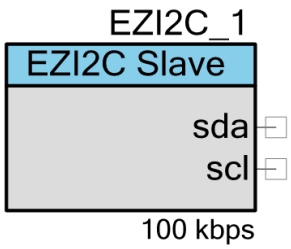
EZI2C 从设备

1.90



特性

- 工业标准 NXP® I²C 总线接口
- 仿真通用 I²C EEPROM 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从设备组件实现基于I²C寄存器的从设备。它与NXP I2C总线规范所定义的I²C标准、快速和超快速模式的设备相兼容[1]。主设备在I²C总线上启动所有通信，并为所有从设备提供时钟。EZI2C从设备支持高达1000 kbps的标准数据速率，且与同一总线上的多个设备兼容。

EZI2C 从设备是 I²C 从设备的唯一实现，主设备和从设备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。该接口表现为主设备与从设备之间的共享存储器。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：模拟毛刺滤波器、I/O V_{OL}/I_{OL}、I/O 迟滞。I²C 模块带有数字毛刺滤波器（在睡眠模式下无效）。通过将各个 I/O 设置为慢速可以达到组件在快速工作模式下的最小下降时间。更多详细信息，请参考器件数据手册的“输入和输出”一节中的 I/O 电气规范。

何时使用 EZI2C 从设备

在 I²C 从设备与 I²C 主设备之间需要共享存储器时使用此组件。可以在代码中将 EZI2C 从设备的缓冲区定义为任何变量、阵列或结构，而无需考虑 I²C 协议。I²C 主设备可以查看该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 从设备的各种输入和输出连接。

sda — 输入/输出

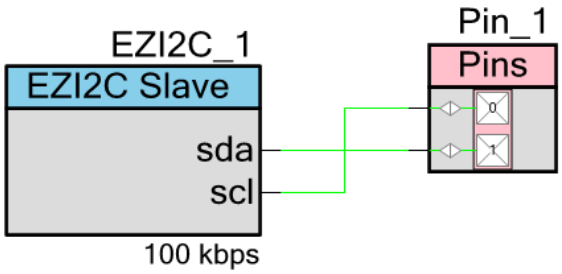
串行数据（SDA）是 I²C 数据信号。这种双向数据信号用于传输或接收所有总线数据。

SCL — 输入/输出

串行时钟（SCL）是主设备生成的 I²C 时钟。虽然从设备从不生成时钟信号，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。

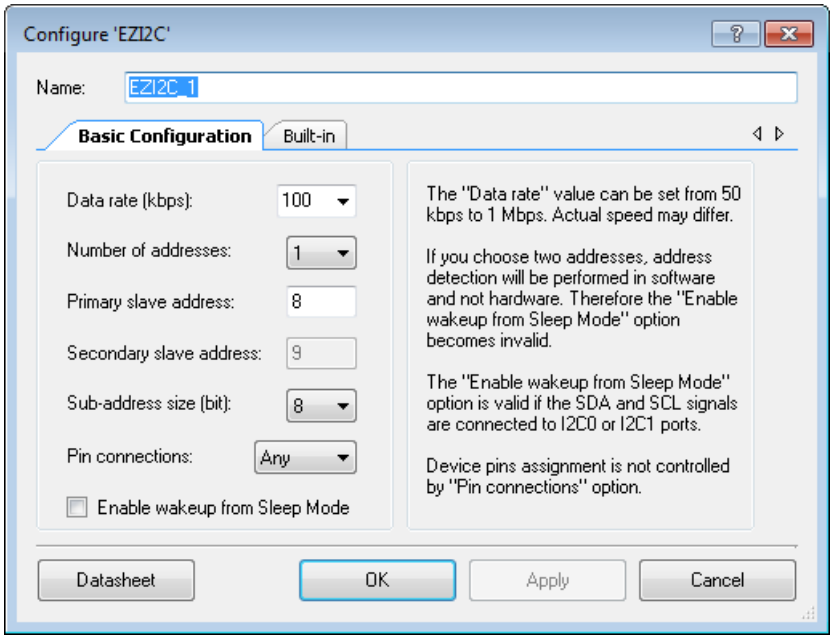
原理图宏信息

组件目录中默认的 EZI2C 从设备组件是使用默认的 EZI2C 从设备设置的示意宏。EZI2C 从设备组件直接连接到配置为 SIO 对的特定引脚。



Component Parameters（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。

Data rate（数据速率）

该参数用于设置 I²C 数据速率，支持的速率高达 1000 kbps；实际速率可能因可用时钟速率和分频器范围而异。标准数据速率为 50、100（默认值）、400 和 1000 kbps。

Number of addresses（地址数）

该选项确定是识别一个（默认值）还是两个独立 I²C 从设备地址。如果识别了两个地址，则将使用软件（而不是硬件）执行地址检测，因此 **Enable wakeup from Sleep Mode**（使能从睡眠模式唤醒）选项不可用。

Primary slave address（主从设备地址）

这是主 I²C 从设备地址（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”。该地址为右对齐的 7 位从设备地址，它不包括读/写位。

Secondary slave address（辅从设备地址）

这是辅 I²C 从设备地址（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”。仅在 **Number of addresses**（地址数）参数被设置为 **2** 时，该地址才有

效。主从设备地址和辅从设备地址必须不同。该地址为右对齐的 7 位从设备地址，它不包括读/写位。

Sub-address Size（辅助地址大小）

该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，则主设备只能访问 0 到 255 之间的数据偏移。您还可以选择 16 位辅助地址大小。这样将允许 I²C 主设备在每个从设备地址访问最大达 65,536 个字节的数据阵列。

Pin connections（引脚连接）

该参数确定要用于 SDA 和 SCL 信号连接的引脚类型。该选项用于补充 **Enable wakeup from Sleep mode**（使能从睡眠模式唤醒）选项，仅在 **Number of addresses**（地址数）选项中选择了单个 I²C 地址时才可用。该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）。

- 如果不需要 **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]），这样您可以配置器件，以便在 I²C 地址匹配时唤醒。

Enable wakeup from Sleep Mode（使能从睡眠模式唤醒）

该参数允许器件在从设备地址匹配时从睡眠模式唤醒。默认情况下，禁用该选项。仅在选择单一 I²C 地址且 SDA 和 SCL 信号连接到 SIO 端口（引脚对 I2C0 或 I2C1）时，地址匹配时的唤醒选项才有效。**Enable wakeup from Sleep mode** 受 PSoC 3 和 PSoC 5LP 的支持。

应用编程接口（API）

通过应用编程接口（API）子程序，您可以使用软件对组件进行配置。下表列出并说明了每个函数的接口。以下各节将对每个函数加以说明。

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。实例名称会成为与该组件实例相关的每个全局函数名称、变量和常量符号的前缀。为了提高可读性，下表中使用了实例名称“EZI2C”。

基本函数

函数	说明
EZI2C_Start()	启动对I²C通信的响应。使能中断。
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操作都需要使能中断。
EZI2C_DisableInt()	禁用中断。EZI2C_Stop() API自动执行该操作。
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()	获取组件活动状态。
EZI2C_Sleep()	停止I²C操作，并保存I²C配置。禁用中断。
EZI2C_Wakeup()	恢复I²C配置，并启动I²C操作。使能中断。
EZI2C_Init()	使用定制器提供的初始值初始化I²C寄存器。
EZI2C_Enable()	激活硬件，并且开始组件操作。
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。

void EZI2C_Start(void)

说明： 这是开始执行组件操作的优选方法。EZI2C_Start()设置initVar变量，并依次调用EZI2C_Init()和EZI2C_Enable()函数。它必须在I²C总线操作之前执行。

EZI2C_Start()调用EZI2C_Enable()之后，EZI2C_Enable()会调用EZI2C_EnableInt()，这样会使能I²C中断。

参数： 无

返回值： 无

其他影响： 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

其他影响： 无

void EZI2C_EnableInt(void)

说明： 使能I²C中断。大部分操作都需要中断。在调用EZI2C_Start() API时会调用该函数。

参数： 无

返回值： 无

其他影响： 无

void EZI2C_DisableInt(void)

说明： 禁用I²C中断。通常情况下，Stop()函数禁用中断后不需要再调用该函数。

参数： 无

返回值： 无

其他影响： 如果在运行I²C时禁用了I²C中断，则I²C总线可能会锁定。

void EZI2C_SetAddress1(uint8 address)

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

参数： address：0到127之间的7位从设备地址。该地址右对齐，不包括读/写位。

返回值： 无

其他影响： 无

uint8 EZI2C_GetAddress1(void)

说明： 返回主要存储器缓冲区的I²C从设备地址。

参数： 无

返回值： 通过SetAddress1或默认I²C地址设置的同一I²C从设备地址。

其他影响： 无

void EZI2C_SetBuffer1(uint16 bufSize、uint16 rwBoundry、volatile uint8 * dataPtr)

说明： 该函数设置从设备数据的缓冲区指针、大小以及读写区域。这是公开给I²C主设备的数据。

参数： bufSize：缓冲区的大小（单位为字节）。

rwBoundry：设置在缓冲区开头可写入的字节数。该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

其他影响： 调用该函数前必须先调用EZI2C_Start()。

uint8 EZI2C_GetActivity(void)

说明： 如果自从上次调用该函数后发生I²C读取或写入，则该函数将返回非零值。在该函数调用结束时，活动标志将复位为零。

读取时会清除读写忙标志，但是只有I²C Stop才能清除“BUSY”标志。

参数： 如果检测到活动状态，则返回非零值。

返回值： I²C活动的状态。

常量	说明
EZI2C_STATUS_READ1	如果检测到第一个地址的读取序列，将设置该值。该值会在读取状态时被清除。
EZI2C_STATUS_WRITE1	如果检测到第一个地址的写入序列，将设置该值。该值会在读取状态时被清除。
EZI2C_STATUS_READ2	如果检测到第二个地址的读取序列（如果使能），将设置该值。该值会在读取状态时被清除。
EZI2C_STATUS_WRITE2	当检测到第二个地址的写入序列（如果使能）时，将设置该值。该值会在读取状态时被清除。
EZI2C_STATUS_BUSY	当检测到“启动”状态时，设置该值。检测到停止状态时会清除该值。
EZI2C_STATUS_ERR	检测到I²C硬件错误时将设置该值。读取状态时会清除该值。

其他影响： 无

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()函数前，需要先调用EZI2C_Sleep()函数。欲了解更多有关功耗管理函数的详细信息，请参考*《系统参考指南》*中“PSoC Creator”一节的内容。

参数： 无

返回值： 无

其他影响： 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。即使在调用EZI2C_Sleep()函数前组件已被使能，EZI2C_Wakeup()函数也将重新使能组件。

参数： 无

返回值： 无

其他影响： 在EZI2C_SaveConfig()或EZI2C_Sleep()前调用该函数会产生意外行为。

void EZI2C_Init(void)

说明： 根据Configure对话框的设置初始化或恢复组件。不需要调用EZI2C_Init()，因为EZI2C_Start()API将调用该函数，这是开始组件操作的优选方法。

参数： 无

返回值： 无

其他影响: 所有寄存器都将根据Configure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将调用该函数，这是开始组件操作的优选方法。调用EZI2C_EnableInt()以使能I²C中断。

参数： 无

返回值： 无

其他影响： 无

void EZI2C_SaveConfig(void)

说明： 此函数会保存组件配置以及非保留寄存器。它还保存当前的组件参数值（该值是在“Configure”对话框中定义的或是通过相应API修改的）。该函数由EZI2C_Sleep()函数调用。

参数： 无

返回值： 无

其他影响： 无

void EZI2C_RestoreConfig(void)

说明： 该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调用EZI2C_Sleep()函数之前的值。

参数： 无

返回值： 无

其他影响： 调用EZI2C_Sleep()或EZI2C_SaveConfig()之前调用该函数会产生意外行为。

可选辅从设备地址 API

仅在使能了两个 I²C 地址时，这些指令才有效。

函数	说明
EZI2C_SetAddress2()	设置辅从设备I²C地址。
EZI2C_GetAddress2()	返回辅从设备I²C地址。
EZI2C_SetBuffer2()	设置辅从设备I²C的缓冲区指针。

void EZI2C_SetAddress2(uint8 address)

说明： 设置辅助存储器缓冲区的I²C从设备地址。该值可以为0到127之间的任何值。仅在用户参数中选择了两个I²C地址时，才提供该函数。

参数： address：0到127之间的7位从设备地址。该地址右对齐，不包括读/写位。

返回值： 无

其他影响： 无

uint8 EZI2C_GetAddress2(void)

说明： 返回辅助存储器缓冲区的I²C从设备地址。仅在用户参数中选择了两个I²C地址时，才提供该函数。

参数： 无

返回值： 通过SetAddress2或默认I²C地址设置的同一I²C从设备地址。

其他影响： 无

void EZI2C_SetBuffer2(uint16 bufSize, uint16 rwBoundry, volatile uint8 * dataPtr)

说明： 该函数设置辅从设备数据的缓冲区指针、大小以及读写区域。这是公开给辅从设备I²C地址的I²C主设备的数据。仅在用户参数中选择了两个I²C地址时，才提供该函数。

参数： bufSize：I²C主设备可读取的缓冲区的大小。

rwBoundry：I²C主设备可读/写的字节数量。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数据缓冲区的数据阵列或结构的指针。

返回值： 无

其他影响： 调用该函数前必须先调用EZI2C_Start()。

全局变量

正常运行时，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启。 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设备地址的I²C主设备的数据指针。
EZI2C_rwOffsetS1	存储用于读取及写入操作的偏移。它在第一个从设备地址的每个写入序列上设置。
EZI2C_rwIndexS1	存储要为第一个从设备地址读取或写入的下一个值的指针。
EZI2C_wrProtectS1	存储第一个从设备地址的只读数据的偏移。
EZI2C_bufSizeS1	存储公开给第一个从设备地址的I²C主设备的数据阵列的大小。
EZI2C_dataPtrS2	存储公开给第二个从设备地址的I²C主设备的数据指针。
EZI2C_rwOffsetS2	存储读写操作的偏移，在第二个从设备地址的每个写入序列进行设置。
EZI2C_rwIndexS2	存储要为第二个从设备地址读取或写入的下一个值的指针。
EZI2C_wrProtectS2	存储第二个从设备地址的只读数据的偏移。
EZI2C_bufSizeS2	存储公开给第二个从设备地址的I²C主设备的数据阵列的大小。
EZI2C_curState	存储I²C状态机的当前状态。
EZI2C_curStatus	存储组件的当前状态。

示例固件源代码

在 Find Example Project（查找示例项目）对话框中，PSoC Creator 提供了大量的示例项目，包括原理图和示例代码。要获取组件特定的示例，请打开组件目录中的对话框或原理图中的组件实例。要查看通用示例，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** 选项来限定可选的项目列表。

更多有关信息，请参考《PSoC Creator 帮助》中主题为“查找示例项目”一节的内容。

MISRA 合规性

本节介绍了 MISRA-C:2004 合规性和本组件的偏差情况。定义了下面两种类型的偏差：

- 项目偏差 — 适用于所有 PSoC Creator 组件的偏差
- 特定偏差 — 仅适用于该组件的偏差

本节提供了有关组件特定偏差的信息。在 *《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。

该 EZI2C 组件具有以下特定偏差：

规则	类别 [2]	规则说明	偏差说明
17.4	R	阵列索引是唯一允许的指针运算形式。	组件使用阵列索引操作访问缓冲区。访问前，先检查缓冲区大小。这是一项安全操作，以防止用户提供的缓冲区大小有错误。
19.7	A	函数应该优先于类似于函数的宏。	因为使用函数宏以实现更高的代码效率而导致了偏差。

该组件有以下嵌入式组件：中断。欲了解 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。

API 储存器的使用情况

根据编译器、器件、所使用的 API 数量以及组件的配置情况的不同，组件的存储器使用情况也不一样。下表提供了给定组件配置中的所有 API 的存储器使用情况。

2 *必须/参考*

下表中的存储器大小是在将相应编译器设置为 Release 模式并且优化选项为 Size 的情况下测得的。对于特定的设计，分析编译器生成的映射文件后可以确定存储器的使用情况。

配置	PSoC 3（Keil_PK51）		PSoC 5LP（GCC）	
	闪存（字节）	SRAM（字节）	闪存（字节）	SRAM（字节）
一个地址	1265	21	1412	25
两个地址	1751	37	1644	43

功能说明

该组件支持具有一个或两个 I²C 地址的 I²C 从设备。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 I²C 硬件是中断驱动的，因此在使用该组件时，必须使能全局中断。即使该组件需要中断，您也不必向 ISR（中断服务子程序）添加任何代码。该组件为所有中断（数据传输）提供服务，与您的代码无关。为该接口分配的存储器缓冲区看上去类似于您的应用与 I²C 主设备之间的简单双端口存储器。

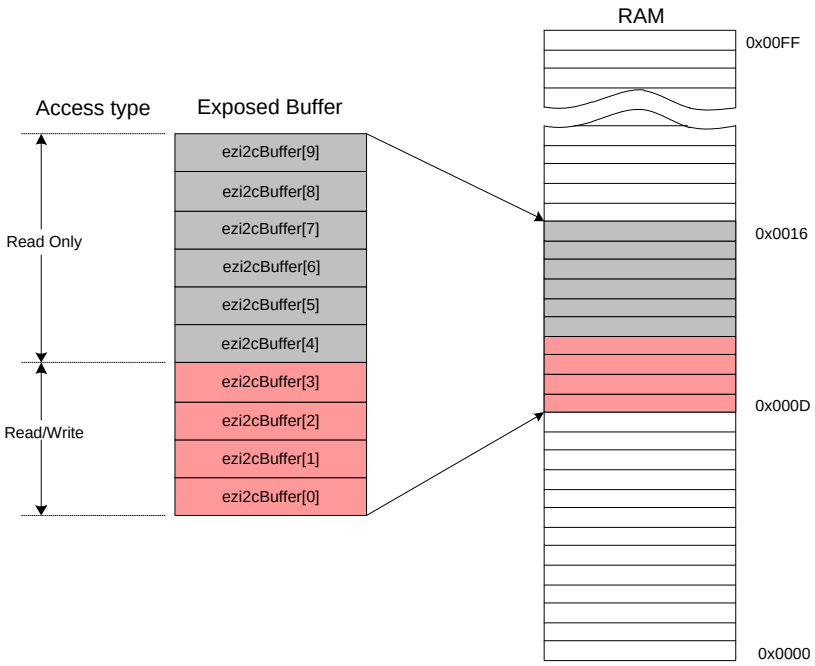
如果需要，可以通过在数据结构中定义信号和指令位置，在主设备与从设备之间创建更高级别的接口。

存储器接口

对于 I²C 主设备，该接口看上去非常类似于通用 I²C EEPROM。EZI2C 接口可以配置为简单变量、阵列或结构，但使用阵列是一个较安全的方法。在某种程度上，它通过 I²C 总线充当您的程序与 I²C 主设备之间的一个或两个共享存储器接口。该组件仅允许 I²C 主设备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，如果主要从设备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)

uint8 ezi2cBuffer[BUFFER_SIZE];


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

图 1. 公开给 I²C 主设备的 EZI2C 缓冲区的存储器表示



为了能配置整个缓冲区为读取和写入访问，缓冲区以及读取/写入边界需要大小相同。例如：

```
EZI2C_SetBuffer1(BUFFER_SIZE, BUFFER_SIZE, ezi2cBuffer);
```

结构体的处理

EZI2C 缓冲区可按照结构体进行设置。接口（I²C 主设备）仅将结构体视为字节阵列，不能访问所定义的区域外的任何存储器。

编译器列出了存储器的结构，并且可以添加额外的字节。这个被称为填装。编译器将添加这些字节，用于使文件结构对齐，从而使之满足 Cortex-M3 的要求。当使用某个结构体时，该应用必须考虑到这种对齐方式。要想避免填装字节，应使用“packed”（封装）属性：

```
struct
{
    uint8 status;
    uint8 data0;
    uint32 data1;
} __attribute__ ((packed)) ezi2cBuffer;

SCB_EzI2CSetBuffer1(sizeof(ezi2cBuffer),       sizeof(ezi2cBuffer),(uint8       *)
&ezi2cBuffer);
```

字节顺序的处理

对于不同架构，数据的传输会以不同的字节顺序进行。因此，对于特定的字节顺序，需要发送额外的代码。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）。更多有关字节顺序的信息，请参见 *《系统参考指南》* 中的“寄存器访问”一节。

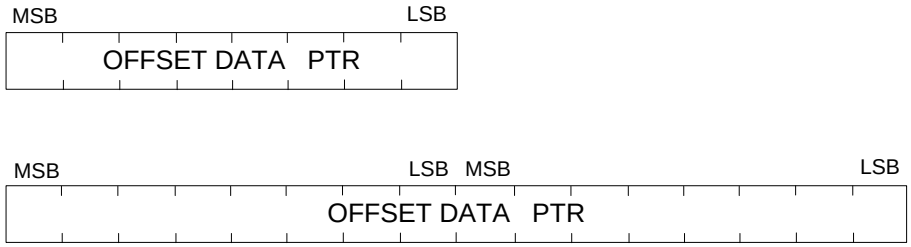
下面的简单示例显示了一个整数（两个字节）的传输。该数的两个字节都可由 I²C 主设备进行读写操作。

```
uint16 ezi2cVariable1;
CY_SET_REG16(&ezi2cVariable1, 0xABCD);
EZI2C_SetBuffer1(2u, 2u, (uint8 *) (&ezi2cVariable1));
```

外部主设备可视的接口

EZI2C 从设备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，主设备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进行编写。8 位的辅助地址用于对 256 字节的缓冲区进行访问；16 位的辅助地址用于对 65536 字节的缓冲区进行访问。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）

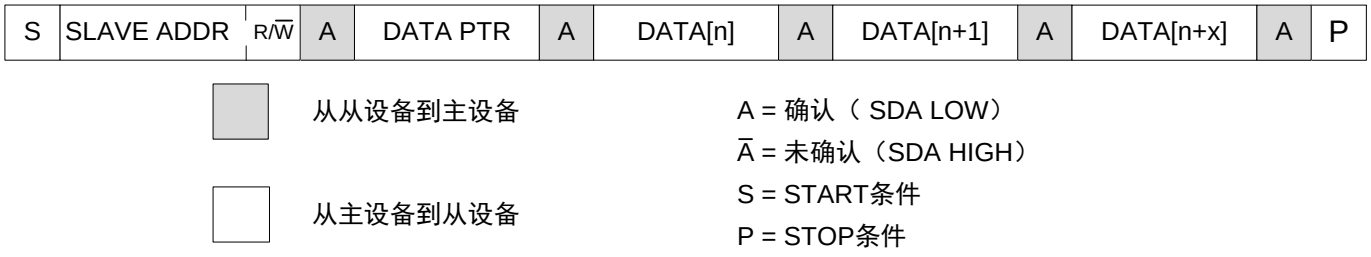


对于写入操作，第一个数据字节始终是偏移数据指针（辅助地址大小的两个字节 = 16）。偏移数据指针后的字节写入偏移数据指针指向的位置。第二个数据字节写入偏移数据指针加一的位置，依此类推，直到写入完成为止。写入操作的长度仅受最大缓冲区读写区域大小的限制。对于写入操作，必须始终提供偏移数据指针。

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。与写入操作相同，每次读取一个字节，偏移数据指针都会递增。新的读取操作将不会从上一次读取操作停止处继续。新的读取操作始终在上一次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该紧密跟随来极大提高读取操作速度。

图 3. 将 x 个字节写入到 I²C 从设备



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。在启动新写入操作前不会更改偏移数据指针。

如果 I²C 主设备尝试跨过 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。在允许的范围外不可读取数据。在允许范围外的任何主设备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. 设置从设备数据指针

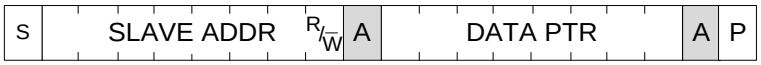


图 5 说明了 8 位偏移数据指针的读取操作。 请记住，数据写入操作始终重新编写偏移数据指针。

图 5. 从 I²C 从设备读取 x 个字节

在复位或加电时，将配置 EZI2C 从设备组件并提供 API，但是必须使用 EZI2C_Start()函数明确使能资源。

在 Philips 网站上提供的完整 I²C 规范中，以及通过参考器件数据手册，可以获得 I²C 总线的详细说明和实现。

数据一致性

虽然数据缓冲区可以包含大于单字节的数据结构，但是主设备的读取或写入操作仍由多个单字节操作组成。这可能会引起数据的一致性问题，因为没有任何机制能够确保多字节读取或写入操作在接口两侧（主设备和从设备）是同步的。例如，考虑一个包含单一的两字节整数的缓冲区。虽然主设备每次读取 2 字节整数的一个字节，但是从设备可能在主设备读取整数的第一个字节（LSB）到要读取第二个字节（MSB）期间已经更新了整个整数。主设备读取的数据可能无效，因为 LSB 读取自原始数据，而 MSB 读取更新的值。

您必须在主设备、从设备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设备或从设备不会进行更新。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

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

可以在睡眠模式输入过程中（通过 CyPmSaveClocks()函数）修改器件时钟配置（总线时钟频率）。必须先（通过 CyPmRestoreClocks()函数）恢复该配置，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_Sleep()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_Wakeup()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_Wakeup()为止。

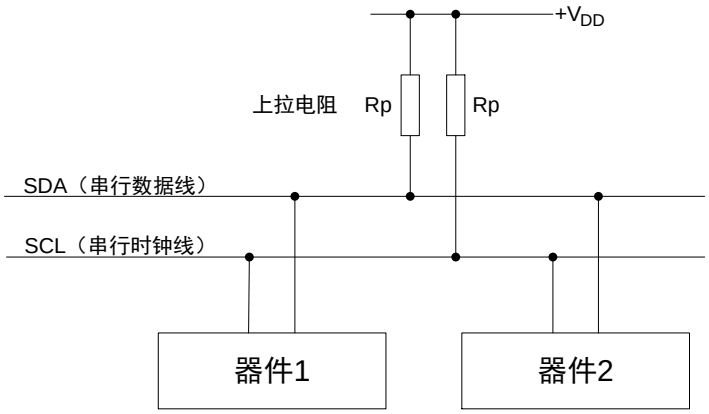
下面是使能 **Enable wake up from Sleep mode**（使能从睡眠模式唤醒）时的正确睡眠模式输入过程：

```
/* Prepares EZI2C to wake up from Sleep mode */
EZI2C_Sleep();

/* Switches to the Sleep mode */
CyPmSaveClocks();
CyPmSleep(PM_SLEEP_TIME_NONE, PM_SLEEP_SRC_I2C);
CyPmRestoreClocks();

/* Prepares EZI2C to work in Active mode */
EZI2C_Wakeup();
```

外部电气连接

如图 6 所示，I²C 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确定。将输出级的任何器件（主设备或从设备）的最小灌电流设置为不超过 3 mA（在 VOLmax = 0.4V 的条件下）。这会将 5 V 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 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 *I²C 总线规范*。

图 6. 器件与 I²C 总线的连接



注意：从赛普拉斯或获得许可的其中一个联营公司处购买 I²C 组件，即可根据 Philips I²C 专利获得一份使用许可，以在符合 Philips 定义的 I²C 标准规范的 I²C 系统中使用这些组件。

时钟选择

时钟连接到系统总线时钟，用户不能更改。

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。您不应该对它进行修改。

使用资源

该组件使用了固定功能 I²C 模块和一个中断。

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

组件勘误表

本节列出了组件的已知问题。

赛普拉斯ID	组件版本	问题	解决方案
197653	v1.90	通过Stop() API函数，I2C固定函数模块被复位。需要重新加载地址寄存器的值才能实现该操作。为了正确加载该地址，Stop() API函数需要检查I2C固定函数模块是否接受了该地址值。	添加一个子程序，用以确保合适地址寄存器值得到恢复。这样可以确保该组件能够按照设计中的总线时钟频率工作。 请参考以下建议。

在 *EZI2C.c* 文件中查找 Stop()函数，该文件位于以下位置：

- 对于 64 位操作系统 — c:\Program Files (x86)\Cypress\PSoC Creator\3.1\PSoC Creator\psoc\content\CyComponentLibrary\CyComponentLibrary.cylib\EZI2C_v1_90\API\
- 对于 32 位操作系统 — c:\Program Files\Cypress\PSoC Creator\3.1\PSoC Creator\psoc\content\CyComponentLibrary\CyComponentLibrary.cylib\EZI2C_v1_90\API\

通过进行下列各步骤，修改 Stop()函数：

1. 声明变量 i

```
uint8 i;
```

2. 添加一个子程序以写入和检查地址寄存器，直到读取值等于写入的值为止。

```
/* Reset fixed-function block */
    `$INSTANCE_NAME`_CFG_REG &= ((uint8) ~`$INSTANCE_NAME`_CFG_EN_SLAVE);
    `$INSTANCE_NAME`_CFG_REG |= `$INSTANCE_NAME`_CFG_EN_SLAVE;

    i = 255u;
    do
    {
        `$INSTANCE_NAME`_ADDR_REG = `$INSTANCE_NAME`_backup.adr;
        i--;
    }
    while ((`$INSTANCE_NAME`_ADDR_REG != `$INSTANCE_NAME`_backup.adr) &&
      (i != 0u));
```

组件更改

本节列出了该组件各版本中的主要更改内容。

版本	更新内容	更改原因/影响
1.90.b	更新数据手册并将其添加到组件勘误表章节。	请参考赛普拉斯ID 197653。
1.90.a	仅更新了数据手册。	API部分没有按顺序排列。也对它进行了更新，使之符合最新模板的要求。
1.90	更新了“MISRA合规性”章节。	该组件具有所描述的特定偏差。
	EZI2C_bufSizeS1 、 EZI2C_wrProtectS1 、 EZI2C_bufSizeS2 和 EZI2C_wrProtectS2 等各种全局变量的类型都从uint8改为uint16。	如果辅助地址大小为8位，将支持长度为256个字节的缓冲区。
	移除了EZI2C_SlaveSetSleepMode()和EZI2C_SlaveSetWakeMode()的相关内容。	这些函数已过时。使用EZI2C_Sleep()和EZI2C_Wakeup()来替换它们。
	已添加了“处理结构”一节。	文档改进。
1.80	添加了MISRA合规性章节。	该组件未进行MISRA合规性验证。
	在某些地方添加了表示不符合NXP I²C规格的脚注。	文档改进。
	修改了唤醒过程的控制流量，以避免禁用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。
	修复了主设备完成读取超过缓冲区大小时的控制流量性能。	由于主设备中的NAK未被检查，所以从设备不可正确地完成数据操作。
1.70.a	更正了图5。	
1.70	添加了PSoC 5LP支持。	

版本	更新内容	更改原因/影响
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()释放总线（如果被锁定）。
	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的可重入函数的可能性。	并非所有API都是真正的可重入函数。组件API源文件中的注释指出了不可真正重入的函数。 对于采用了安全方式（即通过标志或关键节防止同时调用）并且是不可重入的函数，则需要变更该项，从而消除编译器警告。
1.60.b	更正数据手册	
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（使能从睡眠模式唤醒）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。	文档改进。
1.60	更改了使用从设备使能位的方法：EZI2C_Stop()现在不清除该位，并且该位的设置操作已从EZI2C_Enable()移动到EZI2C_Init()。此时，在EZI2C_RestoreConfig()函数中恢复I²C配置寄存器。	用于实现EZI2C_Start() - EZI2C_Stop() - EZI2C_Start()以及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“使能从睡眠模式唤醒”	修改了文本，以保持与要求的一致性。
	更新了组件符号以及目录放置名称：“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。

版本	更新内容	更改原因/影响
	向数据手册添加了特性数据	
	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的子文件夹中	
1.50	已更新了标准数据速率，最高可支持1 Mbps的速率。	允许将I²C总线速度设置高达1 Mbps。
	添加了Keil可重入支持。	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Sleep/Wakeup（睡眠/唤醒）和Init/Enable（初始化/使能）API。	旨在支持低功耗模式并提供常用接口，从而单独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。
	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I²C总线速度计算和生成。
	为Microsoft Windows 7系统优化了窗体高度。	在Windows 7系统中，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。如果发生该情况，请更新到支持您的目标组件的修订版。
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	