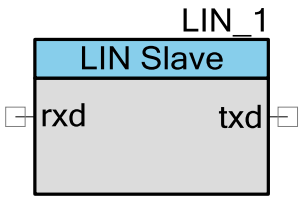
LIN 从器件

1.0



特性

- 完全实现 LIN 2.1 或 2.0 Slave Node（从器件节点）
- 符合 SAE J2602-1 规范
- 自动同步波特率
- 完全实现 Diagnostic Class I Slave Node（诊断 1 类从器件节点）
- 完全支持传输层
- 自动检测总线闲置
- 错误检测
- 自动配置服务处理
- 快速及易于配置的自定义程序
- 导入 **.ncf*/**.ldf* 文件和 **.ncf* 文件导出
- 具有语法检查的 **.ncf*/**.ldf* 文件编辑器

概述

LIN 从器件组件在 PSoC 3 和 PSoC 5 器件上实现了 LIN 2.1 从器件节点。此外，还可以选择使用 LIN 2.0 或 SAE J2602-1 合规性规范。此组件含有 LIN 总线通信所必需的硬件模块，通过 API 函数以及应用代码以轻松地与 LIN 总线通信进行交互。该组件提供的 API 与 LIN 2.1 规范指定的 API 一致。

此组件是灵活性与易于使用的完美组合。提供组件自定义程序，由此可以轻松配置 LIN 从器件的所有参数。

定义

在此数据表中给出的许多定义符合 LIN 2.1 规范。在这些情况下，如需正确了解术语定义，请参见 LIN 2.1 规范的相关章节。

输入/输出连接

本节介绍 LIN 从器件的输入和输出连接。

TXD — 输出

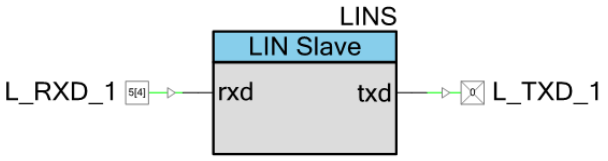
这是数字输出终端。此终端信号是该 LIN 节点发送到 LIN 总线上的数据。

RXD — 输入

这是数字输入终端。此终端信号是物理 LIN 总线上各种 CMOS 格式的信号。注意：此终端通常也接收来自 TXD 终端的所有信号。这是因为 LIN 物理层收发器具有内置回环，可以接收总线上的所有信号，无论这些信号是来自其他某些 LIN 节点，还是来自本身的 LIN 节点。

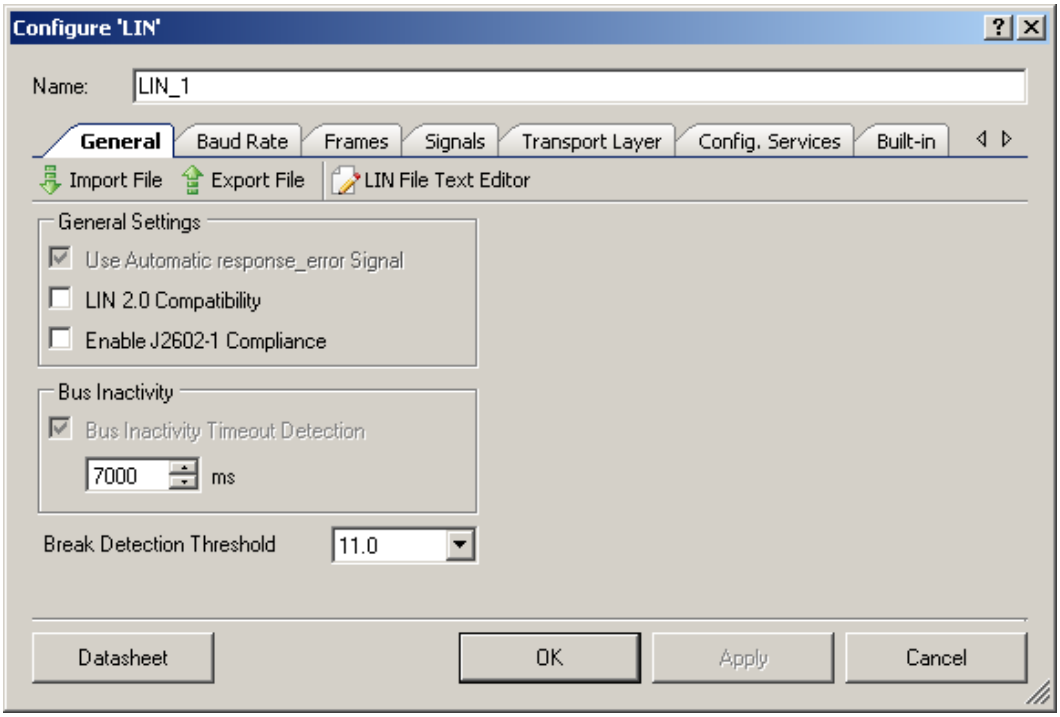
原理图宏信息

默认情况下，PSoC Creator 组件目录包含 LIN 组件的原理图宏。此宏已经包含连接以及配置的引脚组件。该原理图宏具有默认组件配置，如下图所示。



元件参数

将 LIN 从器件组件拖入设计中，双击该组件，打开 **Configure LIN**（配置 LIN）对话框。



General（通用）选项卡

Use Automatic response_error Signal（使用自动 response_error 信号）

该选项卡上的复选框设置自动发送错误信号选项。始终选中此复选框，因此在自定义程序 **Signals**（信号）选项卡中自动添加 1 位信号。此信号的默认名称，即“response_error”。无论何时出现错误响应时，组件均自动设置此信号。此外，成功发送到主机后，组件还自动清除此信号。根据 LIN 2.1 规范，此信号向 LIN 主机提供错误响应通知。

LIN 2.0 Compatibility（LIN 2.0 兼容性）

通过此选项选择此组件是否与 LIN 2.0 规范兼容。此复选框的状态影响自定义程序的其他区域。

Enable J2602-1 Compliance（实现 J2602-1 合规性）

SAE J2602-1 规范与 LIN 2.x 规范类似。它对 LIN 2.x 要求补充了几点限制。然而，此组件支持几个额外特性，从而使其与 J2602-1 完全兼容。此复选框的状态影响自定义程序的其他区域。

Bus Inactivity Timeout Detection（总线闲置超时检测）

此选项控制总线闲置特性及其值的可用性。指定总线闲置时间后，设置相应的状态位。通过 l_ifc_ioctl() 函数的 L_IOCTL_READ_STATUS 操作来获取此位的值。有关详细信息，请参见功能说明。

Break Detection Threshold（中断检测阈值）

此选项用于配置从器件节点中断检测阈值。默认值为本地主导从器件位时间的 11 倍。有关中断检测阈值选择标准的详细信息，请参见 LIN 2.1 规范第 2.3.1.1 节。

General Toolbar（通用工具栏）

这是 **General**（通用）选项卡顶侧的工具栏。此工具栏提供文件操作的访问。

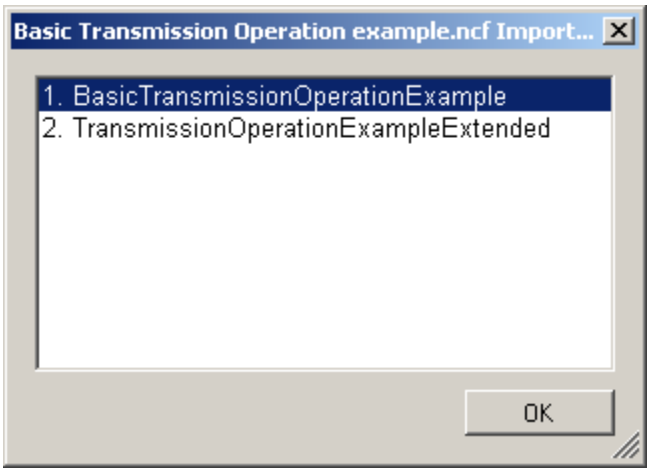
图 1. General Toolbar（通用工具栏）



Import File（导入文件）：单击此按钮，可以导入 LIN 描述文件 (LDF) 或节点能力文件 (NCF)。导入文件配置自定义程序设置，与 NCF/LDF 文件现有节点列表中选中的节点配置相互匹配。

如果导入文件的语法正确无误，则显示可用节点列表。类似列表如图 2 所示。选择其中一个要导入的可用节点说明。

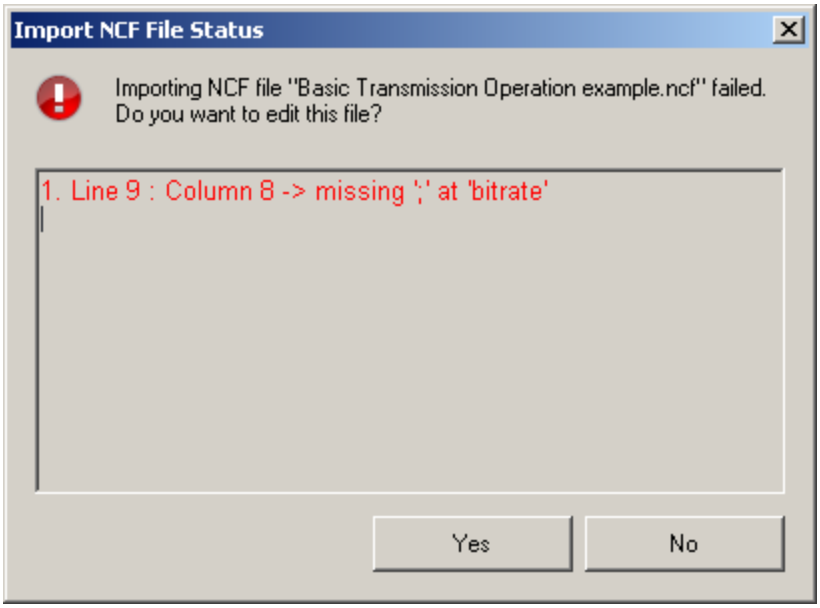
图 2. 要导入的 NCF 文件的可用节点列表



根据 LIN 节点能力语言规范（修订版 2.1）和 LIN 配置语言规范（修订版 2.1）分别验证 **.ncf* 和 **.ldf* 文件的语法。

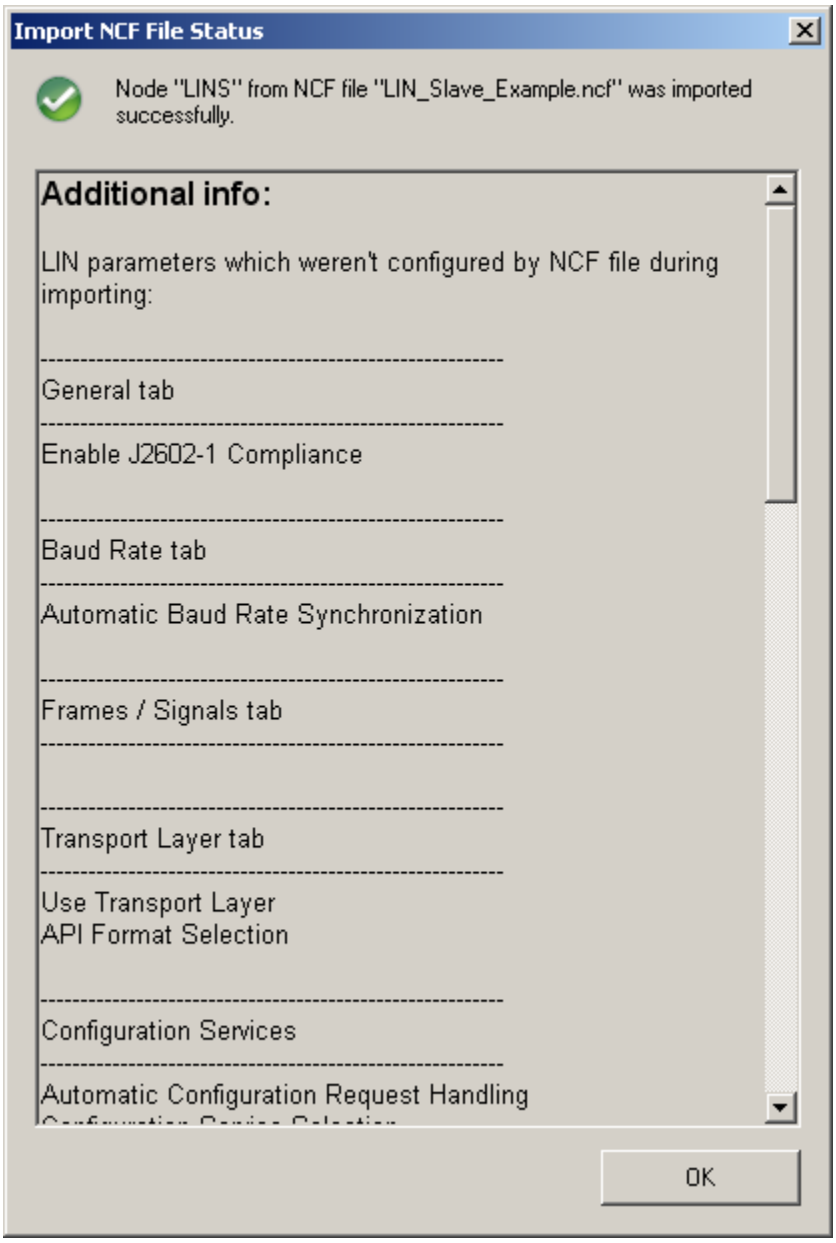
如果导入文件包含错误，则显示对话框窗口，类似于图 3。在此情况下有两个选项：使用 LIN 增强编辑器工具（更多信息，请参见 LIN File Text Editor）编辑导入文件以更正错误，或通过单击 No（否）按钮取消导入。

图 3. NCF 文件导入失败



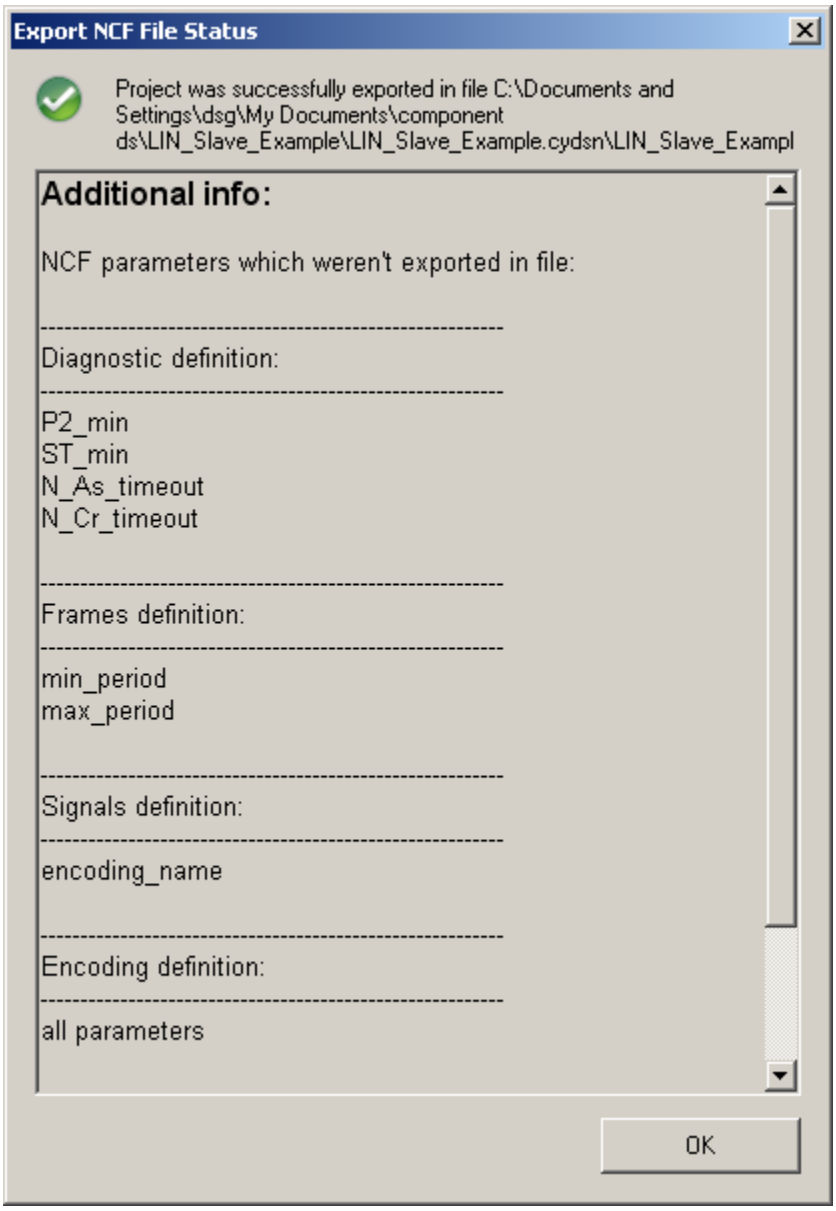
选中要导入的节点，将其导入到自定义程序后，显示一个对话框，描述导入结果（参见图 4）。导入结果包含 LIN 从器件组件参数，在导入过程中，这些参数不受任何影响。

图 4. NCF 文件导入信息



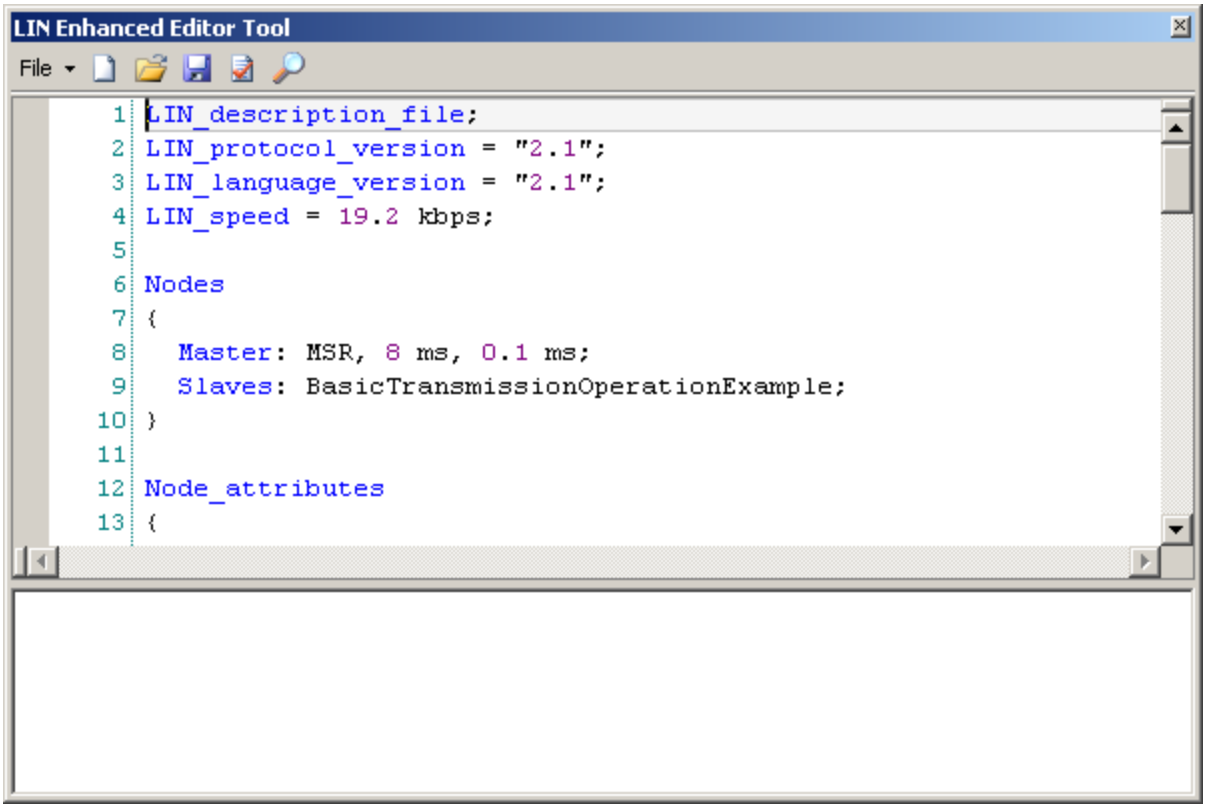
Export File（导出文件）：此工具有助于您将组件配置相关信息保存到节点能力文件 (NCF) 中。

图 5. NCF 文件导出信息



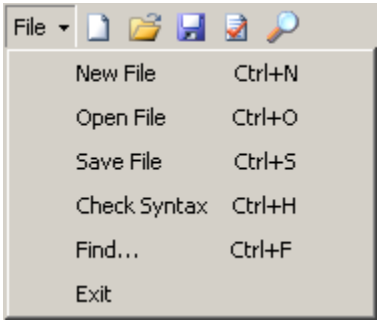
LIN File Text Editor（LIN 文件文本编辑器）：此工具用于消除、编辑和验证 NCF/LDF 文件的语法。根据 *LIN 节点能力语言规范*（修订版 2.1）来验证 **.ncf* 文件的语法。根据 *LIN 配置语言规范*（修订版 2.1）来验证 **.ldf* 文件的语法。

图 6. LIN 文件文本编辑器工具



在 **LIN Enhanced Editor Tool**（LIN 增强编辑器工具）顶侧有一个工具栏（参见图 7）。

图 7. LIN 文件文本编辑器工具栏



New File（新建文件）：创建选定 LIN 文件类型的新文件。

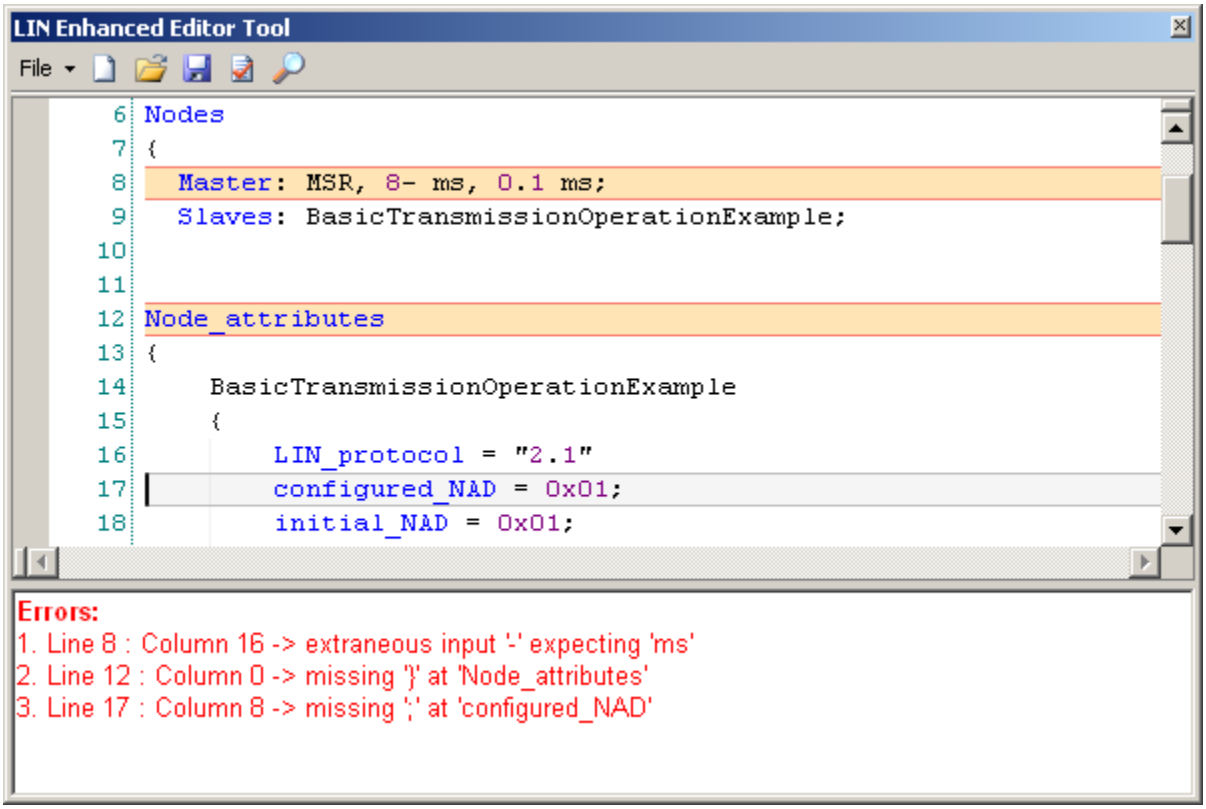
Open File（打开文件）：打开指定的现有 LIN 文件。

Save File（保存文件）：将创建完成的 LIN 文件保存到指定位置。

Check Syntax（检查语法）：通过此控制，可以检查 **.ncf*/**.ldf* 文件的语法是否正确。如果有任何语法错误，则将错误列在编辑器窗口输出区域中，说明所在位置的行数和列数，包含简要错误描述（图 8）。包含错误的代码行突出显示为红色。

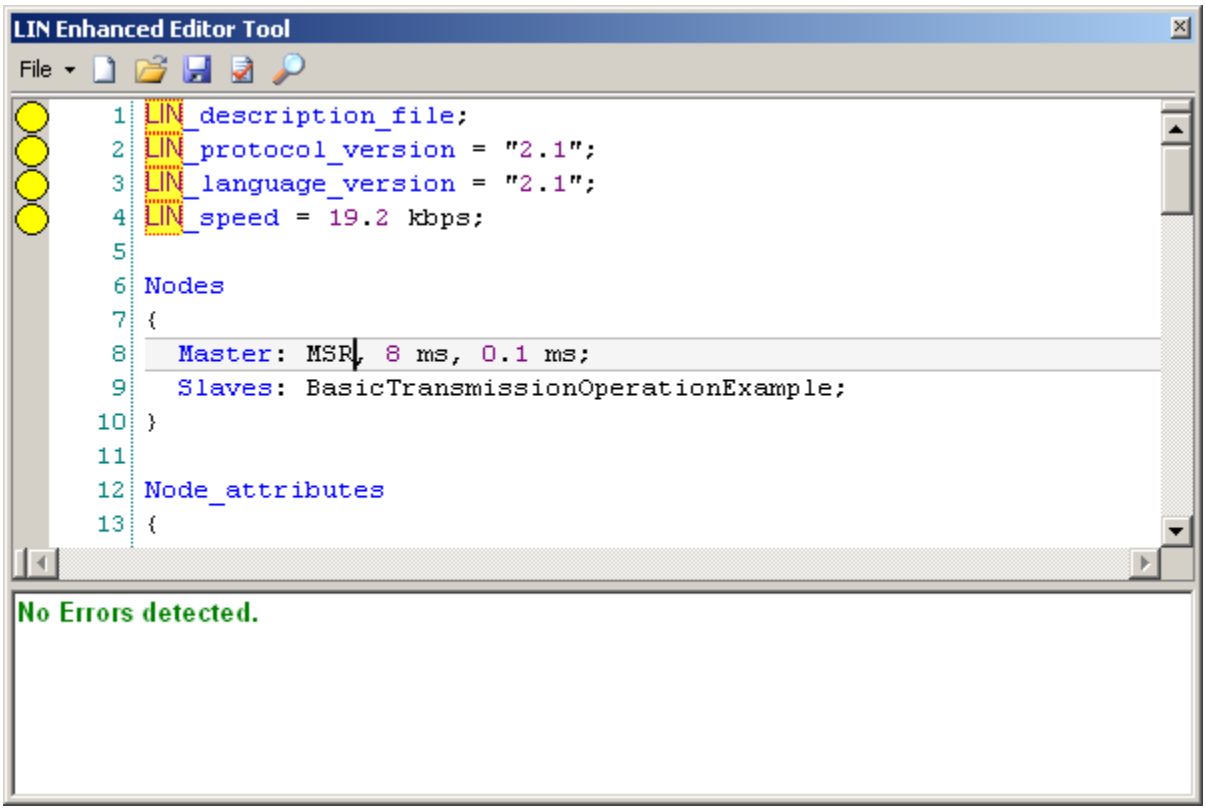
在输出区域，双击错误行，可以导航至文件中包含错误的行。

图 8. LIN 文件语法检查



Find（查找）：通过此工具，可以查找在 LIN 文件中搜索字段中指定的术语。**Find Next**（查找下一个）查找下一个匹配结果。如果选中工具的 **Mark Line**（标记行）复选框，则单击 **Find All**（全部查找）按钮后，包含必要术语的行标有黄色圆圈。单击 **Find All**（全部查找）按钮后，**Style found token**（样式查找结果令牌）复选框启用或禁用查找结果令牌的黄色突出显示功能，如 图 9 所示。**Clear**（清除）按钮删除所有突出显示的令牌。

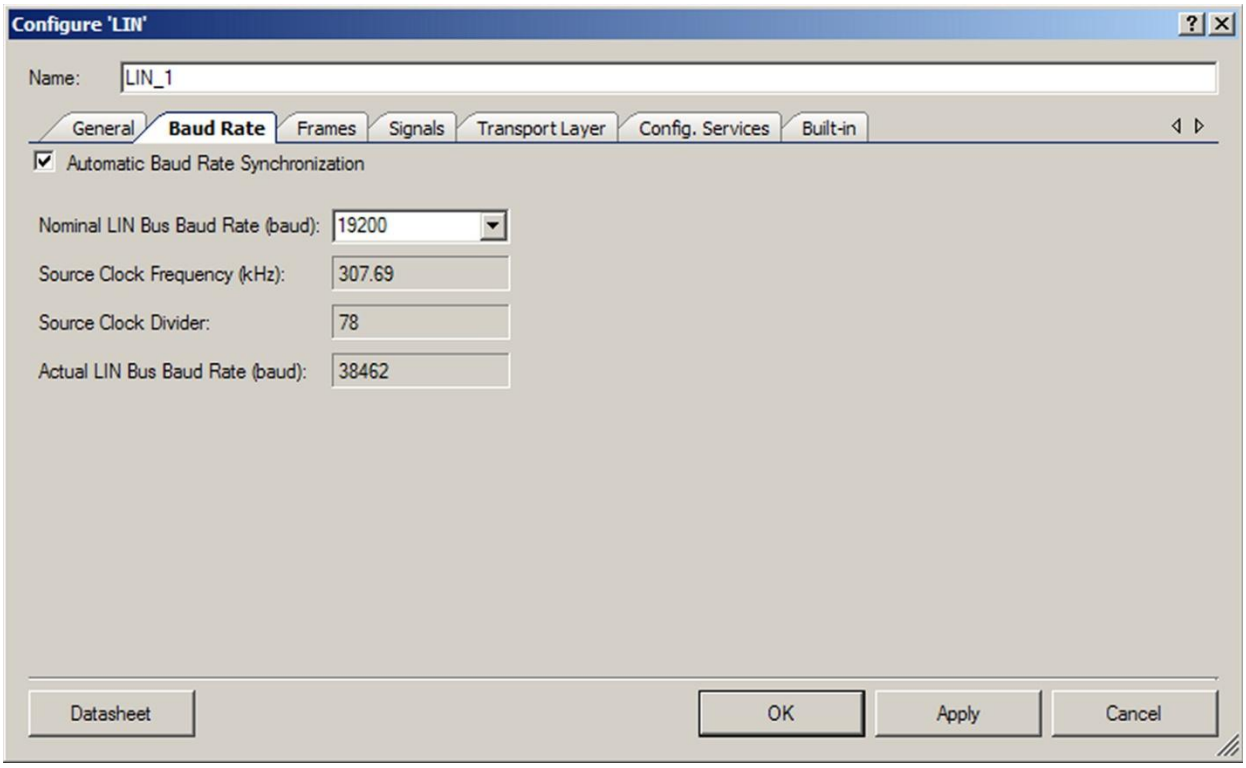
图 9. LIN 文件查找结果



此外，在 LIN 增强编辑器工具的 **File**（文件）菜单（参见图 6）及通过相应的工具栏命令，可以获得所有工具。

Baud Rate（波特率）选项卡

图 10. Configure LIN（配置 LIN）对话框、Baud Rate（波特率）选项卡



Automatic Baud Rate Synchronization（自动同步波特率）

通过此选项，可以启用或禁用自动同步波特率功能。默认情况下，此选项处于启用状态。

如果启用此选项，则组件测试确切的总线波特率，从各个 LIN 帧头的同步字节字段开始。

如果禁用此选项，则组件不测量从同步字节字段开始的波特率。而是接收作为 0x55 数据字节的同步字节字段。

根据 LIN 2.1 规范的要求，频率偏差为 ±1.5 % 或更低的 LIN 从器件节点不需要使用使用自动同步波特率来测量每个帧的同步字节字段。然而，如果 LIN 从器件的频率偏差大于 ±1.5 %，那么从器件节点必须使用自动同步波特率来测量每个帧的同步字节字段。

因此，需要针对衍生 BusClk 的时钟源检查频率偏差规范（这通常是内部主振荡器(IMO)）。

Nominal LIN Bus Baud Rate（额定 LIN 总线波特率）

输入运行此 LIN 从器件节点的额定 LIN 总线波特率。最大值为 20000 波特，最小值为 1000 波特。自定义程序禁止您选择此范围以外的波特率。在下拉列表中，这些值为 19200、10417、9600 和 2400。然而，您可以在组合框中键入介于 1000-20000 之间的任意值。如果修改 **Nominal LIN**

Bus Baud Rate（额定 LIN 总线波特率），则按 **Apply**（应用）按钮获取适合于 **Source Clock Frequency**（源时钟频率）、**Source Clock Divider**（源时钟分频器）和 **Actual LIN Bus Baud Rate**（实际 LIN 总线波特率）字段的新值。

Source Clock Frequency（源时钟频率）

这是时钟频率，重复采样数为 8，可用于数据传输。

Source Clock Divider（源时钟分频器）

这是时钟分频器的值，用来从 BusClk 中获取在 **Source Clock Frequency**（源时钟频率）中指定的时钟频率。

Actual LIN Bus Baud Rate（实际 LIN 总线波特率）

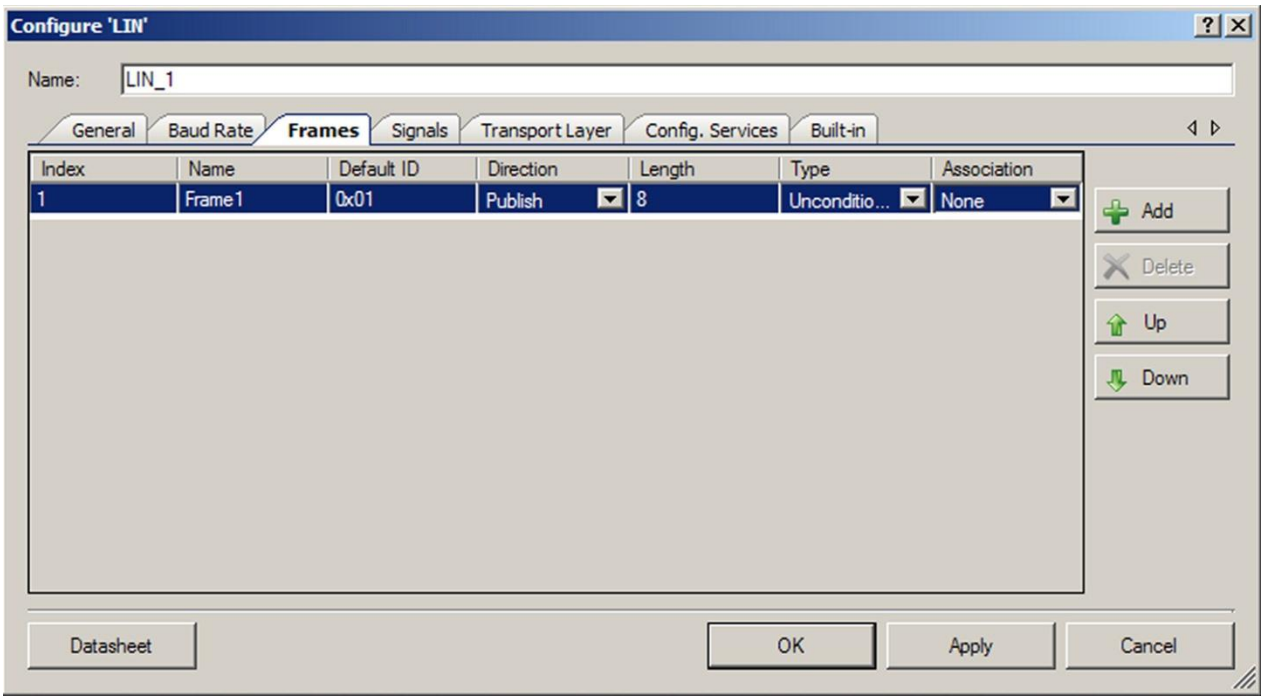
此处显示总线波特率的实际值。LIN 从器件将在此波特率下工作。可以修改 BusClk 的值，从而使 **Nominal LIN Bus Baud Rate**（额定 LIN 总线波特率）等于 **Actual LIN Bus Baud Rate**（实际 LIN 总线波特率）。

Frames（帧）选项卡

此选项卡用来配置 LIN 从器件如何响应由总线上的主控发送的 PID 值。

在此选项卡上配置的设置用于正确生成组件 API 和 ISR 代码。操作期间，LIN 从器件接收 PID（其中内嵌帧 ID），用来确定 LIN 从器件（组件）的响应方法。

图 11. Configure LIN（配置 LIN）对话框、Frames（帧）选项卡



帧配置表

该配置表位于此选项卡中间。该表包含行和列。

每一行对应一个 LIN 帧。注意：此选项卡仅显示“用户”LIN 帧。MRF 和 SRF 帧由此组件来支持，但在此表中不显示。

在数据字段中可能共有 8 个列。

在 **Index**（索引）列中的字段显示每个使用帧的排序编号。无法直接修改这些编号。

在 **Name**（名称）列中的字段用于输入每个帧的名称。可以输入 C 代码中有效的任何字符串。每个帧的名称必须是唯一的。

在 **Default ID**（默认 ID）列中的字段用于定义帧 ID，这是帧在主控请求任意配置之前使用的帧 ID。注意：这些帧 ID 均是动态的。换言之，LIN 主控可以在运行时重新配置帧 ID。在这些单元格中，必须输入 0x00-0x3B 之间的值。可以输入十六进制或十进制格式的值。

Message ID（消息 ID）列不在图 11 中显示。这是因为该列通常是不可见的列。只有在选中自定义程序的 **General**（一般）选项卡中的 **LIN 2.0 Compatibility**（LIN 2.0 兼容性）复选框时，此列才可以显示。可以输入任意 16 位值。可以输入十六进制或十进制格式的值。所有消息 ID 的值均是唯一的。此外，输入到此表中的消息 ID 的值应在整个 LIN 集群中是唯一的。例如，如果其他某些 LIN 从器件有一个帧，其中包含消息 ID 0x000F，则组件不再有任何包含消息 ID 0x000F 的帧。

在 **Direction**（方向）列中的字段定义发送帧数据的方向（相关于此从器件）。**Publish**（发布）表示数据传输；**Subscribe**（订阅）表示数据接收。

在 **Length**（长度）列中的字段定义每个帧所接收和发送的字节数。1-8 之间（包含 1 与 8）的值有效。

在 **Type**（类型）列中的字段用来定义 LIN 帧的类型。LIN 从器件有两种帧类型：**无条件**和**触发事件**类型。当帧为订阅帧时，无法选择触发事件类型。在此情况下，无法修改此单元格。如果将此单元格从**触发事件**更改为**无条件**类型，则必须在 **Association**（关联性）列中将此帧的名称更改为 **None**（无），这里假设其名称在该列的任意单元格中显示。

在 **Association**（关联性）列中的字段用于关联无条件帧与触发事件类型的帧。根据 LIN 规范，触发事件类型的帧至少要有一个与其关联的无条件类型的帧。因此，通过 **Association**（关联性）设置，可以选择与触发事件类型的帧无关的任意无条件类型的帧的名称。此设置的有效值是任意现有的无关联及无条件类型的帧的名称。只有一个无条件类型的帧可以与触发事件类型的帧相互关联。因此，当其中一个单元格包含无条件类型的帧的名称时，此无条件类型的帧的名称无法用于其他任意行。如果触发事件类型的帧与无条件类型的帧相互关联，则二者的长度和方向必须相同。因此，触发事件类型的帧的名称仅在适用这些条件的无条件类型的帧的行中显示。如果单击自定义程序的全局 **OK**（确定）按钮，或通过单击另一个选项卡退出此选项卡，则自定义程序执行检查，以确保不再有与任意无条件类型的帧关联的触发事件类型的帧。

注意： 帧部数不能超过 60。所有帧的总计大小限定为 256 个字节。

选项卡按钮

在此选项卡上共有 4 个按钮。

Add（添加）按钮添加新帧到表中。

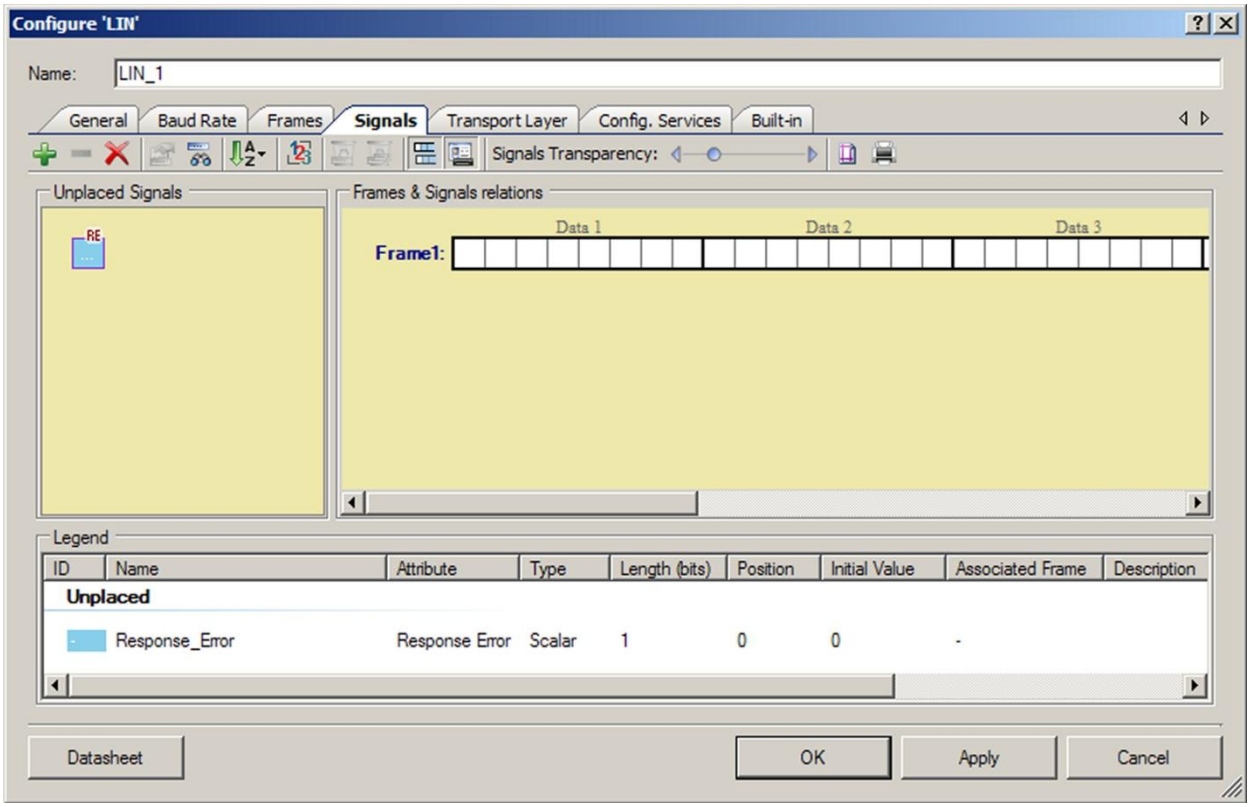
Delete（删除）按钮从表中删除当前选中的帧。相应地更改索引编号字段。如果在此选项卡上删除一个帧，则压缩到其中的任意信号（使用 **Signals**（信号）选项卡配置）均被移至**未安置信号**区域（参见 Signals（信号）选项卡部分中的 Sort Signals（排序信号）按钮）。

您可以使用 **Up**（上箭头）和 **Down**（下箭头）按钮给每个帧的**索引**重新排序编号。

Signals（信号）选项卡

在自定义程序中，此选项卡用于定义压缩到 LIN 帧中的“信号”。

图 12. Configure LIN（配置 LIN）对话框、Signals（信号）选项卡



Frames & Signals relations（帧与信号关联性）

此 **Signals**（信号）选项卡的图形区域用来显示您使用自定义程序定义的帧与信号之间交互式图形。

Frame Graphics（帧图形）：一个帧图形代表在自定义程序 **Frames**（帧）选项卡中定义的各个帧。

Signal Graphics（信号图形）：各个信号图形代表为 LIN 从器件定义的相应信号。信号图形显示为白色状态栏（参见图 12）。通过拖放操作，可以将信号放置在帧顶部。这些信号占据帧的位或字节。

在信号上单击，选中该信号。延期信号导致在工具提示中显示有关该信号的相关信息。

Unplaced Signals（未安置信号）

此图形区域是添加信号后临时存储这些信号的区域，尚未放置信号。信号可以在 **Unplaced Signals**（未放置信号）区域与 **Frames & Signals relations**（帧与信号关联性）区域之间来回移动。

注意：如果在 **Frames**（帧）选项卡上删除一个帧，则压缩到其中的任意信号（使用 **Signals**（信号）选项卡配置）均被移至 **Unplaced Signals**（未安置信号）区域。

response_error

1 位 response_error 信号自动添加在自定义程序的 **Signals**（信号）选项卡中。可以更改 response_error 信号的名称，但却无法从 **Signals**（信号）选项卡中删除该名称。

仅允许有一个 response_error 信号实例，其名称对此组件而言必须是唯一的。response_error 信号是布尔型信号，可以放置在由 LIN 从器件发布的帧上的任意位置。

此信号的目的是向 LIN 主控报告状态信息。

有关此信号的其他信息，请参见 LIN 2.1 规范第 2.7.3 节“向集群报告”。

信号工具栏

这是 **Signals**（信号）选项卡顶部上的工具栏。此工具栏提供管理选项卡上各个信号的简易方法。

图 13. 信号工具栏



1. **Add/Delete**（添加/删除）按钮

   该 **Add Signal**（添加信号）按钮添加信号到 **Unplaced Signals**（未安置信号）区域。**Delete Signal**（删除信号）按钮删除从组件中选择的信号。**Delete All Signals**（删除所有信号）按钮删除全部现有信号。

2. **Signal Properties**（信号属性）按钮

   此控制打开选中信号的 **Signal Properties**（信号属性）窗口。此窗口可以用于更改信号属性。注意：信号属性窗口还可以通过双击信号来访问。

3. **Find Signal**（查找信号）按钮

   通过此按钮，可以搜索某个信号。

4. **Sort Signals**（排序信号）按钮

   此按钮用于排序 **Unplaced Signals**（未安置信号）区域中的所有信号。可以按名称、长度或类型来排序信号。

5. **Renumber Signals**（重新编号信号）按钮

   此按钮用于以升序方式重新编号信号索引值。

6. **Move**（移动）按钮

   Unplace Signal（未安置信号）按钮可以将所选信号从 **Frames & Signals relations**（帧和信号关联性）区域移至 **Unplaced Signals**（未安置信号）区域。

   该 **Unplace All Signals**（未安置信号）按钮可以将所有信号移至 **Unplaced Signals**（未安置信号）区域。

7. **Show/Hide Event-triggered frames**（显示/隐藏触发事件类型的帧）按钮

   通过此按钮，可以显示或隐藏帧图形，其对应于 **Frames & Signals relations**（帧和信号关联性）区域中的触发事件类型的帧。

8. **Show/Hide Legend**（显示/隐藏图例）按钮

   通过此按钮，可以显示或隐藏描述信号属性的图例区域。

9. **Signals Transparency**（信号透明度）滑条

   此滑条用于设置信号图形的透明度。

10. **Print**（打印）按钮

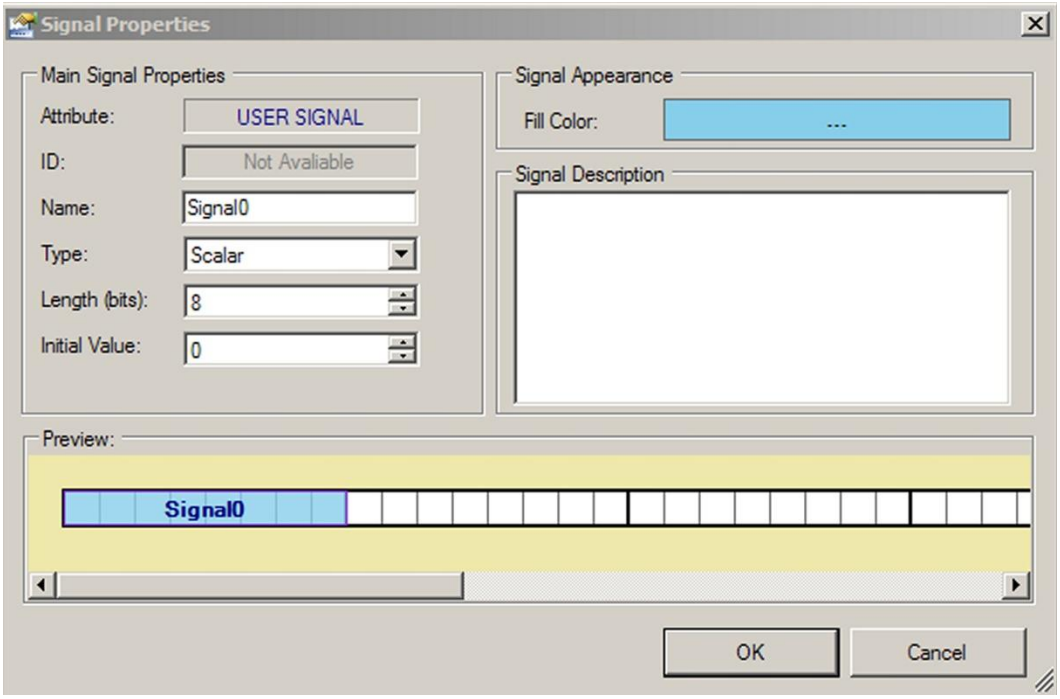
    这些按钮用于打印输出 **Frames & Signals relations**（帧与信号关联性）区域。

Signal Properties（信号属性）窗口

Adding Signals（添加信号）

在工具栏有一个 **Add Signal**（添加信号）按钮。此按钮可以生成新的窗口，用来显示可配置的信号属性选项（参见图 14）。配置完成各个属性后，添加新信号。在此章节中介绍可在此窗口中配置的各种信号属性。

图 14. Signal Properties（信号属性）窗口



Name（名称）

Name（名称）属性用于选择信号的名称。默认信号名称是 Signalx，其中，‘x’等于信号索引编号。输入的信号名称必须是 C 代码中有效的符号名称。

Type（类型）

此属性用于选择信号类型。根据 LIN 2.1 规范共定义两种信号类型。一个**标量**信号长度是 1-16 位，而 **ByteArray** 信号的长度为 1-8 个字节。

Length（长度）

此属性用于选择信号长度。标量信号长度为 1-16 位。ByteArray 信号的长度为 1-8 个字节。

Initial Value（初始值）

此属性用于选择信号的初始值。此值必须以十进制格式输入。

Fill Color（填充颜色）

此控制用于选择信号图形的颜色。

Signal Description（信号描述）

此属性用于输入任何相关描述或有关信号的其他信息。

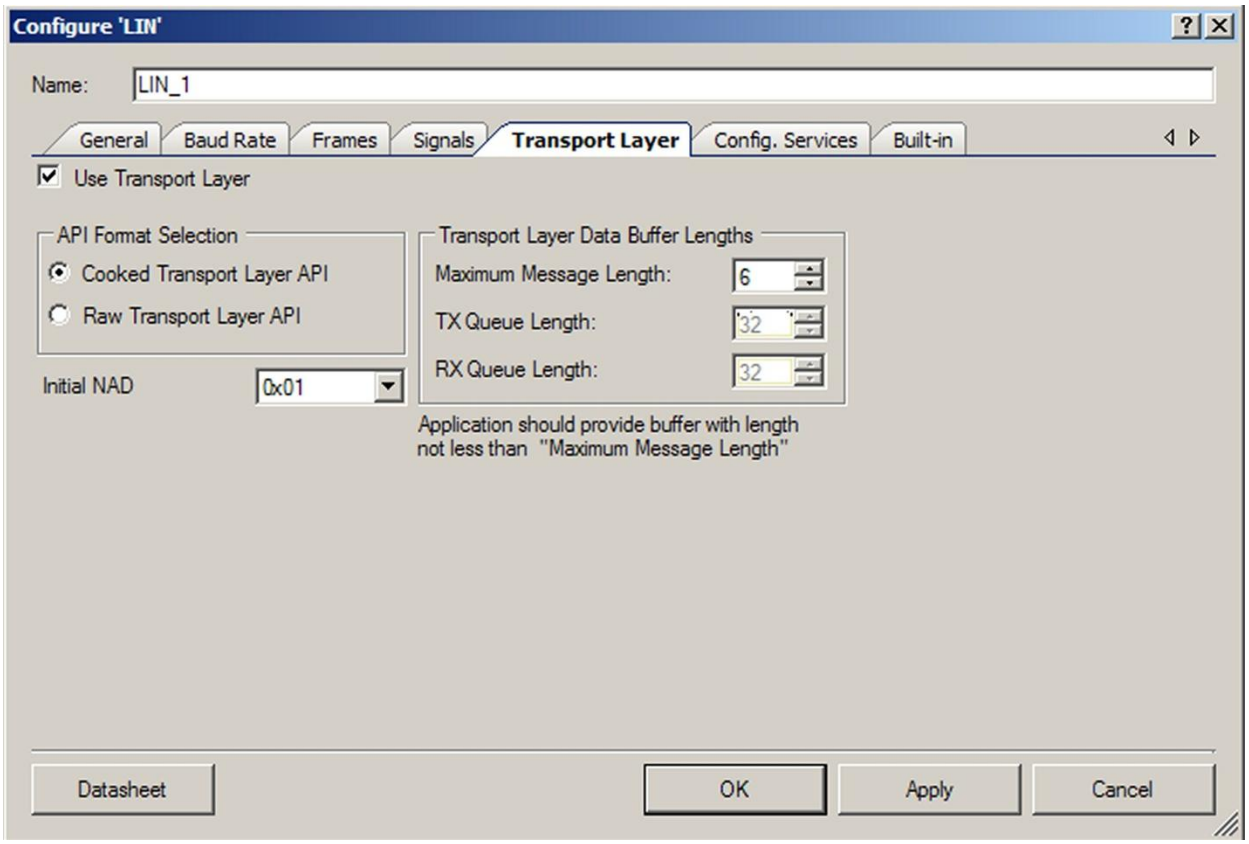
Preview（预览）

此图形区域显示添加信号时信号的外观。

Transport Layer（传输层）选项卡

Transport Layer（传输层）选项卡视图如图 15 所示。

图 15. Configure LIN（配置 LIN）对话框、Transport Layer（传输层）选项卡



Use Transport Layer（使用传输层）

如果未选中 **Use Transport Layer**（使用传输层）复选框，从器件节点将不支持传输层。如果选中，则从器件节点组件将支持传输层。有关传输层的详细信息，请参见 LIN 2.1 规范。

API Format Selection（API 格式选项）

此控制用于选择传输层 API 功能的格式。选项有 **Cooked Transport Layer API**（成熟传输层 API）和 **Raw Transport Layer API**（原始传输层 API）。通常，建议为 LIN 从器件应用选择成熟格式。成熟格式用于发送和接收传输层消息，而每条消息仅使用一个 API 功能。原始格式用来发送和接收组织传输层消息的各个帧，并使用为每个帧调用的一个 API 功能。

LIN 2.1 规范第 7.4 节定义了传输层 API 的这两种格式。

Initial NAD（初始 NAD）

此字段用于选择从器件节点的网络地址 (NAD)。NAD 用于 MRF 和 SRF 帧，以便在一个集群中寻址一个特定的从器件节点。注意：此字段用于选择节点的初始 NAD。可以在运行时更改从器件节点的 NAD。

默认情况下，初始 NAD 值的范围是 0x01-0xFF。为“转到睡眠”命令保留 NAD 的值 0x00。NAD 的值 0x7E 保留为“功能 NAD”，其用于诊断服务。NAD 的值 0x7F 保留为“通配符”NAD。因此，自定义程序限制您在此字段中输入 0x00、0x7E 或 0x7F。

如果选中 J2602-1 Compliance（J2602-1 合规性）复选框，Transport Layer（传输层）选项卡上的 Initial NAD（初始 NAD）的值限定为 0x60-0x6F。默认值为 0x60。根据自定义程序 **Frames**（帧）选项卡上使用的帧数，进一步限制初始值。有关详细信息，请参见表 1。

表 1. 根据从器件节点中使用的帧数限制初始 NAD

帧数	可用的初始 NAD 的值
1 到 4	0x60 到 0x6F
5 到 8	0x60、0x62、0x64、0x66、0x68、0x6A、0x6C、0x6E、0x6F
9 至 16	0x60、0x64、0x68、0x6E、0x6F
16 以上	0x6E、0x6F

Maximum Message Length（最大消息长度）

此属性用于选择此从器件节点所支持的传输层消息的最大长度。最小值为 6，因为在消息中，最多有 6 个传输层消息数据字节共同占用一个帧。此组件仅支持长度高达 4095 字节的传输层消息。注意：实际的传输层消息缓冲区位于节点的应用代码中。

TX 队列长度/ RX 队列长度

只有选中 **Raw Transport Layer API**（原始传输层 API）格式时，这些属性才适用。使用原始 API 格式时，有一条消息“队列”，用来缓冲当前发送或接收的帧响应数据。如果从器件无法快速更新队列，那么队列长度将变得越来越长。如果从器件可以快速更新队列，那么队列可以变短，

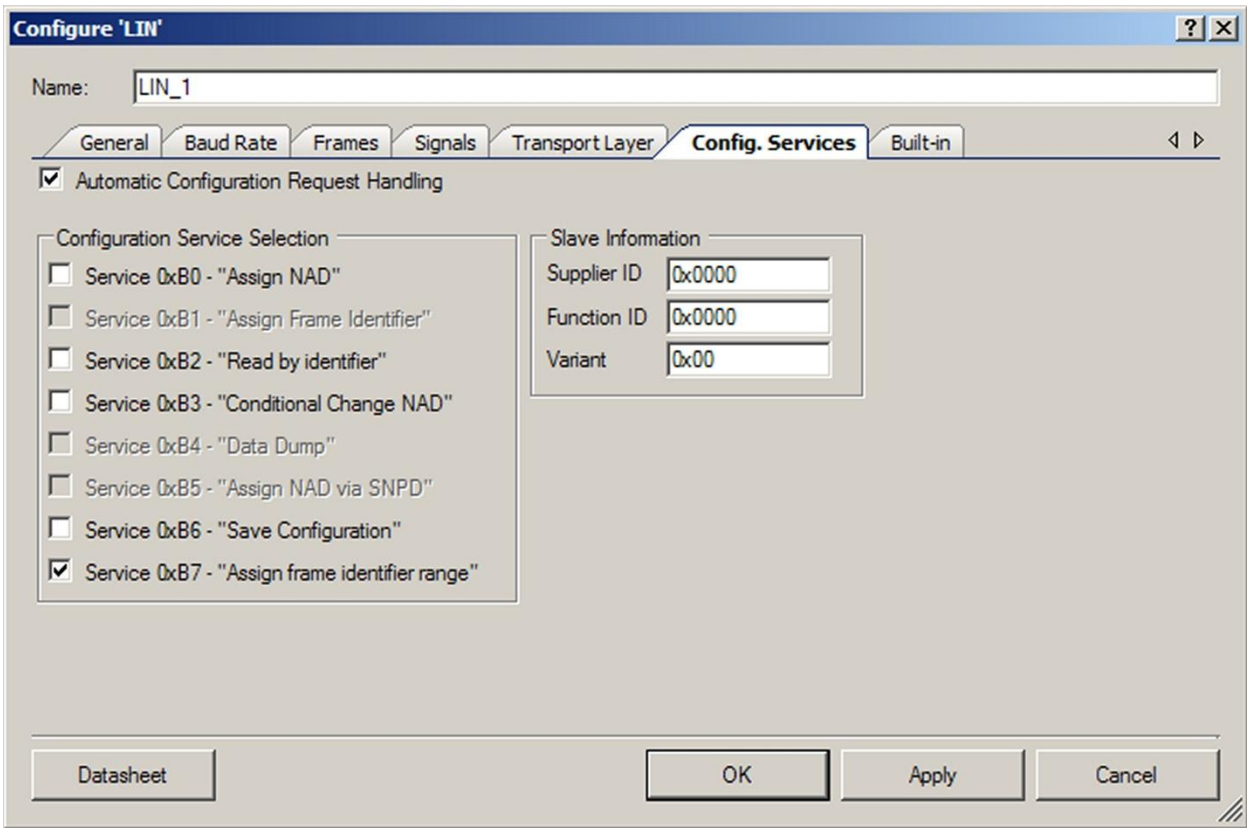
从而减少 RAM 使用量。该组件支持的队列长度为 8-2048，步长为 8 字节。每个队列的默认大小为 32 字节。

Configuration Services（配置服务）选项卡

LIN 2.1 规范用来定义从器件必须支持的配置服务请求（根据 LIN 2.1 规范，某些为必选项，而某些为可选项）。此组件支持所有必选请求和某些可选服务请求。

共有 8 项配置服务请求 (0xB0-0xB7)。在 LIN 2.1 规范的表 4.6 中列出这些服务。此组件支持其中某些服务。您可以选择禁用或启用所支持的单个服务。在 LIN 2.1 规范第 4.2.5 节介绍这些配置服务请求。

图 16. Configure LIN（配置 LIN）对话框、Configuration Services（配置服务）选项卡



配置请求自动处理

指定此组件，以便自动处理配置服务请求。换言之，您不必使用任何 API 或应用代码来处理自主控的这些服务请求。然而，您可以禁用此自动处理，而使用自定义应用代码来处理这些请求。

要简化此选项，在此选项卡上有一个 **Automatic Configuration Request Handling**（自动处理配置请求）复选框。如果选中该复选框，则此选项卡上的其他所有选项均为启用状态。如果未选中该复选框，则此选项卡上的其他所有选项均为禁用状态。

在此选项卡上启用的任何服务均由此组件自动处理。在 LIN 总线工作过程中，无论何时自动处理任何请求，相应的 MRF 和 SRF 帧无法通过传输层 API 获得应用。如果未自动处理服务请求（即在此选项卡上未启用服务请求），则必须使用传输层 API 通过应用程序接收或发送配置服务请求所对应的 MRF 和 SRF 帧。

配置服务选项

在具有复选框的选项卡上列出各个支持的配置服务请求。您可以选择需要自动处理的个别服务请求。

- 服务 0xB0 — “分配 NAD”

  这是 LIN 2.1 规范中的可选服务。

  这是将新的 NAD 值分配到从器件节点的服务请求。

  由于 PSoC 器件的高度可编程性，此组件不可能需要此服务请求。PSoC 在启动之后，可以轻松配置 NAD 到所需的值，这可能无需 LIN 主控请求 NAD 更改。

- 服务 0xB1 — “分配帧标识符”

  这是 LIN 2.1 规范中的过期服务。只有在自定义程序的 **General**（一般）选项卡上选中 **LIN 2.0 Compatibility**（LIN 2.0 兼容性）复选框时，它才处于可用状态。

  此配置服务请求用于更改此从器件节点响应的帧的 ID 值。

  在 LIN 2.1 规范中未介绍此服务。只在 LIN 2.0 规范第 2.5.1 节中对此服务有所介绍。此服务可用于此组件，以便实现向后兼容性。

- 服务 0xB2 — “通过标识符读取”

  根据 LIN 2.1 规范，此配置服务请求是必选选项。此请求用来支持 LIN 主控读取从器件标识信息（供应商 ID、功能 ID、变量）。该组件仅支持此请求的 LIN 产品标识版本。

- 服务 0xB3 — “有条件更改 NAD”

  这是 LIN 2.1 规范中的可选服务。

  它非常类似于“分配 NAD”配置服务。其中，一个主要差异是此服务使用从器件的当前（易失性）NAD，而不使用初始（非易失性）NAD。发生此请求时，从器件对从主控接收的数据字节进行某些逻辑处理，如果处理结果为零，则只更新当前（易失性）NAD。

- 服务 0xB4 — “数据转储”

  此服务请求是 LIN 2.1 规范中的可选项，不由此组件来支持。

- 服务 0xB5 — “通过 SNPD 分配 NAD”

  此服务由 LIN 2.1 规范来支持，但也可以由 J2602-1 规范来支持。此组件不支持该服务。

在 J2602-1 规范中定义了目标复位配置 (0xB5) 服务请求。因此，仅在自定义程序的 General（一般）选项卡上选中 Enable J2602-1 Compliance（启用 J2602-1 合规性）复选框时，0xB5 服务才可用（作为目标复位）。如果通过此从器件来处理目标复位请求，则在 l_ifc_ioctl() 函数的 L_IOCTL_READ_STATUS 操作中设置标志，以便让应用程序知道目标复位应予以发生。有关详细信息，请参见功能说明。

- 服务 0xB6 — “保存配置”

  这是 LIN 2.1 规范中的可选服务请求。

  从器件可以将其配置数据（NAD 值和 PID 值）保存在非易失性存储器（闪存）中。然而，应用代码必须实现实际闪存写入操作。

  当此配置服务请求发生时，在由 l_ifc_read_status() API 函数返回的状态下，设置从器件配置标志。这便通知应用程序应将当前 LIN 从器件节点的配置信息保存到非易失性存储器（闪存）中。

- 服务 0xB7 — “分配帧标识符范围”

  这是 LIN 2.1 规范中的必选配置服务请求。

  通过此服务，LIN 主控可以更改从器件帧的易失性帧的 PID 值。

从器件信息

如果选中 **Automatic Configuration Request Handling**（自动处理配置请求）复选框，则 3 个字段变为可用状态。

这些字段分别是 **Supplier ID**（供应商 ID）、**Function ID**（功能 ID）和 **Variant**（变量）。供应商 ID 是 16 位值，但其有效范围是 0x0000-0x7FFE。功能 ID 也为 16 位，其有效范围为 0x0000-0xFFFE。变量为 8 位，其有效范围为 0x00-0xFF。

这些值用于配置服务请求，以便区分 LIN 集群中各个不同的从器件节点。因此，在某些方面，这些值充当从器件地址类型。

时钟选择

PSoC Creator 计算所需频率和时钟源，并生成实现所需的各种资源。

当禁用 **Automatic Baud Rate Synchronization**（自动同步波特率）选项时，时钟容差必须为 ±1.5%，而未禁用时，时钟容差为 ±14%。如果在此限制下无法生成时钟，则显示一条警告。在此情况下，应修改 DWR 中的主控时钟源。

放置

LIN 组件放置于整个 UDB 阵列中，并且所有放置信息通过 *cyfitter.h* 文件提供给 API。仅有一个组件实例可以放置在各个设计中。

资源

模式	数字模块				API Memory（API 存储器）（字节）		引脚
	数据路径单元	PLD	状态单元	Control/Count7 单元	Flash（闪存）	RAM	
LIN_Slave_Example 项目	4	12	3	3	4321	171	2

应用程序编程接口

应用程序编程接口 (API) 子程序允许您使用软件配置组件。下表列出并介绍各个函数接口。以下各节将更详细地介绍每个函数。

默认情况下，PSoC Creator 将实例名称“LIN_1”分配给指定设计中组件的第一个实例。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“LIN”。

核心 API 函数

初始化子组

函数	说明
l_sys_init()	初始化 LIN 核心。

信号交互函数子组

函数	说明
l_bool_rd()	读取并返回 1 位信号的当前值。
l_u8_rd()	读取并返回 2-8 位信号的当前值。
l_u16_rd()	读取并返回 9-16 位信号的当前值。
l_bytes_rd()	读取并返回信号中所选字节的当前值。
l_bool_w()	将 1 位信号的当前值设置为 v。
l_u8_wr()	将信号当前值设置为 2-8 位。
l_u16_wr()	将信号当前值设置为 9-16 位。
l_bytes_wr()	设置信号中所选字节的当前值。

通知函数子组

函数	说明
l_flg_tst()	返回表示标志当前状态的布尔值。
l_flg_clr()	将标志当前值设置为零。

接口管理函数*子组

函数	说明
l_ifc_init()	初始化 LIN 从器件组件。
l_ifc_wake_up()	传输一个唤醒信号。
l_ifc_ioctl()	控制规范以外的功能。
l_ifc_rx()	LIN 从器件组件自动调用此 API 子程序。
l_ifc_tx()	LIN 从器件组件自动调用此 API 子程序。
l_ifc_aux()	LIN 从器件组件自动调用此 API 子程序。
l_ifc_read_status()	返回指定 LIN 接口的状态。

用户提供的调出子组

函数	说明
l_sys_irq_disable()	禁用组件的所有中断。
l_sys_irq_restore()	恢复组件的所有中断。

节点配置函数

函数	说明
ld_read_configuration()	串行化当前配置，并将其复制到应用程序所提供的区域（数据指针）。
ld_set_configuration()	根据输入参数指定的配置来配置 NAD 和 PID。
ld_read_by_id_callout()	主控节点根据用户定义区域中标识符的请求传输一个读数时使用。

传输层函数

初始化子组

函数	说明
ld_init()	初始化或重新初始化原始和成熟传输层。初始化所有传输层缓冲区。如果正在进行的诊断帧在总线上传输成熟或原始消息，则该函数将不会中止。

原始传输层 API 函数子组

函数	说明
ld_put_raw()	在一个帧上 8 个数据字节传输的调用队列。

函数	说明
ld_get_raw()	将最早接收的诊断帧数据复制到输入参数指定的存储器中。
ld_raw_tx_status()	返回原始帧传输函数的状态。
ld_raw_rx_status()	返回原始帧接收函数的状态。

成熟传输层 API 函数子组

函数	说明
ld_send_message()	将通过数据和长度指定的信息压缩成一个或多个诊断帧。将这些帧通过地址 NAD 传输到主控节点。
ld_receive_message()	准备 LIN 诊断模块以用来接收一条消息并将其存储到根据数据指向的缓冲区。调用时，长度表示最大允许长度。接收完成时，长度变为实际长度，NAD 变为消息中的 NAD。
ld_tx_status()	返回上次执行调用的状态至 ld_send_message()。
ld_rx_status()	返回上次执行调用的状态至 ld_receive_message()。

Non-LIN-Specified API

函数	说明
LIN_Start()	启动组件操作。
LIN_Stop()	启动组件操作。

核心 API 函数：初始化

l_bool l_sys_init()

说明： 初始化 LIN 核心。此函数不执行任何操作，始终返回零。

静态原型： l_bool l_sys_init(void)

动态原型： None（无）

参数： None（无）

Return Value（返回值）： 始终返回 0。

Side Effects（副作用）： None（无）

核心 API 函数：信号交互

在后续所有静态信号 API 调用中，“sss”是信号名称，例如，l_u8_rd_EngineSpeed()。对于后续动态信号 API 调用，“sss”是在应用程序编程接口中定义的信号句柄。

l_bool_rd()

说明： 读取并返回 1 位信号的当前值。如果无效信号句柄传递到该函数，则无需执行任何操作。

静态原型： l_bool l_bool_rd_sss(void)

动态原型： l_bool l_bool_rd(l_signal_handle sss)

参数： sss：要读取信号的信号句柄。

Return Value（返回值）： 返回信号的当前值。

Side Effects（副作用）： None（无）

l_u8_rd()

说明： 读取并返回信号的当前值。如果无效信号句柄传递到该函数，则无需执行任何操作。

静态原型： l_u8 l_u8_rd_sss(void)

动态原型： l_u8 l_u8_rd(l_signal_handle sss)

参数： sss：要读取信号的信号句柄

Return Value（返回值）： 返回信号的当前值。

Side Effects（副作用）： None（无）

l_u16_rd()

说明： 读取并返回信号的当前值。如果无效信号句柄传递到该函数，则无需执行任何操作。

静态原型： l_u16 l_u16_rd_sss(void)

动态原型： l_u16 l_u16_rd(l_signal_handle sss)

参数： Sss：要读取信号的信号句柄

Return Value（返回值）： 返回信号的当前值。

Side Effects（副作用）： 此函数并不保证读取的数据字节是原子操作。如果有必要原子化数据字节，则应用程序必须确保实现这种情况。

l_bytes_rd()

说明： 读取并返回信号中所选字节的当前值。**Start**（开始）和 **Count**（计数）参数的总和从不大于字节阵列长度。注意：当 **Start**（开始）和 **Count**（计数）总和大字节阵列长度时，读取意外**数据**。

如果无效信号句柄传递到该函数，则无需执行任何操作。

假设字节阵列长度为 8 个字节，则编号为 0-7。根据用户选择的阵列读取 2-6 个字节要求 **Start**（开始）参数为 2（跳过字节 0 和 1），**Count**（计数）参数为 5。在此情况下，字节 2 被写入 user_selected_array[0]，所有连续字节以升序格式写入 user_selected_array。

静态原型： void l_bytes_rd_sss(l_u8 start, l_u8 count, l_u8* const data)

动态原型： void l_bytes_rd(l_signal_handle sss, l_u8 start, l_u8 count, l_u8* const data)

参数： sss：要读取信号的信号句柄

开始：从第一个字节开始读取

计数：要读取的字节数

data：指针指向阵列，在此阵列中存储从信号中读取的数据

Return Value（返回值）： None（无）

Side Effects（副作用）： 此函数并不保证读取的数据字节是原子操作。如果有必要原子化数据字节，则应用程序必须确保实现这种情况。

l_bool_wr()

说明： 将值 v 写入信号。如果无效信号句柄传递到该函数，则无需执行任何操作。

静态原型： void l_bool_wr_sss(l_bool v)

动态原型： void l_bool_wr(l_signal_handle sss, l_bool v)

参数： sss：要写入信号的信号句柄
v：要设置信号的值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

l_u8_wr()

说明： 将值 v 写入信号。如果无效信号句柄传递到该函数，则无需执行任何操作。

静态原型： void l_u8_wr_sss(l_u8 v)

动态原型： void l_u8_wr(l_signal_handle sss, l_u8 v)

参数： sss：要写入信号的信号句柄
v：要设置信号的值

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

l_u16_wr()

说明： 将值 v 写入信号。如果无效信号句柄传递到该函数，则无需执行任何操作。

静态原型： void l_u16_wr_sss(l_u16 v)

动态原型： void l_u16_wr(l_signal_handle sss, l_u16 v)

参数： sss：要写入信号的信号句柄；
v：要设置信号的值。

Return Value（返回值）： None（无）

Side Effects（副作用）： 此函数不保证 LIN 主控将自动读取写入的数据字节。如果有必要原子化数据字节，则应用程序必须确保实现这种情况。

l_bytes_wr()

说明：	将所选字节的当前值写入由名称 sss 指定的信号中。虽然器件驱动程序不可能选择在运行时强制执行该操作，但是，开始和计数参数总和从不大于字节阵列的长度。注意：当开始和计数参数总和大字节阵列长度时，读取意外数据。 如果无效信号句柄传递到该函数，则无需执行任何操作。 假设字节阵列长度为 8 个字节，则编号为 0-7。写入此阵列的字节 3 和 4，开始参数为 3（跳过字节 0、1 和 2），计数参数为 2。在此情况下，从 user_selected_array[0] 开始写入字节阵列信号的字节 3，并从 user_selected_array[1] 写入字节 4。
静态原型：	void l_bytes_wr_sss(l_u8 start, l_u8 count, const l_u8* const data)
动态原型：	void l_bytes_wr(l_signal_handle sss, l_u8 start, l_u8 count, const l_u8* const data)
参数：	sss：要写入信号的信号句柄 开始：要写入到的第一个字节 计数：要写入的字节数 data：指针指向阵列，在此阵列中定位传输到 LIN 主控的数据
Return Value（返回值）：	None（无）
Side Effects（副作用）：	此函数不保证 LIN 主控以原子方式读取写入的数据字节。如果有必要原子化数据字节，则应用程序必须确保实现这种情况。

核心 API 函数：通知

通知标志用于同步应用程序与 LIN 核心。这些标志由 LIN 核心自动设置，并且仅可以由应用程序进行测试或清除。通知标志可以与信号、特定帧中的信号（在此情况下，相同信号被压缩到多个帧中）或帧对应。成功发送或接收相应信号或帧时，通过此组件设置标记。

在以下所有标记 API 子程序中，“fff”是标记名称，例如，l_flg_tst_RxEngineSpeed()。对于动态标记 API 子程序，“fff”是在应用程序编程接口中早期定义的信号句柄。

l_flg_tst()

说明：	此函数返回通过名称“fff”指定的标记的当前状态。如果该标记已被清除，则返回“假”，否则返回“真”。如果此子程序返回“真”值，则表示相应信号或帧已成功发送或接收。
静态原型：	l_bool l_flg_tst_fff(void)
动态原型：	l_bool l_flg_tst(l_flag_handle fff)
参数：	fff：标记句柄的名称
Return Value（返回值）：	返回表示由名称“fff”指定的标记的当前状态的 C 布尔值。 “假”：该标记已被清除； “真”：该标记尚未被清除。
Side Effects（副作用）：	None（无）

l_flg_clr()

说明：	清除通过名称“fff”指定的标记。此子程序应用于在测试后（在 l_flg_tst() API 之后）清除标记。该组件不会自动清除通知标记。此子程序是用来清除通知标记的唯一方法。
静态原型：	void l_flg_clr_fff(void)
动态原型：	void l_flg_clr(l_flag_handle fff)
参数：	fff：标记句柄的名称
Return Value（返回值）：	None（无）
Side Effects（副作用）：	None（无）

接口管理函数

这些调用可用来管理特定接口（到达总线的逻辑通道）。每次调用 API 时，按照接口名称（通过“iii”扩展名来表示）标记各个接口，例如，l_ifc_init_MyLinIfc()。对于此组件来说，接口名称与组件实例名称相同。此组件最多支持一个接口。因此，“iii”从未有过一个以上的有效标识符。

l_ifc_init()

说明： l_ifc_init() 初始化通过名称“iii”指定的 LIN 从器件组件实例。它用来设置内部函数，例如波特率，并启动 LIN 从器伯组件使用的数字模块。这是必须执行的第一次调用，然后使用其他任何接口相关的 LIN 从器件 API 函数。

静态原型： l_bool l_ifc_init_iii(void)

动态原型： l_bool l_ifc_init(l_ifc_handle iii)

参数： iii：接口句柄名称

Return Value（返回值）： 如果初始化成功，该函数返回零值，失败时，则返回非零值。

Side Effects（副作用）： None（无）

l_ifc_wake_up()

说明： 此函数用来传输一个唤醒信号。当调用此函数时，直接传输该唤醒信号。调用此 API 函数时，应用程序被阻止，直到在 LIN 总线上传输唤醒信号时为止。CyDelayUs() 函数用作时序源。根据在 PSoC Creator 中输入的时钟配置计算延迟。

静态原型： void l_ifc_wake_up_iii(void)

动态原型： void l_ifc_wake_up(l_ifc_handle iii)

参数： iii：接口句柄名称

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

l_ifc_ioctl()

说明： 此 API 控制其他 API 调用所不涵盖的功能。此函数用来以器件特定的方法控制此组件。

有关此函数所支持的操作，请参见元件参数这一节。

静态原型： l_u16 l_ifc_ioctl_iii(l_ioctl_op op, void* pv)

动态原型： l_u16 l_ifc_ioctl(l_ifc_handle iii, l_ioctl_op op, void* pv)

参数： iii：在 OP 中定义的操作所适用的接口句柄的名称

op：用于指定操作的参数

pv：指针指向指定操作的可选参数集，这些参数必须提供给该函数

下表介绍 l_ifc_ioctl API 函数所支持的可能的操作及其代码值。在表中列出的参数显示现有参数总数及其所具有的数据类型。

“op”操作（符号名称）	值	“pv”参数列表	说明
L_IOCTL_READ_STATUS	0x00u	None（无）	可选状态指示符
L_IOCTL_SET_BAUD_RATE	0x01u	l_u16 l_u16	修改波特率
L_IOCTL_SLEEP	0x02u	None（无）	准备低功耗模式条目的器件
L_IOCTL_WAKEUP	0x03u	None（无）	唤醒后恢复组件状态
L_IOCTL_SYNC_COUNTS	0x04u	None（无）	返回同步字段定时器计数
L_IOCTL_SET_SERIAL_NUMBER	0x05u	l_u8*	更新指针为序号

Return Value（返回值）： 没有为所选操作返回错误代码值。这意味着您必须确保传递到函数的值正确无误。

L_IOCTL_READ_STATUS 操作

在此字节中的首位是一个标记，其表示总线上尚没有针对某些已用时间的信号（启用 **Bus Inactivity Timeout Detection**（总线闲置超时检测）选项时可用）。如果已用时间是过去的某个阀值，则设置此标记。调用此 API 可以清除返回后的所有状态位。第二个位是一个标记，其表示已接收目标复位服务请求 (0xB5)（当启用 J2602-1 合规性时）。

符号名	值	说明
LIN_IOCTL_STS_BUS_INACTIVITY	0x0001u	在总线上未检测到某个已用时间的信号
LIN_IOCTL_STS_TARGET_RESET	0x0002u	已收到目标复位服务请 (0xB5)

L_IOCTL_SET_BAUD_RATE 操作

如果操作成功，返回 0，如果传递给函数的操作参数无效，则返回 1。

L_IOCTL_SLEEP 操作

如果操作成功，返回 0，如果传递给函数的操作参数无效，则返回 1。

L_IOCTL_WAKEUP 操作

如果操作成功，返回 0，如果传递给函数的操作参数无效，则返回 1。

L_IOCTL_SYNC_COUNTS 操作

返回同步字段定时器计数的当前数，即 8 个同步字段字节。

L_IOCTL_SET_SERIAL_NUMBER 操作

如果操作成功，返回 0，如果传递给函数的操作参数无效，则返回 1。

Side Effects（副作用）： None（无）

l_ifc_rx()

说明： LIN 从器件组件自动调用此 API 子程序。因此，此 API 子程序不必由应用程序代码来调用。它仅在此处列出，用来说明 LIN 规范的合规性。

静态原型： void l_ifc_rx_iii(void)

动态原型： void l_ifc_rx(l_ifc_handle iii)

参数： iii：接口句柄名称

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

l_ifc_tx()

说明： LIN 从器件组件自动调用此 API 子程序。因此，此 API 子程序不必由应用程序代码来调用。它仅在此处列出，用来说明 LIN 规范的合规性。

静态原型： void l_ifc_tx_iii(void)

动态原型： void l_ifc_tx(l_ifc_handle iii)

参数： iii：接口句柄名称

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

l_ifc_aux()

说明： LIN 从器件组件自动调用此 API 子程序。因此，此 API 子程序不必由应用程序代码来调用。它仅在此处列出，用来说明 LIN 规范的合规性。

静态原型： void l_ifc_aux_iii(void)

动态原型： void l_ifc_aux(l_ifc_handle iii)

参数： iii：接口句柄名称

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

l_ifc_read_status()

说明： 此函数返回上一个通信的状态。有关 LIN 从器件状态文字中各个信息字段的详细信息，请参见 LIN 2.1 规范。

静态原型： l_u16 l_ifc_read_status_iii(void)

动态原型： l_u16 l_ifc_read_status(l_ifc_handle iii)

参数： iii：接口句柄名称

Return Value（返回值）： 该调用返回状态文字（16 位值），如下表所示：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
上一个帧 PID								0	保存配置	事件触发的帧冲突	总线空闲	转到睡眠	过速	成功传输	响应中出错

状态文字仅根据节点传输或接收的帧来设置（总线活动除外）。调用 API 后清除状态文字。

Side Effects（副作用）： None（无）

用户提供的调出

l_sys_irq_disable()

说明： 此函数禁用组件的所有中断。进入中断掩码位时，它返回状态掩码。此函数基本等效于大多数组件的 DisableInt API。

与这些函数不同，它必须保存返回值，然后使用 l_sys_irq_restore() 函数来正确存储中断状态。强烈建议您在使用此 API 子程序时务必要小心。如果此组件中断禁用时间过长，LIN 通信可能失败。

此子程序应由应用程序来提供。然而，LIN 从器件组件可以自动实现此子程序。必要时，您可以修改子程序中的代码。

静态原型： None（无）

动态原型： l_irqmask l_sys_irq_disable(void)

参数：

Return Value（返回值）： 返回用来定义禁用中断的数字模块的中断寄存器掩码。

Side Effects（副作用）： None（无）

l_sys_irq_restore()

说明： 此函数可以恢复组件中断。它应与 l_sys_irq_disable() 结合使用。此函数基本等效于大多数组件的 EnableInt API；然而，在启动组件时，不应调用该函数。

此子程序应由应用程序来提供。然而，LIN 从器件组件可以自动实现此子程序。必要时，您可以修改子程序中的代码。

静态原型： None（无）

动态原型： void l_sys_irq_restore(l_irqmask previous)

参数： 上一个：用于定义启用中断的数字模块的中断掩码。

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

节点配置函数

ld_read_configuration()

说明： 此函数用于从易失性存储器中读取 NAD 和 PID 值。此函数可用于读取当前配置数据，然后将此数据保存于非易失性（闪存）存储器中。在 LIN 状态寄存器（由 l_ifc_read_status() 返回）中设置“保存配置”位时，应用程序应将配置数据保存于闪存中。

所读取的配置数据是串行字节。第一个字节是从器件的当前 NAD。

下一个字节是从器件向其响应的帧的当前 PID 值。PID 值是有序排列的，其中，所有帧显示在 LDF 或 NCF 文件中。

静态原型： None（无）

动态原型： l_u8 ld_read_configuration(l_ifc_handle iii, l_u8* const data, l_u8* length)

参数： iii：接口句柄名称；

data：从中读取配置数据的阵列

长度：配置数据字节数的大小。该值指向长度指针参数，将其设置为配置数据的实际长度。

Return Value（返回值）： 该函数返回下表中列出的值。

符号名	说明
LD_READ_OK	如果配置数据读取成功，则返回
LD_LENGTH_TOO_SHORT	指向（或被指向）长度指针参数的值小于配置数据的实际长度时返回。

Side Effects（副作用）： None（无）

ld_set_configuration()

说明： 此函数用于设置从器件节点的易失性 NAD 和 PID 的值。这可能用来在运行时修改 NAD 和 PID 值。通常，仅在启动后或主控发出请求后才执行此操作。此外，如果从器件更改 NAD 和/或 PID 的值，主控可能不再与从器件进行通信。

有关配置数据包含的内容及如何存储配置数据的详细信息，请参见 ld_read_configuration function()。

静态原型： None（无）

动态原型： l_u8 ld_set_configuration(l_ifc_handle iii, const l_u8* const data, l_u16 length)

参数： iii：接口句柄名称

data：适用于从器件节点的配置数据阵列

长度：配置数据字节数的大小

Return Value（返回值）： 该函数返回下表中列出的值。

符号名	说明
LD_SET_OK	如果配置数据设置成功，则返回
LD_LENGTH_NOT_CORRECT	如果长度参数的值不等于从器件节点配置数据的值，则返回
LD_DATA_ERROR	如果配置数据设置错误，则返回

Side Effects（副作用）： None（无）

ld_read_by_id_callout()

说明： 主控节点根据用户定义区域中标识符的请求传输一个读数时使用此调出。当收到这样的请求时，从驱动程序中调用从器件节点应用程序。

静态原型： None（无）

动态原型： l_u8 ld_read_by_id_callout (l_ifc_handle iii, l_u8 id, l_u8* data)

参数： iii：接口句柄名称

id：在用户定义区域中的标识符 (32-63)，根据标识符配置请求的读取

frameData：指向包含 5 个字节的数据区域。应用程序使用此区域来设置正向响应。

Return Value（返回值）： 该函数返回下表中列出的值。

符号名	说明
LD_NEGATIVE_RESPONSE	默认返回 API 状态。如果未修改 API 及将 API 重新分配至其他某些状态，则始终返回该状态。
LD_NO_RESPONSE	您可以手动设置此状态集。若设置，则表示不为服务提供任何响应。
LD_POSITIVE_RESPONSE	您可以手动设置此状态集。若设置，则表示为服务提供响应。该响应将指向或被指向 FrameData 参数。

Side Effects（副作用）： None（无）

传输层函数

传输层是 LIN 网络堆栈的更高层级。通过此层级，应用程序可以发送或接收“消息”格式而非“帧”格式的数据。消息可能是使用多个帧发送或接收的许多字节。传输层用于配置服务、诊断服务或自定义及用户定义的实现。

发送和接收传输层消息的 API 具有两种不同格式。即成熟格式和原始格式。此组件仅支持使用一种传输层 API 函数格式。在组件自定义程序的 **Transport Layer**（传输层）选项卡上选中 API 格式。

注意：要使用 LIN 传输层 API 函数，必须在组件自定义程序的 **Transport Layer**（传输层）选项卡上启用传输层的使用。

ld_init()

说明： 此子程序可以初始化或重新初始化从器件节点的传输层。使用任何传输层 API 函数前，必须调用此 API。此外，在从器件节点可以执行任何传输层通信之前，还需调用此 API。如果正在进行的诊断帧在总线上传输成熟或原始消息过程中，调用 API，该消息将中止；而 API 一直等到该消息完成时为止。

静态原型： None（无）

动态原型： void ld_init(l_ifc_handle iii)

参数： iii：接口句柄名称

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

原始传输层 API 函数

ld_put_raw()

说明： 此函数用于支持应用程序代码使用传输层发送数据。它实质上是从用户应用阵列将某些数据复制到帧缓冲阵列。此函数用于一次发送一个完整的传输层消息帧。因此，传输层消息多帧需要多次调用此 API 函数。应经常检查调用此 API 之前缓冲区中是否有放置帧的空间。

静态原型： None（无）

动态原型： void ld_put_raw(l_ifc_handle iii, const l_u8* const data)

参数： iii：接口句柄名称
data：要发送数据的阵列

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

ld_get_raw()

说明： 此函数用于支持应用程序代码使用传输层接收数据。它实质上是从帧缓冲区阵列将某些数据复制到用户应用阵列。此函数用于一次接收一个完整的传输层消息帧。因此，传输层消息多帧需要多次调用此 API 函数。如果接收队列为空，则没有要复制的数据。应经常检查调用此 API 之前缓冲区中是否有放置帧的空间。

静态原型： None（无）

动态原型： void ld_get_raw(l_ifc_handle iii, l_u8* const data)

参数： iii：接口句柄名称

data：复制最早接收的诊断帧数据的目标阵列

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

ld_raw_tx_status()

说明： 使用原始 API 时，此调用返回上次执行的总线帧传输的状态。

静态原型： None（无）

动态原型： l_u8 ld_raw_tx_status(l_ifc_handle iii)

参数： iii：接口句柄名称

Return Value（返回值）：

符号名	说明
LD_QUEUE_EMPTY	传输队列为空。如果上次调用 ld_put_raw() 已经完成，则已完成队列中所有帧的传输。
LD_QUEUE_AVAILABLE	传输队列包含许多条目，但处于未满状态。
LD_QUEUE_FULL	传输队列已满，无法接收更多帧。
LD_TRANSMIT_ERROR	传输过程中，LIN 协议出错，初始化并重新执行传输。

Side Effects（副作用）： None（无）

ld_raw_rx_status()

说明： 使用原始 API 时，此调用返回上次执行的总线帧接收的状态。

静态原型： None（无）

动态原型： l_u8 ld_raw_rx_status(l_ifc_handle iii)

参数： iii：接口句柄名称。

Return Value（返回值）：

符号名	说明
LD_NO_DATA	接收队列为空。
LD_DATA_AVAILABLE	接收队列包含可以读取的数据。
LD_RECEIVE_ERROR	传输过程中，LIN 协议出错。初始化并重新执行传输。

Side Effects（副作用）： None（无）

成熟传输层 API 函数函数

ld_send_message()

说明： 此函数用于支持应用程序代码使用传输层发送数据。它负责将隔那些在多个 SRF 帧过程中自动发送的数据加入队列。此函数用于发送完整的传输层消息。因此，传输层消息多帧仅需要调用一次此 API 函数。长度值必须介于 6-4095 字节之间。

如果消息正在进程中，则调用不返回任何操作。

静态原型： None（无）

动态原型： void ld_send_message(l_ifc_handle iii, l_u16 length, l_u8 nad, const l_u8* const data)

参数： iii：接口句柄名称

长度：要发送数据字节数的大小

nad：发送数据的目标从器件节点的地址

数据：要发送数据的阵列。RSID 的值是数据区域中第一个字节

Return Value（返回值）： None（无）

Side Effects（副作用）： 该调用为异步调用，即在消息发送完成前从不挂起，并且，只要调用 ld_tx_status() return LD_IN_PROGRESS，缓冲区不可能被应用程序更改。

ld_receive_message()

说明： 此函数支持应用程序代码使用传输层接收数据。它负责多个 MRF 帧，并将所有消息数据复制到用户应用缓冲区阵列。此函数用于接收完整的传输层消息。因此，传输层消息多帧仅需要调用一次此 API 函数。长度值必须介于 6-4095 字节之间。

静态原型： None（无）

动态原型： void ld_receive_message(l_ifc_handle iii, l_u16* const length, l_u8* const nad, l_u8* const data)

参数： iii：接口句柄名称

长度：要接收数据字节数的大小

nad：从中接收数据的从器件节点的地址

data：要接收数据的阵列。SID 的值是数据区域中第一个字节。

Return Value（返回值）： None（无）

Side Effects（副作用）： 该调用为异步调用，即在消息接收完成前从不挂起，并且，只要调用 ld_tx_status() 返回 LD_IN_PROGRESS，缓冲区不可能被应用程序更改。

ld_tx_status()

说明： 此函数返回上次调用 ld_send_message() 和在总线上执行传输层数据传输的状态。

静态原型： None（无）

动态原型： l_u8 ld_tx_status(l_ifc_handle iii)

参数： iii：接口句柄名称。

Return Value（返回值）： 可以返回下列值。

符号名	说明
LD_IN_PROGRESS	传输尚未完成。
LD_COMPLETED	传输已成功完成（因此您可以发布新的 ld_send_message call()）。此外，该值还在初始化传输层后返回。
LD_FAILED	传输结束时出错。只有部分数据得以发送。处理更多消息前，必须重新初始化传输层。要查找传输失败的原因，请检查状态管理函数 l_read_status()。
LD_N_AS_TIMEOUT	传输失败，因为 N_As 超时，当前消息传输将中止。请参见 LIN 2.1 规范第 3.2.5 节。

Side Effects（副作用）： None（无）

ld_rx_status()

说明： 此函数返回上次调用 ld_receive_message() 和在总线上执行传输层数据接收的状态。

静态原型： None（无）

动态原型： l_u8 ld_rx_status(l_ifc_handle iii)

参数： Iii：接口句柄名称

Return Value（返回值）： 可以返回下列值：

符号名	说明
LD_IN_PROGRESS	接收尚未完成。
LD_COMPLETED	成功完成接收，所有信息（长度、NAD、数据）均为可用状态。此外，还可以发布新的 ld_receive_message() 调用。此外，该值还在初始化传输层后返回。
LD_FAILED	接收结束时出错。只接收部分数据，该数据不可用。处理更多传输层消息前初始化。要查找接收失败的原因，请检查状态管理函数 l_read_status()。
LD_N_CR_TIMEOUT	接收失败，因为 N_Cr 超时，当前消息接收将中止。请参见 LIN 2.1 规范第 3.2.5 节。
LD_WRONG_SN	接收失败，因为序列号异常。

Side Effects（副作用）： None（无）

Non-LIN-Specified API

LIN_Start()

说明： 启动组件操作。不需要此函数。

静态原型： None（无）

动态原型： l_bool LIN_Start()

参数： None（无）

Return Value（返回值）： 零：初始化成功。
非零：初始化失败。

Side Effects（副作用）： None（无）

LIN_Stop()

说明：	停止组件操作。不需要此函数。
静态原型：	None（无）
动态原型：	l_bool LIN_Stop()
参数：	None（无）
Return Value（返回值）：	None（无）
Side Effects（副作用）：	None（无）

固件源代码示例

PSoC Creator 在“查找示例项目”对话框中提供了很多包括原理图和代码示例的示例项目。要获取组件特定的示例，请打开组件目录中的对话框或原理图中的组件实例。要获取通用的示例，请打开 Start Page（开始页）或 **File**（文件）菜单中的对话框。根据需要，使用对话框中的 **Filter Options**（滤波器选项）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。

PSoC 3 重新进入支持

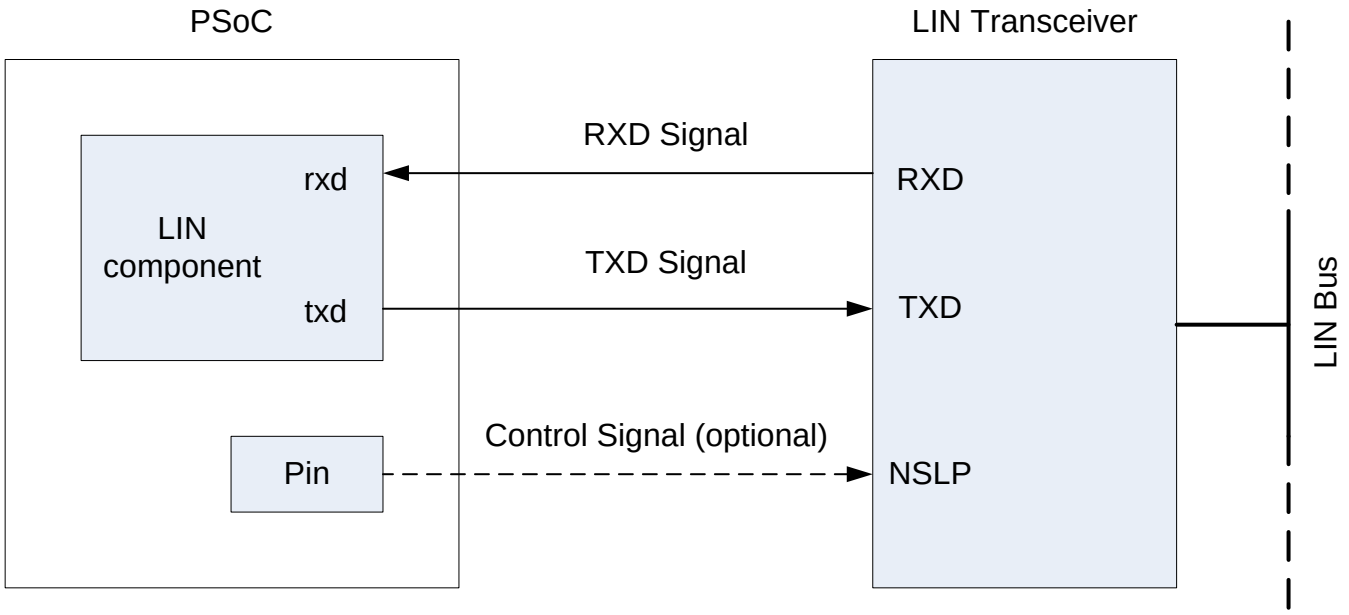
并发调用 CyIntClearPending() 函数，因为这是在 LIN 组件中从两个不同的中断执行调用。默认情况下，即使未重新进入，它也可以执行重新进入支持以在编译过程中去除“MULTIPLE CALL TO FUNCTION”警告。有关详细信息，请参见 PSoC Creator 帮助中的“PSoC 3 中的重入代码”这一主题。此外，组件示例项目已添加了重新进入支持。

PSoC 和 LIN 总线硬件接口

当 PSoC LIN 从器件节点直接与 LIN 总线连接时，需要 LIN 物理层收发器器件。在此情况下，LIN 组件的 TxD 引脚连接至收发器的 TXD 引脚，而 RxD 引脚连接至收发器的 RXD 引脚。需要 LIN 收发器器件，因为 PSoC 电子信号电平与 LIN 总线上的电子信号不兼容。

某些 LIN 收发器器件还具有“启用”或“睡眠”输入信号，用来控制器件的操作状态。LIN 组件不提供此控制信号。而是使用引脚，当需要此信号时，用其将所需信号输出到 LIN 收发器器件。

图 17. PSoC 与 LIN 总线之间的硬件接口



直流和交流电气特性

有关直流和交流电气特性的信息，请参见 *LIN 2.1 规范*的“LIN 物理层规范”这一章。

组件更改

版本 1.0 是 LIN 组件的首次发行版本。

版本	更改说明	更改/影响原因
1.0.a	对数据表进行了少量编辑和更新	

© 赛普拉斯半导体公司，2012。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。