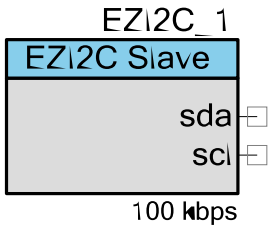
EZI2C Slave
1.60



特点

- 行业标准 Philips I²C 总线接口
- 仿真通用 I²C EEPROM 接口
- 只需两个引脚（SDA 和 SCL）来连接 I²C 总线
- 100/400/1000 kbps 标准数据速率
- 高级 API 只需少量用户编程
- 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

一般说明

EZI2C Slave 组件实现基于 I²C 寄存器的从器件。I²C 总线是 Philips® 开发的基于行业标准的两线硬件接口。主控在 I²C 总线上启动所有通信，并为所有从器件提供时钟。EZI2C Slave 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 Slave 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。该接口显示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。

何时使用 EZI2C Slave

此组件最适合在 I²C 从器件与 I²C 主控之间需要共享存储器模型时使用。EZI2C Slave 缓冲区可以定义为用户代码中的任何变量、数组或结构，无需考虑 I²C 协议。I²C 主控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 Slave 的各种输入和输出连接。

sda – 输入/输出

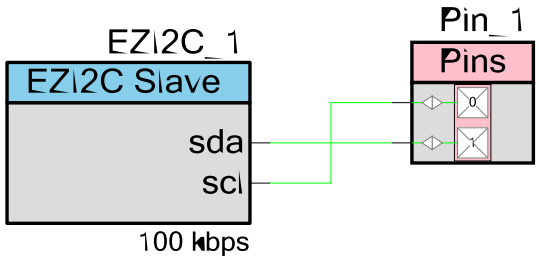
串行数据 (SDA) 是 I²C 数据信号。它是用于传输或接收所有总线数据的双向数据信号。

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。

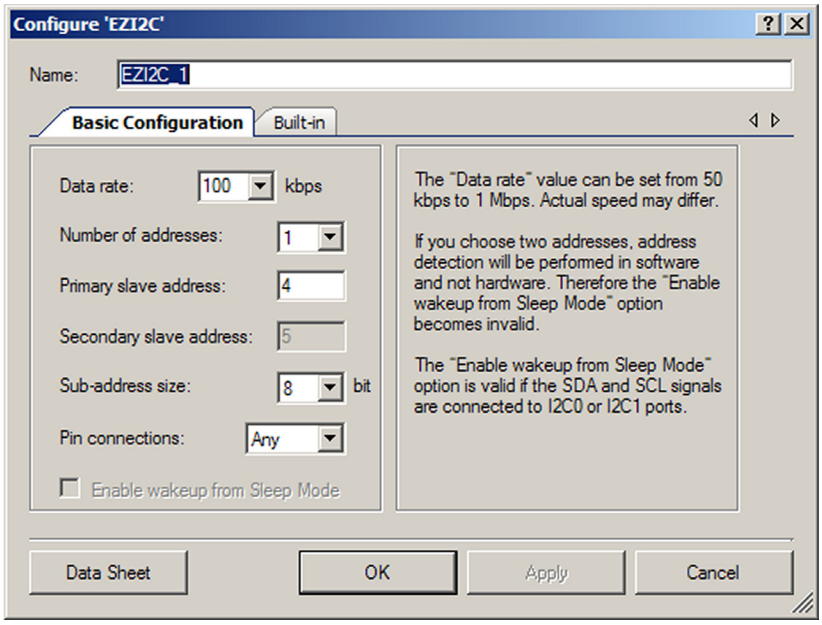
示意宏信息

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



参数和设置

将 EZI2C 组件拖动到您的设计中，双击它可打开“配置”对话框。



EZI2C 组件提供下列参数。

数据速率

此参数用于设置高达 1000 kbps 的 I^2C 数据速率值；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。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，则将在软件（而不是硬件）中执行地址检测，因此**从睡眠模式使能唤醒**选项不可用，变为无效。

主要从器件地址

这是主要 I^2C 从器件地址（默认值为 4）。可以用十进制或十六进制格式输入该值。对于十六进制，应当在数字之前键入“0x”。此地址是 7 位右对齐从器件地址，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 5）。可以用十进制和十六进制格式输入该值。对于十六进制，应当在数字之前键入“0x”。只有当**地址数**参数设置为 2 时，此辅助地址才有效。主要地址和辅助从器件地址必须不同。此地址是 7 位右对齐从器件地址，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 254 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 65,534 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。有三个可能值：任意、I2C0 和 I2C1。默认值为“任意”。

“任意”表示一般用途 I/O (GPIO)。如果不需要**从睡眠模式使能唤醒**，则“任意”应当用于 SDA 和 SCL。如果需要**从睡眠模式使能唤醒**，则必须使用“I2C0”或“I2C1”，它们允许您将器件配置为在 I^2C 地址匹配时唤醒。

从睡眠模式使能唤醒

此参数允许当从器件地址匹配时从睡眠模式唤醒器件。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（I2C0 或 I2C1）时，地址匹配时唤醒选项才有效。

时钟选择

时钟绑定到系统总线时钟，用户不能更改。

资源

对此组件使用了固定函数 I²C 时钟。

模式	数字模块					API 内存（字节）		引脚
	数据路径	宏单元	状态寄存器	控制寄存器	计数器 7	闪存	RAM	
单地址解码（默认值）	不适用	不适用	不适用	不适用	不适用	895	18	2
双地址解码（16 位辅助地址大小）	不适用	不适用	不适用	不适用	不适用	1620	37	2

应用程序编程接口

应用程序编程接口 (API) 子程序使您可以使用软件来配置组件。下表列出并描述了每个函数的接口。后面的章节将更详细地介绍每个函数。

默认情况下，PSoC Creator 将实例名称“EZI2C_1”分配给提供的设计中的第一个组件实例。您可以将其重命名为遵循标识符语法规则的任何唯一值。该实例名称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
void EZI2C_Start(void)	启动对 I²C 流量的响应。（使能中断）
void EZI2C_Stop(void)	停止对 I²C 流量的响应（禁用中断）
void EZI2C_EnableInt(void)	使能中断，这是大多数 I²C 操作所必需的。
void EZI2C_DisableInt(void)	禁用中断。EZI2C_Stop() API 自动执行此操作。
void EZI2C_SetAddress1(uint8 addr)	设置主要 I²C 地址。
uint8 EZI2C_GetAddress1(void)	返回主要 I²C 地址。
void EZI2C_SetBuffer1(uint16 bufSize, uint16 rwBoundry, void * dataPtr);	设置主要 I²C 的缓冲区指针。
uint8 EZI2C_GetActivity(void)	检查组件活动状态。
void EZI2C_Sleep(void)	停止 I²C 操作，保存 I²C 配置（禁用中断）。
void EZI2C_Wakeup(void)	恢复 I²C 配置并启动 I²C 操作。（使能中断）
void EZI2C_Init(void)	使用自定义程序提供的初始值初始化 I²C 寄存器。

函数	说明
void EZI2C_Enable(void)	激活硬件，开始组件操作。
void EZI2C_SaveConfig(void)	保存 EZI2C 组件的当前用户配置。
void EZI2C_RestoreConfig(void)	恢复非保留 I^2C 寄存器。

可选辅助地址 API

只有当使能两个 I^2C 地址时，才提供下列这些命令。

函数	说明
void EZI2C_SetAddress2(uint8 addr)	设置辅助 I^2C 地址。
uint8 EZI2C_GetAddress2(void)	返回辅助 I^2C 地址。
void EZI2C_SetBuffer2(uint16 bufSize, uint16 rwBoundry, void * dataPtr);	设置辅助 I^2C 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**从睡眠模式使能唤醒**时，下列这些函数才可用。

函数	说明
void EZI2C_SlaveSetSleepMode(void)	使能 EZI2C 睡眠地址解码并保存 I^2C 配置。禁用中断。
void EZI2C_SlaveSetWakeMode(void)	禁用 EZI2C 睡眠地址解码，恢复 I^2C 配置，并启动 I^2C 操作。使能中断。

全局变量

正常操作时不需要了解这些变量。

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 I^2C 主控的数据指针。
EZI2C_rwOffsetS1	存储读写操作的偏移，在第一个从器件地址的每个写入序列进行设置。
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。

函数	说明
EZI2C_wrProtectS1	存储第一个从器件地址的只读数据的偏移。
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： 这是开始组件操作的首先方法。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。
此 API 不使能 I²C 中断；但是大多数 I²C 操作需要中断。

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时钟关断。

参数： 无

返回值： 无

副作用： 无

void EZI2C_EnableInt(void)

说明： 使能 I²C 中断。大多数操作需要中断。应当在 EZI2C_Start() API 之后调用。

参数： 无

返回值： 无

副作用： 无

void EZI2C_DisableInt(void)

说明： 禁用 I^2C 中断。通常不需要此函数，因为 Stop 函数禁用中断。

参数： 无

返回值： 无

副作用： 如果在 I^2C 仍运行时禁用 I^2C 中断，则可能导致 I^2C 总线锁定。

void EZI2C_SetAddress1(uint8 address)

说明： 此函数设置主要存储器缓冲区的 I^2C 地址。此值可以为 0 到 127 之间的任何值。

参数： **地址s**：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 I^2C 从器件地址。

参数： 无

返回值： 通过 SetAddress1 或默认 I^2C 地址设置的同一 I^2C 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 I^2C 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： 无

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	如果检测到启动，则检测到停止时清除。
EZI2C_STATUS_ERR	当检测到 I²C 硬件错误时设置，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： **地址**：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 I^2C 地址的 I^2C 主控的数据。只有当在用户参数中选择了两个 I^2C 地址时，才提供此函数。

参数： **bufSize**：公开给 I^2C 主控的缓冲区的大小。

rwBoundry: 设置 I^2C 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 I^2C 数据缓冲区的数据数组或结构的指针。

返回值： 无

副作用： 无

void EZI2C_SlaveSetSleepMode(void)

说明： 这是选择**从睡眠模式使能唤醒**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。将 NAK 所有后续 I^2C 流量，直到器件进入睡眠为止。

参数： 无

返回值： 无

副作用： 如果使能“从睡眠模式唤醒”选项，将禁用 I2C 中断（仅用于 PSoC3 ES3）。

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。应当在从睡眠唤醒后立即调用。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： 如果使能“从睡眠模式唤醒”选项，将使能 I2C 中断（仅用于 PSoC3 ES3）。

void EZI2C_Sleep(void)

说明： 这是未选择**从睡眠模式使能唤醒**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。有关电源管理功能的更多信息，请参考 PSoC Creator *System Reference Guide*（《系统参考指南》）。

参数： 无

返回值： 无

副作用： 无

void 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： 根据自定义程序的“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： 所有寄存器将设置为根据自定义程序的“配置”对话框的值

void EZI2C_Enable(void)

说明： 激活硬件并开始组件操作。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。

参数： 无

返回值： 无

副作用： 无

void EZI2C_SaveConfig(void)

说明： 此函数保存组件配置。这将保存非保留寄存器。此函数还将保存“配置”对话框中定义的或相应 API 修改的当前组件参数值。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： 此函数恢复组件配置。这将恢复非保留寄存器。此函数还将组件参数值恢复为调用 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。

固件源代码示例

PSoC Creator 在“查找示例项目”对话框中提供大量示例项目，其中包括图示和示例代码。若要获取特定于组件的示例，请从“组件目录”或图示组件实例打开该对话框。若要获取一般示例，请从开始页或**文件**菜单打开该对话框。根据需要，在对话框中使用**筛选选项**以缩小可用于选择的项目列表的范围。

有关更多信息，请参考 PSoC Creator 帮助中的“查找示例项目”主题。

功能说明

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。

此组件要求您使能全局中断，因为 I^2C 硬件采用中断驱动。即使此组件需要中断，您也不需要向 ISR（中断服务子程序）中添加任何代码。 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命令位置，在主控与从器件之间创建更高级别接口。

存储器接口

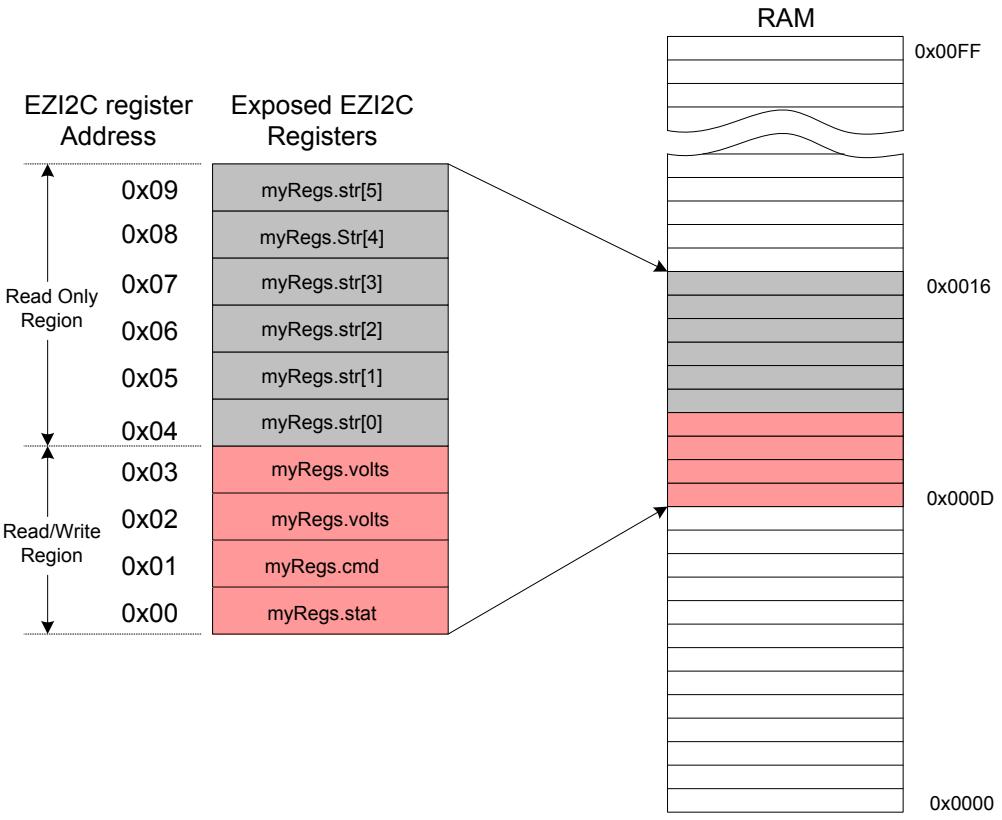
对于 I^2C 主控，该接口看上去类似于通用 I^2C EEPROM。EZI2C 接口可以配置为简单变量、数组或结构。 在某种程度上，它通过 I^2C 总线充当您的程序与 I^2C 主控之间的一个或两个共享存储器接口。该组件仅允许 I^2C 主控访问指定存储器区域，阻止该区域外的任何读取或写入。 例如，如果主要从器件地址的缓冲区按下面的代码示例配置，则存储器中的缓冲区表示可以按下图所示表示。

```
typedef struct _EZI2C_REGS
{
  uint8 stat;                    /* R/W variable */
```

```
    uint8 cmd;                  /* R/W variable */
    int16 volts;                /* R/W variable */
    uint8 str[6];           /* Read only to I2C */
} EZI2C_REGS;

EZI2C_REGS myRegs;
EZI2C_SetBuffer1(sizeof(myRegs), 4, (void *) &myRegs);
```

图 1. 公开给 I²C 主控的 EZI2C 缓冲区的存储器表示



该结构可以包含任何变量组，前提是它在存储器中是连续的且由指针引用。接口（I²C 主控）仅将其视为字节数组，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。

```
EZI2C_SetBuffer1(sizeof(myRegs), 4, (void *) &myRegs);
```

第一个参数设置公开给 I²C 接口的存储器的大小。第二个参数通过设置读/写区域中的字节数，设置读/写区域和只读区域之间的边界。读/写区域在最前面，后面跟随只读区域。在这种情况下，只能写入前 4 个字节，但是 I²C 主控可以读取所有字节。第三个参数是指向数据的指

在下例中，创建了 15 字节数组，并将其公开给 I²C 接口。该数组的前 8 字节为读/写型，其余 7 字节为只读型。

```
char theArray[15u];
```

```
EZI2C_SetBuffer2(15u, 8u, (void *) theArray);
```

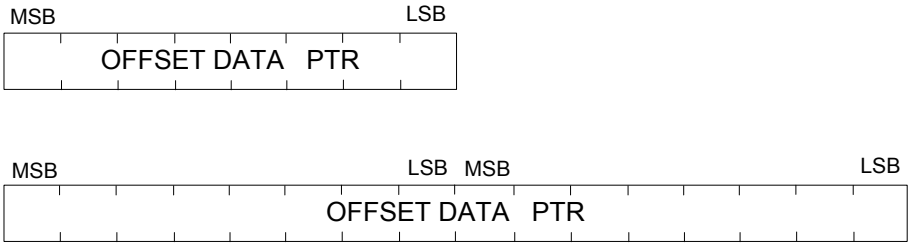
下面的简单示例显示仅公开一个整数（2 字节）。这两个字节都可由 I^2C 主控读写。

```
uint16 myVar;
EZI2C_SetBuffer1(2u, 2u, (void *) (&myVar));
```

外部主控可视的接口

EZI2C Slave 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 I^2C 地址接口包含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 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）

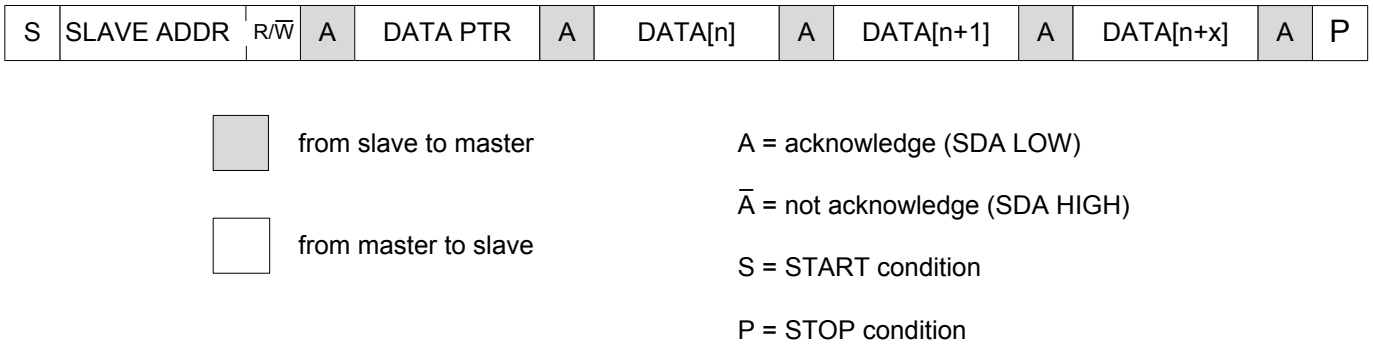


对于写入操作，第一个数据字节始终是偏移数据指针（Sub_Address_Size 的两个字节 = 16）。偏移数据指针后的字节写入偏移数据指针指向的位置。第二个数据字节写入偏移数据指针加一的位置，依此类推，直到写入完成。写入操作的长度仅受最大缓冲区读写区域大小的限制。对于写入操作，必须始终提供偏移数据指针。

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，读取由写入操作（该操作仅包含偏移数据指针，后跟重新启动（或停止/启动））和读取操作组成。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 I²C 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 I²C 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

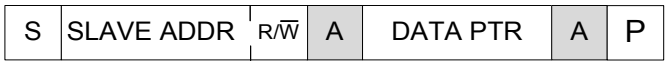
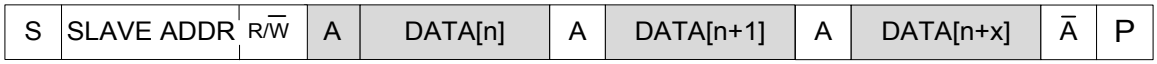


图 5 说明了 8 位偏移数据指针的读取操作。请记住，数据写入操作始终重新编写偏移数据指针。

图 5. 从 I²C 从器件读取 x 字节



在复位或加电时，将配置 EZI2C Slave 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 I²C 规范中，以及通过参考器件数据表，可以获得 I²C 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 2 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。

用户需要在主控、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。EZI2C_GetActivity() 函数可用于帮助开发特定于应用程序的机制。

从睡眠模式唤醒

如果假设使用了**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）。

器件时钟配置（总线时钟频率）可以在睡眠模式输入过程中修改（通过 CyPmSaveClocks() 函数进行修改），但必须在活动模式下继续 I2C 数据操作之前恢复（通过 CyPmRestoreClocks() 函数进行恢复）。

为了满足这些要求，在 EZI2C_SlaveSetSleepMode() 函数中禁用 I2C 中断，在 EZI2C_SlaveSetWakeMode() 中使能 I2C 中断。因此，当发生硬件地址匹配事件时，会通过将 SCL 线路保持为低电平（时钟伸展过程）来暂停数据操作。

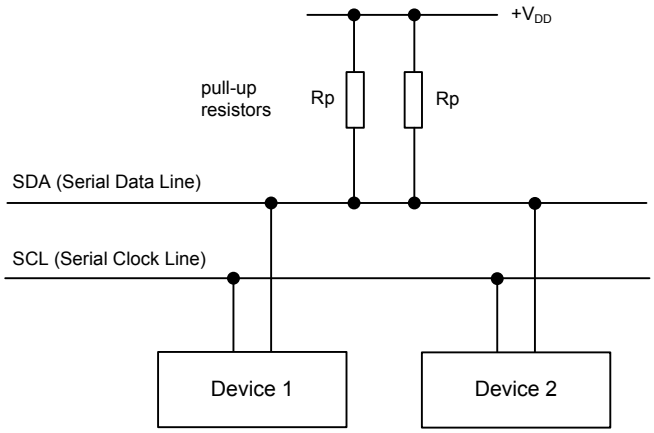
使能**从睡眠模式使能唤醒**功能的情况下的正确睡眠模式输入过程：

```
    /* 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();
```

外部电气连接

根据框图所示，I^2C 总线需要外部上拉电阻。上拉电阻 (RP) 由供电电压、时钟速度和总线电容决定。将输出阶段的任何器件（主控或从器件）的最小灌电流设置为不超过 3 mA（在 VOLmax = 0.4V 的情况下）。这会将 5V 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速度。对于总线电容为 150 pF 的 5V 系统，上拉电阻不应大于 6 kΩ。 有关更多信息，请参见 Philips 网站 www.philips.com 上的 *The I^2C-Bus Specification*（I^2C 总线规范）。

图 6. 器件与 I²C 总线的连接



注意：从赛普拉斯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。

中断服务子程式

中断服务子程式供组件代码自身使用，用户不能修改。

参考

不适用。

直流电和交流电电气特征

EZI2C 直流电规范

参数	说明	条件	最小值	典型值	最大值	单位
	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

EZI2C 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

本节列出组件与以前版本相比的主要更改。

版本	更改说明	更改/影响原因
1.60	更改了使用从器件使能位的方法：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()。现在首先在 EZI2C_RestoreConfig() 函数中恢复 I2C 配置寄存器。	原因是为了实现 EZI2C_Start() - EZI2C_Stop() - EZI2C_Start() and EZI2C_Sleep() - EZI2C_Wakeup() 序列的正确执行结果。预计没有功能影响。
	自定义程序中的标签“I2C 总线速度:”替换为“数据速率:”。“从睡眠模式唤醒”一节添加到了“功能说明”中	I2C 总线规范命名和 I2C/EZI2C 组件之间的一致性。
	自定义程序中的标签“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。
	自定义程序中的标签“从睡眠模式启用唤醒”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	
	对数据手册进行了少量编辑和更新	
1.50.a	将组件移动到组件目录中的子文件夹	
1.50	标准数据速率已更新，最高可支持 1 Mbps。	允许将 I^2C 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/使能 API。	目的是支持低功耗模式，以及提供通用接口以分别控制大多数组件的初始化和使能。
	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 ES3 器件的支持。	应用了必需的更改，以支持 PSoC 3 ES2 和 ES3 器件之间的硬件更改。
	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 EZ I^2C 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。

版本	更改说明	更改/影响原因
	为 Microsoft Windows 7 优化了窗体高度。	在 Windows 7 中，在自定义程序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的子文件夹中。	
	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“配置”对话框。	
	在图示中，数字端口已更改为引脚组件	

© 赛普拉斯半导体公司，2011。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组件。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。

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组件。若将赛普拉斯产品用于生命支持系统应用中，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。