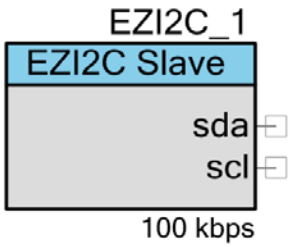
EZI2C 从器件

1.80



性能

- 工业标准 NXP® I²C 总线接口
- 仿真通用 I²CEEPROM 接口
- 只需要两个引脚（SDA 和 SCL）与 I²C 总线连接
- 50/100/400/1000 kbps 标准数据速率
- 高层级的 API 需要最少的用户编程
- 支持使用独立存储器缓冲区对一个或两个地址进行解码
- 存储器缓冲区提供可配置的读/写和只读区域

概述

EZI2C从器件组件实现基于I²C寄存器的从器件。它与I²C标准、快速和超快速模式的设备兼容[1]，装置见NXP I2C-总线规范。主控会在I²C总线上启动所有通信，并为所有从器件器件提供时钟。EZI2C从器件支持高达1000 kbps的标准数据速率，且与同一总线上的多个器件兼容。

EZI2C 从器件是 I²C 从器件的唯一实现，主控和从器件之间的所有通信都在 ISR（中断服务子程序）中处理，不需要与主程序流交互。该接口表现为主控与从器件之间的共享存储器。一旦执行了 EZI2C_Start()函数，则几乎不再需要与 API 交互。

[1] I²C 外设在以下范围内不符合 NXP I²C 规范：模拟短时脉冲滤波器、I/O V_{OL}/I_{OL}、I/O 迟滞。I²C 模块没有数字短时脉冲滤波器（在睡眠模式下无效）。将各 I/O 设置为慢速模式可以满足快速模式的最小下降时间规范。更多详细信息，请参考器件数据手册“输入和输出”一节中的电气规范内容。

何时使用 EZI2C 从器件

在 I²C 从器件与 I²C 主控之间需要共享存储器模型时使用此组件。可以在代码中将 EZI2C 从器件缓冲区定义为任何变量、数组或结构，而无需考虑 I²C 协议。I²C 主机可以查看此缓冲区中的任何变量，修改 EZI2C_SetBuffer1()或 EZI2C_SetBuffer2()函数定义的变量。

输入/输出连接

本节介绍 EZI2C 从器件的各种输入和输出连接。

sda — 输入/输出

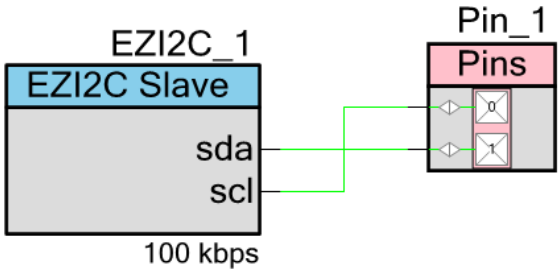
串行数据(SDA)是 I²C 数据信号。这种双向数据信号用于传输或接收所有总线数据。

SCL — 输入/输出

串行时钟(SCL)是主控生成的 I²C 时钟。虽然从器件从不生成时钟信号，但是它可以将其保持在低电平，使总线停顿，直到它准备发送数据或 NAK/ACK 最新数据或地址为止。

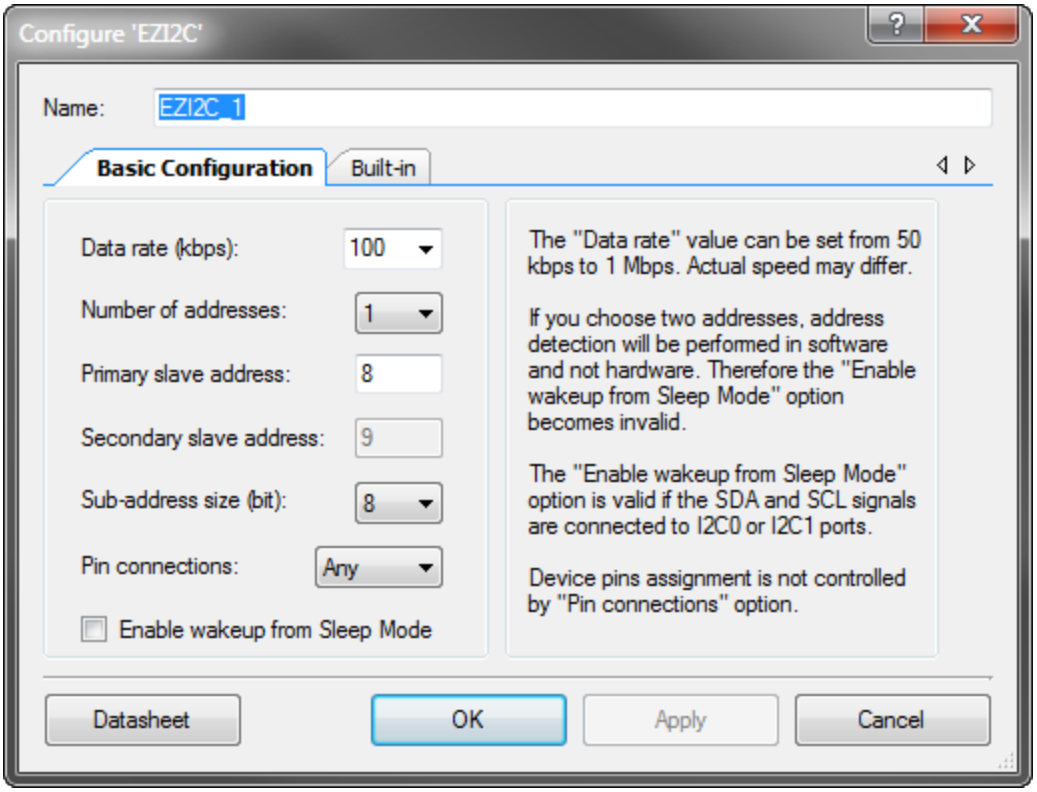
原理图宏信息

组件目录中的默认 EZI2C 从器件是使用具有默认设置的 EZI2C 从器件组件的示意宏。EZI2C 从器件组件连接到引脚组件，该组件配置为 SIO 对。



组件参数

将 EZI2C 组件拖动到您的设计中，双击它可打开 **Configure**（配置）对话框。



EZI2C 组件提供下列参数。

数据速率

此参数用于设置 I^2C 数据速率,速率支持高达 1000 kbps ；实际速率可能因可用时钟速度和分频器范围而异。标准数据速率为 50、100（默认值）、400 和 1000 kbps。

地址数

此选项确定是识别 **1** 个（默认值）还是 **2** 个独立 I^2C 从器件地址。如果识别两个地址，则将用软件（而不是硬件）执行地址检测，因此 **Enable wakeup from Sleep Mode**（启用从睡眠模式唤醒）选项不可用。

主从器件地址

这是主 I^2C 从器件地址（默认值为 **8**）。可以用十进制或十六进制格式输入该值。对于十六进制，在数字之前键入“0x”。此地址为右对齐的 7 位从器件地址，它不包括读/写位。

辅助从器件地址

这是辅助 I^2C 从器件地址（默认值为 **9**）。可以用十进制和十六进制格式输入该值。对于十六进制，在数字之前键入“0x”。只有当**地址数**参数设置为 **2** 时，此辅助地址才有效。主地址和辅从器件地址必须不同。此地址为右对齐的 7 位从器件地址，它不包括读/写位。

辅助地址大小

此选项确定可以访问的数据范围。可以选择 **8** 位（默认值）或 **16** 位辅助地址。如果使用 8 位地址大小，则主控只能访问 0 到 255 之间的数据偏移。您还可以选择 16 位辅助地址大小。这将允许 I^2C 主控在每个从器件地址访问最大达 65,535 字节的数据数组。

引脚连接

此参数确定要用于 SDA 和 SCL 信号连接的引脚类型。此选项用于补充 **Enable wakeup from Sleep mode**（启用从睡眠模式唤醒）选项，仅当在 **Number of addresses**（地址数）选项中选择了单个 I^2C 地址时才可用。此参数包含三个可选值：**Any**、**I2C0** 和 **I2C1**。默认值为 **Any**。

Any（任意）表示一般用途 I/O (GPIO)。如果 **Enable wakeup from Sleep Mode**（使能从睡眠模式唤醒），则应为 SDA 和 SCL 使用 **Any** 值。如果需要 **Enable wakeup from Sleep Mode**（使能从睡眠模式唤醒），则必须使用引脚对 I2C0（P12[4]、P12[5]）或 I2C1（P12[0]、P12[1]），这使您可以配置器件以便在 I^2C 地址匹配时唤醒。

使能从睡眠模式唤醒

此参数允许当从器件地址匹配时从睡眠模式唤醒。默认情况下，禁用此选项。只有当选择单一 I^2C 地址且 SDA 和 SCL 信号连接到 SIO 端口（引脚对 I2C0 或 I2C1）时，地址匹配时唤醒选项才有效。 **Enable wakeup from Sleep mode** 受 PSoC 3 和 PSoC 5LP 的支持。

时钟选择

时钟连接到系统总线时钟，用户不能更改。

应用程序编程接口

通过应用编程接口（API），您可以使用软件进行配置组件。下面的表格列出并介绍了每个函数的接口。以下各节将详细介绍每个函数。

默认情况下，PSoC Creator 将实例名称“EZI2C_1”分配给设计中的第一个组件实例。您可以将其重新命名为任何一个符合标识符语法规则的值。实例名称会成为每个全局函数名称、变量和常量符号的前缀。为增加可读性，下表中使用了实例名称“EZI2C”。

函数	说明
EZI2C_Start()	启动对I²C通信的响应。启用中断。
EZI2C_Stop()	停止对I²C通信的响应。禁用中断。
EZI2C_EnableInt()	使能中断，大部分I²C操作都需要启用中断。
EZI2C_DisableInt()	禁用中断。EZI2C_Stop() API自动执行此操作。
EZI2C_SetAddress1()	设置主I²C地址。
EZI2C_GetAddress1()	返回主I²C地址。
EZI2C_SetBuffer1()	设置主I²C的缓冲区指针。
EZI2C_GetActivity(void)	获取组件活动状态。
EZI2C_Sleep()	停止I²C操作，保存I²C配置。禁用中断。
EZI2C_Wakeup()	恢复I²C配置并启动I²C操作。启用中断。
EZI2C_Init()	使用定制器提供的初始值来初始化I²C寄存器。
EZI2C_Enable()	激活硬件并开始执行组件操作。
EZI2C_SaveConfig()	保存EZI2C组件的当前用户配置。
EZI2C_RestoreConfig()	恢复非保留I²C寄存器。

可选次地址 API

只有当启用两个 I²C 地址时，这个指令才有效。

函数	说明
EZI2C_SetAddress2()	设置次I²C地址。
EZI2C_GetAddress2()	返回次I²C地址。
EZI2C_SetBuffer2()	设置次I²C的缓冲区指针。

可选睡眠/唤醒模式

只有当使用单一地址、SCL 和 SDA 信号路由到 SIO 引脚且选择 **Enable wakeup from Sleep Mode**（使能从睡眠模式唤醒）时，下列这些函数才可用。

函数	说明
EZI2C_SlaveSetSleepMode()	使能EZI2C睡眠地址解码并保存I²C配置。禁用中断。
EZI2C_SlaveSetWakeMode()	禁用EZI2C睡眠地址解码，恢复I²C配置，并启动I²C操作。启用中断。

全局变量

正常运行情况下，不需获得这些变量。

函数	说明
EZI2C_initVar	指示是否已初始化EZI2C。该变量初始化为0，并在第一次调用EZI2C_Start()时设置为1。这允许第一次调用EZI2C_Start()子程序后组件无需重新初始化便可重新启动。 如果需要重新初始化组件，该变量应当在调用EZI2C_Start()子程序之前设置为0。另外，可以通过调用EZI2C_Init()和EZI2C_Enable()函数来重新初始化EZI2C。
EZI2C_dataPtrS1	存储公开给第一个从器件地址的I²C主控的数据指针。
EZI2C_rwOffsetS1	存储用于读取和写入操作的偏移。它在第一个从器件地址的每个写入序列上设置。
EZI2C_rwIndexS1	存储要为第一个从器件地址读取或写入的下一个值的指针。
EZI2C_wrProtectS1	存储第一个从器件地址的只读数据的偏移。

函数	说明
EZI2C_bufSizeS1	存储公开给第一个从器件地址的I²C主控的数据数组的大小。
EZI2C_dataPtrS2	存储公开给第二个从器件地址的I²C主控的数据指针。
EZI2C_rwOffsetS2	存储读写操作的偏移，在第二个从器件地址的每个写入序列进行设置。
EZI2C_rwIndexS2	存储要为第二个从器件地址读取或写入的下一个值的指针。
EZI2C_wrProtectS2	存储第二个从器件地址的只读数据的偏移。
EZI2C_bufSizeS2	存储公开给第二个从器件地址的I²C主控的数据数组的大小。
EZI2C_curState	存储I²C状态机的当前状态。
EZI2C_curStatus	存储组件的当前状态。

void EZI2C_Start(void)

说明： 这是开始执行组件操作的首选方法。EZI2C_Start()设置initVar变量，调用EZI2C_Init()函数，然后调用EZI2C_Enable()函数。它必须在I²C总线操作之前执行。

在EZI2C_Start()调用EZI2C_Enable()之后，EZI2C_Enable()会调用EZI2C_EnableInt()，这会启用I²C中断。

参数： 无

返回值： 无

副作用： 无

void EZI2C_Stop(void)

说明： 禁用I²C硬件并禁用I²C中断。根据需要，禁用活动模式电源模板位或门控时钟。

参数： 无

返回值： 无

副作用： 无

void EZI2C_EnableInt(void)

说明： 启用I²C中断。大部分操作都需要中断。在调用EZI2C_Start() API时调用。

参数： 无

返回值： 无

副作用： 无

void EZI2C_DisableInt(void)

说明： 禁用I²C中断。通常情况下，Stop()函数禁用中断后就不需要此函数了。

参数： 无

返回值： 无

副作用： 如果在运行I²C的时候禁用I²C中断，则I²C总线可能会锁定。

void EZI2C_SetAddress1(uint8 address)

说明： 此函数设置主存储器缓冲区的I²C地址。此值可以为0到127之间的任何值。

参数： address：0到127之间的7位从器件地址。此地址右对齐，不包括读/写位。

返回值： 无

副作用： 无

uint8 EZI2C_GetAddress1(void)

说明： 返回主要存储器缓冲区的I²C从器件地址。

参数： 无

返回值： 通过SetAddress1或默认I²C地址设置的同一I²C从器件地址。

副作用： 无

void EZI2C_SetBuffer1(uint16 bufSize, uint16 rwBoundry, void * dataPtr)

说明： 此函数设置从器件数据的缓冲区指针、大小和读写区域。这是公开给I²C主控的数据。

参数： bufSize：缓冲区的大小（以字节为单位）。

rwBoundry：设置在缓冲区开头可写入的字节数。此值必须小于或等于缓冲区大小。位于偏移rwBoundry和更远位置处的数据是只读的。

dataPtr：数据缓冲区的指针。

返回值： 无

副作用： 调用此函数前必须先调用EZI2C_Start()。

uint8 EZI2C_GetActivity(void)

说明： 如果自从上次调用此函数后发生I²C读取或写入循环，则此函数返回非零值。在此函数调用的结尾，活动标志复位为零。

当读取时读写忙标志会清除，但是“BUSY”标志只由I²C Stop清除。

参数： 如果检测到活动，则返回非零值。

返回值： I²C活动的状态。

常量	说明
EZI2C_STATUS_READ1	设置是否针对第一个地址检测读取序列。当读取状态时清除。
EZI2C_STATUS_WRITE1	设置是否针对第一个地址检测写入序列。当读取状态时清除。
EZI2C_STATUS_READ2	设置是否针对第二个地址检测读取序列（如果使能）。当读取状态时清除。
EZI2C_STATUS_WRITE2	设置是否针对第二个地址检测写入序列（如果使能）。当读取状态时清除。
EZI2C_STATUS_BUSY	当检测到启动时置位。当检测到停止时清除。
EZI2C_STATUS_ERR	当检测到I²C硬件错误时置位。当读取状态时清除。

副作用： 无

void EZI2C_SetAddress2(uint8 address)

说明： 设置辅助存储器缓冲区的I²C从器件地址。此值可以为0到127之间的任何值。只有当在用户参数中选择了两个I²C地址时，才提供此函数。

参数： address：0到127之间的7位从器件地址。此地址右对齐，不包括读/写位。

返回值： 无

副作用： 无

uint8 EZI2C_GetAddress2(void)

说明： 返回辅助存储器缓冲区的I²C从器件地址。只有当在用户参数中选择了两个I²C地址时，才提供此函数。

参数： 无

返回值： 通过SetAddress2或默认I²C地址设置的同一I²C从器件地址。

副作用： 无

void EZI2C_SetBuffer2(uint16 bufSize, uint16 rwBoundry, void * dataPtr)

说明： 此函数设置辅助从器件数据的缓冲区指针、大小和读写区域。这是公开给辅助I²C地址的I²C主控的数据。只有当在用户参数中选择了两个I²C地址时，才提供此函数。

参数： bufSize：I²C主设备可读取的缓冲区的大小。

rwBoundry：设置I²C主控可读写的字节数。 此值必须小于或等于缓冲区大小。位于偏移rwBoundry和更远位置处的数据是只读的。

dataPtr：这是用于I²C数据缓冲区的数据数组或结构的指针。

返回值： 无

副作用： 调用此函数前必须先调用EZI2C_Start()。

void EZI2C_SlaveSetSleepMode(void)

说明： 这是选择**Enable wakeup from Sleep Mode**（使能从睡眠模式唤醒）的情况下准备组件睡眠的首选API。此API使能I^2C睡眠地址解码。它将一直等待，直到所有I^2C流量在完成前停止。所有后续的I^2C流量将被否认，直至该器件进入睡眠模式为止。

参数： 无

返回值： 无

副作用： 无

void EZI2C_SlaveSetWakeMode(void)

说明： 这是将组件恢复为调用EZI2C_SlaveSetSleepMode()时的状态的首先API。它禁用睡眠EZI2C从器件，并重新使能运行时EZI2C。应当刚好在从睡眠唤醒后调用。只有当使用单一I^2C地址时才提供此函数。

参数： 无

返回值： 无

副作用：

void EZI2C_Sleep(void)

说明： 这是未选择**Enable wakeup from Sleep Mode**（使能从睡眠模式唤醒）的情况下准备组件睡眠的首选API。EZI2C_Sleep() API保存当前组件状态。然后它调用EZI2C_Stop()函数并调用EZI2C_SaveConfig()以保存硬件配置。

在调用CyPmSleep()或CyPmHibernate()函数之前调用EZI2C_Sleep()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 EZI2C_Wakeup(void)

说明： 这是将组件恢复为调用EZI2C_Sleep()时的状态的首先API。EZI2C_Wakeup()函数调用_RestoreConfig()函数来恢复硬件配置。如果在调用EZI2C_Sleep()函数之前使能组件，EZI2C_Wakeup()函数也将重新使能组件。

参数： 无

返回值： 无

副作用： 在EZI2C_SaveConfig()或EZI2C_Sleep()之前调用此函数会产生意外行为。

void EZI2C_Init(void)

说明： 根据定制器**Configure**（配置）对话框设置初始化或恢复组件。不需要调用EZI2C_Init()，因为EZI2C_Start()API将调用此函数，这是开始组件操作的首选方法。

参数： 无

返回值： 无

副作用： 所有寄存器都将根据定制器**Configure**（配置）对话框设置为相应的值

void EZI2C_Enable(void)

说明： 激活硬件并开始执行组件操作。不需要调用EZI2C_Enable()，因为EZI2C_Start() API将调用此函数，这是开始组件操作的首选方法。调用EZI2C_EnableInt()以启用I^2C中断。

参数： 无

返回值： 无

副作用： 无

void EZI2C_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修改的当前器件参数值。此函数由EZI2C_Sleep()函数调用。

参数： 无

返回值： 无

副作用： 无

void EZI2C_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为在调用EZI2C_Sleep()函数之前的值。

参数： 无

返回值： 无

副作用： 在EZI2C_Sleep()或EZI2C_SaveConfig()之前调用此函数会产生意外行为。

MISRA 兼容性

本节介绍了MISRA-C:2004规范和本器件的偏差情况。规定了两种类型的偏差：

- 项目偏差 — 适用于所有 PSoC Creator 器件的偏差
- 特定偏差 — 仅适用于本器件的偏差

本节提供了有关器件特定偏差的信息。*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。

还未证明 EZI2C 从设备组件源代码符合 MISRA-C:2004 编码准则。

样例固件源代码

在“Find Example Project”（查找示例项目）对话框中，PSoC Creator 提供了大量示例项目，包括原理图和示例代码。要获取器件示例，请打开器件目录中的对话框或原理图中的器件实例。要查

看通用样例，请打开 Start Page 或 **File** 菜单中的对话框。根据要求，可以通过使用对话框中的 **Filter Options** 项来限定可选的项目列表。

更多有关信息，请参考《PSoC Creator 帮助》部分中主题为 Find Example Project（查找示例项目）中的内容。

功能描述

此组件支持具有一个或两个 I^2C 地址的 I^2C 从器件设备。任一地址都可以访问 RAM、EEPROM 或闪存数据空间中定义的存储器缓冲区。EEPROM 和闪存存储器缓冲区是只读的，而 RAM 缓冲区可以是读写的。这些地址右对齐。

由于 I^2C 硬件是中断驱动的，因此在使用此组件时，必须启用全局中断。即使此组件需要中断，您也不需要向 ISR（中断服务子程序）添加任何代码。模块独立于您的代码为所有中断（数据传输）提供服务。为此接口分配的存储器缓冲区看上去类似于您的应用程序与 I^2C 主控之间的简单双端口存储器。

如果需要，可以通过在数据结构中定义信号和指令位置，在主控与从器件之间创建更高级别接口。

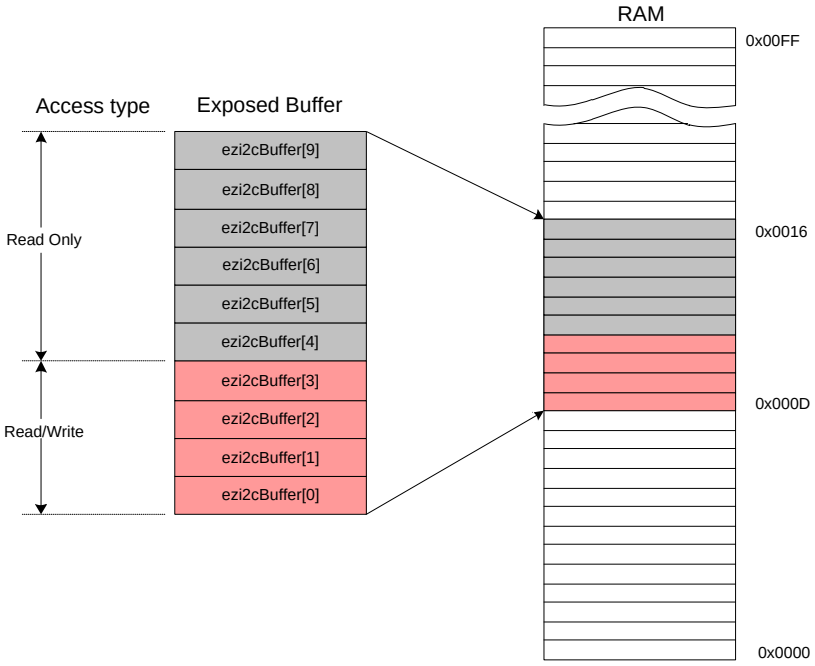
存储器接口

对于 I^2C 主控，该接口看上去非常类似于通用 I^2C EEPROM。EZI2C 接口可以配置为简单变量、数组或结构。在某种程度上，它通过 I^2C 总线充当您的程序与 I^2C 主控之间的一个或两个共享存储器接口。该组件仅允许 I^2C 主控访问指定存储器区域，阻止该区域外的任何读取或写入。例如，如果主要从器件地址的缓冲区按下面的代码示例配置，则存储器中的缓冲区表示可以按图 1 所示表示。

```
#define BUFFER_SIZE (0x0Au)
#define BUFFER_RW_AREA_SIZE (0x04u)

EZI2C_SetBuffer1(BUFFER_SIZE, BUFFER_RW_AREA_SIZE, (void *) ezi2cBuffer);
```

图 1. 公开给 I²C 主控的 EZI2C 缓冲区的存储器表示



接口（I²C 主控）仅将结构视为字节数组，不能访问定义的区域外的任何存储器。使用上述示例结构，提供的 API 用于将数据结构公开给 I²C 接口。

```
char ezi2cBuffer2[15u];
EZI2C_SetBuffer2(15u, 8u, (void *) ezi2cBuffer2);
```

对于不同架构，以不同字节顺序传输数据。因此，需要以特定字节顺序发送额外代码。例如，CY_GET_REGXX()/ CY_SET_REGXX()宏（XX 表示 16/24/32）可以用于匹配低位优先顺序（与器件架构无关）。有关字节顺序的更多信息，请参见*系统参考指南*中的“寄存器访问”一节。

下面的简单示例显示仅公开一个整数（两个字节）。这两个字节都可由 I²C 主控读写。

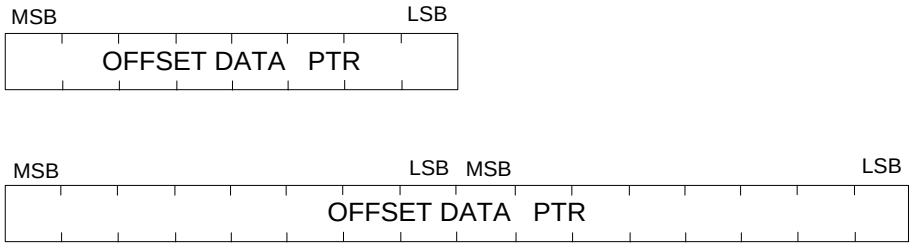
```
uint16 ezi2cVariable1;
CY_SET_REG16(&ezi2cVariable1, 0xABCD);
EZI2C_SetBuffer1(2u, 2u, (void *) (&ezi2cVariable1));
```

外部主控可视的接口

EZI2C 从器件组件支持读写区域的基本读写操作和只读区域的只读操作。两个 I²C 地址接口使用单独偏移数据指针寻址的单独数据缓冲区。根据 **Sub-address size**（辅助地址大小）参数，主控将

偏移数据指针作为写入操作的第一个或前两个数据字节写入。在此讨论的其余部分，将重点介绍 8 位 Sub_Address_Size。

图 2. 8 位和 16 位辅助地址大小（自上而下）

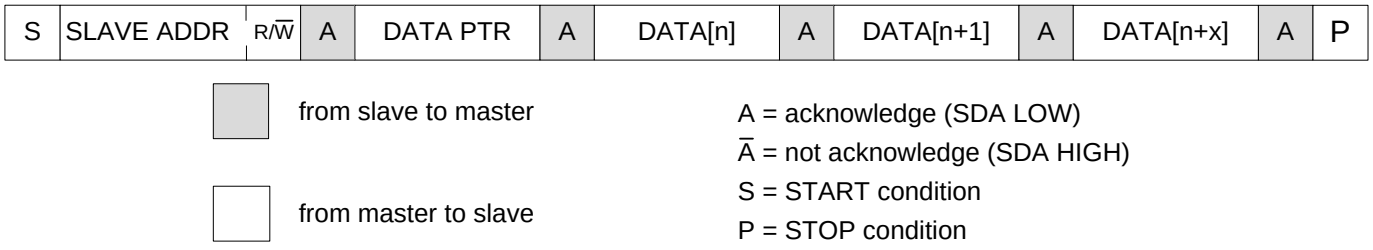


对于写入操作，第一个数据字节始终是偏移数据指针（**Sub-address size**（辅助地址大小）的两个字节 = 16）。偏移数据指针后的字节写入偏移数据指针指向的位置。第二个数据字节写入偏移数据指针加一的位置，依此类推，直到写入完成。写入操作的长度仅受最大缓冲区读写区域大小的限制。对于写入操作，必须始终提供偏移数据指针。

读取操作始终在最近写入操作提供的偏移数据指针处开始。像写入操作一样，每读取一个字节，偏移数据指针都会递增。新的读取操作将不会从上一读取操作停止处继续。新的读取操作始终在上一写入操作偏移数据指针指向的位置处开始读取数据。读取操作的长度仅受数据缓冲区的最大大小限制。

通常，读取包含写入操作（该操作仅包含偏移数据指针，后跟重新启动（或停止/启动））和读取操作。如果偏移数据指针像在重复读取相同数据时那样不需要更新，则第一次写入后不需要其他写入操作。这可以通过允许读取操作彼此紧密跟随来极大提高读取操作速度。

图 3. 将 x 字节写入 I^2C 从器件



例如，如果偏移数据指针设置为 4，则读取操作开始在位置 4 读取数据，并连续读取，直到达到数据末尾或主机完成读取操作为止。无论执行一个还是多个读取操作，都是这样。偏移数据指针在启动新写入操作之前不会更改。

如果 I^2C 主控尝试跨过 EZI2C_SetBuffer1()或 EZI2C_SetBuffer2()函数指定的区域写入数据，则该数据被丢弃，不会影响指定 RAM 区域内外的任何 RAM。不能在允许的范围外读取数据。在允许范围外的任何主控读取请求都会导致返回无效数据。

图 4 说明了 8 位偏移数据指针的数据指针写入。

图 4. 设置从器件数据指针

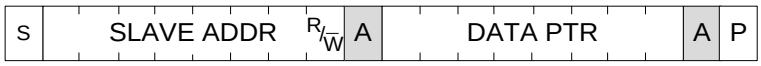


图 5 说明了 8 位偏移数据指针的读取操作。请记住，数据写入操作始终重新编写偏移数据指针。

图 5. 从 I^2C 从器件读取 x 字节



在复位或加电时，将配置 EZI2C 从器件组件并提供 API，但是必须使用 EZI2C_Start()函数明确启用资源。

在 Philips 网站上提供的完整 I^2C 规范中，以及通过参考器件数据表，可以获得 I^2C 总线的详细说明和实现。

数据连贯性

虽然数据缓冲区可以包含大于单字节的数据结构，但是主控读取或写入操作由多个单字节操作组成。这可能会导致数据连贯性问题，因为没有机制可以确保多字节读取或写入将在接口两侧（主控和从器件）同步。例如，考虑一个包含单一两字节整数的缓冲区。虽然主控每次读取 2 字节整数的一个字节，但是从器件可能在主控读取整数的第一个字节(LSB)和要读取第二个字节(MSB)之间的时间内已更新了整个整数。主控读取的数据可能无效，因为 LSB 读取自原始数据，而 MSB 读取自更新的值。

您必须在主控、从器件或二者上提供一个机制，以确保在另一方读取或写入数据时主控或从器件不会进行更新。可以使用 EZI2C_GetActivity()函数开发特定于应用的机制。

从睡眠模式唤醒

如果您想使用**从睡眠模式使能唤醒**功能，可能需要将 I²C 主控设计为处理时钟伸展过程（SCL 保持低电平）。

可以睡眠模式输入过程中修改（通过 CyPmSaveClocks()函数）器件时钟配置（总线时钟频率）。必须先恢复（通过 CyPmRestoreClocks()函数）该配置，然后才能在活动模式下继续 I²C 操作。

要满足这些要求，在进入 EZI2C_SlaveSetSleepMode()前，需要保持 EZI2C 中断的有效状态但必须修改中断处理程序。地址匹配时唤醒触发 EZI2C 中断，而且 EZI2C 唤醒标志被设置，以通知该事件。调用 EZI2C_SlaveSetWakeMode()函数后，中断处理程序将修改为常规 EZI2C，另外将根据 EZI2C 唤醒标志生成中断，以进行输入的数据操作。唤醒后，SCL 线保持低电平，直到调用 EZI2C_SlaveSetSleepMode()为止。

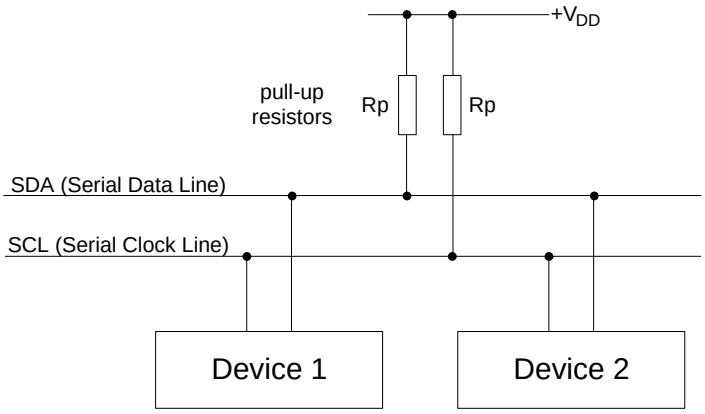
下面是启用 **Enable wake up from Sleep mode**（使能从睡眠模式唤醒）时的正确睡眠模式输入过程：

```
/* Prepares EZI2C to wake up from Sleep mode */
EZI2C_SlaveSetSleepMode();
/* Switches to the Sleep mode */
CyPmSaveClocks();
CyPmSleep(PM_SLEEP_TIME_NONE, PM_SLEEP_SRC_I2C);
CyPmRestoreClocks();
/* Prepares EZI2C to work in Active mode */
EZI2C_SlaveSetWakeMode();
```

外部电路连接

如图 6 所示，I²C 总线需要外部上拉电阻。上拉电阻(R_P)由供电电压、时钟速度和总线电容确定。将输出阶段的任何器件（主设备或从设备）的最小灌电流设置为不超过 3 mA（在 V_{OLmax} = 0.4V 的条件下）。这会将 5-V 系统的最小上拉电阻值限制为大约 1.5 kΩ。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 5-V 系统，上拉电阻不应大于 6 kΩ。更多有关信息，请参见 Philips 网站 www.philips.com 上的 *The I²C-Bus Specification*（*I²C 总线规范*）。

图 6. 器件与 I²C 总线的连接



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。

中断服务子程序

中断服务子程序是组件代码本身使用的程序，您不应该对其进行修改。

资源

该组件使用了固定功能 I²C 模块。

API 存储器大小

根据编译器、器件、所使用的 API 数量以及组件的配置情况，组件的内存大小也不一样。下表介绍的是已给组件所配置的 API 的内存大小。

通过释放模式中所配置的相应编译器来完成测量操作。在释放模式下，可以得到最优化的尺寸。对于特定的设计，分析编译器生成的映射文件后可以确定内存的大小。

配置	PSoC 3 (Keil_PK51)		PSoC 5 (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
一个地址	1019	18	972	22	984	22
两个地址	1679	37	1508	43	1500	43

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。

直流电规范

参数	说明	条件	最小值	典型值	最大值	单位
I_{DD}	模块电流消耗	已使能，针对100 kbps进行配置	–	–	250	µA
		已使能，针对400 kbps进行配置	–	–	260	µA
		从睡眠模式唤醒	–	–	30	µA

交流电规范

参数	说明	条件	最小值	典型值	最大值	单位
	比特率		–	–	1	Mbps

组件更改

本节列出了各版本的主要组件更改内容。

版本	更改内容	更改原因/影响
1.80	已添加MISRA合规性章节。	此器件未进行MISRA合规性验证。

版本	更改内容	更改原因/影响
	在某些地方添加了表示与NXP I²C规格不相兼容的脚注。	文档增强。
	修改了唤醒过程的控制流量，为了避免禁用I²C中断。	PSoC 5 LP要求使能一个I²C中断，以便在发生地址匹配时可以唤醒器件。
	当主机完成读取超过缓冲区大小的范围时，修复控制流量的性能。	由于主器件中的NAK未被检查，所以从器件没有正确完成数据操作。
1.70.a	更正了图5。	
1.70	添加了PSoC 5LP支持。	
1.61	增强了在定制器中配置以及与**Enable wakeup from Sleep Mode**（使能从睡眠模式唤醒）选项相关的选项的验证。	防止使用不支持的模式配置组件。
	更新了针对PSoC 3器件的EZI2C_Stop()实现。	使EZI2C_Stop()释放（如果锁定）。
	将默认I²C地址更新为8和9，以符合I²C总线规范要求。	根据I²C总线规范保留以前使用的地址。
	更新了组件调试器工具窗口支持。	增强了调试窗口支持。
	通过向.cyre文件添加函数名称，增加了将每个函数声明为PSoC 3的重新进入的可能性。	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不能真正重入的函数。 需要此更改为采用安全方式使用（通过标志或关键节防止并发调用）并且不是可重入函数消除编译器警告。
1.60.b	数据手册纠正	
1.60.a	使用有关Enable wakeup from Sleep mode（使能从睡眠模式唤醒）与Number of addresses（地址数）选项之间的依赖关系的信息更新了“引脚连接”一节。	说明该选项用于补充Enable wakeup from Sleep mode（启用从睡眠模式唤醒）选项，而且仅当在Number of addresses（地址数）选项中选择了单个I²C地址时才可用。
	更新了图2、3 和 5以显示位字段。	可视性增强，
	阐明了编写可移植代码（与PSoC器件架构无关）的方法。	文档增强。

版本	更改内容	更改原因/影响
1.60	更改使用从器件启用位的方法：EZI2C_Stop()现在不清除此位，此位的设置操作已从EZI2C_Enable()移动到EZI2C_Init()。现在是在EZI2C_RestoreConfig()函数中恢复I²C配置寄存器。	用于实现EZI2C_Start() - EZI2C_Stop() - EZI2C_Start()和EZI2C_Sleep() - EZI2C_Wakeup()序列的正确结果。预计没有功能影响。
	将定制器中的标签I2C Bus Speed:（I2C总线速度：）替换为Data Rate（数据速率）。向“功能说明”中添加了“从睡眠模式唤醒”一节。	I²C总线规范命名和I²C/EZI2C组件之间的一致性。
	定制器中的标签“连接到的I2C引脚”替换为“引脚连接”	修改了文本以保持与要求的一致性。
	定制器中的标签“使能从睡眠模式唤醒”替换为“从睡眠模式使能唤醒”	修改了文本以保持与要求的一致性。
	更新了组件符号和目录放置名称：“EZ I2C”重命名为“EZI2C”。	修改了文本以保持与要求的一致性。
	解决了当全局变量在代码和ISR中使用时可能被编译器优化的问题。	避免可能导致意外结果的优化问题。
	向数据手册中添加了特性数据	
	对数据表进行了少量编辑和更新	
1.50.a	将组件移动到组件目录的子文件夹中	
1.50	标准数据速率已更新，最高可支持1 Mbps。	允许将I²C总线速度设置为高达1 Mbps。
	添加了Keil重新进入支持。	通过Keil编译器支持PSoC 3，以便能够从多个控制流调用函数。
	添加了睡眠/唤醒和初始化/启用API。	用于支持低功耗模式并提供常用接口，以单独控制大多数组件的初始化和启用。
	添加了组件的XML说明。	这允许PSoC Creator提供一个机制来为此组件创建新的调试器工具窗口。
	添加了对PSoC 3量产器件的支持。	应用了必需的更改，以支持PSoC 3 ES2和生产器件之间的硬件更改。

版本	更改内容	更改原因/影响
	向组件目录添加了默认图示模板。	每个组件应当有一个图示模板。
	修改了EZI2C的总线速度生成。以前它比应有的速度大4倍。在源代码中添加了更多注释以描述总线速度计算。	正确的I^2C总线速度计算和生成。
	为Microsoft Windows 7优化了窗体高度。	在Windows 7中，在定制器启动后会立即出现滚动条。
	使用“将0x前缀用于十六进制”文本，为地址输入框添加了工具提示。	向用户通知十六进制输入的可能性。
1.20.a	将组件移动到组件目录中的子文件夹。	
	向组件中添加了信息，以说明它与芯片修订版的兼容性。	如果组件在不兼容的芯片上使用，该工具将报告错误/警告。如果发生此情况，请更新到支持您的目标器件的修订版。
1.20	更新了“Configure”对话框。	
	在原理图中将数字端口更改为引脚连接	

© 赛普拉斯半导体公司，2013。此处所包含的信息可能会随时更改，恕不另行通知。除赛普拉斯产品的内嵌电路之外，赛普拉斯半导体公司不对任何其他电路的使用承担任何责任。也不根据专利或其他权利以明示或暗示的方式授予任何许可。除非与赛普拉斯签订明确的书面协议，否则赛普拉斯产品不保证能够用于或适用于医疗、生命支持、救生、关键控制或安全应用领域。此外，对于可能发生运转异常和故障并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键器件。若将赛普拉斯产品用于生命支持系统，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

PSoC®是赛普拉斯半导体公司的注册商标，PSoC Creator 和 Programmable System-on-Chip 是赛普拉斯半导体公司的商标。此处引用的所有其他商标或注册商标归其各自所有者所有。

所有源代码（软件和/或固件）均归赛普拉斯半导体公司（赛普拉斯）所有，并受全球专利法规（美国和美国以外的专利法规）、美国版权法以及国际条约规定的保护和约束。赛普拉斯据此向获许可者授予适用于个人的、非独占性、不可转让的许可，用以复制、使用、修改、创建赛普拉斯源代码的派生作品、编译赛普拉斯源代码和派生作品，并且其目的只能是创建自定义软件和/或固件，以支持获许可者仅将其获得的产品依照适用协议规定的方式与赛普拉斯集成电路配合使用。除上述指定的用途之外，未经赛普拉斯的明确书面许可，不得对此类源代码进行任何复制、修改、转换、编译或演示。

免责声明：赛普拉斯不针对此材料提供任何类型的明示或暗示保证，包括（但不仅限于）针对特定用途的适销性和适用性的暗示保证。赛普拉斯保留在不做出通知的情况下对此处所述材料进行更改的权利。赛普拉斯不对此处所述之任何产品或电路的应用或使用承担任何责任。对于可能发生运转异常和故障并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键器件。若将赛普拉斯产品用于生命支持系统中，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

产品使用可能受适用的赛普拉斯软件许可协议限制。