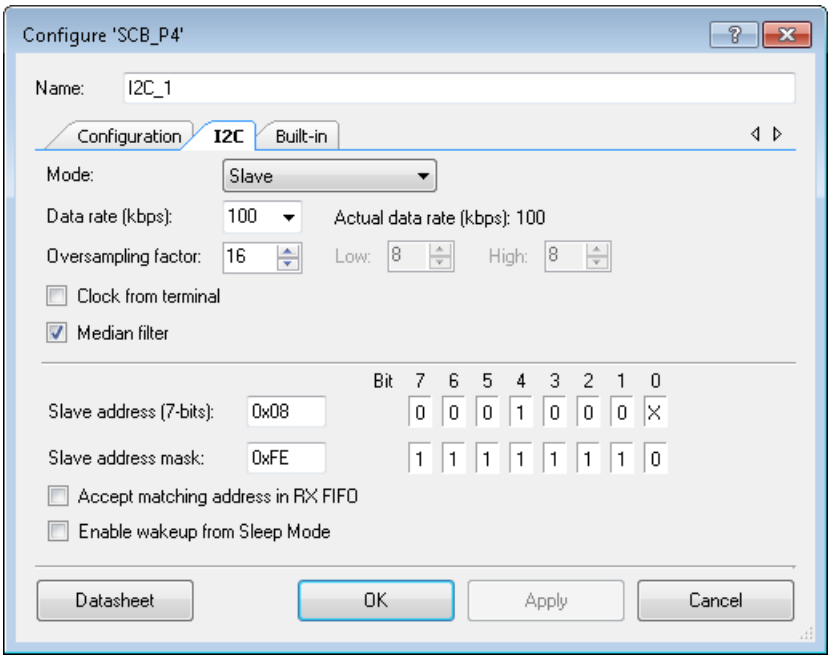
时钟负责运行此模块。根据**终端时钟**参数的不同，终端的存在位将各不相同。

组件内部有接口的特定引脚，因为这些引脚使用专用连接，不作为通用信号连接。欲获得更多信息，请见芯片《技术参考手册》 (TRM) 的“I/O 系统”部分。

[1] 请参见 *I²C-总线规范* （2012 年 2 月修订版 4） 此规范刊登于 www.nxp.com NXP 网站上。

I2C 参数

将 SCB 组件拖放到您的设计上，双击它可打开 **配置** 对话框。



I2C 标签具有下列参数：

模式

此选项确定所支持的模式：从设备、主设备、多主设备或多主从设备。

- **从设备**－仅限从设备操作（默认）
- **主设备**－仅限主设备操作
- **多主设备**－支持总线上多个主设备设备
- **多主从设备**－从设备与多主设备同步操作

数据速率

此参数用于设置 I²C 数据速率值（可达 1000 kbps）；实际速度可能会基于可用的时钟速度和分频器范围而有所不同。标准数据速率为 50、100（默认值）、400 和 1000 kbps。如果设置了**终端时钟**，可忽略**数据速率**参数；输入时钟与 **过采样因子**决定实际数据速率。

实际数据速率

实际数据速率显示了将用当前设置运行的组件上的数据速率。所选数据速率可能和实际数据速率不同。影响实际数据速率计算的因子为：过采样因子、HFCLK 时钟和内部组件时钟的精度。

过采样因子

此参数定义了 I²C SCL 时钟的过采样因子：一个 I²C SCL 时钟内组件时钟的数量。过采样因子由高低过采样因子组成。过采样因子的最大值为 32；最小值取决于中值滤波选项。默认值为 16。

在**从设备**模式中运行时，不使用**过采样因子**（不设置任何寄存器）。仅用于计算要求的时钟频率，以达到过采样量。

低

此参数定义了 I²C SCL 时钟的低过采样因子：一个低周期 I²C SCL 时钟内组件时钟的数量。默认值为 8。

高

此参数定义了 I²C SCL 时钟的低过采样因子：一个高周期 I²C SCL 时钟内组件时钟的数量。默认值为 8。

终端时钟

终端时钟参数允许在内部配置时钟和外部配置时钟之间进行选择，使用外部配置时钟时，数据率取决于外部时钟。启用该选项时，组件不对数据速率进行控制但会基于用户连接的时钟源显示实际数据速率。考虑到过采样的问题，PSoC Creator 会基于**数据速率**这个参数对要求的时钟频率进行计算和配置。

注意：当设置数据率或外部时钟频率值时，请确保 PSoC Creator 可以用当前系统时钟频率提供此值。否则，创建项目时会生成时钟精度范围警告。警告中包含 PSoC Creator 设置的实际时钟值。选择是应当更改系统时钟或组件时钟以满足时钟设置系统要求并达到最佳值。

中值滤波器

该参数 I2C SDA 输入路径上实现 3 抽头中值滤波器。该滤波器可增强信号路径的抗扰度。然而，过采样因子最小值将增加。

从设备地址（7 位）

这是从设备将识别的 I²C 地址。如果不选择从设备操作，则会忽略此参数。您可以选择介于 0 至 127 之间的从设备地址，默认值为 **8**。此地址为右对齐的 7 位从设备地址，它不包括读/写位。

您可以输入十进制或十六进制的值；对于十六进制数值，请在地址之前键入 “0x”。也为您提供了二进制输入格式。

从设备地址屏蔽

地址匹配程序中，此参数用于屏蔽从设备地址的位。地址屏蔽中的 0 位对应读/写方向位，在地址匹配中并不重要。

- 该比特值为 0 －该位不进行地址对比。
- 该比特值为 1 －该位须与 I²C 从设备地址的相应位相互匹配。

例如：从设备地址为 0x36 且从设备地址掩码为 0xDE（地址匹配中，0 位为读/写位，5 位的设置不重要）。匹配的从设备地址为：0x36 和 0x26。

您可以输入十进制或十六进制的值；对于十六进制数值，请在地址之前键入 “0x”。也为您提供了二进制输入格式。

接受 RX FIFO 中的匹配地址

此参数确定是否接受 RX FIFO 中的匹配 I²C 从设备地址。

使能“睡眠模式”唤醒

此选项会在地址匹配时从睡眠模式唤醒系统。接收匹配地址后，I²C 时钟将延长。I²C 从设备执行时钟延长，直至唤醒设备并接收地址。当**模式**为从设备或多主从设备时，此选项方有效。

必须启用 I²C，以便在从设备地址匹配时可唤醒设备，同时切换至睡眠模式。可调用 SCB_Sleep() 应用程序编程接口来完成这一操作；也可参考《系统参考指南》的“电源管理应用程序编程接口”一节。

I2C 应用程序编程接口 (API)

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了每个函数。

默认情况下，PSoC Creator 将实例名称 “SCB_1” 分配给指定设计组件的第一个实例。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。出于可读性考虑，下表中使用的实例名称为 “SCB”。

函数	说明
SCB_Init()	根据自定义程序中的定义参数初始化 SCB 组件。
SCB_Enable()	使能 SCB 组件。
SCB_Start()	启动 SCB 组件。
SCB_Stop()	禁用 SCB 组件。
SCB_Sleep()	准备 SCB 组件进入深度睡眠模式。

函数	说明
SCB_Wakeup()	使 SCB 组件退出深度睡眠模式。
SCB_I2CInit()	配置 SCB 组件，以在 I2C 模式中运行。
SCB_I2CSlaveStatus()	返回从设备状态标志。
SCB_I2CSlaveClearReadStatus()	返回读取状态标志并清除从设备读取状态标志。
SCB_I2CSlaveClearWriteStatus()	返回写状态并清除从设备写状态标志。
SCB_I2CSlaveSetAddress()	设置从设备地址，该值介于 0 至 127 （0x00 至 0x7F） 之间。
SCB_I2CSlaveSetAddressMask()	设置从设备地址掩码，该值介于 0 至 254（0x00 至 0xFE）之间。
SCB_I2CSlaveInitReadBuf()	设置从设备接收数据缓冲区（主设备 <- 从设备）。
SCB_I2CSlaveInitWriteBuf()	设置从设备写缓冲区（主设备 -> 从设备）。
SCB_I2CSlaveGetReadBufSize()	由于调用了 SCB_I2CSlaveClearReadBuf()，因此返回主设备读取的字节数量。
SCB_I2CSlaveGetWriteBufSize()	由于调用了 SCB_I2CSlaveClearWriteBuf()，因此返回主设备所写的字节数量。
SCB_I2CSlaveClearReadBuf()	读取缓冲区计数器重新设为零。
SCB_I2CSlaveClearWriteBuf()	写缓冲区计数器重新设为零。
SCB_I2CMasterStatus()	返回主设备状态。
SCB_I2CMasterClearStatus()	返回主设备状态并清除状态标志。
SCB_I2CMasterWriteBuf()	将引用数据缓冲区写入到指定的从设备地址。
SCB_I2CMasterReadBuf()	从指定的从设备地址读取数据，并将数据放入引用的缓冲区中。
SCB_I2CMasterSendStart()	生成开始条件并发送指定的从设备地址。
SCB_I2CMasterSendRestart()	生成重新开始条件并发送指定的从设备地址。
SCB_I2CMasterSendStop()	生成停止条件。
SCB_I2CMasterWriteByte()	写出单字节。这是一个手动命令，它只应与 SCB_I2CMasterSendStart() 或 SCB_I2CMasterSendRestart() 函数一同使用。
SCB_I2CMasterReadByte()	读取单字节。这是一个手动命令，它只应与 SCB_I2CMasterSendStart() 或 SCB_I2CMasterSendRestart() 函数一同使用。
SCB_I2CMasterGetReadBufSize()	由于调用了 SCB_I2CMasterClearReadBuf()，因此返回读取的字节数量。
SCB_I2CMasterGetWriteBufSize()	由于调用了 SCB_I2CMasterClearWriteBuf()，因此返回所写的字节数量。
SCB_I2CMasterClearReadBuf()	将读取缓冲区指针复位到缓冲区的起点。
SCB_I2CMasterClearWriteBuf()	将写缓冲区指针复位到缓冲区的起点。

全局变量

正常运行情况下，不需获得这些变量。

变量	说明
SCB_initVar	SCB_initVar 表明 SCB 组件是否完成了初始化。该变量初始化为 0，并在第一次调用 SCB_Start() 时设置为 1。这样，第一次调用 SCB_Start() 子程序后，组件不用重新初始化即可重启。 如需重新初始化组件，可在 SCB_Start() 或 SCB_Enable() 函数前调用 SCB_Init() 函数。

I2C 函数应用

函数	从设备	主设备	多主设备	多主从设备
SCB_I2CInit()	+	+	+	+
SCB_I2CSlaveStatus()	+	–	–	+
SCB_I2CSlaveClearReadStatus()	+	–	–	+
SCB_I2CSlaveClearWriteStatus()	+	–	–	+
SCB_I2CSlaveSetAddress()	+	–	–	+
SCB_I2CSlaveSetAddressMask()	+	–	–	+
SCB_I2CSlaveInitReadBuf()	+	–	–	+
SCB_I2CSlaveInitWriteBuf()	+	–	–	+
SCB_I2CSlaveGetReadBufSize()	+	–	–	+
SCB_I2CSlaveGetWriteBufSize()	+	–	–	+
SCB_I2CSlaveClearReadBuf()	+	–	–	+
SCB_I2CSlaveClearWriteBuf()	+	–	–	+
SCB_I2CMasterStatus()	–	+	+	+
SCB_I2CMasterClearStatus()	–	+	+	+
SCB_I2CMasterWriteBuf()	–	+	+	+
SCB_I2CMasterReadBuf()	–	+	+	+
SCB_I2CMasterSendStart()	–	+	+	+
SCB_I2CMasterSendRestart()	–	+	+	+
SCB_I2CMasterSendStop()	–	+	+	+
SCB_I2CMasterWriteByte()	–	+	+	+
SCB_I2CMasterReadByte()	–	+	+	+
SCB_I2CMasterGetReadBufSize()	–	+	+	+
SCB_I2CMasterGetWriteBufSize()	–	+	+	+

函数	从设备	主设备	多主设备	多主从设备
SCB_I2CMasterClearReadBuf()	–	+	+	+
SCB_I2CMasterClearWriteBuf()	–	+	+	+

void SCB_Init(void)

说明： 使 SCB 组件初始化，以便在所选的配置之一中运行：I2C、SPI 或 UART

设置为 “Unconfigured SCB” 时，该函数不进行任何初始化操作。

参数： 无

返回值： 无

副作用： 无

void SCB_Enable(void)

说明： 使能 SCB 组件。组件启动时，不应更改 SCB 配置。禁用组件后，方可更改配置。

设置为 “Unconfigured SCB” 时，该函数不能启动组件。组件必须进行初始化，以便在之前配置的下列模式中运行：I2C、SPI 或 UART。

参数： 无

返回值： 无

副作用： 无

void SCB_Start(void)

说明： 调用 SCB_Init() 与 SCB_Enable()。调用该函数后，组件使能且操作准备就绪。

设置为 “Unconfigured SCB” 时，该函数不能启动组件。组件必须进行初始化，以便在之前配置的下列模式中运行：I2C、SPI 或 UART。

参数： 无

返回值： 无

副作用： 无

void SCB_Stop(void)

说明： 禁用 SCB 组件。

参数： 无

返回值： 无

副作用： 无

void SCB_Sleep(void)

说明： 为组件进入深度睡眠做准备。

“从睡眠模式使能唤醒” 的选择对此函数的实现有影响。

在调用 CyPmSysDeepSleep() 函数之前调用 SCB_Sleep() 函数。进入睡眠模式前不得调用此函数。有关电源管理函数的更多信息，请参考《系统参考指南》。

参数： 无

返回值： 无

副作用： 无

void SCB_Wakeup(void)

说明： 为组件退出深度睡眠做准备。

“从睡眠模式使能唤醒” 的选择对此函数的实现有影响。

进入睡眠模式前不得调用此函数。

参数： 无

返回值： 无

副作用： 在调用 SCB_Sleep() 函数之前，调用 SCB_Wakeup() 函数可能会导致意外行为。

void SCB_ I2CInit(SCB_I2C_INIT_STRUCT *config)

说明： 配置 SCB 以运行 I2C。

本函数**专为**当自定义程序中的SCB配置设为 “未配置 SCB” 时使用。在 I2C 模式中初始化 SCB 后，组件可用 SCB_Start() 或者 SCB_Enable() 函数启动。

本函数使用了一个结构指针来提供设这。该结构含有同样的信息，也可以通过定制化设置提供。

参数： config：结构指针含有如下字段排序列表。这些字段匹配自定义程序的选项。设置的详细说明见自定义程序

字段	说明
mode	I2C 运行模式。以下定义为可用选择： • SCB_I2C_MODE_SLAVE • SCB_I2C_MODE_MASTER • SCB_I2C_MODE_MULTI_MASTER • SCB_I2C_MODE_MULTI_MASTER_SLAVE
oversampleLow	I2C 时钟低相过采样因子。从设备模式操作时忽略。过采样因子必须连同时钟率一起选择，以便生成所需的 I2C 运行速率。
oversampleHigh	I2C 时钟高相过采样因子。从设备模式操作时忽略。
enableMedianFilter	0 – 禁用 1 – 启用
slaveAddr	7 位从设备地址。非子时钟模式忽略。
slaveAddrMask	8 位从设备地址掩码。0 位的值必须为 0。非子时钟模式忽略。
acceptAddr	0 – 禁用 1 – 启用 启用后，Rx FIFO 接收匹配地址。
enableWake	0 – 禁用 1 – 启用 非从设备模式忽略。

返回值： 无

副作用： 无

uint32 SCB_I2CSlaveStatus(void)

说明： 返回从设备通信状态。

参数： 无

返回值： uint32：I²C 从设备的当前状态。

本状态包含读取和写入的状态常量。每个常量是一个位字段值。返回值可能包含多个位，用于指示读取或写入传输的状态。

从设备状态常数	说明
SCB_I2C_SSTAT_RD_CMPLT	从设备读取传输完成。置位表示主设备通过发送一个 NAK[2] 显示完成读取。
SCB_I2C_SSTAT_RD_BUSY	从设备读取传输进行中。主设备向从设备发送一个读取命令时设置，RD_CMPT置位时清除。
SCB_I2C_SSTAT_RD_OVFL	主设备已尝试读取超出缓冲区外的字节。
SCB_I2C_SSTAT_RD_ERR	读取传输过程中从设备在总线上发现错误。错误来源为：开始或停止的条件错误或 SDA 驱动时仲裁丢失。
SCB_I2C_SSTAT_WR_CMPLT	从设备写传输完成。收到一个停止或重新开始条件时设置。
SCB_I2C_SSTAT_WR_BUSY	从设备写入传输进行中。置位表示主设备向从设备发送一个写入命令，将 WR_CMPT 置位可以将该位清零。
SCB_I2C_SSTAT_WR_OVFL	主设备尝试写入的内容已超出缓冲区末尾。
SCB_I2C_SSTAT_WR_ERR	写入传输过程中从设备在总线上发现错误。错误来源为：开始或停止的条件错误或 SDA 驱动时仲裁丢失。

副作用： 无

uint32 SCB_I2CSlaveClearReadStatus(void)

说明： 清除读取状态标志并返回其值。其他状态标志不会受影响。

参数： 无

返回值： uint32：从设备的当前读取状态。常量见 SCB_I2CSlaveStatus() 函数。

副作用： 本次 API 调用不清除 SCB_I2C_SSTAT_RD_BUSY。

[2] NAK 为 negative acknowledgment（否定确认）或 not acknowledged（未确认）的缩写。I²C 文件通常用 NACK 来表示，网络的其它位置则用 NAK。二者意思相同。

uint32 SCB_I2CSlaveClearWriteStatus(void)

说明： 清除写入状态标志并返回其值。其他状态标志不会受影响。

参数： 无

返回值： uint32：从设备的当前写入状态。常量见 SCB_I2CSlaveStatus() 函数。

副作用： 本次 API 调用不清除 SCB_I2C_SSTAT_WR_BUSY。

void SCB_I2CSlaveSetAddress(uint32 address)

说明： 设置 I²C 从设备地址

参数： uint32 address： I²C slave address. 此地址为右对齐的 7 位从设备地址，它不包括读/写位。

本地址的值未就其是否违反 I2C 规范加以检查。推荐地址为8至120之间 (0x08 至 0x78)。

返回值： 无

副作用： 无

void SCB_I2CSlaveSetAddressMask(uint32 addressMask)

说明： 设置 I²C 从设备地址

参数： uint32 addressMask：I²C 从设备地址掩码。

位值为 0 －该位不进行地址对比。

位值为 1－该位须与 I2C 从设备地址的相应位相互匹配。

该值可为 0 至 254 之间任意值 (0x00 至 0xFE)。地址的 LSB 为 R/W 位，无论 addressMask 位为何值该位均忽略。

返回值： 无

副作用： 无

void SCB_I2CSlaveInitReadBuf(uint8 * rdBuf, uint32 bufSize)

说明： 设置缓冲区指针并设置读取缓冲区的大小。此函数还会复位 SCB_I2CSlaveGetReadBufSize() 函数所返回的传输计数。

参数： uint8* rdBuf：指向主设备读取的数据缓冲区的指针。

uint32 bufSize：I²C 主设备可读取的缓冲区的大小。

返回值： 无

副作用： 此函数在总线数据操作期间被调用时，可传输前一个缓冲区位置和当前缓冲开始时的数据。

void SCB_I2CSlaveInitWriteBuf(uint8 * wrBuf, uint32 bufSize)

说明： 设置缓冲区指针并设置写入缓冲区的大小。此函数还能重置由 SCB_I2CSlaveGetWriteBufSize() 函数返回的传输次数。

参数： uint8* wrBuf：指向主设备写入数据的数据缓冲区的指针.

uint32 bufSize：I²C 主设备可写入数据的缓冲区的大小。

返回值： 无

副作用： 此函数在总线数据操作期间被调用时，可接收前一个缓冲区和当前缓冲位置的数据。

uint32 SCB_I2CSlaveGetReadBufSize(void)

说明： SCB_I2CSlaveInitReadBuf() 或 SCB_I2CSlaveClearReadBuf() 函数被调用之后，返回由 I²C 主设备读取的字节数。

最大返回值为读取缓冲区的大小。

参数： 无

返回值： uint32：主设备读取的字节。传输完成前返回值为零。

副作用： 无

uint32 SCB_I2CSlaveGetWriteBufSize(void)

说明： SCB_I2CSlaveInitReadBuf() 或 SCB_I2CSlaveClearReadBuf() 函数被调用之后，返回由 I²C 主设备写入的字节数。

最大返回值为写入缓冲区的大小。

参数： 无

返回值： uint32：主设备写入的字节。如果传输尚未完成，它将返回截至目前已传输的字节数。

副作用： 无

void SCB_I2CSlaveClearReadBuf(void)

说明： 将读取指针复位至读取缓冲区中的第一个字节。主设备读取的下一个字节将成为读取缓冲区的第一个字节。

参数： 无

返回值： 无

副作用： 无

void SCB_I2CSlaveClearWriteBuf(void)

说明： 将写入指针重置为写入缓冲区中的第一个字节。主设备写入的下一个字节将成为写入缓冲区的第一个字节

参数： 无

返回值： 无

副作用： 无

uint32 SCB_I2CMasterStatus(void)

说明： 返回主设备的通信状态。

参数： 无

返回值： uint32：I²C 主设备的当前状态。这个状态包含状态常量。每个常量是一个位字段值。返回值可能包含多个位，用于指示读取或写入传输的状态。

主设备状态常量	说明
SCB_I2C_MSTAT_RD_CMPLT	读取传输完成。 必须检查错误条件位，确保读取传输成功。
SCB_I2C_MSTAT_WR_CMPLT	写入传输完成。 必须检查错误条件位，确保写入传输成功。
SCB_I2C_MSTAT_XFER_INP	传输正在进行。
SCB_I2C_MSTAT_XFER_HALT	传输已停止。I²C 总线等待重启或停止条件生成。
SCB_I2C_MSTAT_ERR_SHORT_XFER	**错误条件**：在传输完所有字节之前写入传输已结束。
SCB_I2C_MSTAT_ERR_ADDR_NAK	**错误条件**：从设备没有确认地址。
SCB_I2C_MSTAT_ERR_ARB_LOST	**错误条件**：主设备在与从设备进行通信时仲裁失败。

SCB_I2C_MSTAT_ERR_ABORT_XFER	**错误条件**：主设备执行“开始”条件生成时，从设备由另一台主设备寻址。主设备自动切换到从设备模式并相应。主设备数据操作没有发生
SCB_I2C_MSTAT_ERR_BUS_ERROR	**错误条件**：总线错误在主设备传输时发生因为开始或停止的条件错误。
SCB_I2C_MSTAT_ERR_XFER	**错误条件**：这是上述所有错误条件的逻辑或的值。

副作用： 无

uint32 SCB_I2CMasterClearStatus(void)

说明： 清除所有状态标志并返回主设备状态。

参数： 无

返回值： uint32：主设备当前状态。有关常量的信息，请参见SCB_I2CMasterStatus() 函数。

副作用： 无

uint32 SCB_I2CMasterWriteBuf(uint32 slaveAddress, uint8 * wrData, uint32 cnt, uint32 mode)

说明： 自动将整个缓冲数据写入到从设备组件中。一旦此函数启用了数据传输，包含的中断服务子程序将处理进一步数据传输。

启用 I²C 中断。

参数： uint32 slaveAddress：右对齐的 7 位从设备地址（有效范围为 8 至 120）。

uint8 wrData：指向发送数据的缓冲区的指针。

uint32 cnt：缓冲区发送的位数。

uint32 mode：传输模式用于定义：（1）传输开始时是否生成了开始或重新开始条件，（2）总线生成停止条件之前，传输是否完成或停止。

传输模式和模式常量可以一起做逻辑或运算。

模式常量	说明
SCB_I2C_MODE_COMPLETE_XFER	执行从开始到停止的完整传输过程。
SCB_I2C_MODE_REPEAT_START	发送“重复开始”，而非“开始”。
SCB_I2C_MODE_NO_STOP	在没有“停止”操作下执行传输。

返回值： **uint32**：错误状态。有关常量的信息，请参见 SCB_I2CMasterSendStart() 函数。

副作用： 此函数清除 SCB_ I2C_MSTAT_WR_CMPLT 状态。

uint32 SCB_I2CMasterReadBuf(uint32 slaveAddress, uint8 * rdData, uint32 cnt, uint32 mode)

说明： 自动读取从设备中的整个缓冲数据。一旦此函数启用了数据传输，包含的中断服务子程序将处理进一步数据传输。

启用 I²C 中断。

参数： uint32 slaveAddress：右对齐的 7 位从设备地址（有效范围为 8 至 120）。

uint8 rdData：用于放置从设备数据的缓冲区的指针。

uint32 cnt：要读取的缓冲区字节数。

uint32 mode：传输模式用于定义：

(1) 在传输开始时是生成开始条件还是重新开始条件

(2) 在总线上产生停止条件之前是完成还是停止该传输。

在传输模式下，可对所有模式常量执行“或”运算。有关常量的信息，请参见 SCB_I2CMasterWriteBuf() 函数。

返回值： uint32：错误状态。有关常量的信息，请参见 SCB_I2CMasterSendStart() 函数。

副作用： 此函数可清除 SCB_ I2C_MSTAT_RD_CMPLT 的状态。

uint32 SCB_I2CMasterSendStart(uint32 slaveAddress, uint32 bitRnW)

说明： 生成开始条件，并发送带读/写位的从设备地址。

禁用 I²C 中断。

参数： uint32 slaveAddress：右对齐的 7 位从设备地址（有效范围为 8 至 120）。

uint32 bitRnW：值为零时发送写入命令，值为非零时则发送读取命令。

返回值： uint32：错误状态。

模式常量	说明
SCB_I2C_MSTR_NO_ERROR	函数完整、无误。
SCB_I2C_MSTR_BUS_BUSY	总线繁忙。未生成开始条件。
SCB_I2C_MSTR_NOT_READY	主设备不是总线上的活动主设备。 可能正在执行从设备操作。
SCB_I2C_MSTR_ERR_LB_NAK	**错误条件**：最后一个字节未被确认。
SCB_I2C_MSTR_ERR_ARB_LOST	**错误条件**：主设备仲裁失败。
SCB_I2C_MSTR_ERR_BUS_ERR	**错误条件**：主设备出现总线错误。总线被错误设置为开始检测或停止检测。
SCB_I2C_MSTR_ERR_ABORT_START	**错误条件**：由于从设备操作启动导致开始条件中止生成。 （该状态仅在多主设备从设备模式下有效。）

副作用： 准备生成开始条件后，该功能被禁用。它存在于发送开始条件和地址且已收到 ACK/NACK 响应或发生某一错误后。

uint32 SCB_MasterSendRestart(uint32 slaveAddress, uint32 bitRnW)

说明： 生成重新开始条件，并发送带读/写位的从设备地址。

调用此函数之前必须生成有效的开始或重新开始条件。如果在调用此函数之前开始或重新开始条件失败，则此函数将不执行任何操作。

参数： uint32 slaveAddress：右对齐的 7 位从设备地址（有效范围为 8 至 120）。

uint32 bitRnW：值为零时发送写入命令，值为非零时则发送读取命令。

返回值： uint32：错误状态。有关常量的信息，请参见 SCB_I2CMasterSendStart() 函数。

副作用： 准备生成开始条件后，该功能被禁用。它存在于发送开始条件和地址且已收到 ACK/NACK 响应或发生某一错误后。

uint32 SCB_I2CMasterSendStop(void)

说明： 在总线上生成 I²C 停止条件。

如果已生成带读取方向的开始或重新开始条件，则至少须读取一字节。

调用此函数之前必须生成有效的开始或重新开始条件。如果在调用此函数之前开始或重新开始条件失败，则此函数将不执行任何操作。

参数： 无

返回值： uint32：错误状态。有关常量的信息，请参见 SCB_MasterSendStart() 命令。

副作用： 此函数正在执行阻止操作，且在生成停止条件或发生错误前不会退出。

uint32 SCB_I2CMasterWriteByte(uint32 theByte)

说明： 向从设备发送一个字节。

调用此函数之前必须生成有效的开始或重新开始条件。如果在调用此函数之前开始或重新开始条件失败，则此函数将不执行任何操作。

参数： uint32 theByte：发送至从设备的数据字节。

返回值： uint32：错误状态。

模式常量	说明
SCB_I2C_MSTR_NO_ERROR	函数完整、无误。
SCB_I2C_MSTR_NOT_READY	主设备不是总线上的活动主设备。 可能正在执行从设备操作。
SCB_I2C_MSTR_ERR_LB_NAK	**错误条件**：最后一个字节未被确认。
SCB_I2C_MSTR_ERR_ARB_LOST	**错误条件**：主设备仲裁失败。
SCB_I2C_MSTR_ERR_BUS_ERR	**错误条件**：主设备出现总线错误。总线错误被错认为开始检测或停止检测。

副作用： 此函数正在执行阻止操作，且在字节传输完成或发生错误之前不会退出。

uint32 SCB_I2CMasterReadByte(uint32 acknNak)

说明： 向从设备中读取一个字节，并确认或否认该传输。

调用此函数之前必须生成有效的开始或重新开始条件。如果在调用此函数之前开始或重新开始条件失败，则此函数将不执行任何操作并返回一个零值。

参数： uint32 acknNak：值为零时发送 NAK，值为非零时则发送 ACK。

返回值： uint32：从从设备中读取的字节。如果发生错误，返回数据的最高有效位将置为 “1”。

副作用： 此函数正在执行阻止操作，且在收到字节或发生错误之前不会退出。

uint32 SCB_I2CMasterGetReadBufSize(void)

说明： 返回通过 SCB_I2CMasterReadBuf() 函数传输的字节数。

参数： 无

返回值： uint32：传输的字节数。如果传输尚未完成，它将返回截至目前已传输的字节数。

副作用： 无

uint32 SCB_I2CMasterGetWriteBufSize(void)

说明： 返回通过 SCB_I2CMasterWriteBuf() 函数传输的字节数。

参数： 无

返回值： uint32：传输的字节数。如果传输尚未完成，则在传输结束前返回零。

副作用： 无

void SCB_I2CMasterClearReadBufSize(void)

说明： 将读取缓冲区指针重置为缓冲区中的首字节。

参数： 无

返回值： 无

副作用： 无

void SCB_I2CMasterClearWriteBufSize(void)

说明： 将写入缓冲区指针重置为缓冲区中的首字节。

参数： 无

返回值： 无

副作用： 无

引导加载程序支持

SCB 组件可用作引导加载程序的通信组件。SCB 仅在 I2C 模式下方可用作引导加载程序。使用以下配置为外部系统与引导加载程序之间的通信协议提供支持：

- 配置：I2C
- I2C 模式：从设备或多主设备从设备

- 数据速率：必须与主机（引导设备）数据速率保持一致。
- 从设备地址：必须与所选主机（引导设备）的从设备地址保持一致。

有关引导加载程序的更多信息，请参考《系统参考指南》。

SCB 组件为使用引导加载程序提供了一组 API 函数。

函数	说明
SCB_CyBtldrCommStart()	启动 I²C 组件并启用其中断。
SCB_CyBtldrCommStop()	禁用 I²C 组件并禁用其中断。
SCB_CyBtldrCommReset()	将读取和写入 I²C 缓冲区设置为其初始状态，然后重置从设备状态。
SCB_CyBtldrCommWrite()	允许调用程序将数据写入引导加载程序主机。该函数将处理轮询以便将数据块完整发送至主机设备。
SCB_CyBtldrCommRead()	允许调用程序读取引导加载程序主机中的数据。该函数将处理轮询以便从主机组件完整接收数据块。

void SCB_CyBtldrCommStart(void)

说明： 启动 I²C 组件并使能其中断。
每个 I²C 写数据操作都将视为引导加载程序的命令。
每个 I²C 读数据操作都将在引导加载程序响应已执行的命令前返回 0xFF。

参数： 无

返回值： 无

副作用： 无

void SCB_CyBtldrCommStop(void)

说明： 禁用 I²C 组件并禁用其中断。

参数： 无

返回值： 无

副作用： 无

void SCB_CyBtldrCommReset(void)

说明： 将读取和写入 I²C 缓冲区设置为其初始状态，然后重置从设备状态。

参数： 无

返回值： 无

副作用： 无

cystatus SCB_CyBtldrCommRead(uint8 pData[], uint16 size, uint16 * count, uint8 timeOut)

说明： 允许调用程序读取引导加载程序主机中的数据。该函数将处理轮询以便从主机组件完整接收数据块。

参数： uint8 pData[]：发送至设备的数据块的指针。

uint16 size：待写入的字节数。

uint16 *count：用于写入实际已写入字节数之变量的指针。

uint8 timeOut：在 10 毫秒内等待的单位数，之后将因超时而返回。

返回值： cystatus：如果未遇到任何问题则返回 CYRET_SUCCESS，或是返回对该问题描述最准确的值。有关更多信息，请参考《系统参考指南》的“返回代码”一节。

副作用： 无

cystatus SCB_CyBtldrCommWrite(const uint8 pData[], uint16 size, uint16 * count, uint8 timeOut)

说明： 允许调用程序将数据写入引导加载程序主机。该函数将处理轮询以便将数据块完整发送至主机组件。

参数： const pData[]：发送至设备的数据块的指针。

uint16 size：待写入的字节数。

uint16 *count：用于写入实际已写入字节数之变量的指针。

uint8 timeOut：在 10 毫秒内等待的单位数，之后将因超时而返回。

返回值： cystatus：如果未遇到任何问题则返回 CYRET_SUCCESS，或是返回对该问题描述最准确的值。有关更多信息，请参考《系统参考指南》的“返回代码”一节。

副作用： 无

I2C 功能描述

此组件支持 I²C 从设备、主设备、多主设备和多主设备从设备等配置。下文简述如何在这些配置下使用该组件。

由于 I²C 硬件被中断驱动，因此您需为该组件使能全局中断。即使该组件需要中断，您也无需向 ISR（中断服务子程序）添加代码。该组件将独立于代码之外为所有中断（数据传输）提供服务程序。为此接口分配的存储器缓冲区类似应用程序与 I²C 主设备/从设备之间的简单双端口存储器。

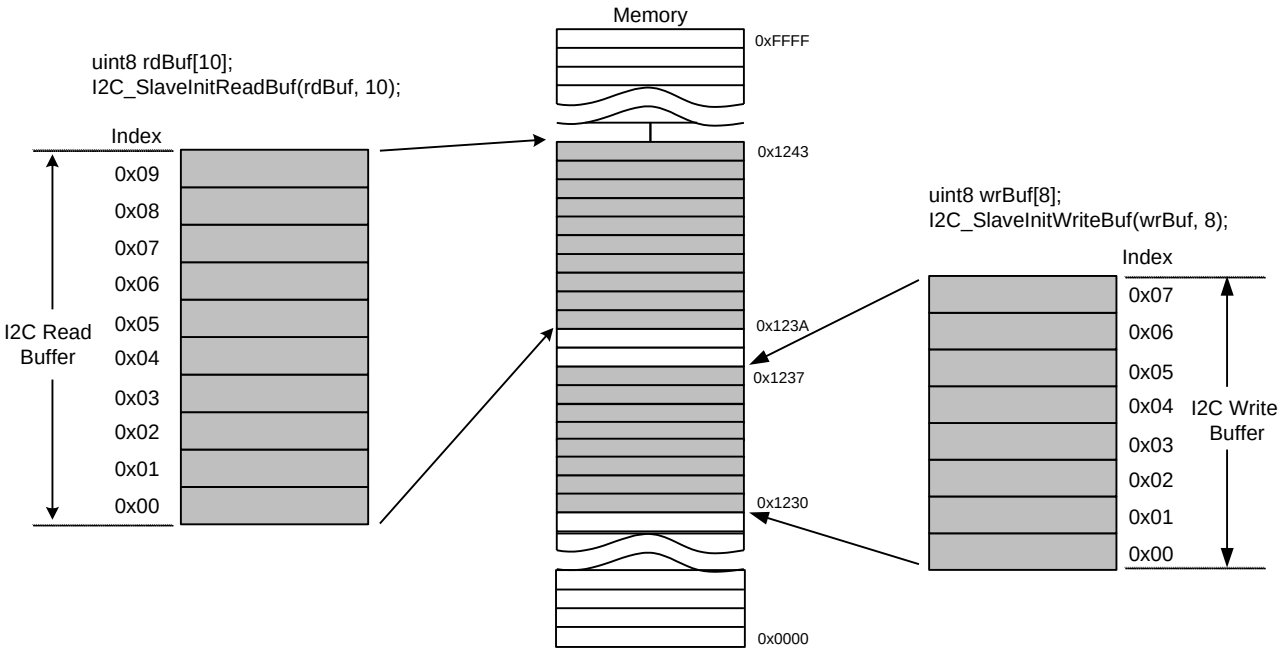
从设备操作

从设备接口由存储器中的两个缓冲区组成。其中一个用于由主设备写入到从设备的数据，另一个则用于主设备由从设备中读取的数据。请牢记，此处的读取和写入均基于 I²C 主设备而言。I²C 从设备的读写缓冲区由下列初始化命令完成设置。这些命令不会分配内存，而是将阵列指针和大小复制到内部组件变量中。由于组件不会自动生成这些阵列，必须对用于缓冲区的阵列进行实例化。同一缓冲区可同时用于读取和写入缓冲区，但须妥当管理数据。

```
void SCB_I2CSlaveInitReadBuf(uint8 * rdBuf, uint32 bufSize)
void SCB_I2CSlaveInitWriteBuf(uint8 * wrBuf, uint32 bufSize)
```

使用上述功能为读取和写入缓冲区设置一个指针和字节计数。这些函数的 bufSize 可能小于或等于实际的阵列大小，但不应大于 rdBuf 或 wrBuf 指针指向的可用存储器大小。

图 1 从设备缓冲区结构



在调用 SCB_I2CSlaveInitReadBuf() 或 SCB_I2CSlaveInitWriteBuf() 函数时，内部索引将被设置为 rdBuf 和 wrBuf 分别指向的阵列中的首个值。当 I²C 主设备读取或写入字节时，该索引会递增，直至偏移小于缓冲区大小。您可以随时为读取缓冲区和写入缓冲区分别调用 SCB_I2CSlaveGetReadBufSize() 或 SCB_I2CSlaveGetWriteBufSize() 以查询已传输的字节数。读取或写入的字节数超过缓冲区中的字节数时，会发生溢出错误。溢出状态将在从设备的状态字节中设置并可通过 SCB_I2CSlaveStatus() API 读取。

要将索引重置为阵列的开始位置，请使用以下命令。

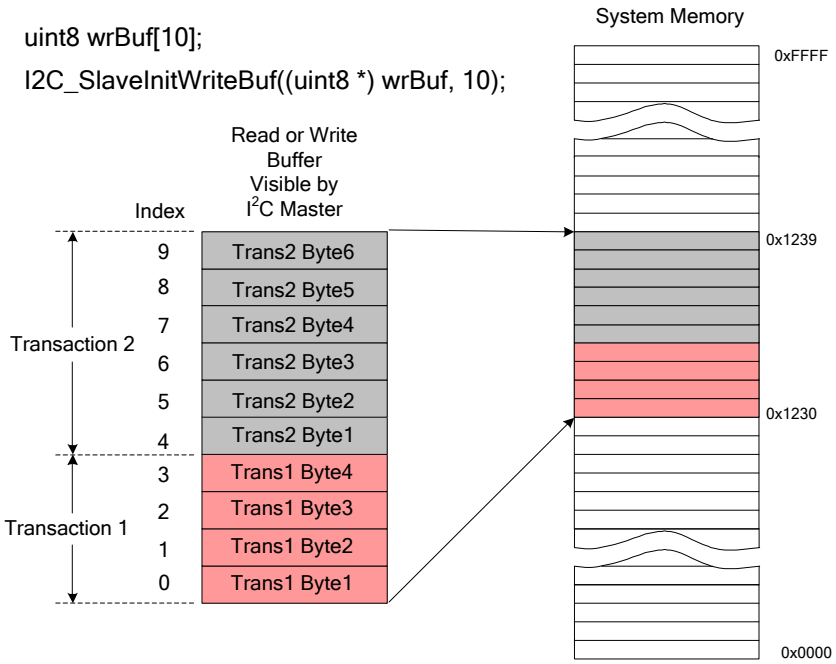
```
void SCB_I2CSlaveClearReadBuf(void)
void SCB_I2CSlaveClearWriteBuf(void)
```

此操作会将索引重置为零。I²C 主设备读取或写入的第二个字节为该阵列的第一个字节。在使用这些清除缓冲区的命令之前，应读取或更新阵列中的数据。

在使用清除缓冲区命令或是阵列索引可能超过阵列大小之前，I²C 主设备的多次读取或写入操作将继续使该索引递增。图 2 的示例中，I²C 主设备已执行两次写入数据操作。第一次写入了 4 个字节，第二次则写入了 6 个字节。其中，第二次操作中的第 6 个字节已被从设备否认，以此表明已到达缓冲区末尾。如果主设备试图在第二次操作中写入第 7 个字节，或在第三次操作时写入其他字节，则在重置缓冲区之前， 每个字节都会被否认并丢弃。

在第一次操作将索引重置为零并导致第二次操作覆盖第一次操作中的数据后，请使用 SCB_I2CSlaveClearWriteBuf() 函数。请务必确保在缓冲区溢出时不会丢失数据。重置缓冲区索引之前，从设备应先处理缓冲区中的数据。

图 2 系统存储器



读取和写入缓冲区均有四个状态位用以指示传输完成、正在传输以及缓冲区溢出等状态。传输开始时，将设置繁忙标志。传输完成时，将设置传输完成标志并清除繁忙标志。若开始第二次传输，则可能会同时设置繁忙标志和传输完成标志。读写状态标志如下表所示。

从设备状态常数	说明
SCB_I2C_SSTAT_RD_CMPLT	从设备读取传输完成。

从设备状态常数	说明
SCB_I2C_SSTAT_RD_BUSY	正在执行从设备读取传输（繁忙）。
SCB_I2C_SSTAT_RD_OVFL	主设备已尝试读取超出缓冲区内字数数的字节。
SCB_I2C_SSTAT_RD_ERR	从设备在读取传输时检测到总线错误。
SCB_I2C_SSTAT_WR_CMPLT	从设备写入传输完成。
SCB_I2C_SSTAT_WR_BUSY	正在执行从设备写入传输（繁忙）。
SCB_I2C_SSTAT_WR_OVFL	主设备尝试写入的内容已超出缓冲区末尾。
SCB_I2C_SSTAT_WR_ERR	从设备写入传输时检测到总线错误。

以下示例代码对写入缓冲区初始化后等待传输完成。当传输完成后，会将数据复制到工作数组中以处理该数据。在许多应用程序中，不必复制数据到另一个地方，但可在原始缓冲区中进行处理。您可以通过使用读取函数和常量来替换写入函数和常量，创建一个与写入缓冲区示例一样的读取缓冲区示例。处理数据可能意味着新数据将传入到从设备缓冲区中，而不是从从设备缓冲区中传出。

```
uint8 wrBuf[10];
uint8 userArray[10];
uint32 byteCnt;
/* Initialize write buffer before call SCB_Start */
SCB_I2CSlaveInitWriteBuf((uint8 *) wrBuf, 10);

/* Start I2C Slave operation */
SCB_I2CStart();

/* Wait for I2C master to complete a write */

for(;;)  /* loop forever  */
{
   /* Wait for I2C master to complete a write */
   if(0u != (SCB_I2CSlaveStatus() & SCB_I2C_SSTAT_WR_CMPLT))
   {
      byteCnt = SCB_I2CSlaveGetWriteBufSize();
      SCB_I2CSlaveClearWriteStatus();
      for(i=0; i < byteCnt; i++)
      {
         userArray[i] = wrBuf[i]; /* Transfer data */
      }
      SCB_I2CSlaveClearWriteBuf();
   }
}
```

注意：所有从设备状态和缓冲区操作 APIs 均不受中断保护并可通过 I2C ISR 修改。执行状态和缓冲区操作过程中，建议禁用 I2C 中断。禁用中断可阻止从设备操作并伸展 SCL 时钟。

主设备和多主设备操作

主设备和多主设备操作基本上是相同的，但有两个例外。当您在多主设备模式下操作时，应始终检查总线以查看其是否繁忙。另一个主设备可能已与另一个从设备进行通信。这种情况下，程序必须等待当前操作完成后才能发出开始数据操作。该程序会查看返回值，如果另一个主设备控制了总线，则该值会设置错误。

第二个不同之处在于，在多主设备模式下可以同时启动两个主设备。如果发生这种情况，其中一个主设备将仲裁失败。必须在每个字节传输完成后检查是否出现这种情况。组件会自动检查这种情况，并在仲裁失败后发出错误响应。

在执行 I²C 主设备操作时可进行如下选择：手动和自动。在自动模式中，将创建缓冲区以保持整个传输。在执行写入操作时，缓冲区预填充要发送的数据。如果从从设备读取数据，则至少需要分配一个具有数据包大小的缓冲区。要在自动模式中将字节阵列写入到从设备，请使用以下函数。

```
uint32 SCB_I2CMasterWriteBuf(uint32 slaveAddress, uint8 * wrData, uint32 cnt,
uint32 mode)
```

SlaveAddr 变量是一个右对齐的 7 位从设备地址，其值介于 0 至 127 之间。组件 API 将会自动将读写标志附加到地址字节的 LSb 中。传输的数组将指向第二个参数 “xferData”。参数 cnt 为要传输的字节数。最后一个参数 “mode（模式）” 将确定传输开始和结束的方式。可以使用重新开始而不是开始来启动数据操作，或在停止序列之前终止数据操作。这些选项将允许执行背对背传输，其中最后一个传输不会发送“停止”命令，而下一个传输会发送“重新开始”而不是“开始”命令。

读取操作与写入操作方法几乎一致。会使用带有相同常量的相同参数。

```
uint32 SCB_I2CMasterReadBuf(uint32 slaveAddress, uint8 * rdData, uint32 cnt,
uint32 mode);
```

两个函数均返回状态。有关 SCB_I2CMasterStatus() 函数返回值的信息，请参见状态表。由于在 I²C 中断代码中读取和写入传输会在后台完成，因此可以使用 SCB_I2CMasterStatus() 函数来确定该传输何时完成。后续代码段显示对从设备的典型写入操作。

```
SCB_I2CMasterClearStatus(); /* Clear any previous status */
SCB_I2CMasterWriteBuf(8u, (uint8 *) wrData, 10u, SCB_I2C_MODE_COMPLETE_XFER);
for(;;)
{
   if(0u != (SCB_I2CMasterStatus() & SCB_I2C_MSTAT_WR_CMPLT))
   {

      /* Transfer complete. Check Master status to make sure that transfer
         completed without errors. */

      break;
   }
}
```

I²C 主设备可手动操作。在此模式中，将使用单个命令来执行写入数据操作的每个部分。

```
status = SCB_I2CMasterSendStart(8u, SCB_I2C_WRITE_XFER_MODE);
```

```
if(SCB_I2C_MSTR_NO_ERROR == status)   /* Check if transfer completed without
errors */
{
   /* Send array of 5 bytes */
   for(i=0; i<5u; i++)
   {
      status = SCB_I2CMasterWriteByte(userArray[i]);
      if(SCB_I2C_MSTR_NO_ERROR != status)
      {
         break;
      }
   }
}
SCB_I2CMasterSendStop();      /* Send Stop */
```

除最后一个字节应该否认外，手动读取数据操作与写入数据操作类似。以下示例显示了一个典型的手动读取数据操作。

```
status = SCB_I2CMasterSendStart(8u, SCB_I2C_READ_XFER_MODE);
if(SCB_I2C_MSTR_NO_ERROR == status)   /* Check if transfer completed without
errors */
{
   /* Read array of 5 bytes */
   for(i=0; i<5u; i++)
   {
      if(i < 4u)
      {
         userArray[i] = SCB_I2CMasterReadByte(SCB_I2C_ACK_DATA);
      }
      else
      {
         userArray[i] = SCB_I2CMasterReadByte(SCB_I2C_NAK_DATA);
      }
   }
}
SCB_I2CMasterSendStop();    /* Send Stop */
```

多主设备从设备模式操作

在此模式中，可以执行多主设备和从设备操作。可以将组件称为从设备，但固件还会启动主设备模式传输。在此模式中，当主设备在地址字节期间仲裁失败时，从设备硬件将检查获胜主设备是否对其进行寻址。如果地址匹配，从设备转为活动状态。

有关主设备和从设备操作示例的更多信息，请参考《从设备操作》从设备操作中的 }
}

注意：所有从设备状态和缓冲区操作 APIs 均不受中断保护并可通过 I2C ISR 修改。执行状态和缓冲区操作过程中，建议禁用 I2C 中断。禁用中断可阻止从设备操作并伸展 SCL 时钟。

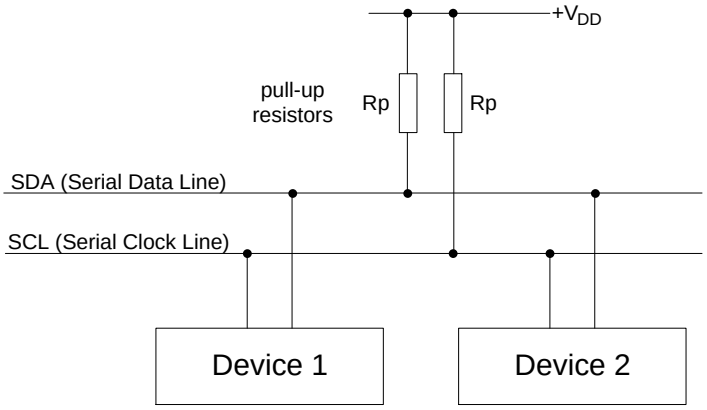
主设备和多主设备操作“主设备和多主设备操作

”一节。

外部电路连接

如图 3 所示，I²C 总线上要有外部上拉电阻。上拉电阻 (RP) 由供电电压、时钟速度和总线电容确定。将输出阶段的任何组件（主设备或从设备）的最小灌电流设置为不超过 3 mA（在 V_{OLmax} = 0.4 V 的条件下）。这会将 5V 系统的最小上拉电阻值限制在 1.5 kΩ 左右。RP 的最大值取决于总线电容和时钟频率。对于总线电容为 150 pF 的 5V 系统，上拉电阻不超过 6 kΩ。有关上拉电阻和其他物理总线规格的更多信息，请参见《 I²C 总线规格》 。

图 3 连接设备到 I²C 总线



注意：从 Cypress 或其获得分许可的其中一个联营公司处购买 I²C 组件，即可根据 Philips I²C 专利权获得一份许可，以便在 I²C 系统中使用这些组件，但前提是该系统符合 Philips 定义的 I²C 标准规范。自 2006 年 10 月 1 日起，Philips Semiconductors 就采用一个新的商标名称－NXP Semiconductors。

低功耗模式

在从设备模式（从设备和多组件从设备）中，I2C 可用做深度睡眠唤醒源。“从睡眠模式使能唤醒”须在定制器或设置的配置结构相应字段中进行检查。外部逻辑进行地址匹配并生成唤醒中断。从设备延展 SCL 直至组件唤醒时间过去，然后 I2C_ISR z 通过设置命令确认地址。在执行 I2C 数据操作到从设备时进入深度睡眠，为保障安全后跟以下代码：

```
CyGlobalIntDisable; /* Disable all interrupts to lock I2C bus state */

/* Check if transaction is in process */
status = (SCB_I2CSlaveStatus() & (SCB_I2C_SSTAT_RD_BUSY |
                                  SCB_I2C_SSTAT_WR_BUSY));
```

```
if(0u == status) /* Bus is free */
{
    SCB_Sleep(); /* Prepare to Deep Sleep */

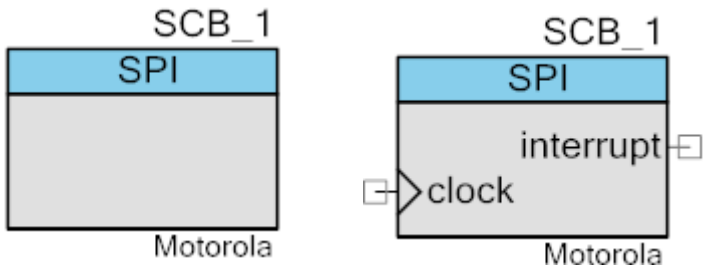
    CySysPmDeepSleep();

    CyGlobalIntEnable; /* Disable all interrupts to unlock I2C bus state */

    SCB_Wakeup(); /* Exit Deep Sleep */
}
else
{
    /* Transaction in progress: do not go to Deep Sleep */
    CyGlobalIntEnable;
}
```

注意：要将组件从睡眠中唤醒，保持 “使能睡眠模式唤醒” 处于未选中状态。在这种情况下，没有理由调用 SCB_Sleep() and SCB_Wake()。

SPI



该组件提供了行业标准的 4 线 SPI 接口。原始的 SPI 协议由 Motorola 定义。该组件支持三种额外模式，允许与任何 SPI 组件进行交流。除了标准的 8 位字长之外，该组件还支持可配置的 4 至 16 位数据宽度，用于在非标准的 SPI 数据宽度下通信。

输入/输出连接

本部分描述了 SCB 组件用的各种各样的输入/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。

时钟 – 输入*

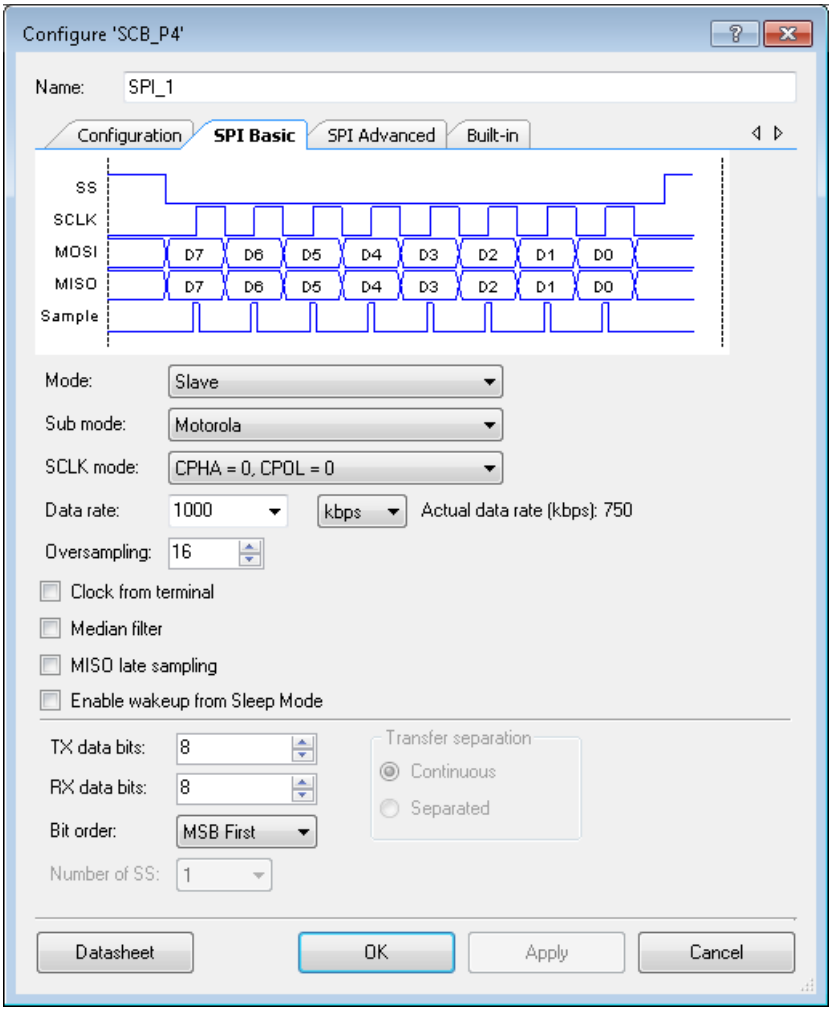
时钟负责运行此模块。根据**终端时钟**参数的不同，终端的存在位将各不相同。

中断 – 输出*

此信号只能与一个中断组件相连，或者不连接。根据**中断**参数的不同，终端的存在位将各不相同。

组件内部有特定接口的引脚，因为这些引脚使用的是专用连接，而且不可作为通用信号用途。欲获得更多信息，请见芯片《技术参考手册》 (TRM) 的 “I/O 系统” 部分。

SPI 基础参数



SPI 基础 选项卡包含下列参数：

模式

该选项决定了 SCB 在何种 SPI 模式中操作。

- **从设备**—仅限从设备操作（默认）
- **主设备**—仅限主设备操作

子模式

该选项决定了支持的 SPI 子模式：Motorola、TI (Start Coincides)、TI (Start Precedes) 或 National Semiconductor 的 Microwire 协议。

- **Motorola**—原始 SPI 协议由 Motorola 定义（默认）。

- **TI (Start Coincides)** —Texas Instruments 的 SPI 协议。
- **TI (Start Precedes)**—Texas Instruments 的 SPI 协议。
- **National Semiconductor** —National Semiconductor 的 Microwire 协议。

SCLK 模式

该参数定义了您想在通信中运用的时钟相位和时钟极性模式。已提供 CPHA 和 CPOL 选择。

- **CPHA = 0，CPOL= 0**—数据在 SCLK 的下降沿被输出。数据在 SCLK 的上升沿被采样。这是默认模式。
- **CPHA = 0，CPOL= 1**—数据在 SCLK 的上升沿被输出。数据在 SCLK 的下降沿被采样。
- **CPHA = 1，CPOL= 0**—数据在 SCLK 的上升沿被输出。数据在 SCLK 的下降沿被采样。
- **CPHA = 1，CPOL= 1**—数据在 SCLK 的下降沿被输出。数据在 SCLK 的上升沿被采样。

终端时钟

通过该参数选择内部时钟或者外部时钟，同时决定数据率。启用该选项时，组件不对数据速率进行控制但会基于用户连接的时钟源显示实际数据速率。考虑到过采样的问题，PSoC Creator 会基于**数据速率**这个参数对要求的时钟频率进行计算和配置。

注意：当设置数据率或外部时钟频率值时，请确保 PSoC Creator 可以用当前系统时钟频率提供此值。否则，创建项目时会生成时钟精度范围警告。警告中包含 PSoC Creator 设置的实际时钟值。选择是应当更改系统时钟还是组件时钟以满足时钟设置系统要求、达到最佳值。

数据速率

此参数用于设置高达 8000 kbps 的 SPI 数据速率值；实际速率可因可用时钟速度和分频器范围而异。标准比特率为 500、1000（默认）、2000、4000 和 2000 kbps 的倍数，最大 16000。如果**来自终端的时钟**这个参数被启用，该参数没有作用。

过采样

过采样因子参数确定每个 SPI 时钟周期内的内部时钟数量。默认值为 **16**。4 到 16 之间的任何整数都是有效设置。

对于**主设备**，该范围根据**中值滤波**和**启用 MISO 近期样本**的设置而变化。

- 6 到 16—**中值滤波器**未选中，**启用延迟 MISO 采样**未选中
- 3 到 16—**中值滤波器**未选中，**启用延迟 MISO 采样**选中

- 8 到 16－**中值滤波器**未选中，**启用延迟 MISO 采样**未选中
- 4 到 16－**中值滤波器**选中，**启用延迟 MISO 采样**选中

在**从设备**模式操作中，不使用**过采样因子**（不设置任何寄存器）。仅用于计算需要的时钟频率，以达到过采样的眼球。对于**从设备**，该范围随**中值滤波器**设置的不同而不同：

- 6 到 16－未选中**中值滤波器**
- 8 到 16－已选中**中值滤波器**

中值滤波器

该参数在 MISO 输入路径上应用 3 抽头中值滤波器。该滤波器可增强信号的抗扰性。然而，过采样因子最小值将增加。默认值为**禁用**。

启用延迟 MISO 采样

通过该选项可更改用于采样 MISO 的 SCLK 边沿（仅适用于**主设备**模式） 默认值为**禁用**。

使能 “睡眠模式” 唤醒

可使用此选项将系统在从设备选择上从睡眠状态唤醒。此选项仅在从设备模式下有效。

TX 数据位

此选项用于定义传输的数据帧中的位宽度。默认位数为单字节（8 位）。4 到 16 之间的任何整数都是有效设置。

RX 数据位

此选项用于定义接收的数据帧中的位宽度。默认位数为单字节（8 位）。4 到 16 之间的任何整数都是有效设置。

注意：对于 **Motorola** 和 **Texas Instruments** 子模式，**TX 数据位**和 **RX 数据位**的数量设置应相同；对于 **National Semiconductor** 子模式，设置可以不同。

位序

位序参数用于定义串行数据的传输方向。设置为 **MSB 优先**时，首先传输最高有效位。设置为 **LSB 优先**时，首先传输最低有效位。

SS 数量

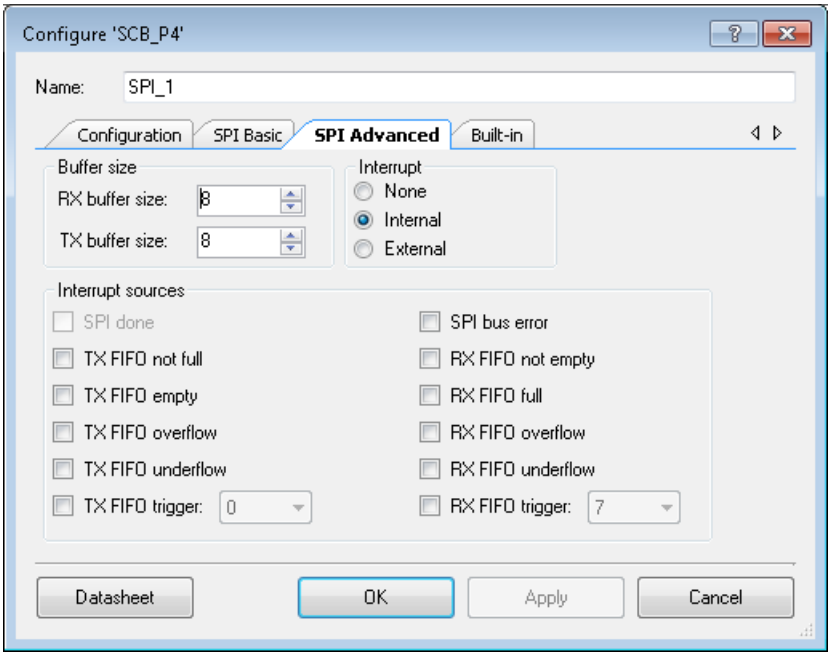
该参数决定 SPI 从设备选择线的数量。线路数量默认值为 1。1 到 4 之间的任何整数都是有效设置。此选项仅在**主设备**模式下有效。

传输分离

传输分离参数决定是否通过取消选择从设备选择来分开各个数据传输。下表中定义了这些模式。有关更多信息，请参见下表。

- **连续**－SS 在传输开始时降低，当传输完成时升高（默认）
- **分离**－通过 SS 取消选择分离每个数据帧 4 -16 位一个 SCLK 周期。

高级 SPI 参数



SPI 高级选项卡包含下列参数：

RX 缓冲区大小

RX 缓冲区大小参数定义为循环接收数据缓冲区分配的存储器大小（以字节/字为单位）。如果此参数设置为 8，则在硬件中实现 RX FIFO 的第 8 字节/字。所有其他值（最大为 2^{32}）使用 8 字节/字 RX FIFO 和由提供的 API 和内部 ISR 控制的软件缓冲区。缓冲区大小限制为可用的存储器。如果 RX 缓冲区大小超过 8 字节/字，中断模式自动设置为“内部”模式。

TX 缓冲区大小

TX 缓冲区大小参数定义为循环传输数据缓冲区分配的存储器大小（以字节/字为单位）。如果此参数设置为 8，则在硬件中实现 TX FIFO 的第 8 字节/字。所有其他值（最大为 2^{32}）使用 8 字节/字 RX FIFO 和由提供的 API 和内部 ISR 控制的软件缓冲区。缓冲区大小限制为可用的存储器。如果 TX 缓冲区大小超过 8 字节/字，中断模式自动设置“内部”模式。

中断

此选项用于决定支持的中断模式：“无”、“内部” 或 “外部”。

- **无**－删除内部中断组件
- **内部**－此选项将中断组件留在 SCB 组件内部。预先定义的内部 ISR 连接到中断。可注册自定义函数，以调用 ISR 的每一个项目。**中断源**选项定义了中断事件，以触发中断。
- **外部**－此选项删除内部中断并提供输出端子。如果需要自定义中断处理程序，仅可连接中断组件。**中断源**选项可设置触发中断输出的中断源。

注意：对于大于 8 字节/字的缓冲区大小，组件自动启用正确内部软件缓冲区操作所需的内部中断源。此外，必须启用全局中断，才能进行正确的缓冲区处理。

中断源

SPI 支持在以下事件上产生中断：

- **SPI 完成**－主设备传输完成事件：TX FIFO 中的所有数据帧被发送，TX FIFO 为空。
- **TX FIFO 未满**－TX FIFO 未满
- **TX FIFO 为空**－TX FIFO 为空
- **TX FIFO 溢出**－尝试写入到已满的 TX FIFO
- **TX FIFO 下溢**－尝试从空的 TX FIFO 中读取
- **TX FIFO 触发**－当 TX FIFO 条目少于该字段数量时，生成发送器触发事件
- **SPI 总线错误**－在 SPI 传输过程中，意外取消选择 SPI 从设备。
- **RX FIFO 不为空**－RX FIFO 不为空
- **RX FIFO 已满**－RX FIFO 已满
- **RX FIFO 溢出**－尝试写入到已满的 RX FIFO
- **RX FIFO 下溢**－尝试从空的 RX FIFO 中读取

- **RX FIFO 触发**—当 RX FIFO 条目少于该字段数量时，生成发送器触发事件

注意

当 **RX 缓冲区大小**超过 8 字节/字时，组件保留 **RX FIFO 不为空**中断源，用于内部软件缓冲区操作。

当 **TX 缓冲区大小**超过 8 字节/字时，组件保留 **TX FIFO 未满**中断源，用于内部软件缓冲区操作。

SPI API

API 允许您利用软件对组件进行配置。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了每个函数。

默认情况下，PSoC Creator 将实例名称 “SCB_1” 分配给指定设计组件的第一个实例。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。出于可读性考虑，下表中使用的实例名称为 “SCB”。

函数	说明
SCB_Init()	根据自定义程序中的定义参数初始化 SCB 组件。
SCB_Enable()	启用 SCB 组件。
SCB_Start()	开始 SCB。
SCB_Stop()	禁用 SCB 组件。
SCB_Sleep()	准备组件进入深度睡眠。
SCB_Wakeup()	为组件退出深度睡眠做准备。
SCB_SpiInit()	为 SPI 操作配置 SCB。
SCB_SpiSetActiveSlaveSelect()	选择活动从设备选择线。仅在主设备模式中适用。
SCB_SpiUartWriteTxData()	在传输缓冲区中放置将于下一个可用总线时间发送的数据。
SCB_SpiUartPutArray()	将要发送的数组放入传输缓冲区
SCB_SpiUartGetTxBufferSize()	返回传输缓冲区中元素的数量。
SCB_SpiUartClearTxBuffer()	清空传输缓冲区和 TX FIFO。
SCB_SpiUartReadRxData()	从接收缓冲区中检索下一个数据元素。
SCB_SpiUartGetRxBufferSize()	返回接收缓冲区中接收数据元素的数量。
SCB_SpiUartClearRxBuffer()	清除接收缓冲区和 RX FIFO。

全局变量

正常运行情况下，不需获得这些变量。

变量	说明
SCB_initVar	SCB_initVar 表明 SCB 组件是否完成了初始化。该变量初始化为 0，并在第一次调用 SCB_Start() 时设置为 1。这样，第一次调用 SCB_Start() 子程序后，组件不用重新初始化即可重启。 如需重新初始化组件，可在 SCB_Start() 或 SCB_Enable() 函数前调用 SCB_Init() 函数。
SCB_rxBufferOverflow	当产生内部软件接收缓冲区溢出时，将 SCB_rxBufferOverflow 置位。

void SCB_Init(void)

说明： 使 SCB 组件初始化，以便在所选的配置之一中运行：I2C、SPI 或 UART。
设置为 “未配置的 SCB” 时，该函数不进行任何初始化操作。

参数： 无

返回值： 无

副作用： 无

void SCB_Enable(void)

说明： 使能 SCB 组件。组件启动时，不应更改 SCB 配置。禁用组件后，方可更改配置。
配置设置为 “Unconfigured SCB” 时，该函数不能启用组件。组件必须进行初始化，以便在之前配置的下列模式中运行：I2C、SPI 或 UART。

参数： 无

返回值： 无

副作用： 无

void SCB_Start(void)

说明： 调用 SCB_Init() 与 SCB_Enable()。调用该函数后，组件启动且运行准备就绪。
配置设置为 “Unconfigured SCB” 时，该函数不能启用组件。组件必须进行初始化，以便在之前配置的下列模式中运行：I2C、SPI 或 UART。

参数： 无

返回值： 无

副作用： 无

void SCB_Stop(void)

说明： 禁用 SCB 组件。

参数： 无

返回值： 无

副作用： 无

void SCB_Sleep(void)

说明： 准备组件进入深度睡眠。

“从睡眠模式使能唤醒” 的选择对此函数的实现有影响。

在调用 CyPmSysDeepSleep() 函数之前调用 SCB_Sleep() 函数。进入睡眠模式前不得调用此函数。有关功耗管理函数的详细信息，请参考 《系统参考指南》。

参数： 无

返回值： 无

副作用： 无

void SCB_Wakeup(void)

说明： 未组件退出深度睡眠做准备。

“从睡眠模式使能唤醒” 选项对该函数的实现有影响。进入睡眠模式前不得调用此函数。

参数： 无

返回值： 无

副作用： 在调用 SCB_Sleep() 函数之前，调用 SCB_Wakeup() 函数可能会导致意外行为。

void SCB_SpiInit(SCB_SPI_INIT_STRUCT *config)

说明： 为 SPI 操作配置 SCB。

本函数**专为**当自定义程序中的 SCB 配置设为 “未配置 SCB” 时使用。在 SPI 模式初始化 SCB 时，可使用 SCB_Start() 或 SCB_Enable() 函数启用组件。

本函数对提供配置设置的结构使用了一个指针。该结构所含信息自定义程序设置同样可提供。

参数： config：针对含有如下字段排序列表的指针。这些字段匹配自定义程序的选项。设置的详细说明见自定义程序

字段	说明
mode	SPI 操作模式。以下定义为可用选择： SCB_SPI_SLAVE SCB_SPI_MASTER
子模式	SPI 操作子模式。以下定义为可用选择： SCB_SPI_MODE_MOTOROLA SCB_SPI_MODE_TI_COINCIDES SCB_SPI_MODE_TI_PRECEDES SCB_SPI_MODE_NATIONAL
sclkMode	决定 Motorola 子模式的 sclk 关系。其他子模式时忽略。以下定义为可用选择： SCB_SPI_SCLK_CPHA0_CPOL0 SCB_SPI_SCLK_CPHA0_CPOL1 SCB_SPI_SCLK_CPHA1_CPOL0 SCB_SPI_SCLK_CPHA1_CPOL1
过采样	SPI 时钟过采样因子。从设备模式操作时忽略。
enableMedianFilter	0 – 禁用 1 – 启用
enableLateSampling	0 – 禁用 1 – 启用 从设备模式时忽略。
enableWake	0 – 禁用 1 – 启用 主设备模式时忽略。
dataBitsRx	RX 方向数据位的位数。 对于 National 子模式，仅允许使用不同的 dataBitsRx 和 dataBitsTx。
dataBitsTx	TX 方向数据位位数。 对于 National 子模式，仅允许使用不同的 dataBitsRx 和 dataBitsTx。
bitOrder	决定位顺序。以下定义为可用选择： SCB_BITS_ORDER_LSB_FIRST SCB_BITS_ORDER_MSB_FIRST
transferSeperation	决定是否在 SS 之间进行连续传输或禁用 SS。从设备模式时忽略。以下定义为可用选择： SCB_SPI_TRANSFER_CONTINUOUS SCB_SPI_TRANSFER_SEPARATED

rxBufferSize	字中的 RX 缓冲区大小： 值 “8” 代表硬件中缓冲或用途。 值大于 8 表示在软件中缓冲。 软件 RX 缓冲区始终保持一个元素为空。缓冲区大小须为较大元素以便对接收缓冲区大小的整个数据包进行接收。
rxBuffer	为 RX 软件缓冲区提供的缓冲区空间： 须提供字中的 rxBufferSize 缓冲区。 如果 dataBitsRx 大于 8，须提供 (2* rxBufferSize) 字节。 如果软件缓冲有指示，须提供 NULL 指针。
txBufferSize	字中的 TX 缓冲区大小： 值 “8” 代表硬件中缓冲的用途。 值大于 8 表示在软件中缓冲。
txBuffer	为 RX 软件缓冲区提供的缓冲区空间： 须提供字中的 rxBufferSize 缓冲区。 如果 dataBitsRx 大于 8，须提供 (2* rxBufferSize) 字节。 如果软件缓冲有指示，须提供 NULL 指针。
enableInterrupt	0 – 禁用 1 – 启用
rxInterruptMask	要在 RX 方向中启用的中断源掩码。此掩码的写入不论 enableInterrupt 字段是何设置。通过提供以下要启用的所有源的逻辑或值启用多个源： SCB_INTR_RX_TRIGGER SCB_INTR _RX_NOT_EMPTY SCB_INTR _RX_FULL SCB_INTR _RX_OVERFLOW SCB_INTR _RX_UNDERFLOW SCB_INTR _SLAVE_SPI_BUS_ERROR
rxTriggerLevel	RX 触发中断的 FIFO 等级。不论是否启用 RX 触发中断，均写入该值。
txInterruptMask	要在 TX 方向中启用的中断源掩码。此掩码的写入不论 enableInterrupt 字段是何设置。通过提供以下要启用的所有源的 逻辑或值启用多个源： SCB_INTR_TX_TRIGGER SCB_INTR_TX_NOT_FULL SCB_INTR_TX_EMPTY SCB_INTR_TX_OVERFLOW SCB_INTR_TX_UNDERFLOW SCB_INTR_MASTER_SPI_DONE
txTriggerLevel	TX 触发中断的 FIFO 等级。不论是否启用 TX 触发中断，均写入该值。

返回值： 无

副作用： 无

void SCB_SpiSetActiveSlaveSelect(uint32 activeSelect)

说明： 选择活动从设备选择线。此函数仅适用于 SPI 主设备操作模式。
主设备应处于下列状态之一，以正确改变激活的从设备选择信号源。
禁用组件
组件已完成所有数据操作（TX FIFO 为空，且已设置 SpiDone 标志）
对于此函数，不对状态进行检查。初始化后，活动从设备选择线为 0。

参数： uint32 activeSelect：以下四种线路可使用“从设备选择”函数。

活动从设备选择常量	说明
SCB_SPIM_ACTIVE_SS0	进行以下数据操作时，从设备选择线路 0 将被激活。
SCB_SPIM_ACTIVE_SS1	进行以下数据操作时，从设备选择线路 1 将被激活。
SCB_SPIM_ACTIVE_SS2	进行以下数据操作时，从设备选择线路 2 将被激活。
SCB_SPIM_ACTIVE_SS3	进行以下数据操作时，从设备选择线路 3 将被激活。

返回值： 无

副作用： 无

void SCB_SpiUartWriteTxData(uint32 txDataByte)

说明： 在传输缓冲区中放置将于下一个可用总线时间发送的数据。
此函数进行阻止操作，并等待直到传输缓冲区有可用空间存放请求数据。

参数： uint32 txDataByte：要传输的数据。

返回值： 无

副作用： 无

void SCB_SpiUartPutArray(const uint16/uint8 wrBuf[], uint32 count)

说明： 将要发送的数组放入传输缓冲区
此函数执行阻止操作，并等待直到传输缓冲区有可用空间存放所有请求数据。
阵列大小可超过传输缓冲区的大小。

参数： const uint16/uint8 wrBuf[]：指向要放置在传输缓冲区中的数组。
uint32 count：要放置在传输缓冲区中的数据元素的数量。

返回值： 无

副作用： 无

uint32 SCB_SpiUartGetTxBufferSize(void)

说明： 返回传输缓冲区中当前元素的数量。

- 禁用的 TX 软件缓冲区：返回 TX FIFO 中用到的条目数量。
- 启用的 TX 软件缓冲区：返回传输缓冲区中当前用到的元素数量。该数目不包含 TX FIFO 中用到的条目数量。TX FIFO 未满前，传输缓冲区大小为 0。

参数： 无

返回值： uint32：准备传输的数据元素的数量。

副作用： 无

void SCB_SpiUartClearTxBuffer(void)

说明： 清除传输缓冲区和 TX FIFO。

参数： 无

返回值： 无

副作用： 无

uint32 SCB_SpiUartReadRxData(void)

说明： 从接收缓冲区中检索下一个数据元素。

- 禁用 RX 软件缓冲区：返回从 RX FIFO 检索的数据元素。如果 RX FIFO 为空，将返回未定义的数据。
- 启用 RX 软件缓冲区：从软件接收缓冲区返回数据元素。如果软件接收缓冲区为空，将返回零值。

参数： 无

返回值： uint32：来自接收缓冲区的下一个数据元素。

副作用： 无

uint32 SCB_SpiUartGetRxBufferSize(void)

说明： 返回接收缓冲区中接收数据元素的数量。

禁用的 RX 软件缓冲区：返回 TX FIFO 中用到的条目数量。

启用的 RX 软件缓冲区：返回放置在接收缓冲区中的元素的数量。

参数： 无

返回值： uint32：接收到的数据元素的数量

副作用： 无

void SCB_SpiUartClearRxBuffer(void)

说明： 清除接收缓冲区和 RX FIFO。

参数： 无

返回值： 无

副作用： 无

SPI 功能描述

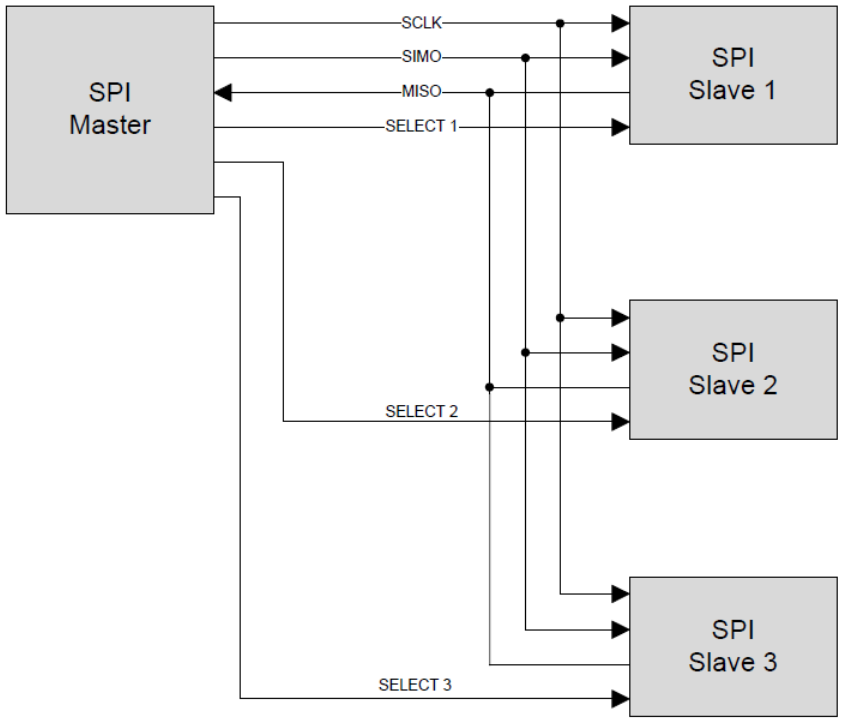
串行外设接口 (SPI) 是同步的串行接口，具有“单主设备多从设备”拓扑结构。原始 SPI 协议由 Motorola 定义。组件在主设备或从设备模式下运行。主设备启动数据帧传输。多个从设备有各自的从设备选择线路。

SPI 接口包括四种信号：

- **SCLK**－串行时钟（主设备输出，从设备输入）。
- **MOSI**－主设备输出，从设备输入（从主设备输出，输入到从设备）。

- **MISO**—主设备输入，从设备输出（输入到主设备，从从设备输出）。
- **SELECT**—从设备选择（主要为有效低电平信号，从主设备输出，输入到从设备）。

图 4 SPI 总线连接示例图



Motorola 子模式操作

原始 SPI 协议由 Motorola 定义。其为全双工协议：传输和接收同时进行。

Motorola SPI 协议具有四种模式，其定义了数据如何在 MOSI 和 MISO 线路上被输出和采样。这些模式由时钟极性 (CPOL) 和时钟相位 (CPHA) 决定。

- **CPHA = 0，CPOL= 0**—数据在 SCLK 的下降沿被输出。数据在 SCLK 的上升沿被采样。
- **CPHA = 0，CPOL= 1**—数据在 SCLK 的上升沿被输出。数据在 SCLK 的下降沿被采样。
- **CPHA = 1，CPOL= 0**—数据在 SCLK 的上升沿被输出。数据在 SCLK 的下降沿被采样。
- **CPHA = 1，CPOL= 1**—数据在 SCLK 的下降沿被输出。数据在 SCLK 的上升沿被采样。

图 5 描述了基于 CPOL 和 CPHA 的 MOSI/MISO 数据输出和采样。

图 5 SPI Motorola 帧格式

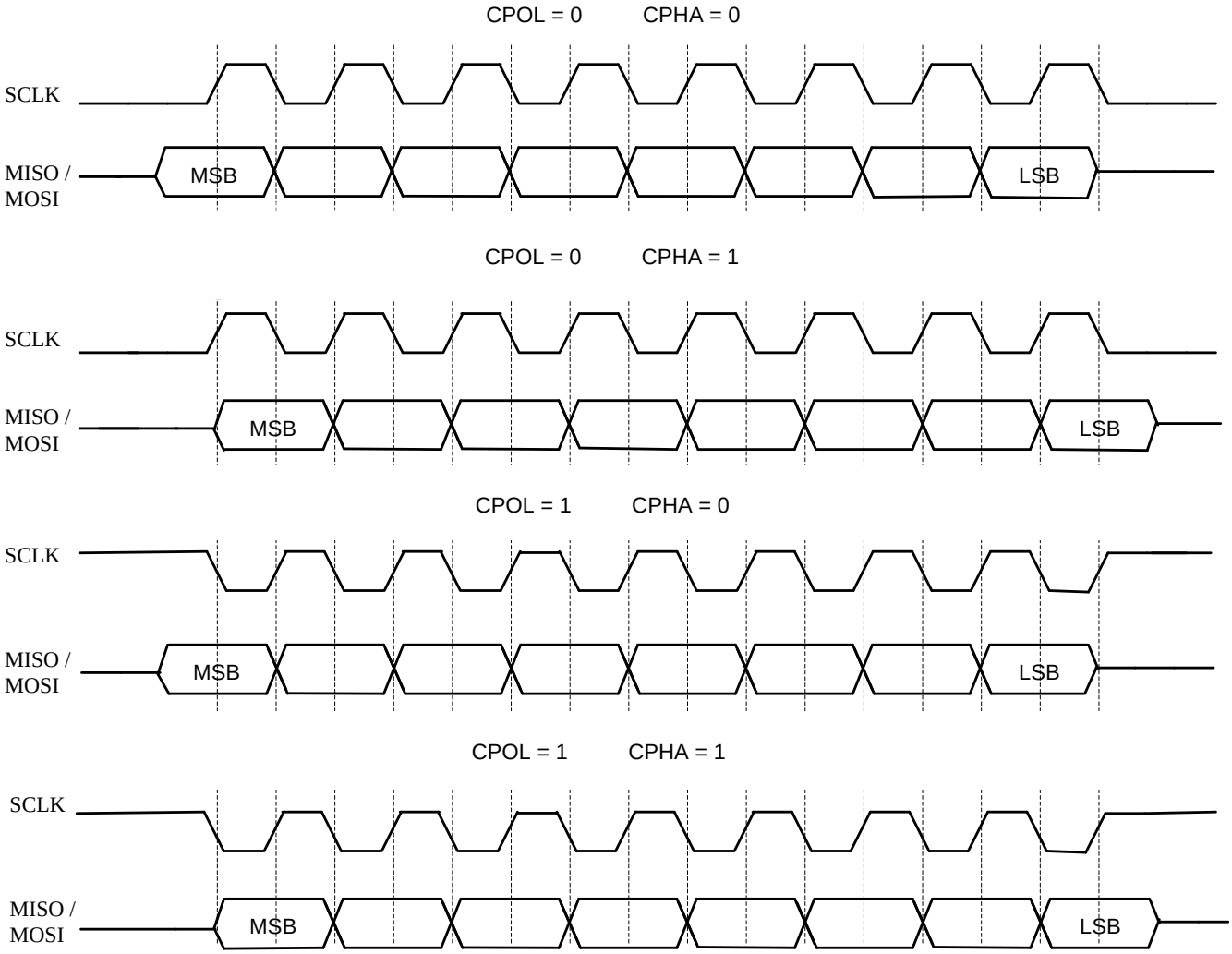
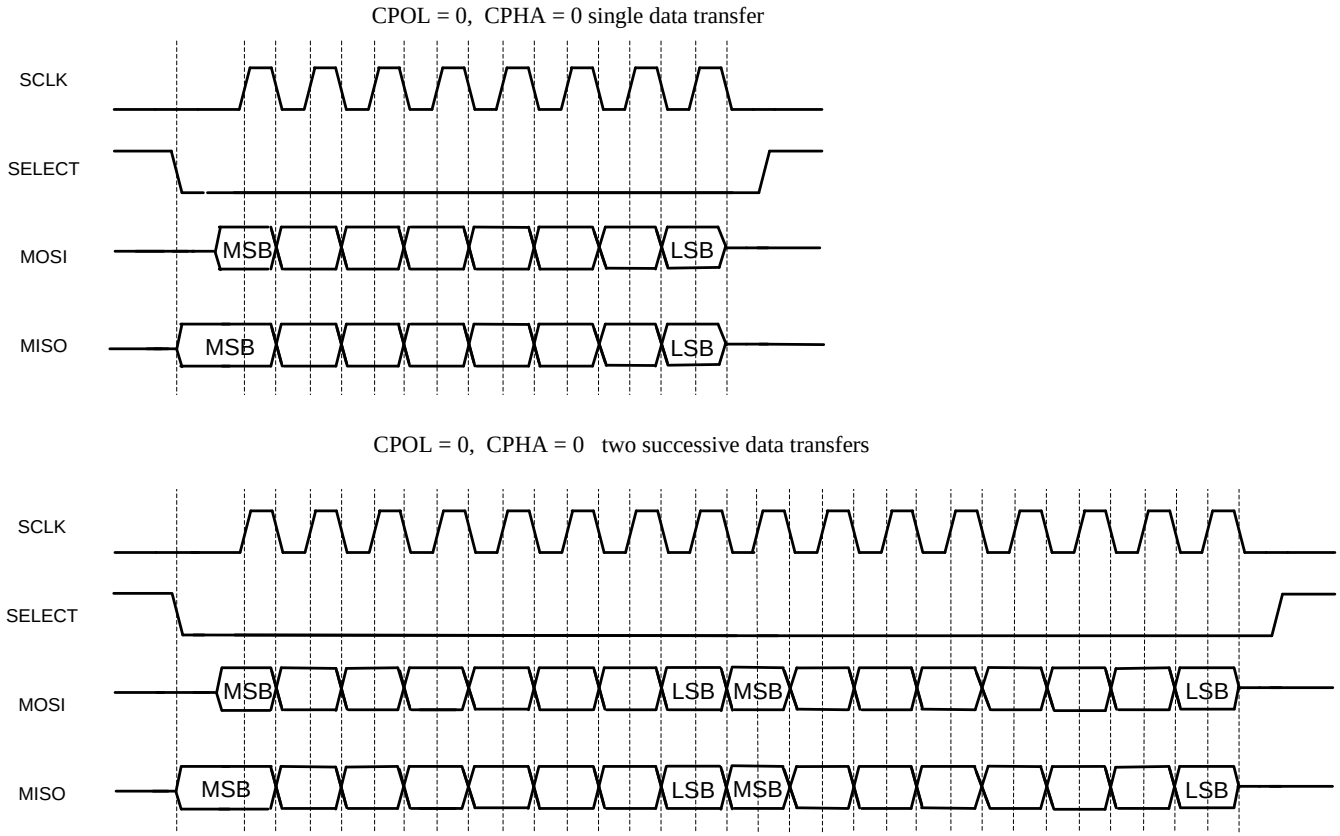


图 6 描述了模式 0（CPOL 为“0”，CPHA 为“0”）下一个 8 位数据传输和两个连续的 8 位数据传输。

图 6 SPI Motorola 数据传输示例图



Texas 子模式操作

Texas 的 SPI 协议重新定义了 SS 信号的使用。其使用该信号（而非有效的从设备选择低电平信号）指示数据传输开始。该协议仅支持 CPHA = 1，CPOL= 0。

单个位传输周期中，高活动脉冲指示传输开始。在一个 SCLK 周期中，该脉冲传输可能优先于第一个数据帧位传输，或与第一个数据位传输同步发生。

图 7 描述了一个 8 位数据传输和两个连续的 8 位数据传输。SS 脉冲传输优先于第一个数据位传输。

注意：第二个数据传输中的 SELECT 脉冲如何与第一个数据传输中的最后数据位同步传输。

图 7 TI（优先）数据传输示例图

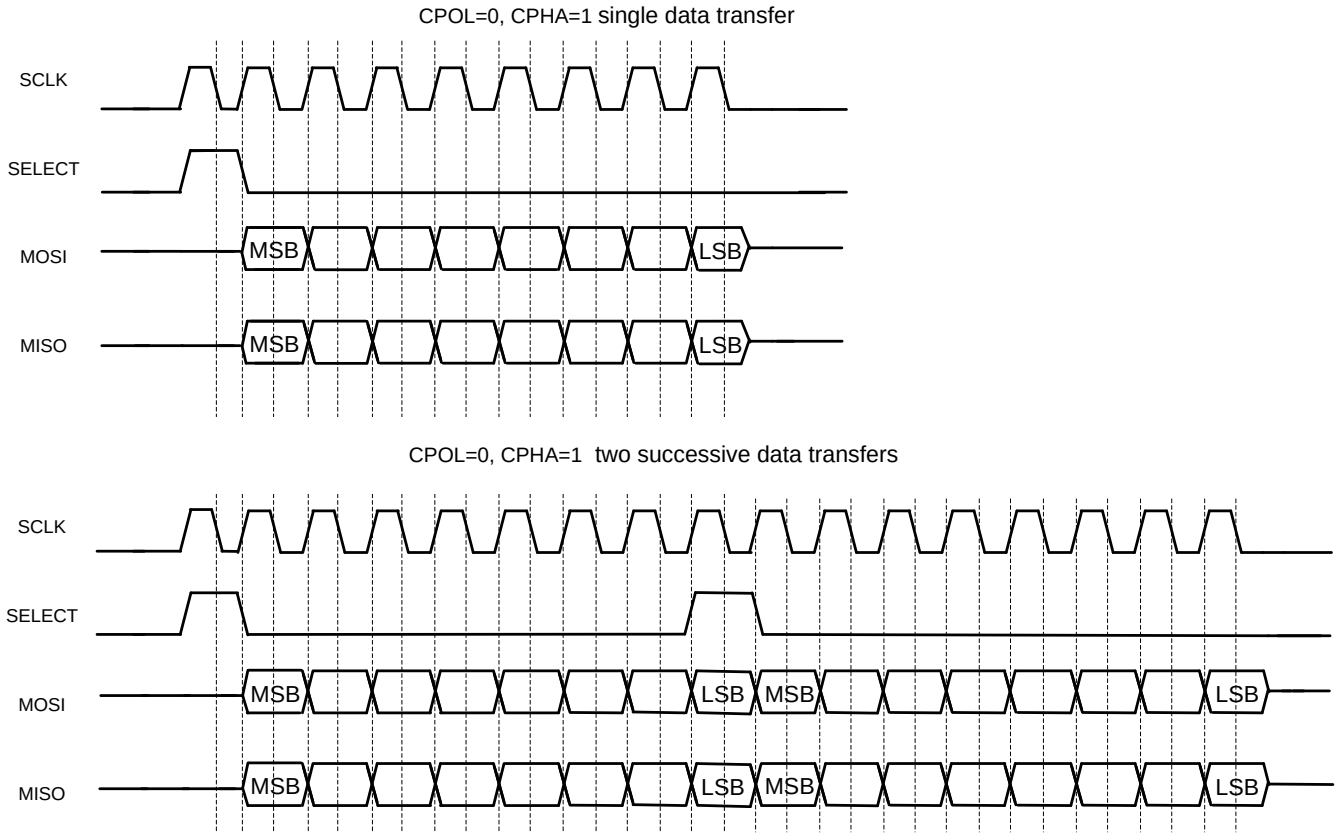
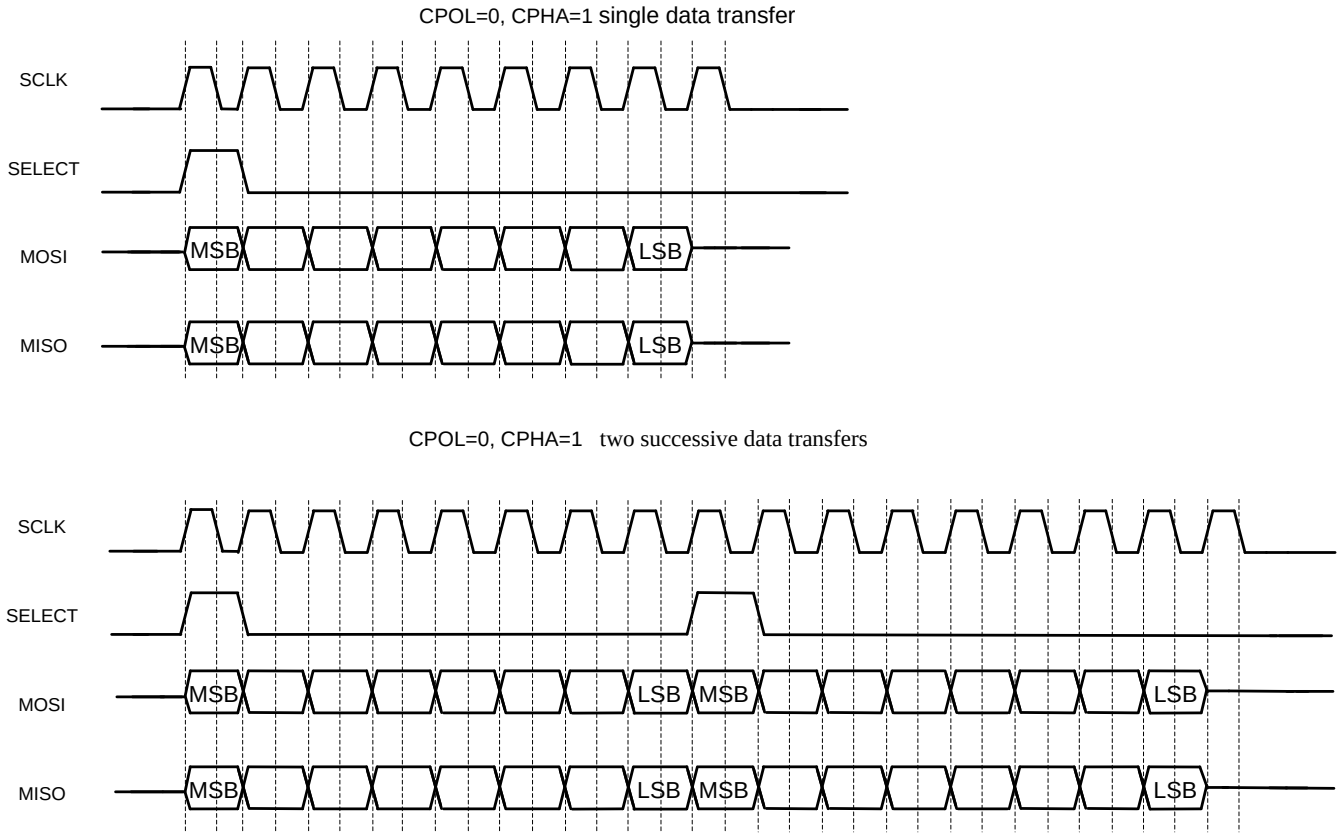


图 8 描述了一个 8 位数据传输和两个连续的 8 位数据传输。SS 脉冲传输同步于第一个数据位传输。

图 8 TI（同步）数据传输示例图



National Semiconductor 的 Micro Wave 子模式操作

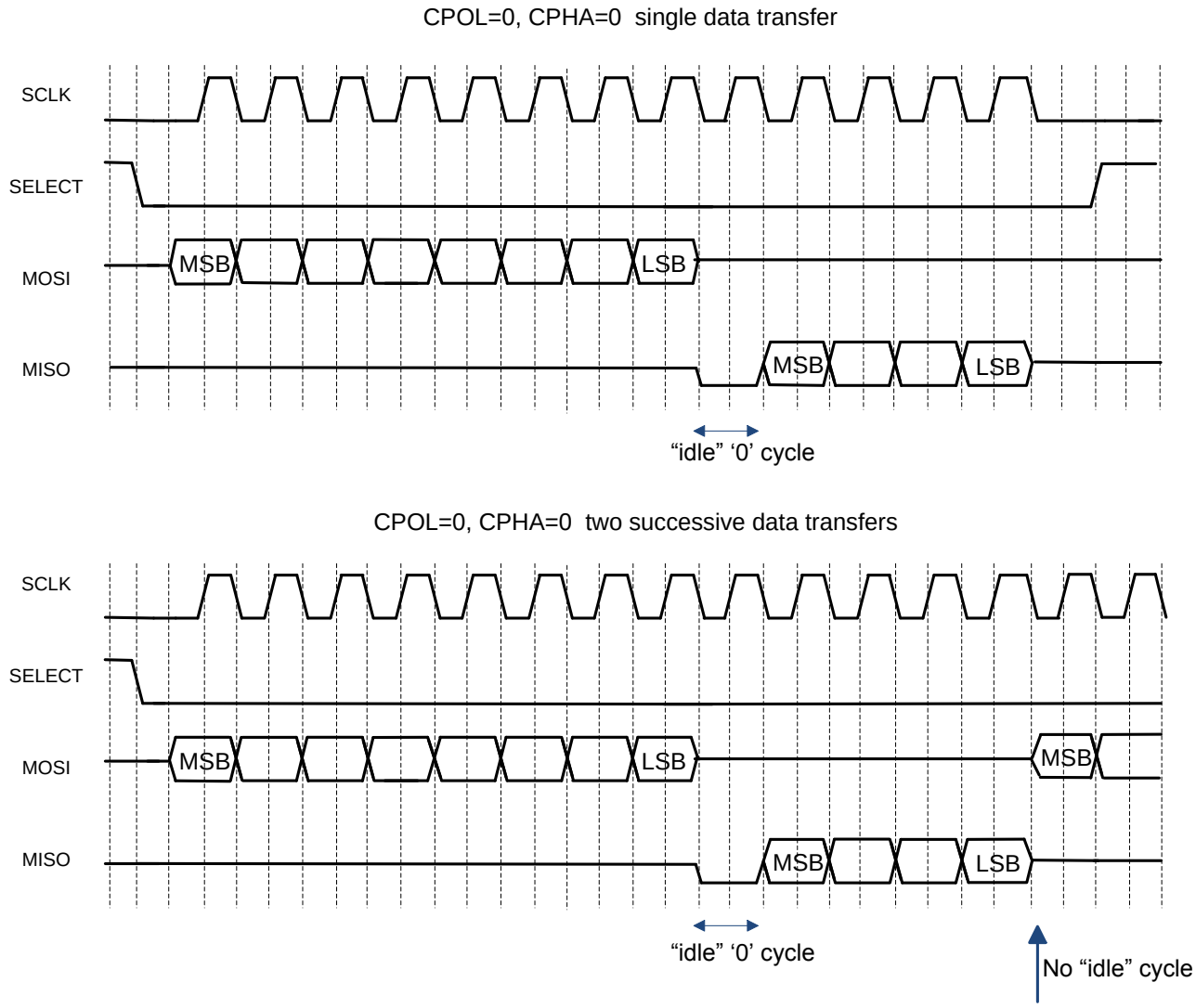
National Semiconductor 的微细线协议为半双工协议。传输和接收交替进行，而非同时进行（传输发生在接收之前）单个 “空闲” 位传输周期中，传输和接收分开进行。该协议仅支持 CPHA = 1，CPOL= 0。

注意：单个“空闲”位传输周期中，连续的数据传输不分开进行。

传输数据的大小可能与接收数据的大小不同。

图 9 描述了一个数据传输和两个连续的数据传输。在这两种情况下，传输数据的传输大小为 8 位，接收数据的传输大小为 4 位.

图 9 National Semiconductor 的微细线数据传输示例图



连续与分开传输

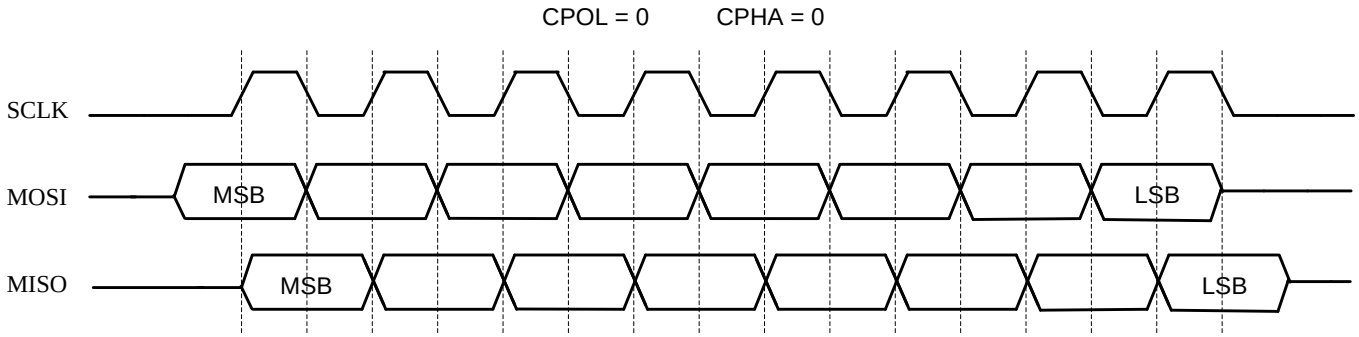
数据分开传输期间，SELECT 线路在分开传输之间从 “0” 变为 “1”，然后又从 “1” 变回 “0”。每个分开的数据传输重复该过程。

多个数据传输可能同时发生，SELECT 线路在分开传输之间无需进行切换。图 10 描述了 SCLK 模式（CPHA=0，CPOL= 0）下两个连续的 8 位数据传输。

MISO 迟采样

在 SCLK 后半周期，MISO 被采样（仅适用于主设备模式）。迟采样技术解决了从主设备向从设备发送 SCLK 以及从从设备向主设备发送 MISO 时的往复路径延迟问题。

图 10 延迟 MISO 采样示例图



软件缓冲

SCB 具有 FIFO 存储器，是一个 16 字 x 16 位 SRAM，具备“字节写入使能”功能。SPI 模式下，FIFO 被拆分成 TX FIFO 和 RX FIFO。各含 8 个条目，每个条目 16 位。每个条目的 16 位宽度用于调节可配置的数据宽度。

内部中断处理程序用于提供软件和硬件 TX/RX 缓冲间的相互操作，而不对您的顶级固件进行任何更改。

您也应当考虑到使用软件缓冲会导致被传输的字之间计时间隔更大，因为中断处理程序需要额外的时间来执行（取决于所选 HFCLK 的值）。

中断

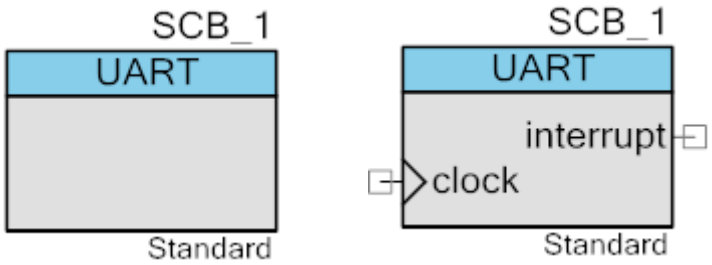
当 **RX 缓冲区大小**或 **TX 缓冲区大小**超过 8 字节/字时，**RX FIFO 非空**和 **TX FIFO 未满**中断源通过内部软件缓冲操作组件保留。**请勿清空或禁用它们，**因为这样做会造成错误的软件缓冲操作。但清除来自其他源的中断事件是用户的责任，因为它们不会被自动清除。那是客户处理程序函数的功能。

假如 **RX 缓冲区大小**或 **TX 缓冲区大小**小于或等于 8 字节/字，使用的不是软件缓冲，而只是硬件 TX 或 RX FIFO。在**内**中断模式中，不会自动清除中断。这是用户的责任。在这种情况下，优先选择**外部**或**无**中断。

低功耗模式

在**从设备**运行模式中，设备的唤醒通过从设备的选择进行。唤醒耗时较长，进行中的 SPI 传输被否定确认，“0xff” 字节发送到 MISO 线。设备唤醒时间过后，主设备必须再次检查组件。

UART



UART 提供异步通信，常用串行异步通信设备为 RS232。SCB 支持三种不同的 UART 类串口协议：

- UART－基本型
- SmartCard－与 UART 类似，但是可以发送否定确认。
- IrDA－对用于红外线通信的调制方案的修改。

输入/输出连接

本部分描述了 SCB 组件各种各样的输入/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。

时钟 – 输入*

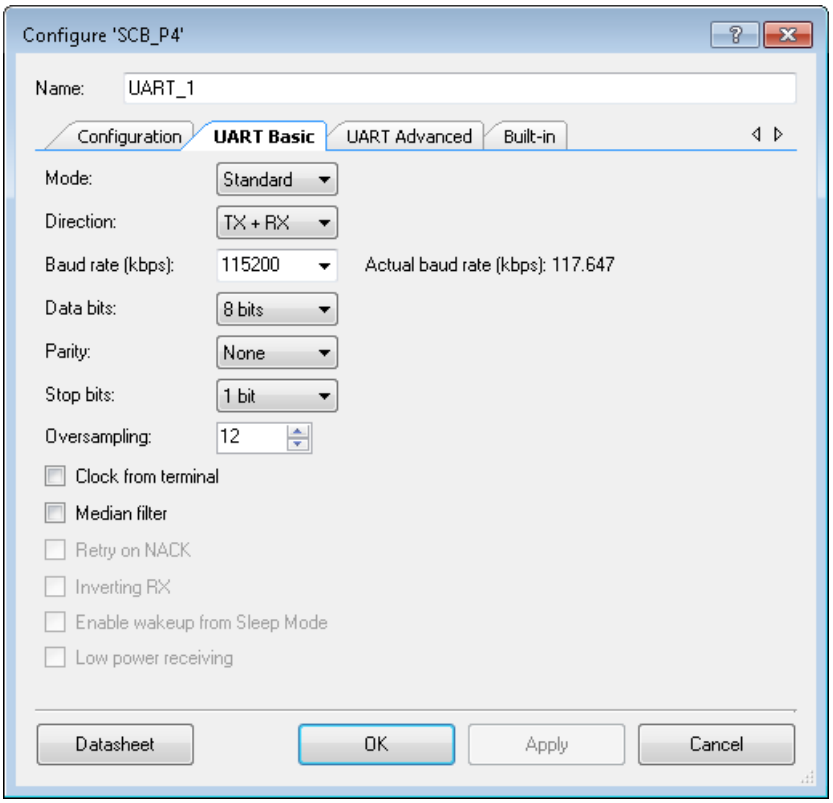
时钟负责运行此模块。根据**终端时钟**参数的不同，终端的存在位将各不相同。

中断 – 输出*

此信号只能与中断组件相连，或者不连接。根据**中断**参数的不同，终端的存在位将各不相同。

组件内部埋有特定接口的引脚，因为这些引脚使用的是专用连接，而且不可作为通用信号用途。欲获得更多信息，请见芯片*《技术参考手册》（TRM）*的 “I/O 系统” 部分。

基本 UART 参数



UART 基本选项卡包含下列参数：

模式

该选项用于决定 UART 的操作模式：标准、SmartCard 或 IrDA。默认模式为**标准**。

方向

该参数定义了要加入到 UART 组件中的功能模式。可设置为双向 **TX + RX**（默认），接收器（**仅适用于 RX**）或发送器（**仅适用于 TX**）。

波特率

该参数定义时钟生成硬件的波特率或位宽配置，最高可达 921600。实际波特率可能因可用时钟速度和分频器范围而不同。默认值为 **115200**。

数据位

该参数定义单个 UART 数据传输在开始到停止期间发送的数据位数。有以下几种选项：**5**、**6**、**7**、**8**（默认）或 **9**。

- 八个数据位为默认配置，即每次传输发送一个字节。
- 9 位模式并不表示发送 9 个数据位；第 9 位会取代奇偶校验位，作为地址或数据的指示符。

奇偶校验

该参数定义传输中奇偶校验位类型功能。可设置为**无**（默认）、**奇校验**或**偶校验**。

停止位

该参数定义发送器中实现的停止位数。该参数可设置为 **1**（默认）、**1.5** 或 **2** 个数据位。

过采样

该参数决定了接口的过采样率。默认值为 **16**。8 至 16 之间的任意整数都是有效设置。

终端时钟

终端时钟参数允许在内部时钟和外部时钟之间进行选择，以生成数据率。启用该选项时，组件不对数据速率进行控制但会基于用户连接的时钟源显示实际数据速率。不启用该选项时，考虑到过采样的问题，PSoC Creator 会基于**波特率**参数对要求的时钟频率进行计算和配置。

注意：当设置数据率或外部时钟频率值时，请确保 PSoC Creator 可以用当前系统时钟频率提供此值。否则，创建项目时会生成时钟精度范围警告。警告中包含 PSoC Creator 设置的实际时钟值。选择是应当更改系统时钟还是组件时钟以满足时钟设置系统要求、达到最佳值。

中值滤波器

该参数在 RX 线的输入路径上应用 3 抽头数字中值滤波。该滤波器可增强信号抗扰度。然而，过采样因子最小值将增加。默认值为**禁用**。

对 NACK 进行重试

该参数用于对 NACK 功能进行重试。收到否定确认时重新发送数据帧。该选项仅适用于 **SmartCard** 模式。

反转 RX

该参数用于对收到的 RX 线信号进行反转。该选项仅适用于 **IrDA** 模式。

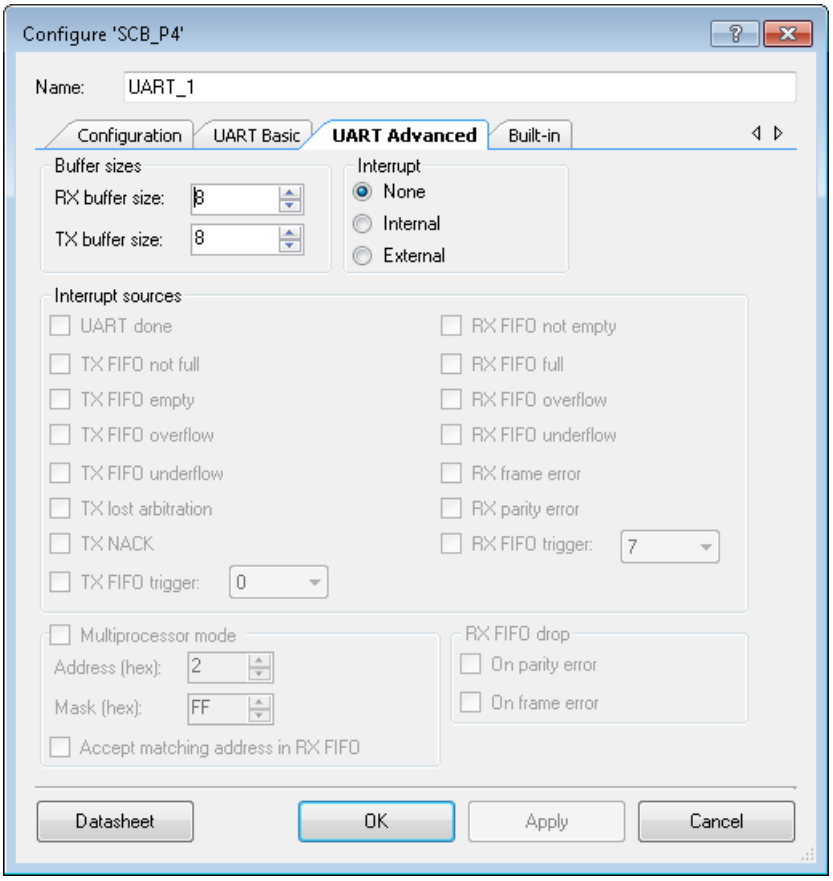
使能 “睡眠模式” 唤醒

该选项可在开始位将系统从睡眠中唤醒。该选项仅在启用 **RX Direction** 时适用。

低功耗接收

该参数用于启用 IrDA 低功耗接收模式。该选项仅在启用 **RX Direction** 时适用。

高级 UART 参数



RX 缓冲区大小

RX 缓冲区大小参数定义为循环接收数据缓冲区分配的存储器大小（以字节/字为单位）。如果此参数设置为 8，则在硬件中实现 RX FIFO 的第 8 字节/字。所有其他值（最大为 2^{32}）使用 8 字节/字 RX FIFO 和由提供的 API 和内部 ISR 控制的软件缓冲区。缓冲区大小受限于可用存储器。如果 RX 缓冲区大小超过 8 字节/字，中断模式自动设置为“内部”模式。

TX 缓冲区大小

TX 缓冲区大小参数定义为循环传输数据缓冲区分配的存储器大小（以字节/字为单位）。如果此参数设置为 8，则在硬件中实现 TX FIFO 的第 8 字节/字。所有其他值（最大为 2^{32}）使用 8 字节/字 TX FIFO 和由提供的 API 和内部 ISR 控制的软件缓冲区。缓冲区大小受限于可用存储器。如果 TX 缓冲区大小超过 8 字节/字，中断模式自动设置“内部”模式。

中断

此选项用于决定支持的中断模式：“无”、“内部” 或 “外部”。

- **无**—删除内部中断组件
- **内部**－此选项将中断组件留在 SCB 组件内部。预先定义的内部 ISR 连接到中断。可注册自定义函数，以调用 ISR 的每一个项目。**中断源**选项定义了中断事件，以触发中断。
- **外部**—此选项删除内部中断并提供输出端子。如果需要客户中断处理程序，中断组件可以与之连接。**中断源**选项可设置触发中断输出的中断源。

注意：对于大于 8 字节/字的缓冲区大小，组件自动启用正确内部软件缓冲区操作所需的内部中断源。此外，必须使能全局中断，才能进行正确的缓冲区处理。

中断源

UART 支持以下事件上的中断：

- **UART 完成**－UART 发送器完成事件：TX FIFO 中的所有数据帧被发送且 TX FIFO 为空
- **TX FIFO 未满**－TX FIFO 未满
- **TX FIFO 为空**－TX FIFO 为空
- **TX FIFO 溢出**－尝试写入到已满的 TX FIFO
- **TX 仲裁失败**－UART 仲裁失败：TX 线上的输出值与 RX 线上的观测值不相同。当发送器和接收器共享 TX/RX 线时，该条件事件十分有用。在 SmartCard 模式下，更是如此。
- **TX NACK**－UART 发送器在 SmartCard 模式下收到否定确认。
- **TX FIFO 下溢出**－尝试从空的 TX FIFO 中读取。
- **TX FIFO 触发**－当 TX FIFO 的条目少于该字段数量时，生成发送器触发事件。
- **RX FIFO 不为空**－RX FIFO 不为空。
- **RX FIFO 已满**－RX FIFO 已满。
- **RX FIFO 溢出**－尝试写入到已满的 RX FIFO。
- **RX FIFO 下溢出**－尝试从空的 RX FIFO 中读取。
- **RX 帧错误**－接收数据帧中包含帧错误。该错误可能是启动或停止位错误：
  - 启动位错误－检测到启动位周期开始后（RX 线从 “1” 变为 “0”），启动位周期中间采样错误（RX 线为 “1”）。

注意：启动位错误在接收到数据帧前进行检测。

- 停止位错误：RX 线采用为 “0”，但是期望值为 “1”。

  注意：停止位错误可能会导致接收连续数据帧失败。停止位错误在接收到数据帧后进行检测。

  注意：如果停止位持续时间等于一位，不对帧错误进行跟踪。

- **RX 奇偶校验错误**－收到的数据帧中包含奇偶校验错误。
- **RX FIFO 触发**－当 RX FIFO 的条目少于该字段数量时，生成发送器触发事件。

注意

当 **RX 缓冲区大小**超过 8 字节/字时，组件保留 **RX FIFO 不为空**中断源。

当 **TX 缓冲区大小**超过 8 字节/字时，组件保留 **TX FIFO 未满**中断。

多处理器模式

该参数用于启用多处理器模式，其中 9 位的第一位表示地址。默认值为**禁用**。要启用该选项须将数据位设置为 9 位。

地址 (hex)

从设备地址。用于在启用多处理器模式时进行匹配。默认值为 0x02。

掩码 (hex)

从设备地址掩码 在与从设备地址进行匹配时使用这些位。默认值为 **0xFF**。

- 位值为 0－该位不进行地址对比。
- 位值为 1－该位须与地址的相应位相互匹配。

接受 RX FIFO 中的匹配地址

该参数决定是否接受 RX FIFO 中的匹配地址。

注意：不匹配的地址不会存入 RX FIFO 中。

RX FIFO 丢弃

为 RX FIFO 提供硬件数据丢弃选项。

- **关于奇偶校验错误**－定义奇偶校验失败时的行为。通过奇偶校验时，收到的数据被发送至 RX FIFO。否则，收到的数据丢失。仅适用于 **标准**和 **SmartCard** 模式。

- **关于帧错误**－定义检测到帧错误时的行为。未捕获帧错误时，收到的数据被发送至 RX FIFO。否则，收到的数据丢失。

UART API

API 允许您利用软件对组件进行配置。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了每个函数。

默认情况下，PSoC Creator 将实例名称 “SCB_1” 分配给指定设计组件的第一个实例。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。出于可读性考虑，下表中使用的实例名称为 “SCB”。

函数	说明
SCB_Init()	根据自定义程序中的定义参数初始化 SCB 组件。
SCB_Enable()	启用 SCB 组件。
SCB_Start()	开始 SCB。
SCB_Stop()	禁用 SCB 组件。
SCB_Sleep()	准备组件进入深度睡眠。
SCB_Wakeup()	为组件退出深度睡眠做准备。
SCB_UartInit()	为 SPI 操作配置 SCB。
SCB_UartPutChar()	在下一个可用总线时间放置传输缓冲区中要发送的一字节数据。
SCB_UartPutString()	在下一个可用总线时间放置传输缓冲区中要发送的以空字符结尾的字符串。
SCB_UartPutCRLF()	向发送缓冲区放置一个字节的数据后，输入回车符 (0x0D) 和换行符 (0x0A)。
SCB_UartGetChar()	从接收缓冲区检索下一个数据元素。
SCB_UartGetByte()	从接收缓冲区检索下一个数据元素
SCB_UartSetRxAddress()	在多处理器模式中为 UART 设置可用硬件检测出的接收器地址。
SCB_UartSetRxAddressMask()	在多处理器模式中为 UART 设置硬件地址掩码。
SCB_SpiUartWriteTxData()	在传输缓冲区中放置将于下一个可用总线时间发送的数据输入。该函数对 SPI 和 UART 通用。
SCB_SpiUartPutArray()	将将要发送的数据数组置于传输缓冲区。该函数对 SPI 和 UART 通用。
SCB_SpiUartGetTxBufferSize()	返回当前传输缓冲区中元素的数量。该函数对 SPI 和 UART 通用。
SCB_SpiUartClearTxBuffer()	清除传输缓冲区和 TX FIFO。该函数对 SPI 和 UART 通用。
SCB_SpiUartReadRxData()	从接收缓冲区检索下一个数据元素。该函数对 SPI 和 UART 通用。
SCB_SpiUartGetRxBufferSize()	返回接收缓冲区中接收的数据元素的数量。该函数对 SPI 和 UART 通用。

函数	说明
SCB_SpiUartClearRxBuffer()	清除接收缓冲区和 RX FIFO。该函数对 SPI 和 UART 通用。

全局变量

正常运行情况下，不需获得这些变量。

变量	说明
SCB_initVar	SCB_initVar 表明 SCB 组件是否完成了初始化。该变量初始化为 0，并在第一次调用 SCB_Start() 时设置为 1。这样，第一次调用 SCB_Start() 子程序后，组件不用重新初始化即可重启。 如果组件需要重新初始化，则首先调用 SCB_Init() 函数，然后再调用 SCB_Start() 或 SCB_Enable() 函数。
SCB_rxBufferOverflow	当产生内部软件接收缓冲区溢出时，设置 SCB_rxBufferOverflow。

void SCB_Init(void)

说明： 使 SCB 组件初始化，以便在所选的配置之一中运行：I2C、SPI 或 UART。
设置为 “未配置的 SCB” 时，该函数不进行任何初始化操作。

参数： 无

返回值： 无

副作用： 无

void SCB_Enable(void)

说明： 使能 SCB 组件。组件启动时，不应更改 SCB 配置。禁用组件后，方可更改配置。
配置设置为 “Unconfigured SCB” 时，该函数不能启用组件。组件必须进行初始化，以便在之前配置的下列模式中运行：I2C、SPI 或 UART。

参数： 无

返回值： 无

副作用： 无

void SCB_Start(void)

说明： 调用 SCB_Init() 与 SCB_Enable()。调用该函数后，组件启动且运行准备就绪。

配置设置为 “Unconfigured SCB” 时，该函数不能启用组件。组件必须进行初始化，以便在之前配置的下列模式中运行：I2C、SPI 或 UART。

参数： 无

返回值： 无

副作用： 无

void SCB_Stop(void)

说明： 禁用 SCB 组件。

参数： 无

返回值： 无

副作用： 无

void SCB_Sleep(void)

说明： 准备组件进入深度睡眠。

“从睡眠模式使能唤醒” 的选择对此函数的使用有影响。

在调用 CyPmSysDeepSleep() 函数之前调用 SCB_Sleep() 函数。

进入睡眠模式前不得调用此函数。

有关功耗管理函数的详细信息，请参考《系统参考指南》中的 PSoC Creator 一节。

参数： 无

返回值： 无

副作用： 无

void SCB_Wakeup(void)

说明： 为组件退出深度睡眠做准备。

“从睡眠模式中唤醒” 的选择会影响此函数的实现。

进入睡眠模式前不得调用此函数。

参数： 无

返回值： 无

副作用： 在调用 SCB_Sleep() 函数之前，调用 SCB_Wakeup() 函数可能会导致意外行为。

void SCB_UartInit(SCB_UART_INIT_STRUCT *config)

说明： 配置 SCB 以进行 UART 操作。

本函数**专为**当自定义程序中的SCB配置设为 “未配置 SCB” 时使用。在 UART 模式中初始化 SCB 后，可通过 SCB_Start() 或 SCB_Enable() 函数启用该组件。

本函数对提供配置设置的结构使用了一个指针。该结构所含信息自定义程序设置同样可提供。

参数： config：针对含有如下字段排序列表的指针。这些字段匹配自定义程序的选项。设置的详细说明见自定义程序

字段	说明
mode	UART 操作模式。以下定义为可用选择： SCB_UART_MODE_STD SCB_UART_MODE_SMARTCARD SCB_UART_MODE_IRDA
direction	UART 操作方向。以下定义为可用选择： SCB_UART_MODE_TX_RX SCB_UART_MODE_RX SCB_UART_MODE_TX
dataBits	数据位数量
parity	决定奇偶校验。以下定义为可用选择： SCB_UART_PARITY_EVEN SCB_UART_PARITY_ODD SCB_UART_PARITY_NONE
stopBits	决定停止位的数量。以下定义为可用选择： SCB_UART_STOP_BITS_1 SCB_UART_STOP_BITS_1_5 SCB_UART_STOP_BITS_2
oversample	UART 过采样因子。 注意： 启用 enableIrdaLowPower 时，过采样因子值被更改。 SCB_UART_IRDA_LP_OVS16 SCB_UART_IRDA_LP_OVS32 SCB_UART_IRDA_LP_OVS48 SCB_UART_IRDA_LP_OVS96 SCB_UART_IRDA_LP_OVS192 SCB_UART_IRDA_LP_OVS768 SCB_UART_IRDA_LP_OVS1536
enableIrdaLowPower	启用 IrDA 低功耗 RX 模式。 0 – 禁用 1 – 启用 当使能时，TX 功能失效。
enableMedianFilter	0 – 禁用 1 – 启用
enableRetryNack	0 – 禁用 1 – 启用 除 SmartCard 外的其他模式被忽略。

enableInvertedRx	0 – 禁用 1 – 启用 除 IrDA 外的其他模式被忽略。
dropOnParityErr	如果检测到奇偶校验错误，将 RX FIFO 中的数据丢失。 0 – 禁用 1 – 启用
dropOnFrameErr	如果检测到帧错误，将 RX FIFO 中的数据丢失。 0 – 禁用 1 – 启用
enableWake	0 – 禁用 1 – 启用 除标准模式外的其他模式被忽略。 UART。须启用 RX 功能。
rxBufferSize	字中的 RX 缓冲区大小： 值“8”代表硬件中缓冲的用途。 如果值超过“8”将产生软件缓冲区。软件 RX 缓冲区始终保持一个元素为空。缓冲区大小须为较大元素以便对接收缓冲区大小的整个数据包进行接收。
rxBuffer	为 RX 软件缓冲区提供的缓冲区空间： 须提供字中的 rxBufferSize 缓冲区。 如果 dataBitsRx 大于 8，须提供 (2* rxBufferSize) 字节。 如果软件缓冲有指示，须提供 NULL 指针。
txBufferSize	字中的 TX 缓冲区大小： 值“8”代表硬件中缓冲的用途。 值大于 8 表示在软件中缓冲。
txBuffer	为 RX 软件缓冲区提供的缓冲区空间： 须提供字中的 rxBufferSize 缓冲区。 如果 dataBitsRx 大于 8，须提供 (2* rxBufferSize) 字节。 如果软件缓冲有指示，须提供 NULL 指针。
enableMultiproc	启用多处理器模式。 0 – 禁用 1 – 启用
multiprocAddr	要在多处理器模式中匹配的 8 位地址。忽略其他模式。
multiprocAddrMask	为匹配多处理器地址进行比较的地址位的 8 位掩码。忽略其他模式。
enableInterrupt	0 – 禁用 1 – 启用

rxInterruptMask	要在 RX 方向中启用的中断源掩码。此掩码的写入不论 enableInterrupt 字段是何设置。通过提供以下要启用的所有源的 逻辑或 值使能多个中断源： SCB_INTR_RX_TRIGGER SCB_INTR_RX_NOT_EMPTY SCB_INTR_RX_FULL SCB_INTR_RX_OVERFLOW SCB_INTR_RX_UNDERFLOW SCB_INTR_RX_FRAME_ERROR SCB_INTR_RX_PARITY_ERROR
rxTriggerLevel	RX 触发中断的 FIFO 等级。不论是否启用 RX 触发中断，均写入该值。
txInterruptMask	要在 TX 方向中启用的中断源掩码。此掩码的写入不论 enableInterrupt 字段是何设置。通过提供以下要启用的所有源的 逻辑或值启用多个源： SCB_INTR_TX_TRIGGER SCB_INTR_TX_NOT_FULL SCB_INTR_TX_EMPTY SCB_INTR_TX_OVERFLOW SCB_INTR_TX_UNDERFLOW SCB_INTR_TX_UART_DONE SCB_INTR_TX_UART_NACK SCB_INTR_TX_UART_ARB_LOST
txTriggerLevel	TX 触发中断的 FIFO 等级。不论是否启用 TX 触发中断，均写入该值。

返回值： 无

副作用： 无

void SCB_UartPutChar(uint32 txDataByte)

说明： 在下一个可用总线时间放置传输缓冲区中要发送的一字节数据。该函数被禁用并等待直至有可用的空间可放置要求数据到传输缓冲区中。

在 UART 多处理器模式中可使用该函数发送 9 位数据。使用 SCB_UART_MP_MARK 添加掩码以便创建地址字节。

参数： uint32 txDataByte：要传输的数据。

返回值： 无

副作用： 无

void SCB_UartPutString(const char8 string[])

说明： 在下一个可用总线时间放置传输缓冲区中要发送的以空字符结尾的字符串。
该函数被禁用并等待直至有可用的空间可放置要求数据到传输缓冲区中。

参数： const char8 string[]：指向要放置在传输缓冲区中的以空字符结尾的字符串阵列。

返回值： 无

副作用： 无

void SCB_UartPutCRLF(uint32 txDataByte)

说明： 向发送缓冲区放置一个字节的数据后，输入回车符 (0x0D) 和换行符 (0x0A)。
该函数被禁用并等待直至有可用的空间可放置要求数据到传输缓冲区中。

参数： uint32 txDataByte：要传输的数据。

返回值： 无

副作用： 无

uint32 SCB_UartGetChar(void)

说明： 从接收缓冲区检索下一个数据元素。该函数专为 ASCII 字符设计并返回一个字符。其中 1 至 255 之间均为有效字符，0 代表出错或没有显示数据。

- 禁用 RX 软件缓冲区：返回从 RX FIFO 检索的数据元素。
- 启用 RX 软件缓冲区：从软件接收缓冲区返回数据元素。

参数： 无

返回值： uint32：来自接收缓冲区的下一个数据元素。1 至 255 之间的 ASCII 字符值为有效字符值。返回零则表示出现错误状况或数据不可用。

副作用： 由于使用 RX FIFO 和软件缓冲区，错误位可能与读取字符不相符。
禁用 RX 软件缓冲区：内部软件缓冲区溢出不做为错误条件。检查 SCB_rxBufferOverflow 以捕获错误条件。

uint32 SCB_UartGetChar(void)

说明： 从接收缓冲区检索下一个数据元素并返回接收字节和错误条件。

禁用 RX 软件缓冲区：返回从 RX FIFO 检索的数据元素。如果 RX FIFO 为空，将返回未定义数据。

启用 RX 软件缓冲区：从软件接收缓冲区返回数据元素。

参数： 无

返回值： uint32：位 15-8 存放状态，位 7-0 存放来自接收缓冲区的下一个数据元素。如果位 15-8 非零，表示已出错。

副作用： 由于使用 RX FIFO 和软件缓冲区，错误位可能与读取字符不相符。

禁用 RX 软件缓冲区：内部软件缓冲区溢出不做为错误条件。检查 SCB_rxBufferOverflow 以捕获错误条件。

void SCB_UartSetRxAddress(uint32 address)

说明： 在多处理器模式中为 UART 设置硬件可检测的接收器地址。

参数： uint32 address：用于硬件地址检测的地址。

返回值： 无

副作用： 无

void SCB_UartSetRxAddressMask(uint32 addressMask)

说明： 在多处理器模式中为 UART 设置硬件地址掩码。

参数： uint32 addressMask：地址掩码

位值为 0 －该位不进行地址对比。

位值为 1—该位须与地址的相应位相互匹配。

返回值： 无

副作用： 无

void SCB_SpiUartWriteTxData(uint32 txDataByte)

说明： 在传输缓冲区中放置将于下一个可用总线时间发送的数据输入。数据传输方向为最低有效位 (LSB)。

此函数进行阻止操作，并等待直到传输缓冲区有可用空间存放请求数据。

在 UART 多处理器模式中可使用该函数发送 9 位数据。使用 SCB_UART_MP_MARK 添加掩码以便创建地址字节。

参数： uint32 txDataByte：要传输的数据。

返回值： 无

副作用： 无

void SCB_SpiUartPutArray(const uint16/uint8 wrBuf[], uint32 count)

说明： 将要发送的数组放入传输缓冲区

此函数执行阻止操作，并等待直到传输缓冲区有可用空间存放所有请求数据。

阵列大小可超过传输缓冲区的大小。

参数： const uint16/uint8 wrBuf[]：指向要放置在传输缓冲区中的数组。

uint32 count：要放置在传输缓冲区中的数据元素的数量。

返回值： 无

副作用： 无

uint32 SCB_SpiUartGetTxBufferSize(void)

说明： 返回传输缓冲区中当前元素的数量。

- 禁用的 TX 软件缓冲区：返回 TX FIFO 中用到的条目数量。
- 启用的 TX 软件缓冲区：返回传输缓冲区中当前用到的元素数量。该数目不包含 TX FIFO 中用到的条目数量。传输缓冲区的大小在 TX FIFO 已满前为零。

参数： 无

返回值： uint32：准备传输的数据元素的数量。

副作用： 无

void SCB_SpiUartClearTxBuffer(void)

说明： 清除传输缓冲区和 TX FIFO。

参数： 无

返回值： 无

副作用： 无

uint32 SCB_SpiUartReadRxData(void)

说明： 从接收缓冲区中检索下一个数据元素。

- 禁用 RX 软件缓冲区：返回从 RX FIFO 检索的数据元素。如果 RX FIFO 为空，将返回未定义的数据。
- 启用 RX 软件缓冲区：从软件接收缓冲区返回数据元素。如果接收软件缓冲区为空，将返回零值。

参数： 无

返回值： uint32：来自接收缓冲区的下一个数据元素。

副作用： 无

uint32 SCB_SpiUartGetRxBufferSize(void)

说明： 返回接收缓冲区中接收数据的数量。

禁用的 RX 软件缓冲区：返回 TX FIFO 中用到的条目数量。

启用的 RX 软件缓冲区：返回放置在接收缓冲区中的元素的数量。

参数： 无

返回值： uint32：接收到的数据元素的数量

副作用： 无

void SCB_SpiUartClearRxBuffer(void)

说明： 清除接收缓冲区和 RX FIFO。

参数： 无

返回值： 无

副作用： 无

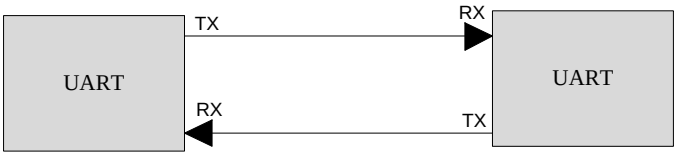
UART 功能描述

通用异步发送器/接收器 (UART) 协议为异步串行接口。UART 的发送和接收接口由两种信号组成：

- **TX**－发送器
- **RX** －接收器

注意：SCB 不支持与流量控制相关联的 RS232 边带信号，例如 DTR（数据终端准备）和 DCD（数据载波检测）等。

图 11UART 典型连接



标准模式操作

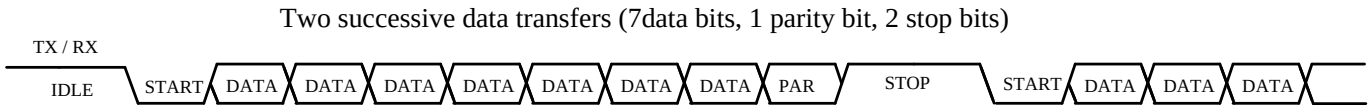
标准 UART 通过 “点对点” 拓扑结构定义。

典型的 UART 传输位由“开始位”、多个 “数据位”、可选的 “奇偶检验位” 和一个或多个 “停止位” 组成。“开始位” 值始终为 “0”；“数据位” 值取决于传输的数据；“奇偶校验位” 值设置为可保证对 “数据位” 进行奇数或偶数校验的值；“停止位” 值为 “1”。“奇偶校验位” 由发送器生成，可被接收器用来检测单个位传输错误。不传输数据时，TX 线为 “1”，也就是与 “停止位” 值相同。

可通过在 TX 线上将 “1” 变为 “0” 实现从 “停止位” 到 “开始位” 的切换。接收器可利用这种切换与发送器的时钟保持同步。在每次数据传输开始时进行同步，这样即使在发送器和接收器时钟间出现频率偏移时也可实现无差错的数据传输。要求的时钟精度取决于数据传输大小。

停止周期或连续数据传输间的“停止位”数量通常在发送器和接收器之间达成一致，通常在 1-3 个传输周期范围内。

图 12 UART 协议



UART 第 9 个数据位的使用

第 9 位被送到奇偶校验位的位置，它常被用于定义发送的数据是否是地址还是标准数据。奇偶校验位中的标记 (1) 表示发送的是地址，奇偶校验位中的空格 (0) 表示发送的是数据。数据流为 “起

始位、数据位、奇偶校验、停止位”，与其他奇偶校验模式类似，但该位在传输前必须由用户软件控制，而非根据数据位的值计算得出。

```
tx_data = 0x31;
tx_data |= 0x100; /* Set 9th bit to indicate 'address' */
UART_SpiUartWriteTxData(tx_data)
```

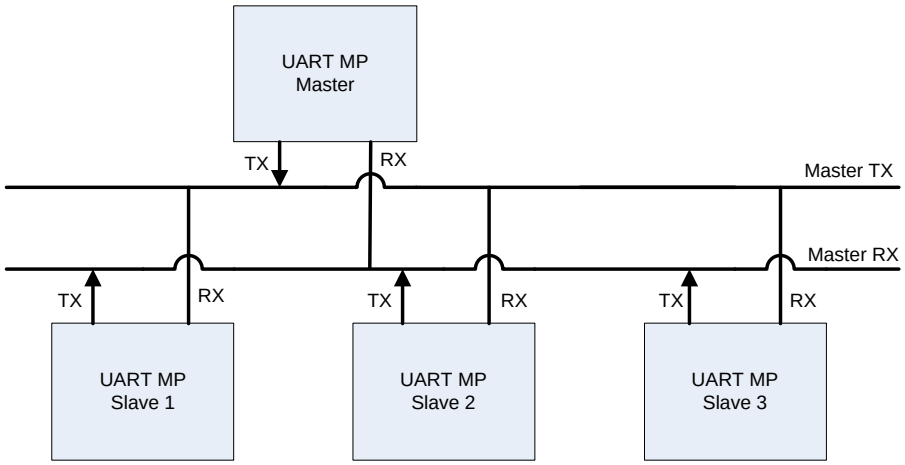
多处理器模式运行

用 “单一主设备多从设备” 拓扑结构对此模式进行定义。多处理器模式又称 UART 9 位协议，标准 UART 协议使用 5 到 8 位数据段。

多处理器模式的主要性能是：

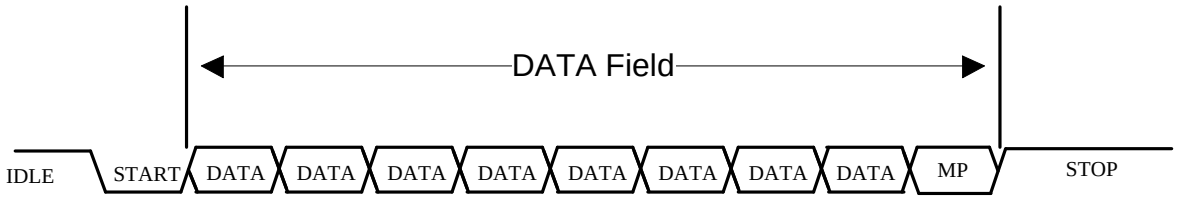
- 单个拥有多个从设备的主设备（多分支网络）
- 每个从设备都由独特的地址进行识别
- 使用 9 位数据段，把第 9 位（MSB）作为地址/数据标志。设为“1”时，它表示地址字节；设为 “0” 时，它表示数据字节。
- 奇偶校验位被禁用

图 13 多处理器总线连接



为了使多处理器模式能按下面的选项配置 UART：**模式**：标准，**数据位**： 9 位，**奇偶校验**：无。

图 14 多处理器模式中的 UART 数据帧



因为多分支网络的数据链路层是一个用户定义协议，所以它为数据帧的格式提供了一种灵活的定义方法。

地址帧中的所有位都能用来表示从设备地址。另外，一些位可表示地址，而其他位可表示对从设备的命令，还有一些位可表示以下数据帧中数据的长度。

在多处理器模式中，SCB 可用作主设备或从设备。

当 UART 充当从设备时。将收到的地址与**地址**和**掩码**的逻辑或结果匹配。**在 RX FIFO 中接受匹配地址**检查后，匹配后的地址写在 RX FIFO 中。如果有匹配，随后收到的数据发送到 RX FIFO。如果没有匹配，随后收到的数据会被丢失，直到收到下一个地址作对比。

SmartCard (ISO7816) 模式运行

ISO7816 是异步串行接口，用 “单主设备单从设备” 拓扑结构进行定义。组件中只支持主设备（读卡器）函数。

SCB 用异步字符传输提供基础物理层支持，且只提供标准 ISO7816[3] 引脚列表中的 “I/O” 引脚。通过在 TX 和 RX 控制模块间内部复用，SCB 中 UART 的 TX 线将与 SmartCard 的 I/O 线相连接。

更高级的协议实现留给来自用户层面的固件来处理。

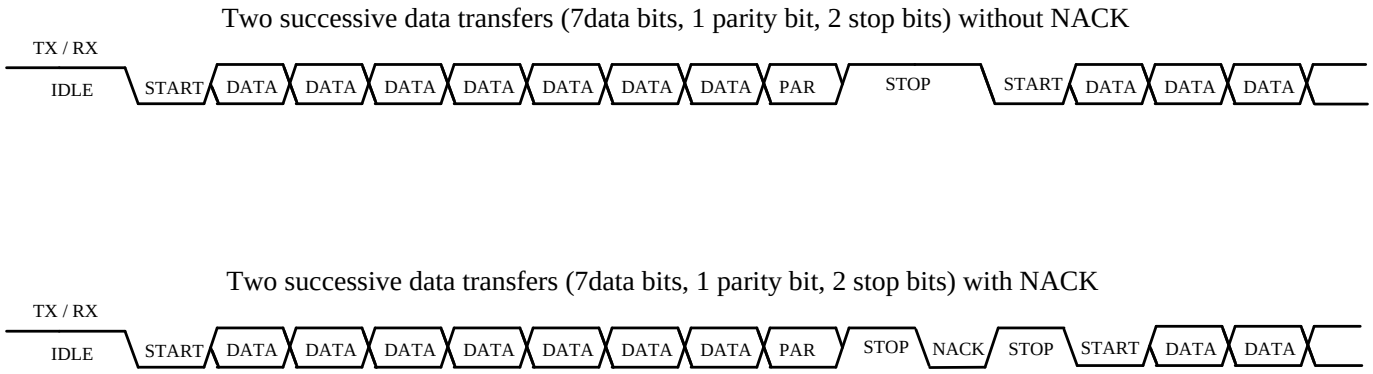
SmartCard 数据传输

SmartCard 传输与 UART 传输类似，外加可能从接收器发送到发送器的否定确认 (NACK)。NACK 总是设为 “0”。发送器和接收器在不同时刻驱动同一 I/O 线路。图 15 阐释了 SmartCard 协议。

通常，协议的实现使用的是带上拉电阻的三态驱动器，这样当线路未被驱动时，其值为 “1”（该值与没传输数据时或 “停止位” 的值相同）。

[3]请参考 *ISO/IEC 7816-3:2006－标识卡－集成电路卡－第 3 部分：带触点的卡－电气接口及传输协议* (1997)，参见 ISO 网站 www.iso.org。

图 15 SmartCard 数据传输示例



SmartCard 传输中让发送器驱动“起始位”、“数据位” 和 “奇偶校验位”。这些位之后，通过释放总线进入停止期。线路中释放结果为 “1”（即 “停止位” 的值）。半个位的传输期进入停止期后，接收器可能驱动线路上的 NACK （“0” 的值）一个到两个位的传输期。发送器观察到此 NACK，通过延长停止期的方式做出反应，延长时间为一个位的传输期。要想让此协议奏效，停止期应大于一个位的传输期。

注意：带 NACK 的数据传输比不带 NACK 的数据传输所用的时间长一个位的传输期。

协议 T=0 和 T=1

ISO7816 规范中有两种传输协议，T=0（异步字符的半双工传输）和 T=1（异步模块的半双工传输）。在物理层中，T=1 模块是用异步字符实现的。

复位应答 (ATR)

根据定义，复位应答是通过卡（从设备）发送到接口设备（读卡器或主设备）的字节的序列值，作为对复位的回应。在 I/O 线路上，每个字节都在异步字符中传输。

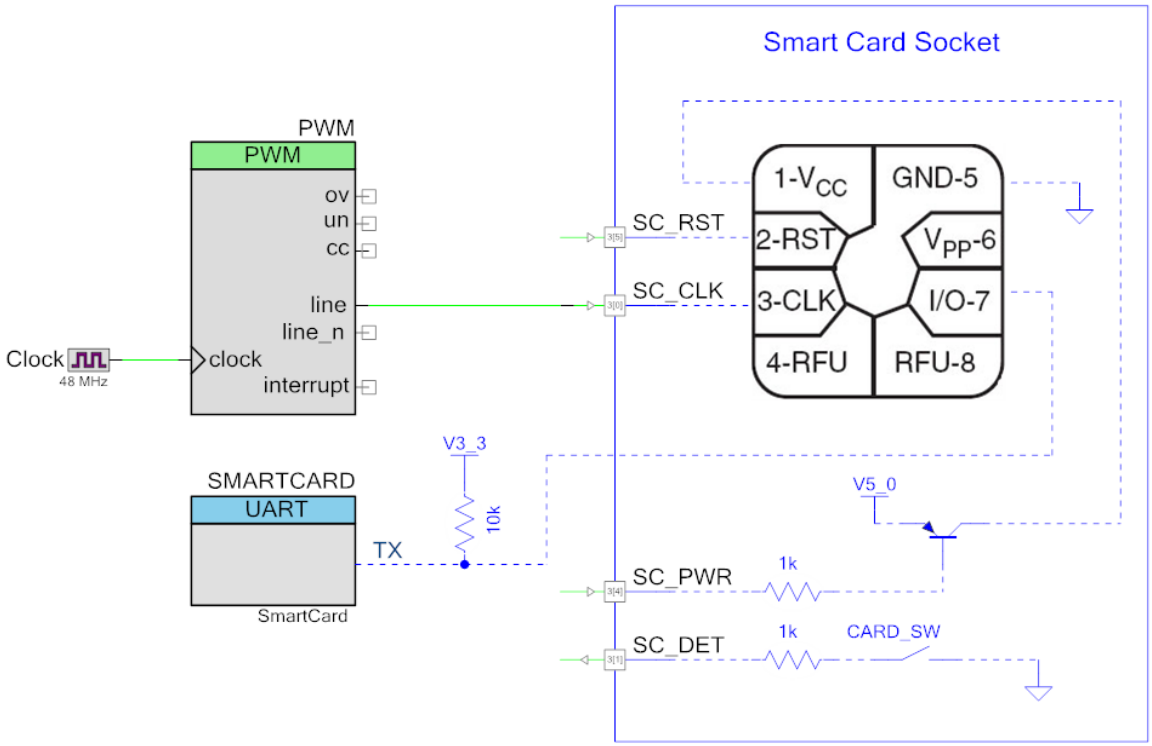
通信设置的过程（ATR，复位应答）、PPS（协议及参数选择）、操作条件类型的选择、操作模式选择和转换、NACK 上的再传输以及其他高级协议的实现都留给用户固件处理。

SmartCard 读卡器实现示例

您必须考虑如何用其他可用的系统资源为 “RST” 信号、“CLK” 信号、卡检测信号和卡电源控制信号实现一个完整的 SmartCard 系统。

图 16 是用 TCPWM 和引脚组件实现 SmartCard 读卡器功能的示例。

图 16 SmartCard 读卡器实现示例



UART 与 I/O 卡触点相连接。必须把上拉电阻连接到此线路。SC_RST、SC_CLK 为标准卡触点。SC_DET 由于卡片插入检测。SC_PWR 用于控制卡电源的开或关。请参考 *ISO7816 规范*了解更多详细信息。

IrDA 模式运行

用“点对点”拓扑结构定义 IrDA。SCB 仅从基础物理层为[4] IrDA 提供支持，速率为 1200 bps 到 115200 bps。物理层负责定义数据传输的硬件收发器。更高级的协议实现留给来自用户层面的固件来处理。

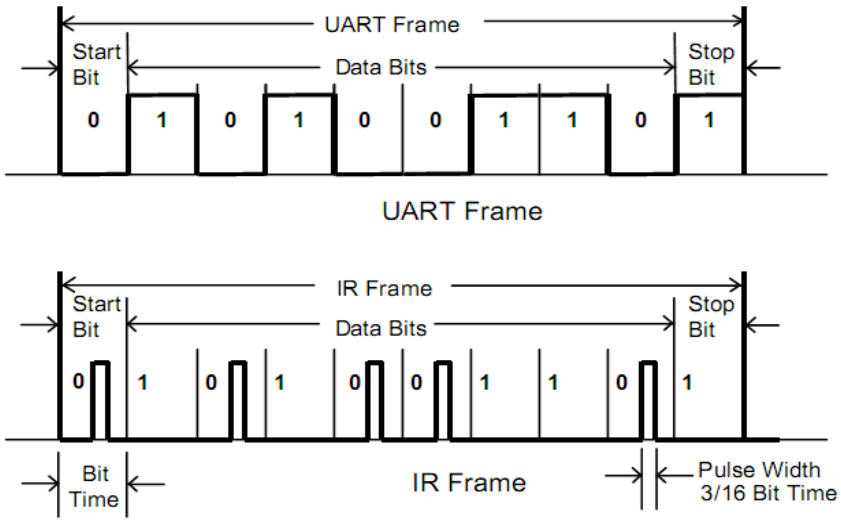
IrDA 对传输速率的最低求仅为 9600 bps。所有的传输都必须从这个速率开始以确保兼容性。更高的速率是建立链接后端口协商的问题。

IrDA 协议给 UART 信号增加了一种调制方案。在发送器端，对比特位进行调制。在接收器端，对比特位进行解调。调制方案使用的是反相归零 (RZI) 格式。线路上短的 “1” 脉冲表示的比特值 “0”，线路上 “0” 脉冲表示的比特值 “1”。IrDA 使用 3/16 RZI 调制。

图 17 展示了 UART 帧和 IR 帧，均包含 1 个起始位和 8 个数据位，没有奇偶校验位，以停止位结束。

[4] 请参考 *IrPHY（IrDA 物理层链接规范）*（修订版 1.4，2001 年 5 月），参见 IrDA 网站 www.irda.org

图 17 UART 帧和 IR 帧示例



过采样选择

IrDA 使用 3/16 RZI 调制，因此通过配置**过采样**，采样时钟的频率应设置为所选**波特率**的 16 倍。**IrDA 的过采样**应一直为 16。

标准传输与低功耗传输

IrDA 运行模式有两种：

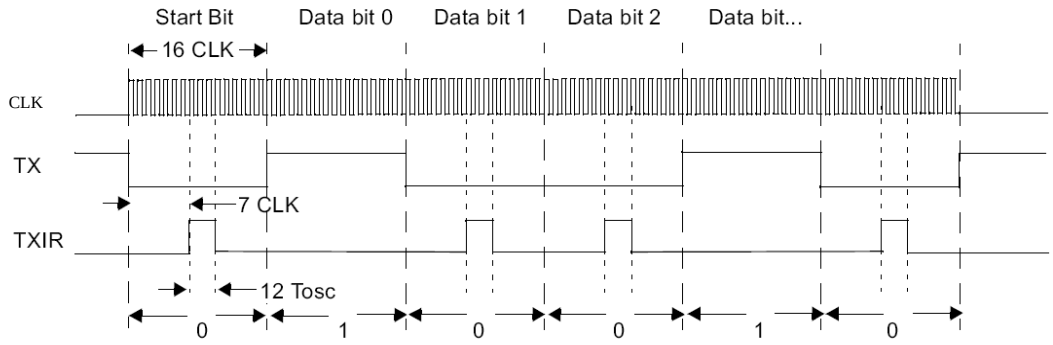
- **标准传输**－脉冲宽度约为位周期的 3/16（所有波特率）
- **低功耗传输**－脉冲宽度可能比位周期的 3/16 更短（通常降至 1.62 µs，最小 1.41 µs）（针对小于 115200 bps 的波特率）。仅支持 **RX** 方向。

反转 RX

此选择用于支持下述两种可能的解调方案。

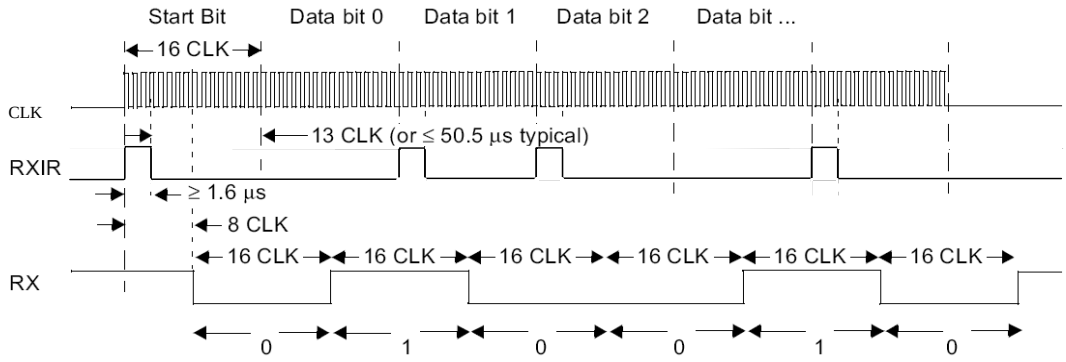
按照 IrPHY 规范，IR 帧调制（编码）方案如图 18 所示。

图 18 IR 帧调制方案



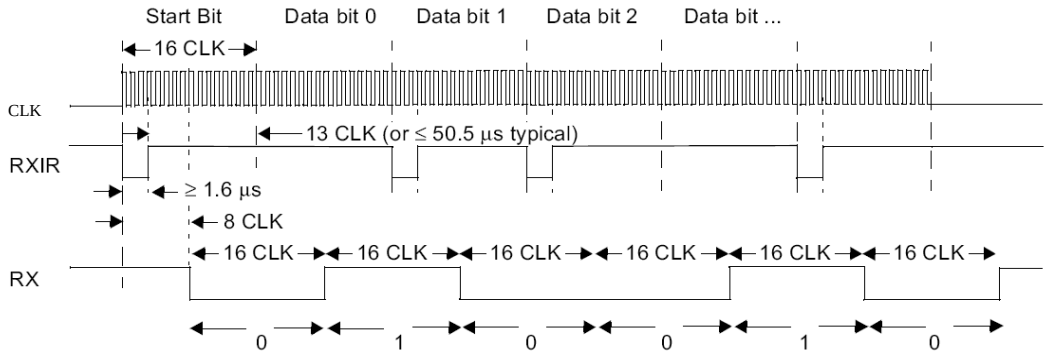
IR 帧解调（解码）方案如图 19 所示。RXIR 线路的电压电平为低电平预设，高电平有效。

图 19 IR 帧解调方案 1（高电平有效）



在应用中，来自 IrDA 收发器的 RXIR 帧输出通常是上拉式，高电平预设，低电平有效。图 20 表示另一种解调方案。

图 20 IR 帧解调方案 2（低电平有效）



软件缓冲器

SCB 有 FIFO 存储器，FIFO 存储器是一种 16 词 x 16 位的 SRAM，允许写入字节。在 UART 模式中，FIFO 被拆分为 TX FIFO 和 RX FIFO。各含 8 个条目，每个条目 16 位。每个条目的 16 位宽度用于调节可配置的数据宽度。

内部中断处理程序用于提供软件和硬件 TX/RX 缓冲间的互动，而不对您的顶级固件进行任何更改。

您也应当考虑到使用软件缓冲会导致被传输的字之间计时间隔更大，因为中断处理程序需要额外的时间来执行（取决于所选 HFCLK 的值）。

中断

当 **RX 缓冲区大小**或 **TX 缓冲区大小**超过 8 字节/字时，**RX FIFO 非空**和 **TX FIFO 未满**中断源通过内部软件缓冲操作组件保留。**请勿清除或禁用它们，**因为这样做会造成错误的软件缓冲操作。但清除来自其他源的中断是用户的责任，因为它们不会被自动清除。那是客户处理程序函数的功能。

假如 **RX 缓冲区大小**或 **TX 缓冲区大小**小于或等于 8 字节/字，使用的不是软件缓冲，而只是硬件 TX 或 RX FIFO。在**内**中断模式中，不会自动清除中断。这是用户的责任。这种情况下应首选**外**中断模式。

低功耗模式

设备通过由引入起始位产生的 RX GPIO 下降沿事件唤醒。唤醒受两大限制：

- 唤醒数据操作的第 1 个数据位必须为 “1”。UART 跳过起始位并在第 1 个数据位同步。
- 设备的唤醒时间必须小于一个位的持续时间。在其它情况下收到的数据不正确。

- **注意**：RX GPIO 中断限制使用来自放置 UART rx 引脚的端口的所有 GPIO 中断。

常用 SCB 组件信息

中断 API

这些函数是大多数 SCB 模式的常用函数。

默认情况下，PSoC Creator 将实例名称 “SCB_1” 分配给指定设计组件的第一个实例。您可以将其重命名为遵循标识符语法规则的任何唯一值。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。为便于阅读，下表使用的实例名称为 “SCB”。

函数	说明
SCB_EnableInt()	当使用内部中断时，将启用 NVIC 中断。
SCB_DisableInt()	当使用内部中断时，将禁用 NVIC 中断。
SCB_GetInterruptCause()	返回显示当前触发中断源的位掩码。
SCB_SetCustomInterruptHandler()	注册一个由内部中断处理程序调用的函数。
SCB_SetTxInterruptMode()	配置触发中断事件的 TX 中断请求寄存器的位。
SCB_GetTxInterruptMode()	返回 TX 中断掩码
SCB_GetTxInterruptSourceMasked()	返回由中断掩码屏蔽的 TX 中断请求寄存器
SCB_GetTxInterruptSource()	返回待处理 TX 中断源的位掩码
SCB_ClearTxInterruptSource()	清除待处理 TX 中断源的位掩码
SCB_SetTxInterrupt()	从 TX 中断源的位掩码生成中断事件
SCB_SetRxInterruptMode()	配置哪些位的 RX 中断请求寄存器位才可触发中断事件
SCB_GetRxInterruptMode()	返回 RX 中断掩码
SCB_GetRxInterruptSourceMasked()	返回由中断掩码屏蔽的 RX 中断请求寄存器
SCB_GetRxInterruptSource()	返回待处理 RX 中断源的位掩码
SCB_ClearRxInterruptSource()	清除待处理 RX 中断源的位掩码
SCB_SetRxInterrupt()	从 RX 中断源的位掩码生成中断事件
SCB_SetMasterInterruptMode()	配置多少位主设备中断请求寄存器才可触发中断事件。
SCB_GetMasterInterruptMode()	返回主设备中断掩码
SCB_GetMasterInterruptSourceMasked()	返回由中断掩码屏蔽的主设备中断请求寄存器
SCB_GetMasterInterruptSource()	返回待处理主设备中断源的位掩码
SCB_ClearMasterInterruptSource()	清除待处理主设备中断源的位掩码
SCB_SetMasterInterrupt()	从主设备中断源的位掩码生成中断事件
SCB_SetSlaveInterruptMode()	配置多少位的从设备中断请求寄存器才可触发中断事件
SCB_GetSlaveInterruptMode()	返回从设备中断掩码
SCB_GetSlaveInterruptSourceMasked()	返回被中断掩码屏蔽的从设备中断请求寄存器
SCB_GetSlaveInterruptSource()	返回待处理从设备中断源的位掩码
SCB_ClearSlaveInterruptSource()	清除待处理从设备中断源的位掩码
SCB_SetSlaveInterrupt()	用组件中断源的位掩码生成中断事件

函数	说明
SCB_SetI2CExtClkInterruptMode()	配置外部时钟模式的 I2C 中断请求寄存器的哪些比特位才可触发中断事件
SCB_GetI2CExtClkInterruptMode()	返回外部计时的 I2C 中断掩码
SCB_GetI2CExtClkInterruptSourceMasked()	返回由中断掩码屏蔽的外部计时的 I2C 中断请求寄存器
SCB_GetI2CExtClkInterruptSource()	返回待处理外部计时的 I2C 中断源的位掩码
SCB_ClearI2CExtClkInterruptSource()	清除待处理的外部计时的 I2C 中断源的位掩码
SCB_SetSpiExtClkInterruptMode()	配置哪些外部时钟模式的 SPI 中断请求寄存器的比特位才可触发中断事件
SCB_GetSpiExtClkInterruptMode()	返回外部计时的 SPI 中断掩码
SCB_GetSpiExtClkInterruptSourceMasked()	返回由中断掩码屏蔽的外部计时的 SPI 中断请求寄存器
SCB_GetSpiExtClkInterruptSource()	返回待处理外部计时的 SPI 中断源的位掩码
SCB_ClearSpiExtClkInterruptSource()	清除待处理的外部计时的 SPI 中断源的位掩码

中断函数应用

函数	I2C	SPI	UART
SCB_EnableInt()	+	+	+
SCB_DisableInt()	+	+	+
SCB_GetInterruptCause()	+	+	+
SCB_SetCustomInterruptHandler()	+	+	+
SCB_SetTxInterruptMode()	+	+	+
SCB_GetTxInterruptMode()	+	+	+
SCB_GetTxInterruptSourceMasked()	+	+	+
SCB_GetTxInterruptSource()	+	+	+
SCB_ClearTxInterruptSource()	+	+	+
SCB_SetTxInterrupt()	+	+	+
SCB_SetRxInterruptMode()	+	+	+
SCB_GetRxInterruptMode()	+	+	+
SCB_GetRxInterruptSourceMasked()	+	+	+
SCB_GetRxInterruptSource()	+	+	+
SCB_ClearRxInterruptSource()	+	+	+

函数	I2C	SPI	UART
SCB_SetRxInterrupt()	+	+	+
SCB_SetMasterInterruptMode()	+	+	–
SCB_GetMasterInterruptMode()	+	+	–
SCB_GetMasterInterruptSourceMasked()	+	+	–
SCB_GetMasterInterruptSource()	+	+	–
SCB_ClearMasterInterruptSource()	+	+	–
SCB_SetMasterInterrupt()	+	+	–
SCB_SetSlaveInterruptMode()	+	+	–
SCB_GetSlaveInterruptMode()	+	+	–
SCB_GetSlaveInterruptSourceMasked()	+	+	–
SCB_GetSlaveInterruptSource()	+	+	–
SCB_ClearSlaveInterruptSource()	+	+	–
SCB_SetSlaveInterrupt()	+	+	–
SCB_SetI2CExtClkInterruptMode()	+	–	–
SCB_GetI2CExtClkInterruptMode()	+	–	–
SCB_GetI2CExtClkInterruptSourceMasked()	+	–	–
SCB_GetI2CExtClkInterruptSource()	+	–	–
SCB_ClearI2CExtClkInterruptSource()	+	–	–
SCB_SetSpiExtClkInterruptMode()	–	+	–
SCB_GetSpiExtClkInterruptMode()	–	+	–
SCB_GetSpiExtClkInterruptSourceMasked()	–	+	–
SCB_GetSpiExtClkInterruptSource()	–	+	–
SCB_ClearSpiExtClkInterruptSource()	–	+	–

void SCB_EnableInt(void)

说明： 当使用内部中断时，将使能 NVIC 的中断。当使用外部中断时，必须用中断组件的 API 来启用中断。

参数： 无

返回值： 无

副作用： 无

void SCB_DisableInt(void)

说明： 当使用内部中断时，将禁用 NVIC 的中断。当使用外部中断时，必须用中断组件的 API 来禁用中断。

参数： 无

返回值： 无

副作用： 无

uint32 SCB_GetInterruptCause(void)

说明： 返回显示当前触发中断源的位掩码。根据多个中断寄存器的条件生成中断对该操作模式有用。

参数： 无

返回值： uint32：已触发包含下列条件OR的掩码：

中断原因常量	说明
SCB_INTR_CAUSE_MASTER	来自主设备的中断
SCB_INTR_CAUSE_SLAVE	来自从设备的中断
SCB_INTR_CAUSE_TX	来自 TX 的中断
SCB_INTR_CAUSE_RX	来自 RX 的中断

副作用： 无

void SCB_SetCustomInterruptHandler(void (*func) (void))

说明： 注册一个由内部中断处理程序调用的函数。首先调用已注册函数，然后内部中断处理程序在中断返回之前执行软件缓存区管理等操作。用户不可中断软件缓存区操作。仅支持一个是自定义处理程序，即：最近期调用提供的函数。初始化期间，未注册任何自定义处理程序。

参数： func：指向要寄存的函数的指针。NULL （空）值表示删除当前自定义中断处理程序。

返回值： 无

副作用： 无

void SCB_SetTxInterruptMode(uint32 interruptMask)

说明： 配置哪些 TX 中断请求寄存器的比特位才可触发中断事件

参数： uint32 interruptMask：将被启用的 TX 中断源的位掩码。

中断原因常量	说明
SCB_INTR_TX_TRIGGER	发送器 FIFO 中的输入条目少于触发电平指定的值
SCB_INTR_TX_NOT_FULL	发送器 FIFO 未满
SCB_INTR_TX_EMPTY	发送器 FIFO 为空
SCB_INTR_TX_OVERFLOW	尝试写入到一个已满的发送器 FIFO
SCB_INTR_TX_UNDERFLOW	尝试从一个空的发送器 FIFO 中读取
SCB_INTR_TX_UART_NACK	UART 接收到 SmartCard 模式的 NACK
SCB_INTR_TX_UART_DONE	UART 传输已完成并且 TX FIFO 为空
SCB_INTR_TX_UART_ARB_LOST	UART 的 TX 线路上的值与 RX 线路上的值不匹配

返回值： 无

副作用： 无

uint32 SCB_GetTxInterruptMode(void)

说明： 返回 TX 中断掩码

参数： 无

返回值： uint32：已启用的 TX 中断源的掩码（返回值请参阅 SCB_SetTxInterruptMode() 函数）

副作用： 无

uint32 SCB_GetTxInterruptSourceMasked(void)

说明： 返回由中断掩码屏蔽的 TX 中断请求寄存器

参数： 无

返回值： uint32：仅已启用的 TX 中断源的状态（返回值请参阅 SCB_SetTxInterruptMode() 函数）

副作用： 无

uint32 SCB_GetTxInterruptSource(void)

说明： 返回待处理 TX 中断源的位掩码

参数： 无

返回值： uint32：TX 中断源的状态（返回值请参阅 SCB_SetTxInterruptMode() 函数）

副作用： 无

void SCB_ClearTxInterruptSource(uint32 interruptMask)

说明： 清除待处理 TX 中断源的位掩码

参数： uint32 interruptMask：将被清除的待处理 TX 中断源的位掩码（返回值请参阅 SCB_SetTxInterruptMode()）

返回值： 无

副作用： 无

void SCB_SetTxInterrupt(uint32 interruptMask)

说明： 从 TX 中断源的位掩码生成中断事件

参数： uint32 interruptMask：将生成中断事件的 TX 中断源的位掩码（返回值请参阅 SCB_SetTxInterruptMode()）

返回值： 无

副作用： 无

void SCB_SetRxInterruptMode(uint32 interruptMask)

说明： 配置哪些 RX 中断请求寄存器的比特位才可触发中断事件

参数： uint32 interruptMask：将被启用的 RX 中断源的位掩码。

中断原因常量	说明
SCB_INTR_RX_TRIGGER	接收器 FIFO 中的输入多于触发电平指定的值
SCB_INTR_RX_NOT_EMPTY	接收器 FIFO 不为空
SCB_INTR_RX_FULL	接收器 FIFO 已满
SCB_INTR_RX_OVERFLOW	尝试写入到一个已满的接收器 FIFO
SCB_INTR_RX_UNDERFLOW	尝试从一个空的接收器 FIFO 中读取
SCB_INTR_RX_FRAME_ERROR	检测到 UART 成帧错误
SCB_INTR_RX_PARITY_ERROR	检测到 UART 奇偶校验错误

返回值： 无

副作用： 无

uint32 SCB_GetRxInterruptMode(void)

说明： 返回 RX 中断掩码

参数： 无

返回值： uint32：已启用的 RX 中断源掩码（返回值请参阅 SCB_SetTxInterruptMode() 函数）

副作用： 无

uint32 SCB_GetRxInterruptSourceMasked(void)

说明： 返回由中断掩码屏蔽的 RX 中断请求寄存器

参数： 无

返回值： uint32：仅已启用的 RX 中断源的状态（返回值请参阅 SCB_SetRxInterruptMode() 函数）

副作用： 无

uint32 SCB_GetRxInterruptSource(void)

说明： 返回待处理 RX 中断源的位掩码

参数： 无

返回值： uint32：RX 中断源的状态（返回值请参阅 SCB_SetRxInterruptMode() 函数）

副作用： 无

void SCB_ClearRxInterruptSource(uint32 interruptMask)

说明： 清除待处理 RX 中断源的位掩码

参数： uint32 interruptMask：将被清除的待处理 RX 中断源的位掩码（返回值请参阅 SCB_SetRxInterruptMode()）

返回值： 无

副作用： 无

void SCB_SetRxInterrupt(uint32 interruptMask)

说明： 从 RX 中断源的位掩码生成中断事件

参数： uint32 interruptMask：将生成中断事件的 RX 中断源的位掩码（返回值请参阅 SCB_SetRxInterruptMode()）

返回值： 无

副作用： 无

void SCB_SetMasterInterruptMode(uint32 interruptMask)

说明： 配置多少位主设备中断请求寄存器才可触发中断事件。

参数： uint32 interruptMask：将被启用的 RX 中断源的位掩码。

中断原因常量	说明
SCB_INTR_MASTER_SPI_DONE	SPI 主设备传输已完成并且 TX FIFO 为空
SCB_INTR_MASTER_I2C_ARB_LOST	I2C 主设备仲裁失败
SCB_INTR_MASTER_I2C_NACK	I2C 主设备否定确认
SCB_INTR_MASTER_I2C_ACK	I2C 主设备确认
SCB_INTR_MASTER_I2C_STOP	I2C 主设备生成“停止”
SCB_INTR_MASTER_I2C_BUS_ERROR	I2C 主设备总线错误（意外检测启动和停止条件）

返回值： 无

副作用： 无

uint32 SCB_GetMasterInterruptMode(void)

说明： 返回主设备中断掩码

参数： 无

返回值： uint32：已启用的主设备中断源的掩码（返回值请参阅 SCB_SetMasterInterruptMode() 函数）

副作用： 无

uint32 SCB_GetMasterInterruptSourceMasked(void)

说明： 返回由中断掩码屏蔽的主设备中断请求寄存器

参数： 无

返回值： uint32：仅已启用的主设备中断源的状态（返回值请参阅 SCB_SetMasterInterruptMode() 函数）

副作用： 无

uint32 SCB_GetMasterInterruptSource(void)

说明： 返回待处理主设备中断源的位掩码

参数： 无

返回值： uint32：主设备中断源的状态（返回值请参阅 SCB_SetMasterInterruptMode() 函数）

副作用： 无

void SCB_ClearMasterInterruptSource(uint32 interruptMask)

说明： 清除待处理主设备中断源的位掩码

参数： uint32 interruptMask：将被清除的待处理主设备中断源的位掩码（返回值请参阅 SCB_SetMasterInterruptMode()）

返回值： 无

副作用： 无

void SCB_SetMasterInterrupt(uint32 interruptMask)

说明： 从主设备中断源的位掩码生成中断事件

参数： uint32 interruptMask：将生成中断事件的主设备中断源的位掩码（返回值请参阅 SCB_SetMasterInterruptMode() ）

返回值： 无

副作用： 无

void SCB_SetSlaveInterruptMode(uint32 interruptMask)

说明： 配置多少位的从设备中断请求寄存器才可触发中断事件

参数： uint32 interruptMask：将被启用的从设备中断源的位掩码。

从设备中断源	说明
INTR_SLAVE_I2C_ARB_LOST	从设备仲裁失败：SDA 线路驱动值不同于 SDA 线路观察值。
INTR_SLAVE_I2C_NACK	收到从设备的否定确认。
INTR_SLAVE_I2C_ACK	收到从设备的确认。
INTR_SLAVE_I2C_WRITE_STOP	本从设备写入传输的 “停止” 或 “重复启动” 事件（进行了地址匹配）。
INTR_SLAVE_I2C_STOP	用于本从设备（读取或写入）传输的 “停止” 或 “重复启动” 事件（进行了地址匹配）。
INTR_SLAVE_I2C_START	收到 I2C 从设备 “启动” 。
INTR_SLAVE_I2C_ADDR_MATCH	收到 I2C 地址匹配。
INTR_SLAVE_I2C_GENERAL	收到从设备一般呼叫地址。
INTR_SLAVE_I2C_BUS_ERROR	从设备总线错误（意外检测 “启动” 和 “停止” 条件）。
INTR_SLAVE_SPI_BUS_ERROR	在 SPI 传输中的预期时间取消选中 SPI 从设备。

返回值： 无

副作用： 无

uint32 SCB_GetSlaveInterruptMode(void)

说明： 返回从设备中断掩码

参数： 无

返回值： uint32：已启用的从设备中断源掩码（返回值请参阅 SCB_SetSlaveInterruptMode() 函数）

副作用： 无

uint32 SCB_GetSlaveInterruptSourceMasked(void)

说明： 由中断掩码屏蔽的从设备中断请求寄存器

参数： 无

返回值： uint32：仅已启用的从设备中断源的状态（返回值请参见 SCB_SetSlaveInterruptMode() 函数）

副作用： 无

uint32 SCB_GetSlaveInterruptSource(void)

说明： 返回待处理从设备中断源的位掩码

参数： 无

返回值： uint32：从设备中断源的状态（返回值请参见 SCB_SetSlaveInterruptMode() 函数）

副作用： 无

void SCB_ClearSlaveInterruptSource(uint32 interruptMask)

说明： 清除待处理从设备中断源的位掩码

参数： uint32 interruptMask：将被清除的待处理从设备中断源的位掩码（返回值请参见 SCB_SetSlaveInterruptMode() 函数）

返回值： 无

副作用： 无

void SCB_SetSlaveInterrupt(uint32 interruptMask)

说明： 用组件中断源的位掩码生成中断事件

参数： uint32 interruptMask：将生成中断事件的从设备中断源的位掩码（返回值请参见 SCB_SetSlaveInterruptMode() 函数）

返回值： 无

副作用： 无

时钟选择

SCB 由一个专用时钟连接计时。根据操作模式，组件可按自定义配置计算该时钟的频率，或由外部提供。

由于 “未配置” 模式定制器不知道终端运行模式，在这种情况下，时钟必须外部提供。

MISRA 合规性

本节介绍了 MISRA-C:2004 合规性和本组件的偏差情况。规定了两种类型的偏差：

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

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”章节中介绍了项目偏差以及有关 MISRA 合规性验证环境的信息。

SCB 组件具有以下特定偏差：

MISRA-C：2004 规则	规则类[5]	规则描述	偏差描述
1.1	必需	此规则规定，代码必须符合 C C ISO/IEC 9899：1990 标准。	控制结构嵌套（说明）超过 15－程序未严格达到 ISO:C90。 实际上，大多数编译器支持更自由的嵌套限制，所以只有当要求严格操作时才涉及此限制。通过比较，ISO:C99 规定模块的限制为 127 嵌套级。 支持的编译器（GCC 4.1.1、RVDS 和 MDK）支持更多的控制结构嵌套。
17.4	必需	数组索引应为唯一允许的指针算法的形式。	组件使用数组索引操作访问缓冲区。访问前缓冲区大小被选中。这是安全的操作，除非用户提供缓冲区大小有误。
19.7	A	使用时，函数应优先于类函数宏。	偏差由于函数宏用于允许更高效的代码。

该组件配有以下嵌入式构件：引脚与中断。MISRA 合规性与特定偏差的相关信息，请参见相应组件数据手册。

固件源代码示例

PSoC Creator 在“查找示例项目”对话框中提供了包括原理图和代码示例的许多示例项目。要获取组件示例，请打开组件目录中的对话框或原理图中的组件实例。要获取一般示例，请打开 Start Page （起始页）或 File （文件）菜单的对话框 。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。

[5] 必须/参考

中断服务子程序

SCB 支持多种事件上的中断，取决于操作模式。所有的中断事件一起执行“或”运算，然后发送至中断控制器，因此在任何给定的时间，SCB 只能生成一个中断请求。任何已启用的中断源被触发时，此信号将变为高电平。

如“输入/输出连接”章节所述，当内部操作不需要此信号时，有些模式将此信号显示为终端信号。如果内部操作需要此信号，终端信号就不存在。

通过使用各种中断 API，软件可在单个中断服务子程序中服务多个中断事件。

PSoC Creator 生成必要的中断服务子程序，以处理内部操作。但是，可以使用 *SCB_SetCustomInterruptHandler()* 函数寄存一个自定义函数。在内部中断处理程序执行诸如软件缓冲管理功能的任何操作时，将首先调用该用户函数。只支持一个自定义处理程序。

注意：用户管理的“中断源”不会被自动清除。这是用户的责任。将 “1” 写入到相应的位位置，可清除中断源。清除中断源的首选方法是使用 API （例如：`SCB_ClearRxInterruptSource()`）

```
void CustomInterruptHandler(void);
void main()
{
    /* Register custom function */
    SCB_SetCustomInterruptHandler(&CustomInterruptHandler);

    /* Initialize SCB component in UART mode.
    * The SCB_INTR_RX_PARITY_ERROR is already enabled in GUI:
    * UART Advanced Tab.
    */
    SCB_Start()
    CyGlobalIntEnable; /* Enable global interrupts. */
    for(;;)
    {
        /* Place your application code here. */
    }
}
/* User interrupt handler to insert into SCB interrupt handler.
* Note: SCB interrupt set to Internal in GUI.
*/
void CustomInterruptHandler(void)
{
    if(0u != (SCB_GetRxInterruptSourceMasked() & SCB_INTR_RX_PARITY_ERROR))
    {
        /* Interrupt sources does not clear automatically if it is managed by
        * user. The interrupt sources clearing becomes user responsibility.
        */
        SCB_ClearRxInterruptSource(SCB_INTR_RX_PARITY_ERROR);

        /*
        * Add user interrupt code to manage SCB_INTR_RX_PARITY_ERROR.
        */
    }
}
```

TX FIFO 中断源

以下中断源具有特定的行为：TX FIFO 为空、TX FIFO 未满和 TX FIFO 触发。这些中断源触发 TX FIFO 的当前状态，并保持到清除操作为止。当 SCB 组件被禁用时，TX FIFO 变为空并且 TX 中断源触发其状态。如果 TX FIFO 为空，清除这些中断源没有任何意义，因为它们将自行恢复。恢复操作需要一个时钟周期，因此在此期间清除中断源。当填充 TX FIFO 时，最好监测其中的条目数量，而不是在每个字节输入到 TX FIFO 后尝试清除未满的 TX FIFO 或触发中断源。恢复时间导致误未满 TX FIFO 被误清除或触发中断源。若要启动 TX FIFO 中断源处理，建议采用如下流程：在 TX FIFO 中填入数据，清除已触发的“旧”中断源并启用它。

为了清除中断源，将 “1” 写入到相应的位位置。

注意：TX FIFO 触发中断源行为取决于触发电平值。

RX FIFO 中断源

以下中断源具有特定的行为：RX FIFO 为空、RX FIFO 已满和 RX FIFO 触发。这些中断源触发 RX FIFO 的电流状态并保持到清除操作为止。当 SCB 组件被禁用时，RX FIFO 变为空，且可以清除触发中断源。当 RX FIFO 为空，清除这些中断源没有任何意义，因为他们将得到恢复。恢复操作需要一个时钟周期，因此在此期间清除中断源。当从 RX FIFO 读取数据时，最好监测其中的条目数量，而不是在每个读取字节后尝试清除未满的 RX FIFO 或触发中断源。恢复时间导致误清除不为空的 RX FIFO 或触发中断源。若要启动 RX FIFO 中断源处理，建议采用如下流程：清除触发的“旧”中断源并启用它。在大多数情况下 ，不需要清除操作。

为了清除中断源，将 “1” 写入到相应的位位置。

注意：RX FIFO 触发中断源行为取决于触发电平值。

放置

SCB 按固定功能模块放置，且所有的放置信息将通过 *cyfitter.h* 文件提供给 API。

寄存器

有关寄存器的详细信息，请参见芯片的《技术参考手册》（TRM）。

资源

USB 作为固定功能模块实现。

模式		资源类型	
		SCB 固定模块	中断
未配置的 SCB		1	1
I²C	从设备	1	1

模式		资源类型		
		SCB 固定模块		中断
	主设备	1		1
	多主设备	1		1
	多主从设备	1		1
SPI	从设备	硬件缓冲区	1	–
		软件缓冲区	1	1
	主设备	硬件缓冲区	1	–
		软件缓冲区	1	1
UART	标准	硬件缓冲区	1	–
		软件缓冲区	1	1
	标准（多处理器模式）	硬件缓冲区	1	–
		软件缓冲区	1	1
	SmartCard	硬件缓冲区	1	–
		软件缓冲区	1	1
	IrDA	硬件缓冲区	1	–
		软件缓冲区	1	1

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。下表提供了某一组件配置中所有 API 的存储器用途。

采用按“释放”模式配置的相关编译器进行了测量，大小采用了优化设定。对于特定设计，可以分析编译器生成的映射文件，确定存储器的用途。

模式		PSoC 4 (GCC)	
		闪存	RAM
未配置的 SCB		3850	38
I^2C	从设备	1680	46
	主设备	2866	49
	多主设备	2866	49
	多主从设备	4058	49

模式			PSoC 4 (GCC)	
			闪存	RAM
SPI	从设备	硬件缓冲区 [6]	384	12
		软件缓冲区 [7]	1062	53
	主设备	硬件缓冲区	420	12
		软件缓冲区	1098	53
UART	标准	硬件缓冲区	568	12
		软件缓冲区	1222	53
	标准（多处理器模式）	硬件缓冲区	622	12
		软件缓冲区	1284	69
	SmartCard	硬件缓冲区	580	12
		软件缓冲区	1234	53
	IrDA	硬件缓冲区	580	12
		软件缓冲区	1230	53

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。

I^2C 直流规范

参数	说明	最小值	典型值	最大值	单位	条件
I_{I2C1}	在 100 KHz 下，模块电流消耗	–	-	10.5	µA	
I_{I2C2}	在 400 KHz 下，模块电流消耗	–	-	135	µA	
I_{I2C3}	在 1 Mbps 下，模块电流消耗	–	-	310	µA	
I_{I2C4}	I^2C 在深度睡眠模式下启用	–	-	1.4	µA	

[6]硬件缓冲区－RX 与 TX 缓冲区大小等于 8 字节。中断模式为“无”

[7]软件缓冲区－RX 与 TX 缓冲区大小等于 9 字节。在这种情况下，内部中断被自动启用。

I²C 交流电规范

参数	说明	最小值	典型值	最大值	单位	条件
F_{I2C1}	比特率	–	–	1	Mbps	

UART 直流规范

参数	说明	最小值	典型值	最大值	单位	条件
I_{UART1}	100 Kbits/秒下的模块耗电量	–	-	9	µA	
I_{UART2}	1000 Kbits/秒下的模块耗电量	–	-	312	µA	

UART 交流规范

参数	说明	最小值	典型值	最大值	单位	条件
F_{UART}	比特率	–	–	1	Mbps	

SPI 直流规范

参数	说明	最小值	典型值	最大值	单位	条件
I_{SPI1}	在 1 Mbits/sec 下，模块电流消耗	–	-	360	µA	
I_{SPI2}	4 Mbits/秒下的模块耗电量	–	-	560	µA	
I_{SPI3}	8 Mbits/秒下的模块耗电量	–	-	600	µA	

SPI 交流规范

参数	说明	最小值	典型值	最大值	单位	条件
F_{SPI}	SPI 操作频率（主设备；6X 过采样））	–	–	8	MHz	

SPI 主设备交流规范

参数	说明	最小值	典型值	最大值	单位	条件
T_{DMO}	Sclock 驱动沿后的 MOSI 有效	–	–	15	ns	
T_{DSI}	Sclock 捕获沿前的 MISO 有效。全时钟、MISO 迟采样	20	–	–	ns	
T_{HMO}	关于从设备捕获沿的先前 MOSI 数据保持时间	0	–	–	ns	

SPI 从设备交流规范

参数	说明	最小值	典型值	最大值	单位	条件
T_{DMI}	Sclock 捕获沿前的 MOSI 有效	40	–	–	ns	
T_{DSO}	Sclock 驱动沿后的 MISO 有效	–	–	42 + 3 × FCPU	ns	
T_{DSO_ext}	在外部时钟中的 Sclock 驱动沿后的 MISO 有效。时钟模式	–	–	48	ns	
T_{HSO}	先前的 MISO 数据保持时间	0	–	–	ns	
$T_{SSELSCK}$	到第一个 SCK 有效沿的 SSEL 有效	100	–	–	ns	

组件修改

本节列出了组件与以前版本相比的主要修改。

版本	更改说明	更改/影响原因
1.0.a	编辑组件数据手册以匹配 GUI。	
1.0	SCB 组件第一次释放	

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。