LIN 从器件

1.30

特性



- 全面实现 LIN2.1 或 2.0 从器件节点
- 支持符合 SAE J2602-1 规范
- 自动同步波特率
- 全面实现一个诊断类 I 从器件节点
- 全面支持传输层
- 自动检测总线闲置
- 检测全部错误
- 对自动配置的服务进行处理
- 快速及易于配置的定制器
- 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 5LP 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。此组件包含在 LIN 总线上进行通信所需的硬件模块和允许应用代码轻松地与 LIN 总线通信进行交互的 API。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。提供组件定制器，由此可以轻松配置 LIN 从器件的所有参数。此组件最多支持一个接口（从逻辑通道到总线的）。

定义

在此数据手册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 规范的指定章节。

输入/输出连接

本节介绍了 LIN 从器件的输入和输出连接。

TXD — 输出

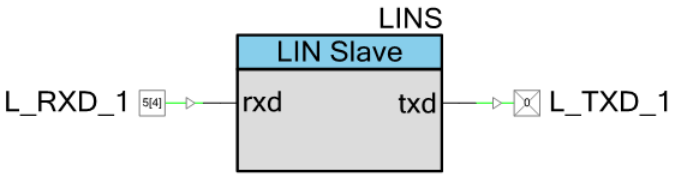
这是数字输出终端。此终端的信号是该LIN节点发送到LIN总线上的数据。

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。

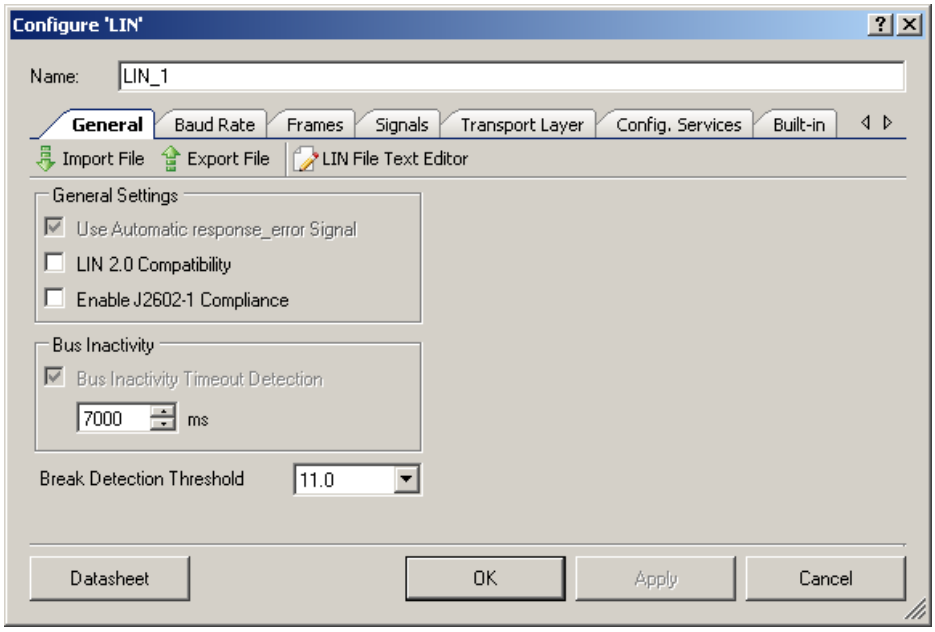
原理图宏信息

默认情况下，PSoC Creator组件目录包含LIN组件的原理图宏。此宏包含已经连接及配置的引脚组件。该原理图宏的默认组件配置，如下所示。



组件参数

将 LIN 从器件组件拖入在您的设计中，并双击它以打开 **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）

该选项卡上的此复选框用于设置自动错误信号的选择。此复选框始终被选中，因此在定制器的 **Signals**（信号）选项卡上自动添加了 1 位信号。此信号的默认名称为“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，成功发送到主设备后，组件还自动清除此信号。根据 LIN 2.1 规范，此信号向 LIN 主设备提供错误响应通知。

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定制器的其他区域。

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

SAE J2602-1 规范与 LIN 2.x 规范并行。它对 LIN 2.x 要求增加了几点限制。然而，此组件支持几个额外特性，从而使其与 J2602-1 兼容。此复选框的状态影响定制器的其他区域。

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

此选项控制总线空闲检测特性的可用性及其值。指定总线空闲特定时间后，相应的状态位将被设置。该位的值可以通过 l_ifc_ioctl()函数的 L_IOCTL_READ_STATUS 操作获得。有关详细信息，请参见功能说明的内容。

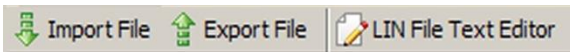
Break Detection Threshold（中断检测阀值）

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

General（通用）工具栏

General 选项卡顶侧有一个工具栏。此工具栏提供文件操作的访问。

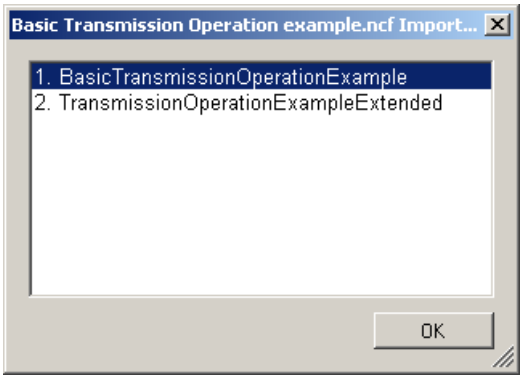
图 1. General （通用）工具栏



Import File（导入文件）：单击此按键，可以导入 LIN 描述文件（LDF）或节点性能文件（NCF）。导入的文件配置定制器设置，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。

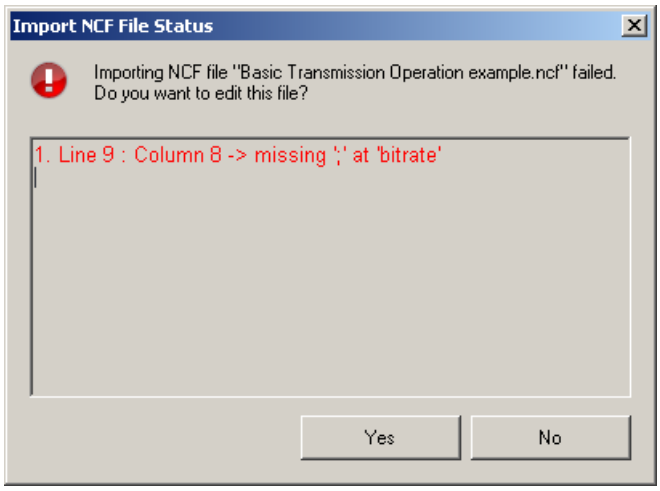
图 2. 要导入的 NCF 文件的可用节点列表



根据 LIN 节点性能语言规范（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。

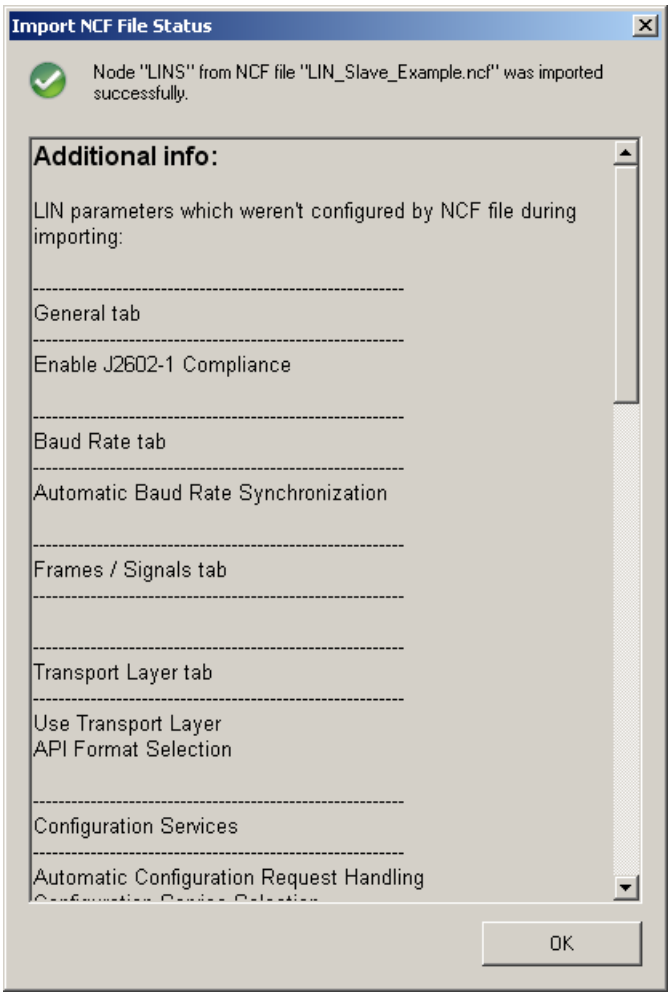
如果导入文件包含错误，则显示对话框窗口，类似于图 3。在此情况下有两个选项：使用 LIN 增强编辑器工具（更多信息，请参见 LIN **File Text Editor**）编辑导入的文件以更正错误，或通过单击 **No**（否）按键取消导入。

图 3. NCF 文件导入失败



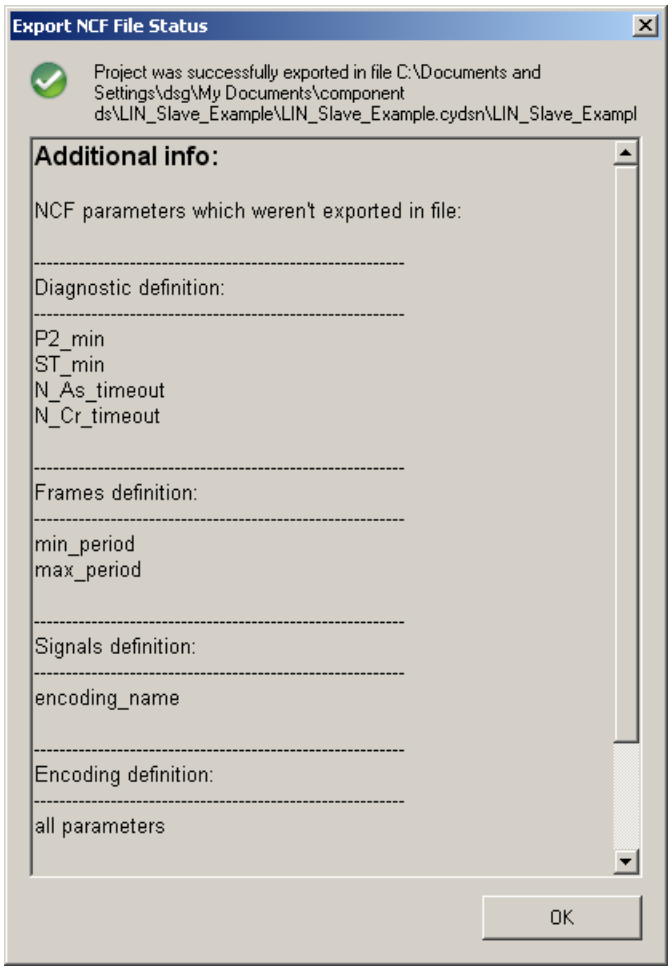
选中要导入的节点，并且将其导入到定制器完成后，会显示一个用于描述导入结果的对话框（请参见图 4）。导入结果包含 LIN 从器件组件参数，在导入过程中，这些参数不受任何影响。

图 4. NCF 文件导入信息



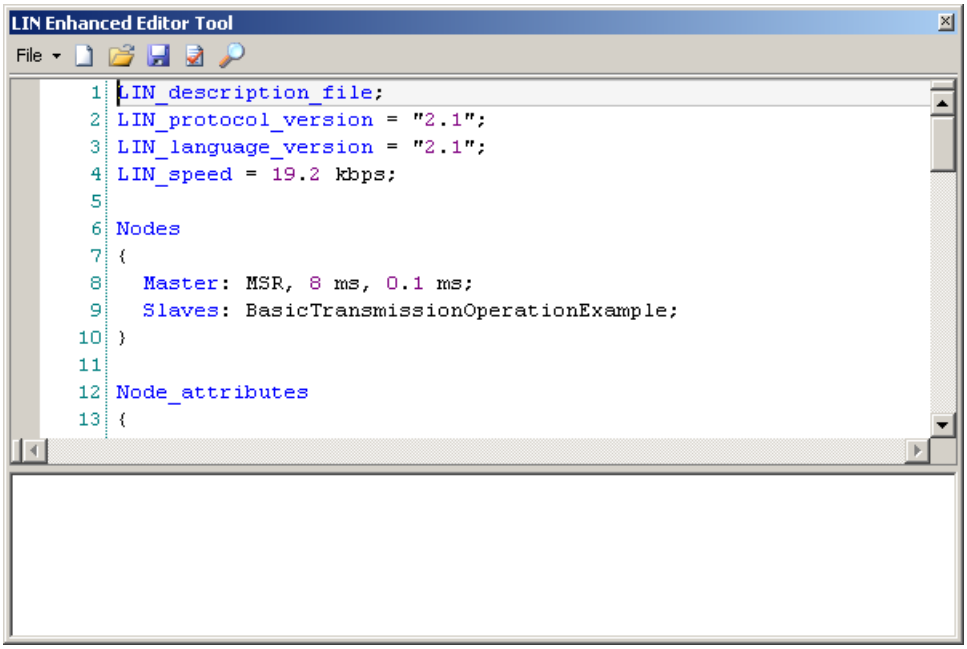
Export File（导出文件）：此工具可以将组件配置相关信息保存到节点性能文件（NCF）中。

图 5. NCF 文件导出信息



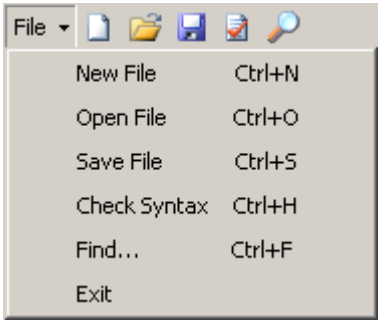
LIN File Text Editor（LIN 文件文本编辑器）：此工具用于消除、编辑和验证 NCF/LDF 文件的语法。根据 *LIN 节点性能语言规范*（修订版 2.1）来验证**.ncf* 文件的语法。根据 *LIN 配置语言规范*（修订版 2.1）来验证**.ldf* 文件的语法。

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）顶部有一个工具栏（请参见图 7）。

图 7. LIN 文件文本编辑器工具栏

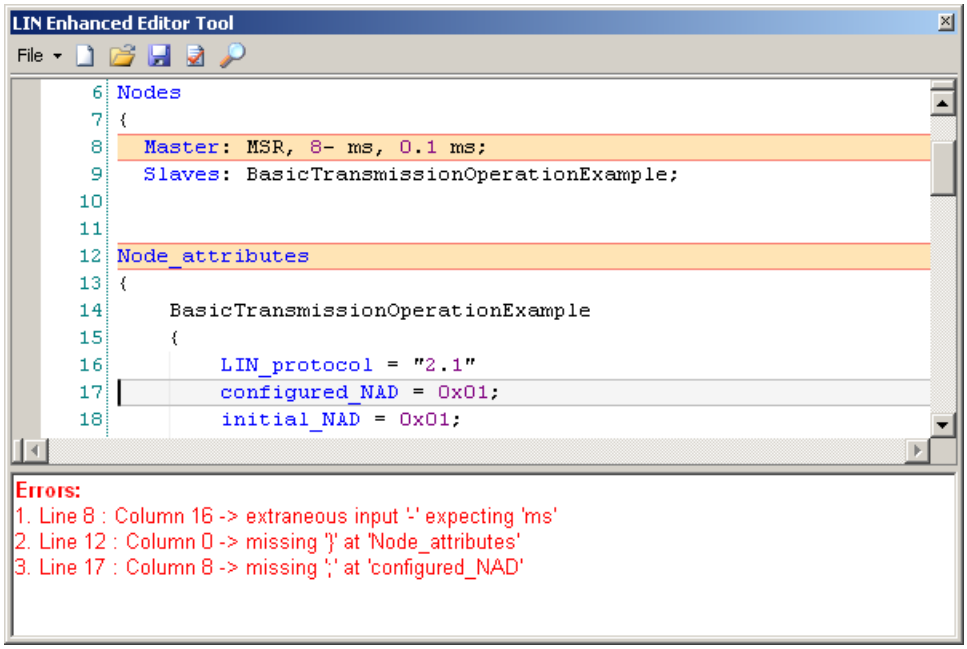


- **New File**（新建文件）：创建选定 LIN 文件类型的新文件。
- **Open File**（打开文件）：打开指定的现有 LIN 文件。
- **Save File**（保存文件）：将已创建的 LIN 文件保存到指定位置中。

- **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，错误将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。

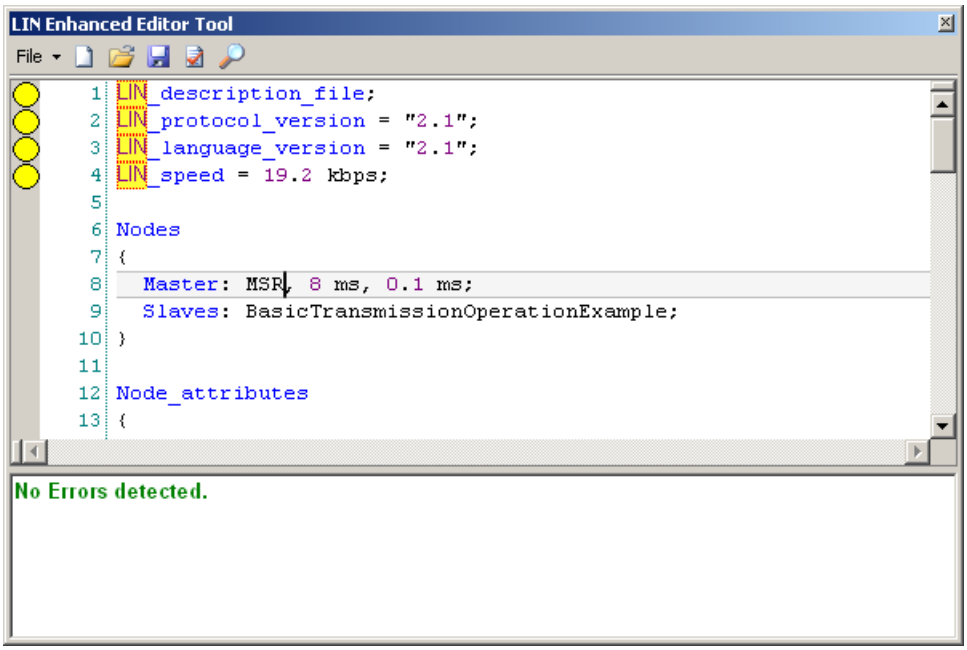
  在输出区域，双击错误行，可以导航至文件中包含错误的行。

图 8. LIN 文件语法检查



- **Find**（查找）：通过此工具，可以查找在 LIN 文件中搜索指定的术语。**Find Next**（查找下一个）按键定位下一个匹配结果。如果选中该工具的 **Mark Line**（标记行）复选框，则单击 **Find All** 按键后，包含必要术语的行标有黄色圆圈。单击 **Find All** 按键后，**Style found token** 复选框能够禁用黄色突出显示查找结果的功能，如图 9 所示。**Clear**（清除）按键清除所有突出显示的标记。

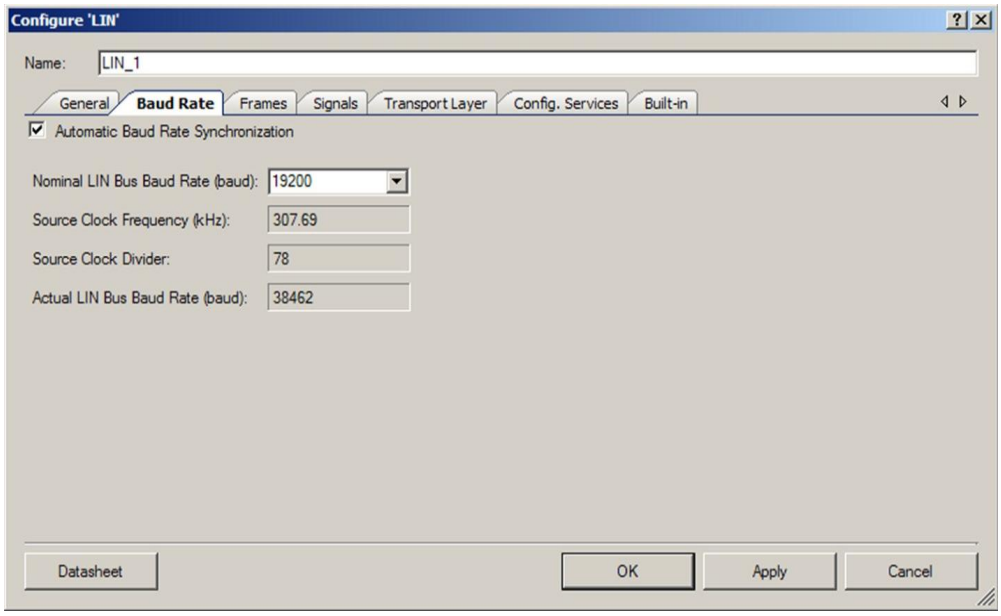
图 9. LIN 文件查找结果



所有工具在 LIN 增强编辑器工具的 **File**（文件）菜单（请参见图 6）或通过相应的工具栏可以获得。

Baud Rate（波特率）选项卡

图 10. Configure LIN 对话框 – Baud Rate 选项卡



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

通过此选项，可以使能或禁用自动同步波特率。默认情况下，此选项处于使能状态。

如果使能此选项，则组件测量确切的总线波特率，从各个 LIN 帧头的同步字节字段。

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。而是接收为 0x55 数据的同步字节的字段。

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 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）

必须输入运行此 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。定制器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。但是，您也可以在组合框中键入介于 1000 到 20000 之间的任意值。如果修改了 **Nominal LIN Bus Baud Rate（额定LIN 总线波特率）**，然后按 **Apply**（应用）按键获取 **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（时钟源频率）

这是时钟频率，过采样数为 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**时钟源频率**中指定的时钟频率。

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

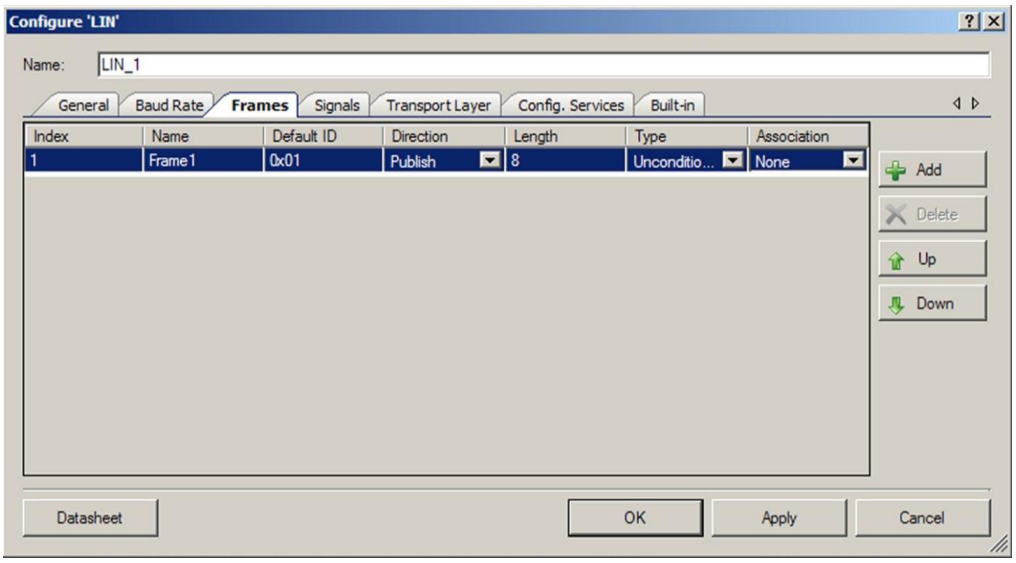
此处显示总线波特率的实际值。LIN 从器件将在此波特率下工作。可以修改 BusClk 的值，从而使**额定 LIN 总线波特率**等于**实际 LIN 总线波特率**。

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设备发送的 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 对话框 – Frames 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。不能直接修改这些编号。

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，这是在主控任何配置请求之前使用的帧 ID。注意：这些帧 ID 均是动态的。换言之，LIN 主设备可以在运行时重新配置帧 ID。在这些单元格中，必须输入 0x00 到 0x3B 之间的值。可以输入十六进制或十进制格式的值。

消息 ID 列不在图 11 中显示。这是因为该列通常是不可见的。只有在选中定制器的 **General** 选项卡中的 **LIN 2.0 Compatibility** 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，则组件不应该有任何包含消息 ID 0x000F 的帧。

在 **Direction**（方向）列中的字段定义发送帧数据的方向（相对于此从器件）。**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 到 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：**Unconditional**（无条件）和 **Event-Triggered**（事件触发）类型。当帧为订阅帧时，无法选择事件触发类型。在此情况下，无法修改此单元格。如果将此单元格从 **Event-Triggered**（事件触发）更改为 **Unconditional**（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。

注意：如果使能了 J2602-1 合规性，则无法访问 **Event-Triggered**（触发事件）帧。

在 **Association**（关联性）列中的字段用于关联无条件帧与事件触发帧。根据 LIN 规范，事件触发帧至少要有一个与其关联的无条件帧。因此，通过 **Association**（关联性）设置，可以选择与事件触发帧无关的任意无条件帧的名称。此设置的有效值是任意现有的无关联及无条件帧的名称。只有一个无条件帧可以与事件触发帧相互关联。因此，当其中一个单元格包含无条件帧的名称时，此无条件帧的名称无法用于其他任意行。如果事件触发帧与无条件帧相互关联，则两者的长度和方向必须相同。因此，事件触发帧的名称仅在适用这些条件的无条件帧行中显示。如果单击定制器的全局 **OK**（确定）按键，或通过单击另一个选项卡退出此选项卡，则定制器执行检查，以确保不再有与任意无条件帧关联的事件触发帧。

注意：总帧数不能超过 60。所有帧的总计大小限定为 256 个字节。

选项卡按键

在此选项卡上共有 4 个按键。

Add（添加）按键将新帧添加到表中。

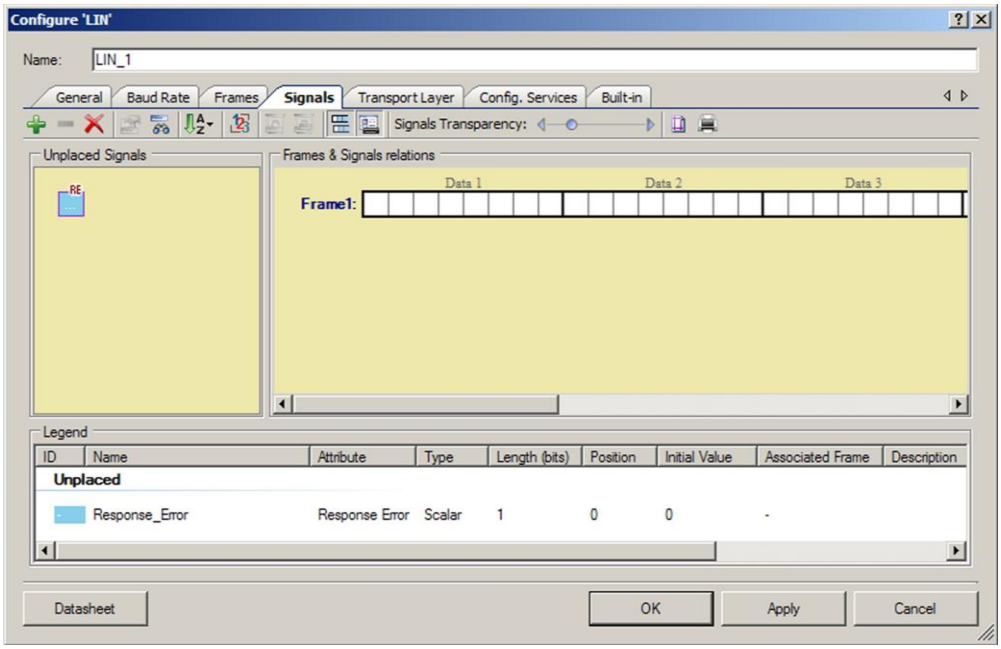
Delete（删除）按键从表中删除当前选中的帧。相应地更改索引编号字段。如果在此选项卡上删除一个帧，则任何打包到其中的信号（使用 **Signals**（信号）选项卡配置）均被移至 **Unplaced Signals**（未安置信号）区域（请参见 Signals（信号）选项卡中的 Sort Signals（排序信号）按键一节）。

您可以使用 **Up**（上箭头）和 **Down**（下箭头）按键给每个帧的索引编号重新排序。

Signals（信号）选项卡

在定制器中，此选项卡用于定义打包到 LIN 帧中的“信号”。

图 12. Configure LIN 对话框 – Signals 选项卡



Frames & Signals relations（帧和信号关联性）

此 **Signals** 选项卡的图形区域用来显示您使用定制器定义的帧与信号之间交互式图形。

Frame Graphics（帧图形）：一个帧图形代表在定制器 **Frames** 选项卡中定义的各个帧。

Signal Graphics（信号图形）：各个信号图形表示 LIN 从器件定义的相应信号。一个信号图形显示为白色条块（请参见图 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。翻转一个信号可以在工具提示中显示有关该信号的相关信息。

Unplaced Signals（未放置信号）

此图形域是添加信号后临时存储这些信号的区域，但尚未放置信号。信号可以在**未放置信号**区域与**帧和信号关联性**区域之间来回移动。

注意：如果在 **Frames** 选项卡上删除一个帧，则任何打包到其中的信号（使用 **Signals** 选项卡配置）均被移至**未放置信号**区域。

response_error

1 位的 Response_error 信号是自动添加到定制器的 **Signals** 选项卡中。Response_error 信号的名称是可以更改的，但不能从 **Signals**（信号）选项卡中删除它。

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 主设备报告状态信息。

有关此信号的其他信息，请参见 LIN 2.1 规范第 2.7.3 节“Reporting to the Cluster”（向集群报告）中的内容。

信号工具栏

这是 **Signals** 选项卡顶部的工具栏。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



Add/Delete（添加/删除）按键

该 **Add Signal**（添加信号）按键将信号添加到**未放置信号**区域。**Delete Signal**（删除信号）按除从组件中删除选择的信号。**Delete All Signals**（删除所有信号）按键删除全部现有信号。

Signal Properties（信号属性）按键

此控制打开选中信号的**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打开。

Find Signal（查找信号）按键

通过此按键，可以搜索某个信号。

Sort Signals（排序信号）按键

此按键用于对**未放置信号**区域中的所有信号进行排序。可以按名称、长度或类型来对信号进行排序。

Renumber Signals（重新编号信号）按键

此按键用于以升序方式重新编号信号索引值。

Move（移动）按键

未放置信号按键可以将所选信号从**帧和信号关联性**区域移至**未放置信号**区域。

该**未放置所有信号**按键可以将所有信号移至**未放置信号**区域。

Show/Hide Event-triggered frames（显示/隐藏事件触发帧）按键

通过此按键，可以显示或隐藏帧图形，其对应于**帧和信号关联性**区域中事件触发帧。

Show/Hide Legend（显示/隐藏图例）按键

通过此按键，可以显示或隐藏描述信号属性的图例区域。

Signals Transparency（信号透明度）滑条

此滑条用于设置信号图形的透明度。

Print（打印）按键

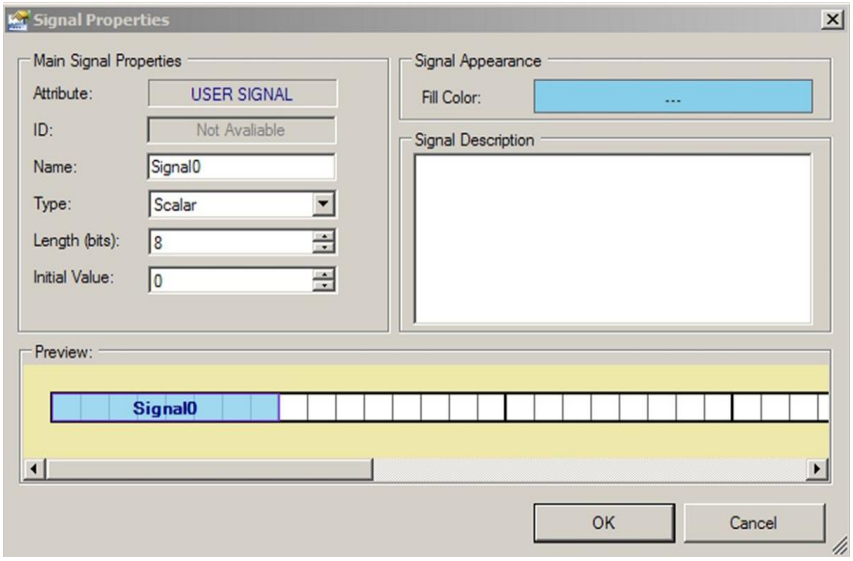
这些按键用于打印输出**帧和信号关联性**区域。

Signal Properties（信号属性）窗口

Adding Signals（添加信号）

在工具栏上有一个**添加信号**按键。此按键可以生成新的窗口，用来显示可配置的信号属性选项（请参见图 14）。完成配置各个属性后，添加一个新信号。本节介绍可在此窗口中配置的各种信号属性。

图 14. 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’表示信号索引编号。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 规范中定义，有两种信号类型。一个 **Scalar**（标量）信号长度为 1-16 位，而 **ByteArray** 信号长度为 1-8 个字节。

Length（长度）

此属性用于选择信号长度。Scaler 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。

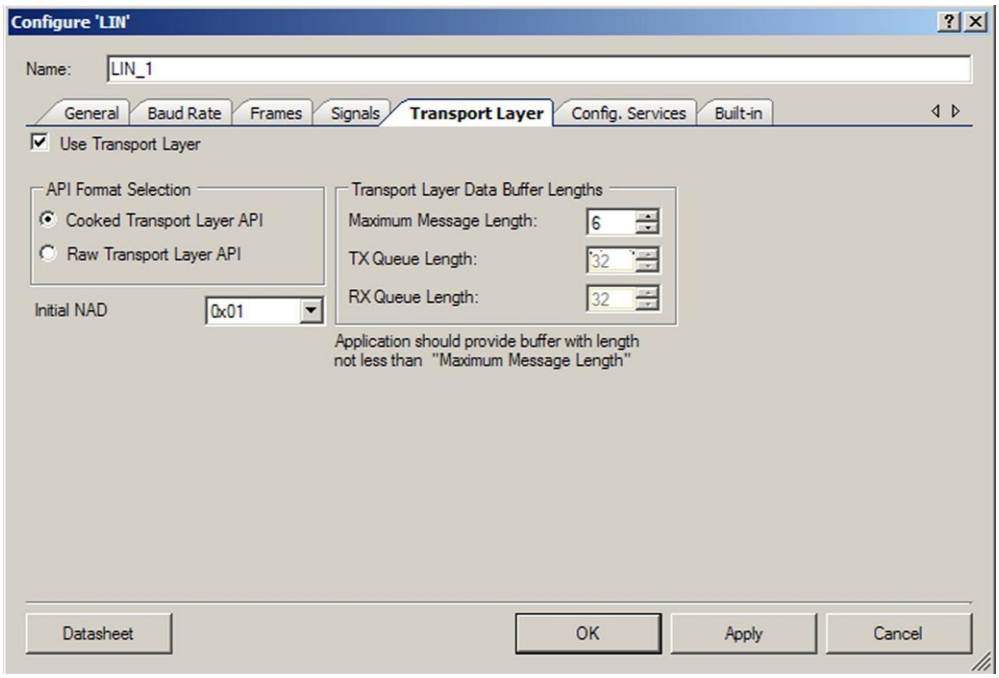
Preview（预览）

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

图 15 显示的是 Transport Layer 选项卡视图。

图 15. Configure LIN 对话框 – Transport Layer 选项卡



Use Transport Layer（使用传输层）

如果未选中 **Use Transport Layer** 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 函数的格式。有 **Cooked Transport Layer API**（Cooked 传输层 API）和 **Raw Transport Layer API**（Raw 传输层 API）等选项。通常，建议为 LIN 从器件应用选择 cooked 格式。cooked 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 函数。raw 格式用来发送和接收组织传输层消息的各个帧，并使用一个 API 函数调用每个帧。

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”指令保留 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，定制器限制您在此字段中输入 0x00、0x7E 或 0x7F。

如果选中“J2602-1 合规性”复选框，“传输层”选项卡上的初始 NAD 的值限定为 0x60-0x6F。默认值为 0x60。根据定制器的 **Frames** 选项卡上使用的帧数，将进一步限制初始值的范围。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

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

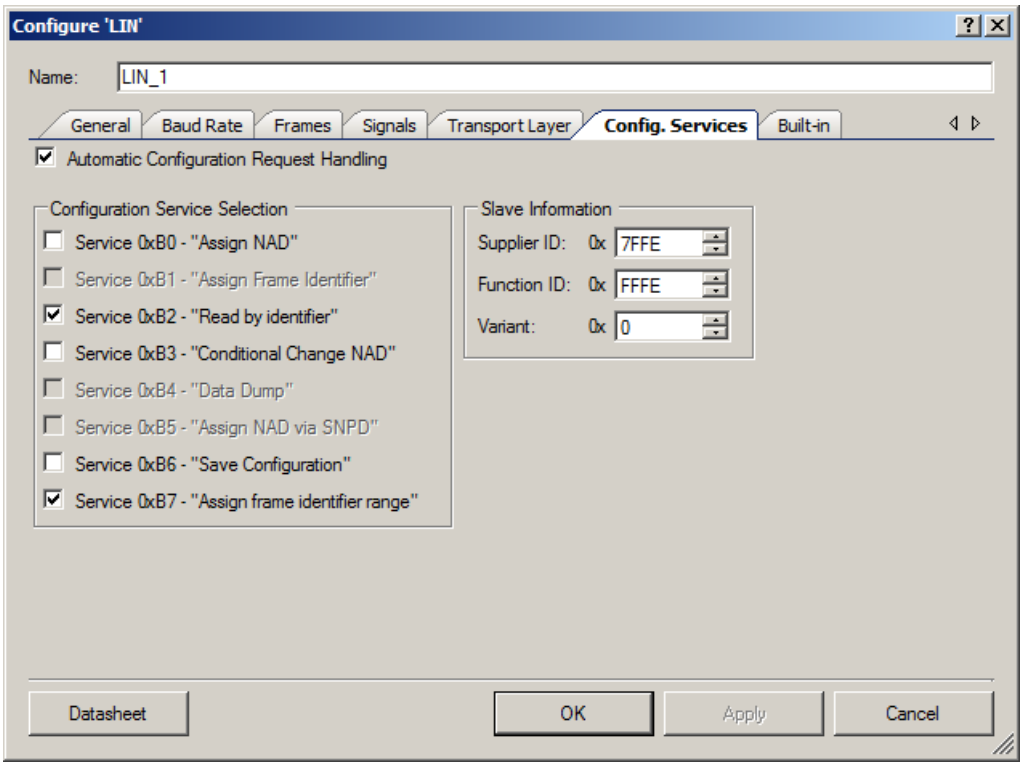
只有选中**原始传输层 API** 格式时，这些属性才能适用。使用原始 API 格式时，有一条消息“队列”，用来缓冲当前发送或接收的帧响应数据。如果从器件无法快速更新队列，那么队列长度将变得越来越长。如果从器件可以快速更新队列，那么队列可以变短，从而减少 RAM 使用量。该组件支持的队列长度为 8-2048，步长为 8 字节。每个队列的默认大小为 32 字节。

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 至 0xB7）。在 LIN 2.1 规范中的表 4.6 列出这些服务。此组件支持其中某些服务。您可以单独选择禁用或使能所支持的每个服务。在 LIN 2.1 规范中的第 4.2.5 节介绍这些配置服务请求。

图 16. Configure LIN 对话框 – Configuration Services 选项卡



Automatic Configuration Request Handling（自动处理配置请求）

此组件设计以便自动处理配置服务请求。换言之，您不必使用任何 API 或应用代码来处理来自主设备的这些服务请求。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有效状态。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。

在此选项卡上使能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。

Configuration Service Selection（配置服务选项）

在选项卡上列出各个支持的配置服务请求和复选框。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，此组件可能不需要此服务请求。PSoC 启动后，可以轻松配置 NAD 为所需的值，这可能无需 LIN 主设备请求 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。只有在定制器的 **General** 选项卡上选中 **LIN 2.0 Compatibility** 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。仅在 LIN 2.0 规范的第 2.5.1 节中介绍了该服务。在组件中使用它是为了向下兼容性。

- 服务 0xB2 — “通过标识符读取”

  根据 LIN 2.1 规范，此配置服务请求是强制的。此请求用来支持 LIN 主设备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，从器件对从主设备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，此组件不支持此服务。

- 服务 0xB5 — “通过 SNPD 分配 NAD”（目标复位）

  “通过 SNPD 分配 NAD”（0xB5）服务不受 LIN 2.1 规范的支持。但是，当在 **General** 选项卡上选中了 **Enable J2602-1 Compliance**（启用 J2602-1 合规性）复选框时，该服务（0xB5）仍能够作为由此组件支持的目标复位。

  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 操作中设置一个标志，以便让应用程序知道目标复位应该发生。有关详细信息，请参见 API 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）存储在非易失性存储器（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 函数返回的状态中设置配置标志。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 主设备可以更改从器件帧的易失性帧的 PID 值。

Slave Information （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 到 0x7FFE。功能 ID 也为 16 位，其有效范围为 0x0000 到 0xFFFE。变量为 8 位，其有效范围为 0x00 到 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，这些值作为从器件地址类型。

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 中的主设备时钟源。

放置

仅有一个组件实例可以放置在每个设计中。

应用编程接口

通过应用编程接口（API），您可以使用软件对组件进行配置。下表列出并介绍各个函数接口。以下各节将更加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。您可以将该实例重新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”。

内核 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位长度信号的当前值。
l_u8_rd()	读取并返回2到8位长度信号的当前值。
l_u16_rd()	读取并返回9到16位长度信号的当前值。
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到8位信号值。
l_u16_wr()	将信号当前值设置为9到16位信号值。
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()	将标志的当前值设置为零。

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。

用户提供的调用子组

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()	当主设备节点通过带有标识的标识符请求在用户定义区域中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raw和cooked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cooked或raw消息，则不会中止。

raw 传输层 API 函数子组

函数	说明
ld_put_raw()	在一个帧上将8个数据字节列队传输。
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raw帧传输函数的状态。

函数	说明
ld_raw_rx_status()	返回raw帧接收函数的状态。

Cooked 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打包成一个或多个诊断帧。这些帧和NAD一起传输到主设备节点。
ld_receive_message()	准备LIN诊断模块，以便用来接收一条消息并将其存储到根据数据指向的缓冲区中。调用时，length指定允许的最大长度。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()。

非 LIN 指定的 API（Non-LIN-Specified API）

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

内核 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： 无

参数： 无

返回值： 始终返回零。

其他影响： 无

内核 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”是在应用编程接口中定义的信号句柄。

l_bool_rd()

说明： 读取并返回1位长度信号的当前值。如果无效信号句柄传递到该函数，则不执行任何操作。

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

返回值： 返回信号的当前值。

其他影响： 无

l_u8_rd()

说明： 读取并返回信号的当前值。如果将无效信号句柄传递到该函数，则不执行任何操作。

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

返回值： 返回信号的当前值。

其他影响： 无

l_u16_rd()

说明： 读取并返回信号的当前值。如果将无效信号句柄传递到该函数，则不执行任何操作。

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

返回值： 返回信号的当前值。

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）参数的和不能不大于字节阵列长度。注意：当**Start**和**Count**的和大于字节阵列长度时，则读取意外**数据**。

如果将无效信号句柄传递到该函数，则不执行任何操作。

假设字节阵列的长度为 8个字节，则其编号为0到7。如果从用户选择阵列读取编号2-6中的字节，则要求**start**参数为2（跳过字节0和1），**count**参数为5。在这种情况下，字节2被写入到user_selected_array[0]中，所有连续字节以升序排列方式写入到user_selected_array中。

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

start：从第一个字节开始读取

count（计数）：要读取的字节数

data（数据）：指向阵列的指针，在此阵列中存储从信号中读取的数据

返回值： 无

其他影响： 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。如果将无效信号句柄传递到该函数，则不执行任何操作。

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

返回值： 无

其他影响： 无

l_u8_wr()

说明： 将值v写入信号。如果将无效信号句柄传递到该函数，则不执行任何操作。

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

返回值： 无

其他影响： 无

l_u16_wr()

说明： 将值v写入信号。如果将无效信号句柄传递到该函数，则不执行任何操作。

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。

返回值： 无

其他影响： 此函数不保证LIN主设备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，开始和计数参数的和不能大于字节阵列的长度。注意：当开始和计数参数的和大于信号字节阵列长度时，则影响到意外的存储器区域。

如果将无效信号句柄传递到该函数，则不执行任何操作。

假设字节阵列长度为8个字节，则编号为0到7。写入此阵列中的字节3和字节4时，要求“start”参数为3（跳过字节0、1和2），并“count”参数为2。在此情况下，将字节3写入到user_selected_array[0]中，并将字节4写入user_selected_array[1]中。

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

start：要写入到的第一个字节

count：要写入的字节数

data：指向阵列的指针，在此阵列中取传输到LIN主设备的数据

返回值： 无

其他影响： 此函数不保证LIN主设备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。

内核 API 函数：通知

通知标志用于将应用程序与 LIN 核心同步。这些标志由 LIN 核心自动设置，并且仅可以由应用程序进行检测或清除。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）或一个帧通信。成功发送或接收相应信号或帧时，一个标记将由此组件设置。

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

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。

false：该标记已被清除；

true：该标记尚未被清除。

其他影响： 无

l_flg_clr()

说明： 清除通过名称“fff”指定的标记。测试此子程序后（l_flg_tst() API之后），它应用于清除标记。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

返回值： 无

其他影响： 无

接口管理函数

这些调用可用来管理特定接口（去到总线的逻辑通道）。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从器件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：接口句柄的名称

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。

其他影响： 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，直至唤醒信号在LIN总线上被传送为止。CyDelayUs()函数作为时序源使用。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：接口句柄的名称

返回值： 无

其他影响： 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，请参见组件参数一节的内容。

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参数中定义的操作所适用的接口句柄名称

op：用于指定操作的参数

pv：指向指定操作的可选参数集的指针，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。在表中列出现有参数数量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	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	无	器件准备进入低功耗模式
L_IOCTL_WAKEUP	0x03u	无	唤醒后恢复组件状态
L_IOCTL_SYNC_COUNTS	0x04u	无	返回同步字段定时器计数的当前数
L_IOCTL_SET_SERIAL_NUMBER	0x05u	l_u8*	更新指针为序号

返回值： 没有为所选操作返回的错误代码值。因此，您必须确保传递到函数的值是正确无误的。

L_IOCTL_READ_STATUS操作

此字节中的首位是一个标记，其表示在过去的特定时间后总线上没有信号（使能**总线闲置超时检测**选项时可用该标记）。如果过去的时间超过某个阀值，则设置此标记。调用此API可以清除返回后的所有状态位。第二位是一个标记，其表示已接收到一个目标复位服务请求(0xB5)（当使能J2602-1合规性时）。

符号名称	值	说明
LIN_IOCTL_STS_BUS_INACTIVITY	0x0001u	在总线上未检测到信号持续一定时间
LIN_IOCTL_STS_TARGET_RESET	0x0002u	已收到目标复位服务请求（0xB5）

返回值：（续）

L_IOCTL_SET_BAUD_RATE操作

如果操作成功，则返回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，则返回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，则返回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，则返回0；如果传递给函数的操作参数无效，则返回1。

其他影响： 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：接口句柄的名称

返回值： 无

其他影响： 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：接口句柄的名称

返回值： 无

其他影响： 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：接口句柄的名称

返回值： 无

其他影响： 无

l_ifc_read_status()

说明： 此函数返回上一个通信的状态。有关LIN从器件状态字中各个状态信息字段的详细信息，请参考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

返回值： 该调用返回状态字（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	总线活动	转到睡眠	溢出	成功传输	响应中出错

状态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后清除状态字。

其他影响： 无

用户提供的调用

l_sys_irq_disable()

说明： 此函数禁用组件的所有中断。它返回包含中断掩码位的状态掩码。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。如果此组件中断被禁用的时间过长，LIN通信可能失败。

此子程序应由应用程序提供。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： 无

动态原型： l_irqmask l_sys_irq_disable(void)

参数：

返回值： 返回禁用的数字模块中的寄存器掩码。

其他影响： 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： 无

动态原型： void l_sys_irq_restore(l_irqmask previous)

参数： previous：中断掩码用于定义使能数字模块中的哪些中断。

返回值： 无

其他影响： 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一串字节。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： 无

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

参数： iii：接口句柄的名称；

pData：读取配置数据的阵列

length：配置数据字节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。

返回值： 该函数返回下表中列出的值。

符号名称	说明
LD_READ_OK	如果配置数据读取成功，则返回。
LD_LENGTH_TOO_SHORT	当长度指针参数指向的值小于配置数据的实际长度时，则返回。

其他影响： 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。运行时，这可以用于修改NAD和PID值。通常，仅在启动后或主设备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，主设备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： 无

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

参数： iii：接口句柄名称

pData：适用于从器件节点的配置数据阵列

length：配置数据字节的大小

返回值： 该函数返回下表中列出的值。

符号名称	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

其他影响： 无

ld_read_by_id_callout()

说明： 当主设备节点传输根据标识符的读取请求和用户定义区域中的标识符时，则使用此调出。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。

注意：此函数内部没有执行部分。根据需要的功能来创建此函数的执行部分， 并覆盖此函数的LD_NEGATIVE_RESPONSE的默认返回值。

静态原型： 无

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

参数： iii：接口句柄名称

id：在用户定义区域中的标识符 (32到63)，根据标识符配置请求的读取

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

返回值： 该函数返回下表中列出的值。

符号名称	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。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。若设置，则表示为服务提供响应。该响应将通过FrameData参数指向指针。

其他影响： 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 函数具有两种不同格式。有一种 cooked 格式和一种 raw 格式。此组件仅支持使用一种传输层 API 函数格式。在组件定制器的 **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 LIN 从器件组件定制器的 **Transport Layer** 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，也必须调用此API。如果正在进行的诊断帧在总线上传输cooked或raw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： 无

动态原型： void ld_init(l_ifc_handle iii)

参数： iii：接口句柄名称

返回值： 无

其他影响： 无

Raw 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。它实质上将用户应用阵列上的某些数据复制到帧缓冲阵列上。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： 无

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

参数： iii：接口句柄名称

ld_data：要发送数据字节的阵列。

返回值： 无

其他影响： 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。它实质上将帧缓冲区阵列上的某些数据复制到用户应用阵列上。此函数用于一次接收一个完整的传输层消息帧。因此，多帧传输层消息需要多次调用此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： 无

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

参数： iii：接口句柄名称

ld_data：将复制最早接收的诊断帧数据的阵列

返回值： 无

其他影响： 无

ld_raw_tx_status()

说明： 使用Raw API时，此调用返回上次执行的总线帧传输的状态。

静态原型： 无

动态原型： l_u8 ld_raw_tx_status(l_ifc_handle iii)

参数： iii：接口句柄名称

返回值：

符号名称	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。

其他影响： 无

ld_raw_rx_status()

说明： 使用Raw API时，此调用返回上次执行的总线帧接收的状态。

静态原型： 无

动态原型： l_u8 ld_raw_rx_status(l_ifc_handle iii)

参数： iii：接口句柄名称。

返回值：

符号名称	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。

其他影响： 无

Cooked 传输层 API 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，多帧传输层消息仅需要调用一次此API函数。长度值必须介于6到4095字节之间。

如果消息正在进程中，则此调用不返回任何操作。

静态原型： 无

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

参数： iii：接口句柄名称

length：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

ld_data：要发送数据的阵列。RSID的值是数据区域中第一个字节

返回值： 无

其他影响： 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。它负责接收多个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，多帧传输层消息仅需要调用一次此API函数。长度值必须介于6到4095字节之间。

静态原型： 无

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

参数： iii：接口句柄名称

length：要接收数据字节数的大小

nad：接收数据的从器件节点的地址

ld_ata：要接收数据的阵列。SID的值是数据区域中第一个字节。

返回值： 无

其他影响： 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() return 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送的状态。

静态原型： 无

动态原型： l_u8 ld_tx_status(l_ifc_handle iii)

参数： iii：接口句柄名称。

返回值： 可以返回下列值。

符号名称	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()）。初始化传输层后也将返回该值。
LD_FAILED	传输将在出错时结束。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	因为发生N_As超时，所以传输失败，以及当前消息传输将中止。请参见LIN 2.1规范中第3.2.5节。

其他影响： 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： 无

动态原型： l_u8 ld_rx_status(l_ifc_handle iii)

参数： Iii：接口句柄名称

返回值： 可以返回下列值：

符号名称	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。初始化传输层后也将返回该值。
LD_FAILED	接收将在出错时结束。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	因为发生N_Cr超时，所以接收失败，以及当前消息接收将中止。请参见LIN 2.1规范中第3.2.5节。
LD_WRONG_SN	接收失败，因为序列号异常。

其他影响： 无

非LIN指定API

LIN_Start()

说明： 启动组件操作。不需要此函数。

静态原型： 无

动态原型： l_bool LIN_Start()

参数： 无

返回值： Zero：初始化成功。
Nonzero：初始化失败。

其他影响： 无

LIN_Stop()

说明： 停止组件操作。不需要此函数。

静态原型： 无

动态原型： l_bool LIN_Stop()

参数： 无

返回值： 无

其他影响： 无

MISRA 合□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

本节介绍了有关组件特定偏差的信息。*系统参考指南*的“MISRA 合规性”章节中介绍项目偏差以及有关 MISRA 合规性验证环境的信息。

LIN 从器件组件具有以下特定偏差：

MISRA-C:2004 规则	规则类别（必须(R)/建议(A)）	规则说明	偏差描述
1.1	R	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。
8.7	R	如果仅从单个函数中访问对象，则应在模块范围内定义这些对象。这样会使对象和变量的范围最小。	发生这种错误，是因为内部变量中的某些配置仅适用于一个函数。 LIN_LinSlaveConfig, LIN_prevPci, messageIdTable

MISRA-C:2004 规则	规则类别（必须(R)/建议(A)）	规则说明	偏差描述
11.4	A	不同对象指针类型间不能实现转换。	LIN 2.1规范中第7.2.5.4节定义了l_ifc_ioctl()，包括以下原型：l_u16 l_ifc_ioctl （l_ifc_handle iii、l_ioctl_op op和void* pv）。 根据运算，可以将“pv”参数转换为指向“unsigned char”（无符号字符）或“unsigned short (l_u16)”（无符号short型(l_u16)）的指针。
13.7	R	不允许结果不变的布尔运算。	根据组件的设置，该组件可以进行条件检查，其结果是不变的。例如，如果执行ld_read_by_id_callout()函数，除非用户覆盖LD_NEGATIVE_RESPONSE，否则此函数将始终返回它。然而，该组件在它的源代码中对所有三个可能返回值进行条件检查。
14.1	R	没有不可访问的代码。这些代码是指在任何情况下都不会被执行的代码。	这与上面的13.7相同。根据组件的设置，该组件可以进行条件检查，其结果是不变的。它会导致不可访问的代码。
16.7	A	如果没有使用指针进行更改寻址的对象，则应将函数原型中的指针参数声明为指向常量的指针。	LIN 2.1规范中第7.2.5.4节定义l_ifc_ioctl()，包括以下原型：l_u16 l_ifc_ioctl （l_ifc_handle iii、l_ioctl_op op和void* pv）。 在某些情况下，根据配置，可能无法修改“pv”参数。
17.4	R	阵列索引是唯一允许的指针运算形式。但仍禁止未声明为阵列的指针进行递增。	为了符合LIN 2.1规范，该组件会定义将该指针作为参数使用的多个API函数。这些指针用于定义数据阵列，同时，通过阵列索引来访问数据。
19.7	A	函数应优先于类函数宏使用	以下宏用于提高性能： LIN_SWAP_U8_TO_U16()

固件源代□示例

PSoC Creator 在“Find Example Project”（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。要查看特定组件实例，请打开“Component Catalog”中的对话框或原理图中的组件示例。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》部分中主题为“查找示例项目”的内容。

PSoC 3 可重入支持

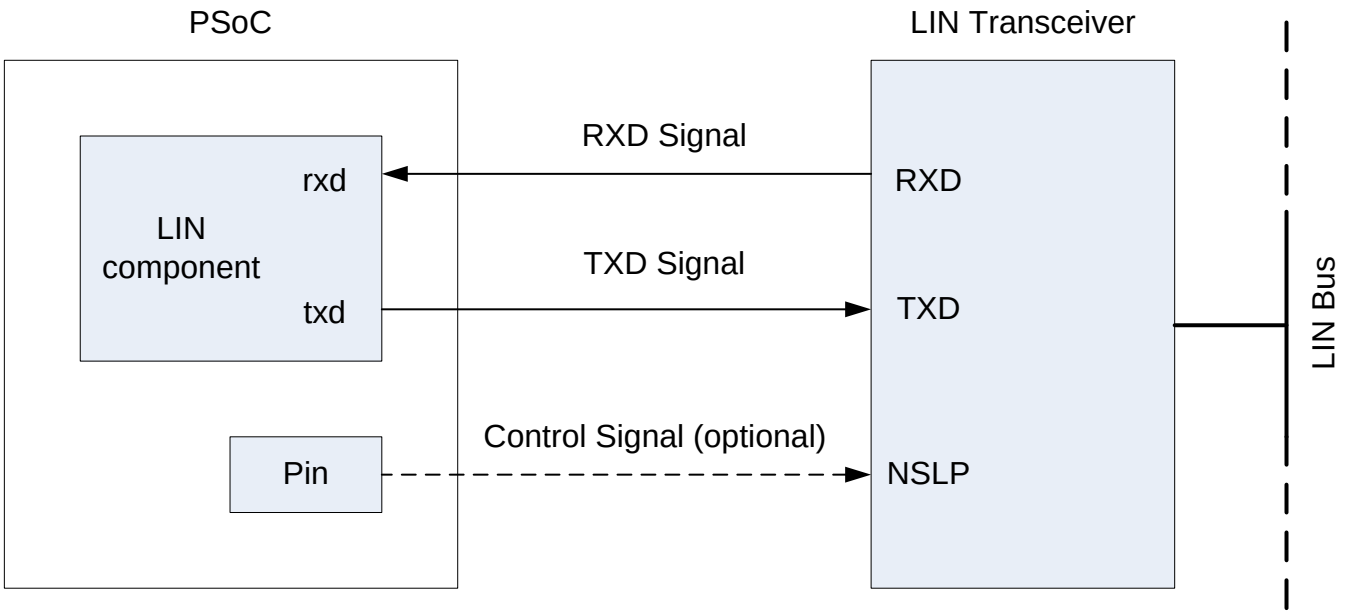
可以并发调用 CyIntClearPending()函数，因为它在 LIN 组件中从两个不同的中断执行调用。默认情况下，即使不是可重入函数，但可以将它设为可重入函数，以在编译过程中去除“MULTIPLE CALL TO FUNCTION”警告。有关详细信息，请参见 PSoC Creator 帮助中“PSoC 3 中的可重入代码”这一主题的内容。此外，组件示例项目已添加了可重入支持。

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。由于 PSoC 电子信号电平与 LIN 总线上的电子信号不兼容，所以需要 LIN 收发器器件。

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。而当需要此信号时，使用一个引脚将所需信号输出到 LIN 收发器。

图 17. PSoC 与 LIN 总线之间的硬件接口



资源

LIN 组件放置在整个 UDB 阵列中。该器件利用以下资源。

配置	资源类型					
	数据路径单元	宏单元	状态单元	控制单元	DMA通道	中断
LIN_Slave_Example工程	4	42	3	3	–	2

API 存□器占用情况

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，组件所用的存储器大小也不一样。下表提供了给定组件配置中的所有 API 的存储器使用情况。

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组件占用存储器的大小。

配置	PSoC 3（Keil_PK51）		PSoC 5LP（GCC）	
	闪存字节	SRAM 字节	闪存字节	SRAM 字节
LIN_Slave_Example工程	5710	174	4492	218

直流和交流的□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”章节中的内容。

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。

直流特性

参数	说明	最小值	典型值[1]	最大值	单位
I_{DD}	组件电流消耗	–	190	–	µA

1. 未包括设备IO和时钟分配的电流。这些值是在温度为25 °C时的值。

口件更改

本节列出了各版本的主要组件更改内容。

版本	更改说明	更改原因/影响
1.30.a	对数据手册进行了少量编辑。	
1.30	从组件中删除了对PSoC 5系列器件的支持。	
	下面的变量被声明为“static”（静态）： LIN_Slave_FindPidIndex(); LIN_Slave_EndFrame(); LIN_Slave_SetAssociatedFlags(); LIN_Slave_GetEtFlagValue(); LIN_Slave_ClearEtFlagValue(); LIN_Slave_ProcessMrf(); LIN_Slave_LinProductId(); LIN_Slave_MessageId().	MISRA 相关内容发生了更改。这些函数仅旨在用于本内部组件。
	45个全局变量被声明为“static”（静态）	MISRA 相关内容发生了更改。这些变量旨在用于本内部组件。
	更新了“MISRA合规性”章节。	此组件进行MISRA合规性验证。
	更新了“API存储器使用情况”章节。	添加了新的API存储器使用数据。
1.20	已添加了MISRA合规性章节。	此组件未进行MISRA合规性验证。
	将LIN从器件的时钟和中断组件更新到最新版本	
1.10	更新了器件特性数据。	
	添加了PSoC 5LP支持。	
	向“.cyre”文件中包括的所有组件API添加了CYREENTRANT关键词。	并非所有API都是真正可重入的函数。组件API源文件中的注释指出了适用的函数。 对于采用了安全方式并且是不可重入的函数，则需要该项变更，这样可以消除编译器警告：通过标志或关键节防止同时调用。
	根据组件的配置，已修改了0xB5服务的说明，这样可以使服务的占用更加清晰。	
1.0.a	对数据手册进行了少量编辑和更新	

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，WICED，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。