



**LIN 规范包**  
版本 2.0  
2003年9月23日；P1

# **LIN规范包**

版本2.0

本规范是在“ASIS”的基础上形成的，不可以作为任何索赔的依据。

© LIN 协会 2003.

版权所有。未经授权的复印、演示、或对本文件其他方面的使用，都是违反法律和知识产权的。

LIN是一个注册商标®.

本文件的任何分发都登记在案。

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 修订记录

**LIN 规范包**  
版本 2.0  
2003年9月23日； P2

### 1. 修订记录

发行号	日期	备注
LIN 1.0	1999-07-01	LIN 规范的初始版本
LIN 1.1	2000-03-06	
LIN 1.2	2000-11-17	
LIN 1.3	2002-12-13	
LIN 2.0	2003-09-16	主要修订

联系方式: **H.-Chr. v. d. Wense**, 摩托罗拉有限公司, **Schatzbogen 7, D-81829 慕尼黑, 德国**  
电话: **+49 (89) 92103-882** E-Mail: **H.Wense@Motorola.com**



## LIN

**LIN 规范包**  
版本 2.0  
2003年9月23日; P3

## 2. LIN

LIN (Local Interconnect Network, 串行通信网络) 是专门为汽车开发的一种低成本串行通信网络。它是现有汽车多元化网络公文包的一个补充。LIN 将成为层级式机动车网络执行的一个能动性因素, 能够提高质量、降低车辆成本。标准化意味着将会减少目前市场上杂乱的低端多元化解决方案, 并降低机动车电子产品在开发、生产、服务及物流领域的费用。

### 2.1 范围

LIN 标准包括输送协议规范、传输媒介、开发工具之间的接口、以及用于软件编程的接口。从硬件和软件的角度来说, LIN 可以提高网络节点的互用性, 并可以提高可预见的 EMC 行为。

### 2.2 特点和可能性

LIN 是由一系列通信协议组成, 能够有效地支持分布式汽车应用领域内的机电一体化节点控制。

LIN 总线的主要属性如下:

- 拥有多从机概念的单主机
- 基于通用 UART/SCI 接口硬件的低成本硅的实现。UART/SCI 接口硬件相当于软件中的一个等同体, 或可以作为纯态机。
- 从机节点不需石英或陶瓷振荡器就能实现自同步
- 信号传播时间可预先计算出来的确定性信号传播,
- 实现低成本单线
- 传输速率最高可达 20Kb/s
- 基于应用交互的信号

本规范的目的是希望能够在标准范围内、在任意两个 LIN 操作之间获得兼容, 也就是说从应用接口、API 一直到物理层。

LIN提供了一套可以节约成本而且非常有效的总线通信。该通信系统不需要带宽和CAN多功能性。单接线驱动器/接收器的规范遵循ISO9141标准, 并对EMI的行为做了某些改进。

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



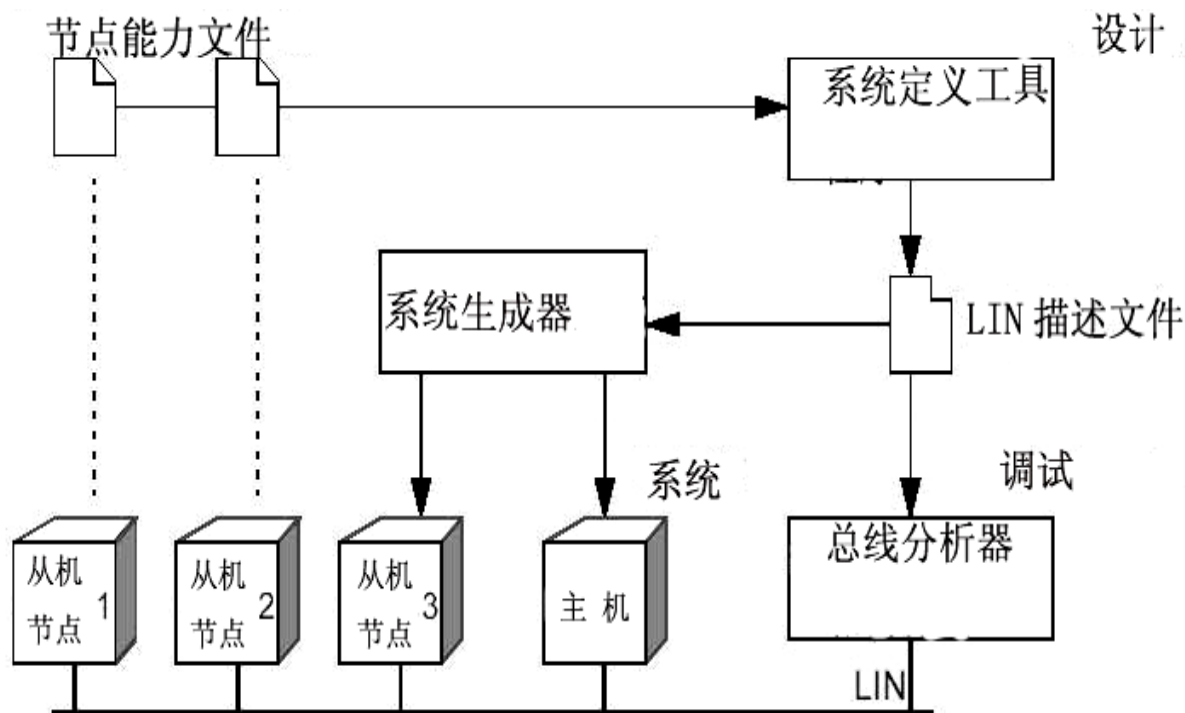
## 2.3 工作流程概念

LIN 的工作流程在设计时，考虑到了设计和开发工具的无缝链接操作。它可以提高 LIN 机群的开发速度和可靠性。

LIN 配置语言也考虑到了节点的安全转包生产，且不会破坏 LIN 的系统功能，也就是信息不兼容性或网络超载。LIN 配置语言是调试 LIN 仪表板的一个非常有用的工具，包括未成节点的仿真系统。

LIN 节点能力语言是 LIN2.0 的一个新特征。它可以为离架从机节点的规格提供标准化的句法。这不仅可以简化标准节点的获得，而且使机群的生成自动成为了可能。因此，机群上节点的即插即用功能将会成为现实。

下图便是工作流程的其中一个应用：



从机节点与主机节点连接，形成LIN机群。相应的节点能力文件被系统定义工具解析，从而在系统的定义过程中产生LIN描述文件（LDF）。LDF被系统产生器分解后，会自动地在指定节点（如上图中的主节点和从节点3）中产生与LIN相关的功能。此外，LIN的总线分析器/仿真器工具也可以利用LDF进行机群的调试。



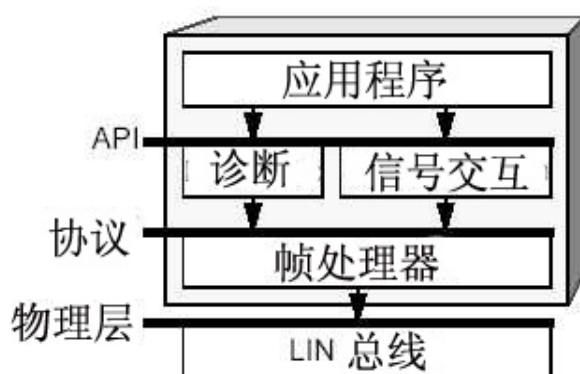
## LIN

**LIN 规范包**  
版本 2.0  
2003年9月23日; P5

### 2.4 节点概念

上述流程可以产生完整的 LIN 机群交互模块，而开发商只需要提供执行节点逻辑功能的应用程序即可。虽然，LIN 的许多规范采用的是具有诸多功能的软件设备，但也促进了其他方案的实现。在后一个例子中，LIN 的文件结构只能被视为一个描述模块：

LIN 的机群接口内的一个节点可以使用帧收发器到达物理总线。这些帧无法直接被应用程序获取；在两者之间需添加基于信号的交互层。作为一个补充，诊断接口便存在于应用程序和帧处理器之间。如下文：



### 2.5 操作概念

#### 2.5.1 主机和从机

LIN 机群由 1 个主机任务和几个从机任务组成。主机节点<sup>1</sup>同时包含主机任务和从机任务。其他的所有节点只包括一个从机任务。下文便是一个 LIN 机群的应用。该机群包含一个主机节点和两个从机节点的：



注1：一个节点可能会参与多个机群。如果节点有多个LIN总线接口，那么这里的“节点”只涉及到一个节点的单总线接口



主任务会指定某个具体的帧应在总线上输送的时间。从机任务提供数据，由每个帧来传送。

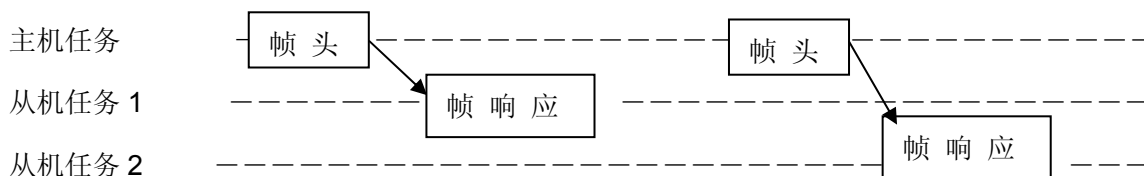
主机任务和从机任务都是帧处理器的一部分。见 2.4 章节。

### 2.5.2 帧

帧由一个帧头（由主机任务提供）和一个帧响应（由从机任务提供）组成。

帧头包括同步间隔场、同步场和标识符场三个部分；标识符场位置在同步场之后。标识符定义了帧的目的。而指派提供与标识符相关的帧响应的从机任务就会输送该目的，过程如下文所示。帧响应包括数据场和校验和场。

如果从机任务对标识符有关的数据感兴趣，那么它就可以接收这个响应，核实校验和，并使用这些被传送过来的数据。



上述过程会产生下列希望得到的特点：

- 系统弹性：节点可以被添加到 LIN 机群上，而无需改变其他从机节点上的硬件或软件。
- 信息路由：信息的内容由标识符<sup>2</sup>定义。
- 多点传送：无论有多少个节点，它们都可以同时接受信息，且在单个帧上进行操作。

### 2.5.3 数据输送

一个帧可以输送两种类型的数据：信号或诊断报文。

#### 信号：

信号指得是标量值或字节数组。这些标量值或字节数组打包后放入帧的数据场里面。对所有拥有相同标识符的帧来说，一个信号在数据场内的位置总是一样的。

注 2：与 CAN 相似。



LIN

**LIN 规范包**  
版本 2.0  
2003年9月23日；P7

#### 诊断报文：

诊断报文的输送是在具有两个保留标识符的帧里面完成的。数据场的判读取决于数据场本身以及通讯节点的状态。

#### **2.5.4 进度表**

（处于主机节点的）主机任务会根据进度表输送帧头。进度表具体规定了每个帧头的标识符以及一个帧与下一个帧之间的间隔。主机应用可以使用不同的进度表，并在它们之间进行挑选。

## **2.6 文件概述**

LIN 规范包包含下列规范：

- 《LIN 物理层规范》对物理层进行了描述，包括比特率、时钟偏差等等。
- 《LIN 协议规范》对 LIN 数据连接层进行了描述。
- 《LIN 诊断和配置规范》描述了为诊断报文和节点配置提供的服务。该服务可以置于数据连接层的上面。
- 《LIN 的 API 规范》描述了网络与应用程序之间的接口，包括诊断模块。
- 《LIN 的节点能力语言规范》对用于描述离架从机节点的模板进行讲解。离架从机节点可以与即插即用工具一起使用，从而自动创立了 LIN 描述文件。

## **2.7 历史和背景**

1999 年 7 月，LIN 版本 1.0 发布。该规格受到了当时一些汽车公司所使用的 VLITE 总线的重要影响。2000 年，LIN 标准被两次更新，从而于 2000 年 11 月产生了 LIN 1.2。2002 年 11 月，LIN 协会发布了 LIN 1.3 版本的标准。这些修正主要是针对物理层进行，其目的就是为了改进节点之间的兼容性。

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



**LIN**

**LIN 规范包**  
版本 2.0  
**2003年9月23日； P8**

全新的 LIN 2.0 版本向我们呈现了 LIN 规范从其鼻祖——LIN 1.3——开始的革命性发展。除了 2.7.1 章节所述的特例外，专门为 LIN 2.0 和 LIN 2.3 设计的节点将会在彼此之间进行通讯。

同时，LIN 2.0 规范被全部整改。原本不明朗的问题现在得到了解决，而需要修改的地方也被相应地修改了。

LIN 2.0 是 LIN 规范的调整，它反映了最新的发展趋势；特别是离架从机节点的使用。SAE J2602 任务组三年来对 LIN 规范的不断投入才使这次重大修订获得了成功。LIN 2.0 同时也包括了一些新的特点，主要是配置/诊断标准化的支持和特定的节点能力文件。这两个特点的目标在于简化离架从机节点的使用。

### **2.7.1 与 LIN 1.3 的兼容性**

LIN 2.0 是 LIN 1.3 的扩展集，是最新研发产品的推荐版本。

一个 LIN 2.0 主机节点能够处理包括 LIN 1.3 从机和/或 LIN 2.0 从机的机群。主机不会向 LIN 1.3 的从机要求获得任何新的 LIN 2.0 特点。

- 增强校验和，
- 重新配置和诊断，
- 波特率自动探测，
- “响应错误”状态监控，

LIN 2.0 从机节点无法与 LIN 1.3 主机节点操作（需对 LIN 1.3 主机节点进行配置）。

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com





LIN

**LIN 规范包**  
版本 2.0  
2003年9月23日；P9

### 2.7.2 LIN 1.3 与 LIN 2.0 之间的差异

下文所列的项目是 LIN 1.3 与 LIN 2.0 的不同点。文件更名和问题澄清化则不包括在本章节中。

- 支持字节数组，信号可以达到 8 个字节。
- 可以删除信号组（被字节数组取代）。
- 规范包括了比特率的自动探测功能。
- 对 LIN 1.3 的传统校验和进行了改进（包括受保护的标识符）。
- 对零星帧进行了定义。
- 网络管理的定时用秒来定义，而不是用比特数来定义。
- 简化状态管理，并直接向网络汇报。应用标准化。
- 增加了强制节点配置命令以及其他选择命令。
- 增加了诊断功能。
- 每个节点的“LIN 产品 ID”标准化。
- 用于微控制器节点的强制 API。节点是 C 语言编程的。
- API 发生变化，能够反映下列变化：字节数组、休眠、唤醒以及状态读取。
- 添加了诊断 API。
- 添加了节点能力语言规范。
- 更新了配置语言规范，以便能够反映所做的变化；添加了节点属性、节点结构、字节数组、零星帧以及配置。

## 2.8 参考文献

[1] 《道路车辆——诊断系统——数据信息交换要求》， 国际标准 ISO9141， 第一版， 1989

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



LIN

**LIN 规范包**  
版本 2.0  
2003年9月23日；P10

### 3. LIN 术语表

下列术语通常出现在《LIN 2.0 规范包》里面。本术语表内，每个词汇的描述都被简化；此外，还提供了一些主要文件和章节的参考文献。它们的缩写如下：

**PHY LIN Physical Layer Specification (LIN物理层规范)**

**PROT LIN Protocol Specification (LIN协议规范)**

**DIAG LIN Diagnostic and Configuration Specification (LIN诊断和配置规范)**

**CLS LIN Configuration Language Specification (LIN配置语言规范)**

**API LIN API Specification (LIN API规范)**

**NCL LIN Node Capability Language Specification (LIN节点能力语言规范)**

**主动节点** 机群中的节点像机群一样彼此之间进行通讯。[PROT 5]

**总线接口** 与机群中物理总线相连的节点逻辑产品（收发器、UART 等等）

**字节场** LIN总线上的每个节点可以在一个字节场里发送；字节场包括起始位和终止位。[PROT 2.1]

**校验和模型** 确认了两个校验和模型：**传统校验和**和**增强校验和**。增强校验和包括校验和内部受保护的标识符，而传统校验和则不包括。[PROT 2.1.5]

**传统校验和** 早期LIN版本所使用的校验和，用于帧的诊断：它只是将所有的数据字节进行统计。[PROT 2.1.5]

**机群** 机群指得是 LIN 总线以及所有的节点。


**数据** LIN的帧响应会输送1-8个字节。这些字节被统称为数据。[PROT 2.1.4]


**数据字节** 数据中的其中一个字节。[PROT 2.1.4]


**诊断帧** 主机请求帧和从机响应帧被称为诊断帧。[PROT 2.3.4]

**增强校验和** 新型校验和，其性能相对较好：除了数据字节外，它还包括总数中的受保护标识符。增强校验和用于LIN 2.0从机节点的通信。[PROT 2.1.5]

**联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国**  
**电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com**

	<b>LIN</b>	<b>LIN 规范包</b> 版本 2.0 <b>2003年9月23日； P11</b>
主机节点	主机节点不仅包括一个从机任务，还包括主机任务。主机任务负责将总线上的所有帧头发送出去，也就是说，它会控制总线的定时和进度表。	
事件触发帧	事件触发帧像一个“占位符”，允许多个从机节点提供响应。当信号不怎么变化时，这个帧非常有用。[PROT 2.3.2]	
帧	所有信息将会以帧的形式打包发送；帧包括帧头和帧响应。[PROT 2]	
帧槽	LIN总线上为某个特定帧的传送所保留的时间。与进度表上的某个条目相符合。PROT 2.2]	
休眠命令	这是一个特殊诊断帧，可以强迫从机节点进入休眠状态。[PROT 5.2] [API 2.5.4]	
帧头	帧的第一部分；由主机任务发送。[PROT 2.1]	
标识符	帧在0-63范围内的ID。[PROT 2.1.3]	
LIN描述文件	LDF是在系统定义过程中产生的，并在系统生成过程中或由调试工具解析。[CLS] [NCL 1.1]	
LIN 产品辨识	每个LIN节点的独特号码。[DIAG 2.4]	
主机请求帧	主机请求帧有标识符60，可以为主机节点所发行的诊断帧所用。[PROT 2.3.4] [DIAG]	
主机任务	主机任务负责将总线上所有的帧头发送出去，也就是说，它会控制总线的定时和进度表。[PROT 4.1]	
信息标识符	从机节点内的每个帧都有一个独特的16位信息号码。在字节配置过程中，这个号码会与受保护的标识符联合，然后在与节点的常规通讯中进行应用。[DIAG 2.5.1]	
NAD	节点诊断地址。诊断帧通过无线电输送，而NAD则指定有地址标注的从机节点。NAD既是实体地址也是逻辑地址。[DIAG 2.3.2]	
联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国 电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com		

	<b>LIN</b>	<b>LIN 规范包</b> 版本 2.0 <b>2003年9月23日； P12</b>
节点	广泛来说，节点便是 ECU（电子控制单元）。不过，单个 ECU 可以与多个 LIN 机群连接；对后者来说，“节点”这个词汇用“总线接口”来表示，更合适。	
节点能力文件	NCF文件用于描绘LIN总线上的从机节点。该文件可用于系统定义。[NCL 1.1]	
受保护的标识符	标识符（6 位）与两个奇偶位组成。[PROT 2.1.3]	
发布	每个信号（或无条件帧）都有一个配套的发布服务器；与认购不同的是，该节点是信息源。[PROT 2.1.4] [PROT 4.2]	
请求	主机节点将请求放在节点配置和诊断运输层的从机节点上。[DIAG 2.3.1] [DIAG 3.3.1]	
保留帧	保留帧有一个不可使用的标识符：63（0X3f）。[PROT 2.3.6]	
响应	(1) LIN帧包括帧头和响应[PROT 2.1]，也叫做帧响应。  (2) ISO请求的回复信息被称之为响应。[DIAG 2.3.1] [DIAG 3.3.1]。它同时也被成为诊断响应。	
进度表	进度表决定LIN总线上的通行状况。[PROT 3] [CLS 2.5] [API 2.4]	
从机节点	节点只包含从机任务，也就是说，它并不包括主机任务。	
从机响应帧	从机响应帧有标识符61，可以被从机节点所发布的诊断帧所使用。[PROT 2.3.4] [DIAG]	
从机任务	从机任务负责倾听所有总线上的帧头，并做出相应地反映，也就是说，它既不会发布一个帧响应，也不会认购（或忽视）一个帧响应。[PROT 4.2]	
休眠模式	机群中不发生任何通信行为。[PROT 5]	
信号	信号指得是 LIN 机群在使用载波帧时的一个被运输值。[PROT 1]	
联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国 电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com		

	LIN	LIN 规范包 版本 2.0 2003年9月23日； P13
载波帧	载运信号的帧应有一个范围在 0-59 的标识符（0X3b）。无条件帧、零星帧以及事件触发帧都是载波帧。[PROT 2.1.3]	
零星帧	零星帧与无条件帧很相似，是一个载波帧。不过，如果发布服务器对其中一个信号进行了更新，零星帧只能在自己的帧槽里面转移。[PROT2.3.3]	
认购	认购是发布的反义词，也就是接受信号（或接受一个载波帧）。[PROT 2.1.4] [PROT 4.2]	
系统定义	创立LIN描述文件的过程。[NCL 1.1.2]	
系统产生	机群中目标节点向LIN描述文件转变的过程。[NCL 1.1.1]	
无条件帧	载波帧总是在其指定的帧槽里面发送。[PROT 2.3.1]	
用户定义的帧	标识符为62的帧。其目的或用途不受LIN规范控制。[PROT 2.3.5]	



## 目录

**LIN 规范包**  
版本 2.0  
2003年9月23日; P14

### 目录

#### 规范包

1. 修订记录	1
2. LIN	2
2.1 范围	3
2.2 特性和可能性	3
2.3 工作流程概念	4
2.4 节点概念	5
2.5 操作概念	5
2.5.1 主机和从机	5
2.5.2 帧	6
2.5.3 数据运输	6
2.5.4 进度表	7
2.6 文件概述	7
2.7 历史和背景	7
2.7.1 与 LIN 1.3 的兼容性	8
2.7.2 LIN 1.3 和 LIN2.0 之间的差异	9
2.8 参考文献	9
3 LIN 术语表	10
目录	14

#### 协议规范

1. 信号管理	2
1.1 信号型号	2
1.2 信号一致性	2
1.3 信号包	2
2 帧传输	3
2.1 帧结构	3
2.1.1 间隔	4
2.1.2 同步字节	4
2.1.3 受保护的标识符	4
2.1.4 数据	5
2.1.5 校验和	6
2.2 帧槽	6
2.3 帧的型号	7
2.3.1 无条件帧	7
2.3.2 事件触发帧	7
2.3.4 零星帧	9
2.3.4 诊断帧	10

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 目录

**LIN 规范包**  
版本 2.0  
2003年9月23日; P15

2.6.1	Assign NAD(分配 NAD)	9
2.6.2	Conditional change NAD(有条件变化的 NAD)	9
2.6.3	数据转储	10
3.	诊断	11
3.1	基于信号的诊断	11
3.2	用户定义的诊断	11
3.3	诊断运输层	11
3.3.1	PDU 结构	12
3.3.2	被定义的要求	14
3.3.3	ISO 定时限制	14
3.3.4	程序表	15
4	参考文献	16

## 物理层规范

1.	振荡器偏差	2
2.	位定时要求和同步程序	3
2.1	位定时要求	3
2.2	同步程序	3
3.	总线驱动器/接收器	4
3.1	总配置	4
3.2	物理接口的电压定义	4
3.3	信号规范	5
3.4	直流电路参数	7
3.5	交流电路参数	9
3.6	线特性	11
3.7	ESD/EMI 的符合条件	12

## 应用程序接口规格

1.	介绍	2
1.1	操作概念	2
1.1.1	系统产生	2
1.1.2	API	2
2.	内核 API	4
2.1	驱动器和机群管理	4
2.1.1	l_sys_init	4
2.2	信号交互	4
2.2.1	信号类型	4
2.2.2	标量信号读取	5
2.2.3	标量信号写入	5
2.2.4	字节数组读取	5
2.2.5	字节数组写入	6



## 目录


### LIN 规范包

版本 2.0

2003年9月23日; P16

2.3	通知	6
2.3.1	l_flg_tst	6
2.3.2	l_flg_clr	7
2.4	进度管理	7
2.4.1	l_sch_tick	7
2.4.2	l_sch_set	8
2.5	接口管理	9
2.5.1	l_ifc_init	9
2.5.2	l_ifc_connect	9
2.5.3	l_ifc_disconnect	9
2.5.4	l_ifc_goto_sleep	10
2.5.5	l_ifc_wake_up	10
2.5.6	l_ifc_ioctl	11
2.5.7	l_ifc_rx	11
2.5.8	l_ifc_tx	12
2.5.9	l_ifc_aux	12
2.5.10	l_ifc_read_status	13
2.6	用户提供的标注	15
3.	节点配置	16
3.0.1	id_is_ready	16
3.0.2	id_check_response	17
3.0.3	id_assign_NAD	17
3.0.4	id_assign_frame_id	18
3.0.5	id_read_by_id	18
3.0.6	id_conditional_change_NAD	19
4.	诊断运输层	20
4.1	内部回调接口	20
4.1.1	id_put_raw	20
4.1.2	id_get_raw	21
4.1.3	id_raw_tx_status	21
4.1.4	id_raw_rx_status	22
4.2	加工好的 API	22
4.2.1	id_send_message	22
4.2.2	id_receive_message	23
4.2.3	id_tx_message	23
4.2.4	id_rx_message	24
5.	案例	25
5.1	LIN 内核 API 的使用	25
5.2	LIN描述文件	25



电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com		
	目录	<b>LIN 规范包</b> 版本 2.0 2003年9月23日; P17
2.3	通知	6
2.3.1	l_flg_tst	6
2.3.2	l_flg_clr	7
2.4	进度管理	7
2.4.1	l_sch_tick	7
2.4.2	l_sch_set	8
2.5	接口管理	9
2.5.1	l_ifc_init	9
2.5.2	l_ifc_connect	9
2.5.3	l_ifc_disconnect	9
2.5.4	l_ifc_goto_sleep	10
2.5.5	l_ifc_wake_up	10
2.5.6	l_ifc_ioctl	11
2.5.7	l_ifc_rx	11
2.5.8	l_ifc_tx	12
2.5.9	l_ifc_aux	12
2.5.10	l_ifc_read_status	13
2.6	用户提供的标注	15
3.	节点配置	16
3.0.1	id_is_ready	16
3.0.2	id_check_response	17
3.0.3	id_assign_NAD	17
3.0.4	id_assign_frame_id	18
3.0.5	id_read_by_id	18
3.0.6	id_conditional_change_NAD	19
4.	诊断运输层	20
4.1	内部回调接口	20
4.1.1	id_put_raw	20
4.1.2	id_get_raw	21
4.1.3	id_raw_tx_status	21
4.1.4	id_raw_rx_status	22
4.2	加工好的 API	22
4.2.1	id_send_message	22
4.2.2	id_receive_message	23
4.2.3	id_tx_message	23
4.2.4	id_rx_message	24
5.	案例	25
5.1	LIN 内核 API 的使用	25
5.2	LIN 描述文件	25
联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国 电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com		



## 目录

**LIN 规范包**  
版本 2.0  
2003年9月23日; P17

### 节点能力语言规范

1. 介绍	2
1.1 即插即用流程	2
1.1.1 系统产生	2
1.1.2 系统定义	3
1.1.3 调试	3
2. 节能力文件定义	4
2.1 全球定义	4
2.1.1 节点能力语言版本号的定义	4
2.2 节点定义	4
2.3. 一般定义	4
2.3.1 LIN 协议版本号的定义	5
2.3.2 LIN 产品识别	5
2.3.3 比特率	5
2.3.4 非网络参数	5
2.4 诊断定义	5
2.5 帧定义	6
2.5.1 帧的属性	7
2.5.2 信号的定義	7
2.5.3 信号编码类型的定义	8
2.6 状态管理	8
2.7 自由文本	9
3. 句法摘要	10
4. 样本文件	11

### 配置语言规范

1. 介绍	2
2. LIN 描述文件的定义	3
2.1 全球定义	3
2.1.1 LIN 协议版本号的定义	3
2.1.2 LIN 语言版本号的定义	3
2.1.3 LIN 速度的定义	3
2.2 节点定义	3
2.2.1 参与的节点	4
2.2.2 节点的属性	4
2.2.3 节点结构定义	5
2.3 信号定义	6
2.3.1 标准信号	6
2.3.2 诊断信号	6
2.3.3 信号组	7
2.4 帧定义	7

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 目录

**LIN 规范包**  
版本 2.0  
2003年9月23日; P19

2.4.1	动力帧的定义	7
2.4.2	无条件帧	7
2.4.3	零星帧	9
2.4.4	事件触发帧	9
2.4.5	诊断帧	10
2.5	进度表的定义	11
2.6	附加信息	13
2.6.1	信号编码类型定义	13
2.6.2	信号再现的定义	15
3.	句法摘要	16

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## **LIN 协议规范**

### **版本 2.0**

本规范是在“ASIS”的基础上形成的，不可以作为任何索赔的依据。

© LIN 协会 2003.

版权所有。未经授权的复印、演示、或对本文件其他方面的使用，都是违反法律和知识产权的。

LIN是一个注册商标®.

本文件的任何分发都登记在案。



## 信号管理

**LIN 协议规范**  
版本 2.0  
2003年9月23日；P2

### 1. 信号管理

信号是在帧的数据场里面输送的。多个信号可以被打包成一个帧，只要这些信号彼此之间不会相互重叠。

每个信号都有自己的生成器，即它总是由机群里面的同一个节点写入。0、1 或多个节点可以认购该信号。

#### 1.1 信号类型

信号有标量值和字节数组两种类型。

标量值的有 1-16 位，一个位的标量信号被称为布尔信号。2-16 位的标量信号被视为无符号（无正负）整数。除此之外的数据判读，即偏移和校准，均不在本范围内。

一个字节数组是由 1-8 个字节组成的。对 LIN 规范来说，任何对此类字节数组的判读均不在范围内。特别是当再现实体要比字节数组里的字节要大的时候，这对字节顺序更加适用。

#### 1.2 信号一致性

标量信号的写入或读取必须是原子操作的。也就是说，应用程序无法接受一个部分被更新的信号值。不过，信号与信号之间或一个字节数组内的字节之间无需保持一致性，

#### 1.3 信号打包

信号会最先通过LSB进行发送，最后才是MSB。对帧内部的标量信号包来说，唯一的原則就是：标量信号<sup>1</sup>可以穿过一个字节的最大界限。字节数组内的每个字节应形成一个单帧字节。该单帧字节的首位是编号最小的数据字节（2.1.4章节）

注 1：如果信号是按照字节排列在一起的，或信号没有穿过字节界限，那么信号的打包或解压行为在基于软件的字节内操作时更有效

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



2. 帧的转移

在LIN总线上进行转移的实体就是帧

帧的发送时间是每个字节发送时间的总和，再加上响应间隙时间和字节间隙时间。字节间隙时间是前一个字节终止位的末端到下一个字节起始位的末端之间的时间。这两个字节不能是负数。

2.1 帧结构

表2.1就是一个帧的结构。该帧是由一个间隔和4-8个字节场组成，这些字节场以数字标注。

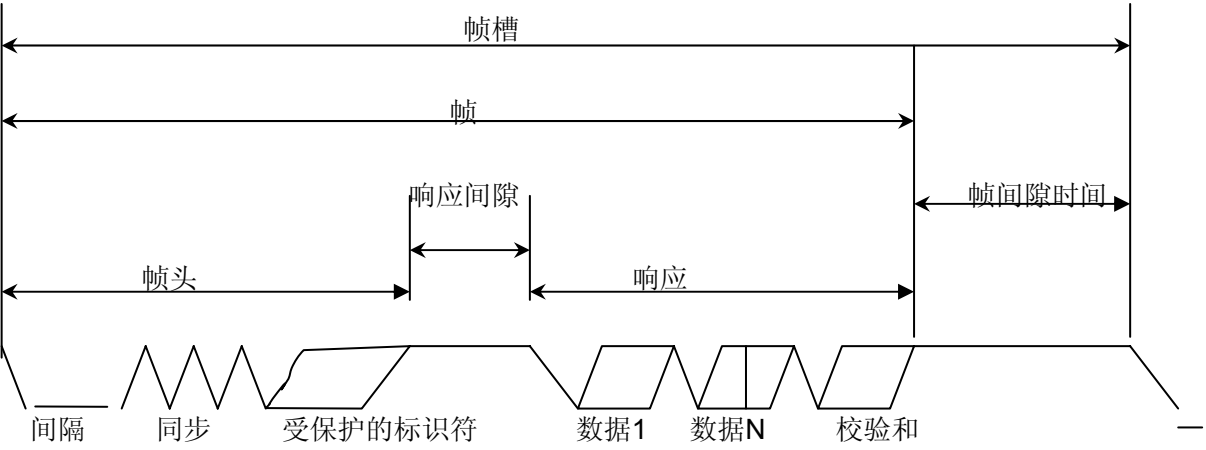


图2.1: LIN的帧结构

每个字节场<sup>2</sup>是作为字节串行被输送的，如图2.2所示。最先发送的数据是LSB，最后才是MSB。起始位是一个值为0的比特（显性），而终止位也是一个值为0的比特(隐性)。

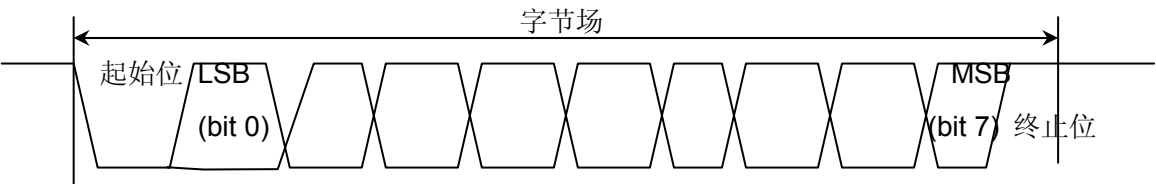


图2.2: 字节场的结构

注2: 不包括间隔字节场。见2.1.1章节



# 帧的转移

**LIN 协议规范**  
版本 2.0  
2003年9月23日；P4

## 2.1.1 间隔

间隔符号用来标识一个新帧的起始点。这是唯一一个不必符合表2.2的场。间隔总是由主机任务（位于主机节点）产生，是一个至少有13个比特的显性值，包括起始位、间隔定界符等等，见图2.3。间隔定界符的长度应至少为一个标定比特。

从机节点需使用标定比特数为11的间隔检测阈<sup>3</sup>。



图2.3：分隔场

## 2.1.2 同步字节

同步场是一个数据值为0 x 55的字节场，见图2.4。

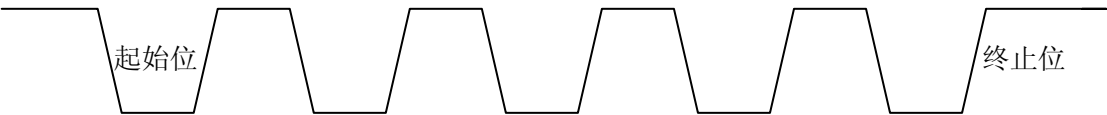


图2.4：同步字节场

从机任务要求能够探测到间隔/同步符号序列，即使这只是个字节场（假设字节场彼此之间是分离的<sup>4</sup>）。如果发现了字节场，间隔/同步符号序列的探测工作将会终止正在进行的帧转移<sup>5</sup>，而启动新帧的处理工作。

## 2.1.3 受保护的标识符

受保护的标识符包括2个子场、标识符以及标识符的奇偶校验。0-5比特是标识符，6-7比特是奇偶校验。

注3：有时钟偏差的从机节点比FTOL\_SYNCH要好。见LIN物理层。表1.2（典型的晶体或磁共鸣器）可以使用9.5比特的间隔检测阈

注4：用于探测间隔/同步序列的一个特性，即使间隔部分被数据字节叠加。该特性虽然很需要，但并不是必需的。

注 5：Reponse\_error 和响应错误需在节点处理帧的过程进行设置，见第 5 章节。



## 帧的转移

**LIN 协议规范**  
版本 2.0  
2003年9月23日； P5

### 标识符

标识符（ID）有6个比特，其值的范围是0-63。标识符可分为4类：

- 载波帧的值，其范围是0-59（0 x 3b）。
- 60 (0x3c) 和61 (0x3d) 可用来载运诊断数据。
- 62(0x3e)专门用于用户定义的扩展部分。
- 63(0x3f)专门用于以后协议的改进。

### 奇偶校验

奇偶校验是在标识符比特的基础上进行计算的，见方程（1）和（2）：

$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4 \quad (1)$$

$$P1 = \neg (ID1 \oplus ID3 \oplus ID4 \oplus ID5) \quad (2)$$

### 制图

比特（从ID0到ID5、P0和P1）的制图，见图2.5



图2.5：受保护标识符字节场的标识符和奇偶校验（图解）

#### 2.1.4 数据


帧可以携带1-8比特的数据。对拥有指定标识符的帧来说，其所包含的字节的数量应与发布服务器和所有认购器保持一致。数据比特是在比特场里面进行输送的，见图2.2

对那些超过1个字节的数据实体来说，LSB被包含在第一个被发送的字节里面，而实体MSB则被包含在最后一个被发送的字节里面（小头）。数据场用数据1、数据2、……、数据8标注，见图2.6。



图 2.6： 8 个数据字节帧里面的数据字节编号



	<h1>帧的转移</h1>	<b>LIN 协议规范</b> 版本 2.0 2003年9月23日；P6
<h2>2.1.5 校验和</h2> <p>帧的最后一个场是校验和。校验和场里面包含取向的8个比特和，<u>它们将转移<sup>6</sup>所有数据字节或所有数据字节和受保护的标识符</u>。仅根据数据字节进行计算的校验和被称为传统的校验和，它可以与LIN1.3从机进行通讯。</p> <p>根据数据字节和标识符进行计算的校验和被称为增强校验和，可以与LIN2.0从机进行通讯，见图2.2。</p> <p>主机节点可以控制进行传统或增强校验和的使用，并由帧标识符决定；传统校验和可以与LIN1.3从机进行通讯；增强校验和可以与LIN2.0从机进行通讯。</p> <p>值为60 (0x3c)到63 (0x3f)的标识符经常使用传统校验和。</p> <h2>2.2 帧槽</h2> <p>每个预定的帧会在总线上分配一个槽。槽的间隙时间必须足够长，以便能够在最糟的情况下载运帧。</p> <p>输送一个帧的标定值必须与发送的比特数量相配，也就是说无响应空间、无字节空间且无帧间隙空间。因此：</p> $T_{Header\_Nominal} = 34 * T_{Bit} \tag{3}$ $T_{Response\_Nominal} = 10 * (N_{Data} + 1) * T_{Bit} \tag{4}$ $T_{Frame\_Nominal} = T_{Header\_Nominal} + T_{Response\_Nominal} \tag{5}$ <p>方程中，Tbit是输送一个比特的标定值，见LIN物理层的定义。</p> <p>字节之间最大的空间是标定输送时间的40%。该时间段被帧头（主机任何）和帧响应（从机帧）隔开，于是产生：</p> $T_{Header\_Maximum} = 1.4 * T_{Header\_Nominal} \tag{6}$ $T_{Response\_Maximum} = 1.4 * T_{Response\_Nominal} \tag{7}$ $T_{Frame\_Maximum} = T_{Header\_Maximum} + T_{Response\_Maximum} \tag{8}$ <p>每个帧槽应大于或等于TFrame_Maximum</p> <p>注 6：运载的 8 个比特和等于所有值的和；如果总和大于或等于 256，那么该总和需减去 255（这并不等于数模-255 或数模-256）。</p>		
联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国		

电话: +49 (89) 92103-882 E-Mail: [H.Wense@Motorola.com](mailto:H.Wense@Motorola.com)



# 帧的转移

LIN 协议规范  
版本 2.0  
2003年9月23日； P7

注：

所有认购节点能够接收零耗量（zero-overhead）的帧，即长度为TFrame\_Nominal。  
THeader\_Maximum规定了间隔符号的最大长度。

## 2.3 帧的类型

帧的类型指得是那些能够有效输送帧的前提条件。只有帧类型中的一部分才可以用于特定用途，而且还受下列条件的约束。注意，并不是所有的节点或机群都要支持所有本章节指定的帧类型。

帧中所有未被使用/定义的比特均为隐性比特。

### 2.3.1 无条件帧

无条件帧总是携带信号，其标识符值的范围为0-59（0x3b）。  
无论什么时候，只要当分配在无条件帧上的帧槽被（主机任务）处理了，无条件帧的帧头才可以被输送。无条件帧（从机任务）的分布器应向帧头提供响应。无条件帧的所有认购器应接收帧，并使它可以被应用（前提是没有发现任何错误）。

图2.7便是3个无条件帧的序列

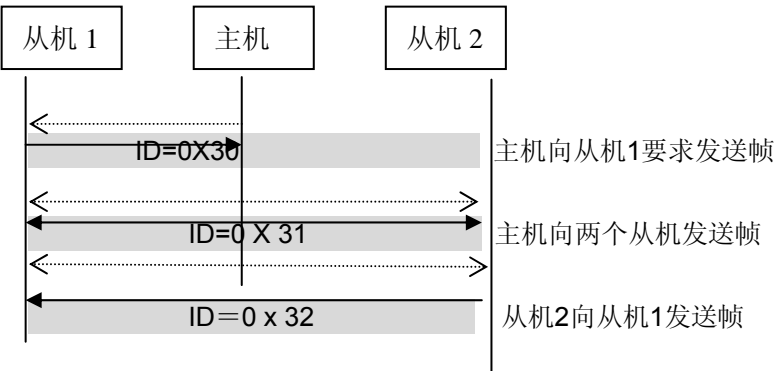


图2.7：三个无条件帧的转移。帧的转移活动通常是由主机启动的。它由一个单发布服务器和一个或多个认购器组成。

### 2.3.2 事件触发帧

事件触发帧的目的是增加LIN机群的灵敏性，从而不会向事件触发较少的多个从机节点分配过多的总线带宽。

事件触发帧会载运一个或多个无条件帧的数据场，且事件触发帧的标识符应在 0-59 (0x3b)的范围内。被载运的无条件帧的第一个数据字节应与其受保护的标识符的字节一样。也就是说，被载运的信号最多只能有 7 个字节。

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



# 帧的转移

**LIN 协议规范**  
版本 2.0  
**2003年9月23日； P8**

如果多个无条件帧与一个事件触发帧（这属正常情况）发生关联，那么它们的长度应该保持一致，且应使用相同的校验和模型（LIN1.3和LIN2.0帧的混合体是不被接受的）；更甚至，它们应该被不同的从机任务发布。

当分配到事件触发帧的帧槽被处理时，事件触发帧的帧头就可以被正常输送（输送条件见下文）。一旦帧所携带的其中一个信号被更新了，那么与无条件帧相关联的发布服务器只能向该帧帧头响应。

如果无任何从机向帧头响应，那么其余的帧槽将沉寂，而帧头也无法被探测。

如果有多个从机任务向同一个帧槽内的帧头响应，那么会发生碰撞事故。主机只有再要求所有相关的无条件帧解决这个碰撞问题之后，才能再次请求事件触发帧。

如果其中一个碰撞从机节点在撤回过程中不打乱输送，那么主机将无法探测到该节点。从机需重新尝试发送响应，直到成功为止，否则该响应将会丢失。

（如果校验和有效的话）所以事件触发帧的认购器需接收帧，并像所有关联的无条件帧已经被接收了一样，对它的数据进行应用。

图2.8是事件触发帧序列的一个例子：

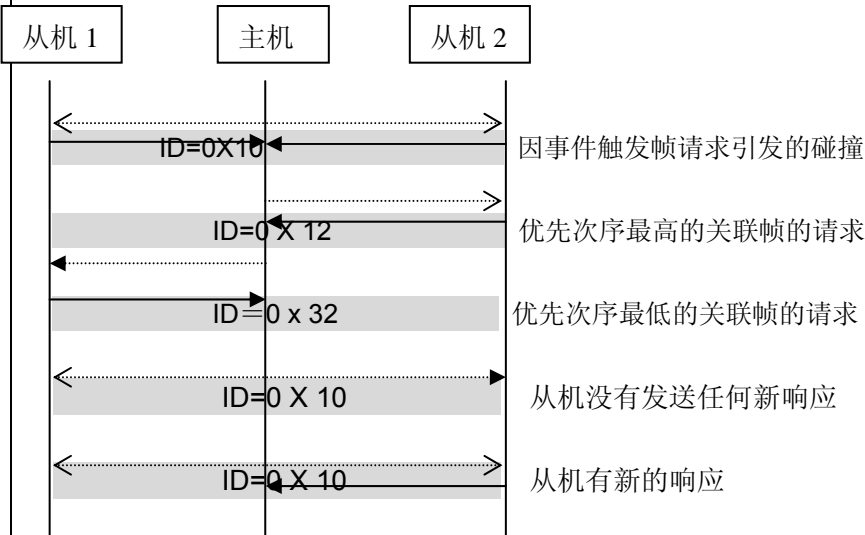


图 2.8: ID 为 0 X 10 的帧是与无条件帧 0 X 11 和 0 X12 相关的事件触发帧。在每 5 个帧槽之间，其他帧可以根据进度表的定义进行输送。



## 帧的转移

**LIN 协议规范**  
版本 2.0  
2003年9月23日； P9

### 例子：

事件触发帧的一个典型应用是监测四门中控锁系统的门把。通过使用事件触发帧来检测所有4个门，系统就可以显示较好的响应次数，同时也可以使总线负担最小化。如果有多个乘客同时按下同一个门把，那么在这种情况下，系统不会错过任何乘客的按下指令，而所花时间可能会长一点。不过上述情况的发生概率很低。

### 注：

如果将增强校验和用于事件触发帧，那么在校验和的计算过程中，就会用到帧头内被输送的受保护标识符。

### **2.3.3 零星帧**

零星帧的用途是将一些动力行为混入确定的实时进度表，且同时不会失去进度表其他部分的决定作用。

零星帧经常携带的信号以及它们的标识符的范围值是0-59(0x3b).

当主机任务获知，帧中所载运的信号已经被更新时，零星帧的帧头就可以被发送了，不过只能在与之相关的帧槽里面进行发送。零星帧的发布服务器应向帧头产生响应。（如果校验和有效的话）所以零星帧的认购器应接收帧并使用它的数据。

如果多个零星帧与同个帧槽发生关联（这属正常情况），优先次序最高<sup>7</sup>的零星帧（当信号被更新时）应在该帧槽里面输送。如果与帧槽关联的零星帧无任何更新信号，那么该帧槽将不发出任何声音。

如果主机任务知道了载运的信号已经被更新，那么该请求会将主机节点变成零星帧的常规发布服务器。即使事件触发帧里面发生了碰撞，主机任务也会意识到相关无条件帧的存在。

图2.9是零星帧序列的一个例子。

注 7：见《LIN 配置语言规范》的第 2.4.3 章节

**联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国**  
**电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com**



# 帧的转移

**LIN 协议规范**  
版本 2.0  
2003年9月23日；P10

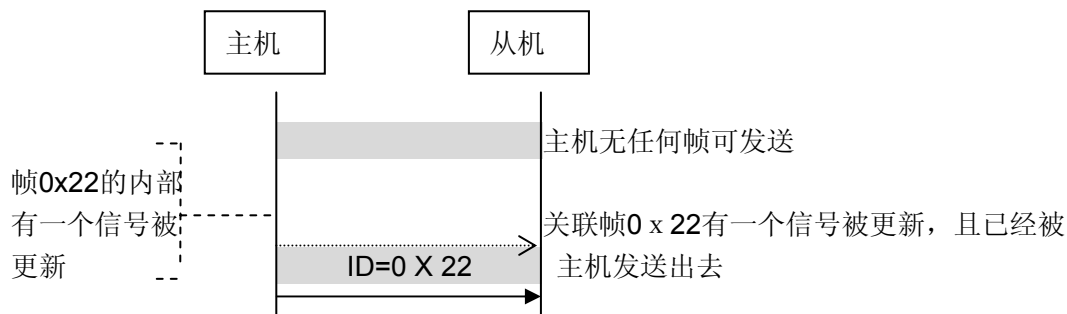


图2.9: 正常情况下，零星帧槽是空的。上图第二个槽内的一个关联帧（0x22）已经被更新。根据进度表的进程，其他帧可以在帧槽之间进行转移，tYy

## 2.3.4 诊断帧

诊断帧通常携带诊断或配置数据，且它们通常包含8个数据字节。标识符值为60 (0x3c)的帧被称为主机请求帧，值为61 (0x3d)的帧被称为从机响应帧。数据的判读见《LIN诊断和配置规范》。

在产生诊断帧的帧头之前，主机任务会询问其诊断模块是否需要发送帧或总线是否处于无声状态。根据诊断模块，从机任务可以发布并认购响应。

## 2.3.5 用户定义的帧

用户定义的帧可以载运任何形式的信息。它们的标识符是62 (0x3e)。只有当分配给帧的帧槽被处理时，其帧头才可以被输送。

## 2.3.6 保留帧

保留帧不可用于LIN2.0机群。它们的标识符是63(0x3f).



## 进度表

**LIN 协议规范**  
版本 2.0  
2003年9月23日； P11

### 3 进度表

LIN协议的一个重要属性便是进度表的使用。进度表可以确保总线永远都不会被超载。它们同时也是确保信号周期性的一个重要因素。

主机任务可以启动LIN机群内部的所以转移行为，这也使确定性的行为成为了可能。主机有责任确保操作模式内部的所以相关帧有足够的时间进行转移。

#### 3.1 槽的分布

这部分确定了进度表应该遵循的所有要求。大多数要求的基本原理是：提供一个无冲突标准或其他的能够简单有效地执行LIN协议的准则。

与零星帧或事件触发帧相关的无条件帧不可以被分配在与零星帧或事件触发帧一样的进度表里面。

帧槽必须有一个足够长的间隙时间，让主机任务和TFrame\_Maximum有时间完成输送。

Frame\_Maximum的定义见方程（8）。正如方程下面的注解所标注的那样，如果有发布服务器的支持，TFrame\_Maximum的值可能会有所减少。

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 4 任务行为模式

本章定义了LIN节点的行为模式。该行为模式是在主机任务/从机任务的概念上建立起来的。没有必要执行拥有三个独立状态机的主机节点或拥有两个独立状态机的从机节点，因为它们非常有可能会结合成一个节块。

### 4.1 主机任务状态机

主机任务负责生成正确的帧头，根据进度表决定哪个帧应该被发送，并维持帧之间的正确定时。主机任务状态机如图4.1所示：

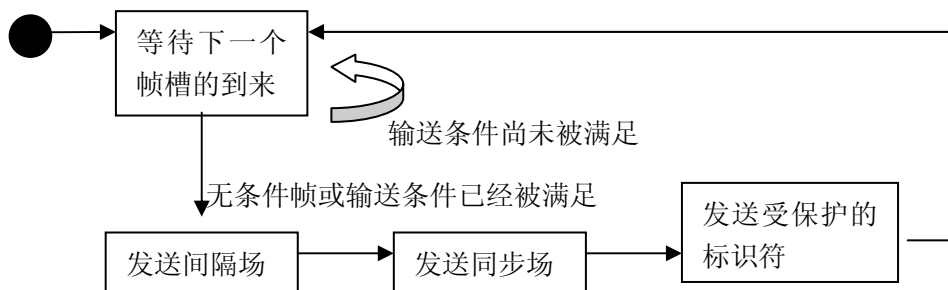


图4.1：主机任务完整的状态机

上述状态机并没有描绘如果对标识符进行选择。

#### 注：

由于没有要求对主机任务状态机内部的错误进行检测。因此，内部可能会发生错误，比如，在发送隐性比特的时候，探测到的显性比特将会使从机忽视帧头。

### 4.2 从机任务状态机

当从机任务充当发布服务器的角色的时候，它才负责发送帧响应；且当从机任务是认购器时，它才负责接收帧响应。从机任务有两个状态机模型。

- 间隔和同步探测器
- 帧处理器

#### 4.2.1 间隔和同步探测器

从机任务要求能够与帧的受保护标识符场在一开始的时候保持同步。也就是说，它必须能够正确的收到受保护符场。在整个帧里面，它必须在要求的比特率偏差内保持同步。见《LIN 物理层规范》第一部分的要求。为了达到这个目的，每个帧在开始的时候，都会伴随着间隔场和同步字节场序列。这个





## 任务行为模式

**LIN 协议规范**  
版本 2.0  
2003年9月23日； P13

序列在整个LIN通讯过程中是唯一的，且能够提供足够的信息，让从机任务探测到一个新帧的开始，且能够在标识符场一开始的时候便保持同步。

### 4.2.2 帧处理器

帧处理过程由两个状态组成：休眠状态和激活状态。激活状态包含5个子状态。只要间隔和同步程序 (BreakAndSynch)被激活，系统就会进入接收标识符 (Receive Identifier) 的激活子状态。这表明，通过探测到一个新的间隔和同步序列，帧的处理程序将会被终止。帧的处理器状态机如图4.2所示。

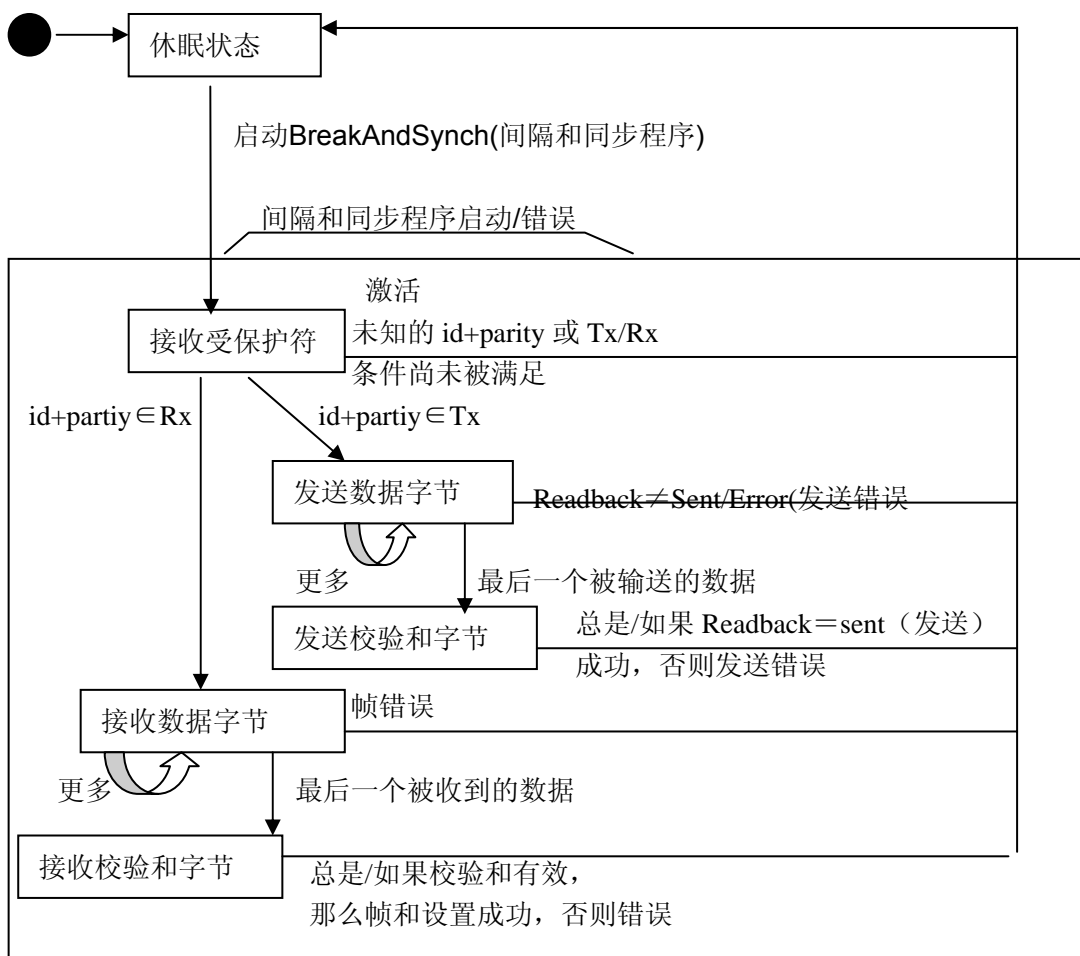


图4.2：帧处理器状态机

错误和成功的定义请参考第6章节的状态管理描述。



## 任务行为模式

**LIN 协议规范**  
版本 2.0  
**2003年9月23日； P14**

如果回读核对（read back）和发送的数据之间不匹配，那么应该在字节场（包含了这个不匹配问题）完成前，探测到该不匹配问题。一旦探测到该问题时，发送行为将立即终止。

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 网络管理

LIN 协议规范

版本 2.0

2003年9月23日；P15

### 5. 网络管理

LIN机群内部的网络管理仅仅是指机群的唤醒和休眠。其他网络管理的特点，举例来说，有配置检测和“跛行回家”（limp-home）管理。这两个是应用程序。

#### 5.1 唤醒

任何处于休眠LIN机群状态的节点可以请求唤醒程序<sup>8</sup>。让总线处于显性状态 $250\mu s$ 至 $5ms$ ，便可以产生唤醒请求。

每个从机节点（与电源相连）应可以探测到唤醒请求（主脉冲不长于 $150\mu s$ <sup>9</sup>），且会在 $100ms$ 内随时等候总线命令。该 $100ms$ 是从主脉冲的终止边缘开始计算的。主机被唤醒，且当从机节点准备好时<sup>10</sup>，主机节点才开始发送帧头，开始探测造成唤醒请求发布的缘由。

如果主机未在收到唤醒请求的 $150ms$ 内发送帧头，那么发送请求的节点可以尝试发送一个新的唤醒请求。三次请求（均失败）后，节点应该等待至少 $1.5$ 秒后才能发送第四个唤醒请求。

#### 5.2 进入休眠状态

处于激活机群的所有从机节点会通过发送一个诊断主机请求帧（帧的标识符= $0x3c$ ）而被迫进入休眠模式<sup>11</sup>。该请求帧的第一个数据字节等于 $0$ <sup>12</sup>。这种特殊的诊断帧的使用，被称之为“进入休眠命令”。如果LIN总线处于未激活状态已经超过 $4$ 秒了，从机节点也会自动进入休眠模式。

注8：主机可以发送一个间隔符号，比如，发送一个普通的帧头，因为该间隔将充当唤醒脉冲的角色。

注9： $150\mu s$ 的检测阈若结合一个 $250\mu s$ 的脉冲生成，便可产生一个探测余量。该探测余量足够大，可以容纳未被校准的从机节点。

注10：（从唤醒开始）需花费 $100ms$ 的时间，除非主机有附加信息，比如，造成唤醒的原因只是因为机群里面的其中一个从机。

注11：休眠状态仅覆盖机群。节点内的应用程序仍然是激活的。

注12：正常情况下，第一个数据字节被判读为节点地址，即NAD；不允许地址为零。

注13：未被激活的定义如下：隐性位和显性位之间的比特值无任何转移。

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



### 5.3 能源管理

图 5.1 的状态图向我们展示了 LIN 节点能源管理的行为模型。本文件中指定的 LIN 协议行为仅适用于操作状态。

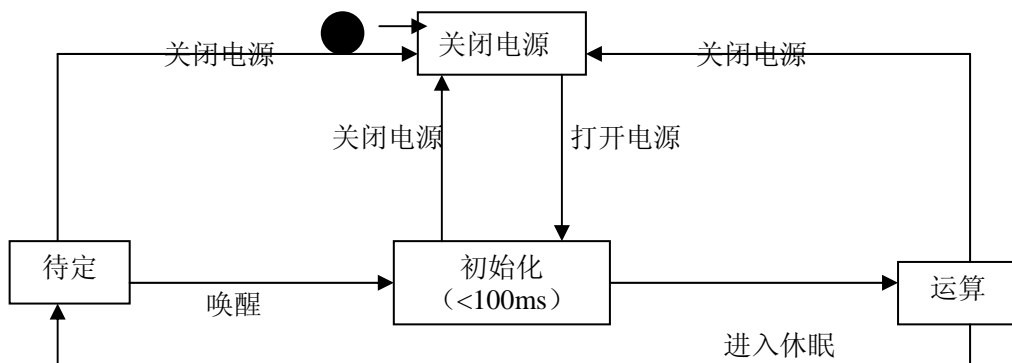


图 5.1: LIN 节点的能源管理



## 状态管理

**LIN 协议规范**  
版本 2.0  
2003年9月23日；P17

### 6. 状态管理

状态管理的目的是在操作过程中检测错误。该目的有两层含义：

- 提供快速替换错误单元的方法；
- 发生错误时，为节点提供进入limp-home模式的方法。

除了本章节中所陈述的状态管理功能，节点还可以提供更为详细的错误信息，虽然根据本规范，该功能尚未被标准化。

#### 6.1 概念

中心机群的状态管理程序是在主机节点内完成的。主机节点可以监控每个节点发回的状态报告，将报告进行过滤/整合，然后总结是否有一个或多个节点有错误。

每个节点的应用程序也可以监测与LIN总线的交互。如果适用的话，该程序可以用于limp-home模式的进入。

#### 6.2 事件触发帧

第2.3.2章节的事件触发帧可以允许碰撞现象。因此，总线错误，如帧错误，不会影响到状态位（这既不是一个成功的转移，也不是一个错误响应）。当然，如果在关联的无条件帧内部发生错误，它将被视为一个错误。

#### 6.3 向网络汇报（Reporting to the network）

向网络汇报这个行为是为主机节点的处理程序专门设置的，它同时还可以监测机群。只有从机节点才会被要求向网络汇报它们的状态。

每个从机应向其中一个被输送帧的主机节点发送状态位信号，即Response\_Error。每当被节点接收到的帧或节点输送的帧的响应场存在错误的时候，Response\_Error将会被设置。每次输送之后，系统都会清理Response\_Error。

在这个单比特的基础上，主机节点总结如下：

Response\_Error = False  $\implies$  节点可以正确操作  
Response\_Error = True  $\implies$  节点有间歇出现的问题  
节点没有响应  $\implies$  该节点 (或总线或主机)有严重的问题

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



由于主机节点也可以根据“帧尚未被发送”这个事实获得信息，因此Response\_Error这个状态位无法被放在事件触发帧里面。除了这个限制外，任何有节点输送的帧都可以用来携带Response\_Error状态位。

主机节点应用程序有责任整合并过滤每个独立的状态报告，并将不同从机节点收到的报告进行整合<sup>14</sup>。

注：

Response\_Error对执行帧的收发器进行性能符合测试（协议引擎），而不受应用程序和信号交互层的束缚。

如果需要的话，从机节点可以提供较多的状态信息，但单个Response\_Error位应时刻都存在。

6.4 各自节点内部的报告

本章节适用于基于软件的节点，不过我们推荐，基于ASIC的状态机的操作应使用相同的概念。

在各自的节点里面，节点可以提供两个状态位用于状态管理：它们是error\_in\_response 和 successful\_transfer。各自的节点应用也能够收到节点认可的最后一个帧的受保护标识符。

——每当节点收到的帧或节点发送的帧，其响应场里有错误时，就会设置error\_in\_response，也就是说，在同样的情况下，也会设置Response\_Error信号。

——当节点已经成功地将帧转移后，也就是说，帧要么已经被接收到，要么已经被发送出去，就会设置successful\_transfer。

每次读取后，这两个状态位就被清理掉。

这两个状态位可以根据表6.1进行判读。

表6.1：节点的内部错误判读

响应中的错误	成功的转移	判读
0	0	无通讯
1	1	间歇通讯（有些被成功地转移，有些则转移失败）
0	1	完全通讯
1	0	错误的通讯（只有失败的转移）

注 14:比如，如果有多个节点没有响应，主机可能会总结它本身，而不是所有从机是这个错误的节点。



## 状态管理

**LIN 协议规范**

版本 2.0

**2003年9月23日； P19**

节点应用程序有责任整合和过滤每个独立的状态报告。

注：

各自节点内部的报告程序，在《LIN API 规范》里面都是标准化的，可以自动生成应用程序。这些应用程序能够对完整的 LIN 驱动器模块自动地执行性能符合测试，包括单个信号交互层。

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 附件

**LIN 协议规范**  
版本 2.0  
2003年9月23日; P20

## 7 附件

### 7.1 数字属性表

表7.1: 数字属性表的定义

属性	最小	最大	单位	参考	备注
标量信号尺寸	1	16	BIT	第1.1章节	
字节数组尺寸	1	8	BYTE	第1.1章节	
间隔长度	13		Tbit	第2.1.1章节	
间隔检测阈	11	11	Tbit	第2.1.1章节	详见脚注3
唤醒请求期限	0.25	5	ms	第5.1章节	
从机初始化时间		100	ms	第 5.1 章节	
每两次唤醒请求之间的沉积时间	150		ms	第 5.1 章节	
每三次唤醒后的沉寂时间	1.5		ms	第 5.1 章节	

### 7.2 有效标识符表

ID[0....5]		P0 =	P1 = $\neg ID1 \oplus ID3 \oplus$	ID场	ID 场		
Dec	十六位制	$ID0 \oplus ID1 \oplus ID2 \oplus ID4$	$ID4 \oplus ID5$	7 6 5 4 3 2 1 0	Dec	十六位制	
0	0 x 00	0	1	1 0 0 0 0 0 0 0	128	0 x 80	
1	0 x 01	1	1	1 1 0 0 0 0 0 1	193	0 x C1	
2	0 x 02	1	0	0 1 0 0 0 0 1 0	66	0 x 42	
3	0 x 03	0	0	0 0 0 0 0 0 1 1	3	0 x 03	
4	0 x 04	1	1	1 1 0 0 0 1 0 0	196	0 x C4	
5	0 x 05	0	1	1 0 0 0 0 1 0 1	133	0 x 85	
6	0 x 06	0	0	0 0 0 0 0 1 1 0	6	0 x 06	
7	0 x 07	1	0	0 1 0 0 0 1 1 1	71	0 x 47	
8	0 x 08	0	0	0 0 0 0 1 0 0 0	8	0 x 08	
9	0 x 09	1	0	0 1 0 0 1 0 0 1	73	0 x 49	
10	0 x 0A	1	1	1 1 0 0 1 0 1 0	303	0 x CA	
11	0 x 0B	0	1	1 0 0 0 1 0 1 1	139	0 x 8B	
12	0 x 0C	1	0	0 1 0 0 1 1 0 0	76	0 x 4C	
13	0 x 0D	0	0	0 0 0 0 1 1 0 1	13	0 x 0D	
14	0 x 0E	0	1	1 0 0 0 1 1 1 0	142	0 x 8E	
15	0 x 0F	1	1	1 1 0 0 1 1 1 1	207	0 x CF	
16	0 x 10	1	0	0 1 0 1 0 0 0 0	80	0 x 50	
17	0 x 11	0	0	0 0 0 1 0 0 0 1	17	0 x 11	
18	0 x 12	0	1	1 0 0 1 0 0 1 0	146	0 x 92	
19	0 x 13	1	1	1 1 0 1 0 0 1 1	211	0 x D3	
20	0 x 14	0	0	0 0 0 1 0 1 0 0	20	0 x 14	
21	0 x 15	1	0	0 1 0 1 0 1 0 1	85	0 x 55	

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com





# 附件

**LIN 协议规范**  
版本 2.0  
2003年9月23日; P21

ID[0....5] Dec 十六位制		P0 = ID0 ⊕ ID1 ⊕ ID2 ⊕ ID4	P1 = ¬ID1 ⊕ ID3 ⊕ ID4 ⊕ ID5	ID场 7 6 5 4 3 2 1 0	ID 场 Dec 十六位制	
22	0 x 16	1	1	1 1 0 1 0 1 1 0	214	0 x D5
23	0 x 17	0	1	1 0 0 1 0 1 1 1	151	0 x 97
24	0 x 18	1	1	1 1 0 1 1 0 0 0	216	0 x D8
25	0 x 19	0	1	1 0 0 1 1 0 1 0	153	0 x 99
26	0 x 1A	0	0	0 1 0 1 1 0 1 0	26	0 x 1A
27	0 x 1B	1	0	0 1 0 1 1 0 1 1	91	0 x 5B
28	0 x 1C	0	1	1 0 0 1 1 1 0 0	156	0 x 9C
29	0 x 1D	1	1	1 1 0 1 1 1 0 1	221	0 x DD
30	0 x 1E	1	0	0 1 0 1 1 1 1 0	94	0 x 5E
31	0 x 1F	0	0	0 0 0 1 1 1 1 1	31	0 x 1F
32	0 x 20	0	0	0 0 1 0 0 0 0 0	32	0 x 20
33	0 x 21	1	0	0 1 1 0 0 0 0 1	97	0 x 61
34	0 x 22	1	1	1 1 1 0 0 0 1 0	226	0 x E2
35	0 x 23	0	1	1 0 1 0 0 0 1 1	163	0 x A3
36	0 x 24	1	0	0 1 1 0 0 1 0 0	100	0 x 64
37	0 x 25	0	0	0 0 1 0 0 1 0 1	37	0 x 25
38	0 x 26	0	1	1 0 1 0 0 1 1 0	166	0 x A6
39	0 x 27	1	1	1 1 1 0 0 1 1 1	231	0 x E7
40	0 x 28	0	1	1 0 1 0 1 0 0 0	168	0 x A8
41	0 x 29	1	1	1 1 1 0 1 0 0 1	233	0 x E9
42	0 x 2A	1	0	0 1 1 0 1 0 1 0	106	0 x 6A
43	0 x 2B	0	0	0 0 1 0 1 0 1 1	43	0 x 2B
44	0 x 2C	1	1	1 1 1 0 1 1 0 0	236	0 x EC
45	0 x 2D	0	1	1 0 1 0 1 1 0 1	173	0 x AD
46	0 x 2E	0	0	0 0 1 0 1 1 1 0	46	0 x 2E
47	0 x 2F	1	0	0 1 1 0 1 1 1 1	111	0 x 6F
48	0 x 30	1	1	1 1 1 1 0 0 0 0	240	0 x F0
49	0 x 31	0	1	1 0 1 1 0 0 0 1	177	0 x B1
50	0 x 32	0	0	0 0 1 1 0 0 0 1	50	0 x 32
51	0 x 33	1	0	0 1 1 1 0 0 1 1	115	0 x 73
52	0 x 34	0	1	1 0 1 1 0 1 0 0	180	0 x B4
53	0 x 35	1	1	1 1 1 1 0 1 0 1	245	0 x F5
54	0 x 36	1	0	0 1 1 1 0 1 1 0	118	0 x 76
55	0 x 37	0	0	0 0 1 1 0 1 1 1	55	0 x 37
56	0 x 38	1	0	0 1 1 1 1 0 0 0	120	0 x 78
57	0 x 39	0	0	0 0 1 1 1 0 0 1	57	0 x 39
58	0 x 3A	0	1	1 0 1 1 1 0 1 0	186	0 x BA
59	0 x 3B	1	1	1 1 1 1 1 0 1 1	251	0 x FB

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 附件

**LIN 协议规范**  
版本 2.0  
2003年9月23日；P22

ID[0...5]		P0 =	P1 = $\neg ID1 \oplus ID3 \oplus$	ID场	ID 场	
Dec	十六位制	$ID0 \oplus ID1 \oplus ID2 \oplus ID4$	$ID4 \oplus ID5$	7 6 5 4 3 2 1 0	Dec	十六位制
60 <sup>a</sup>	0 x 3C	0	0	0 0 1 1 1 1 0 0	60	0 x 3C
61 <sup>b</sup>	0 x 3D	1	0	0 1 1 1 1 1 0 1	125	0 x 7D
62 <sup>c</sup>	0 x 3E	1	1	1 1 1 1 1 1 1 0	254	0 x FE
63 <sup>d</sup>	0 x 3F	0	1	1 0 1 1 1 1 1 1	191	0 x BF

- a. 标识符60 (0x3C) 是为主机请求命令帧预留的 (见第2.3.4章节)。  
b. 标识符61 (0x3D) 是为从机响应命令帧预留的 (见第2.3.4章节)。  
c. 标识符62 (0x3E) 是为用户定义扩展帧预留的 (见第2.3.5章节)。  
d. 标识符63 (0x3F) 是为以后LIN扩展模板预留的 (见第2.3.6章节)。

### 7.3 校验和计算的例子

下文是4字节的校验和计算方法。如果字节是4数据字节或是受保护的标识符，且3个数据字节并不是很重要的话，那么计算方法是一样的。

Data=0x4A, 0x55, 0x93, 0xE5

	十六位制	CY	D7	D6	D5	D4	D3	D2	D1	D0
0x 4A	0x4A		0	1	0	0	1	0	1	0
+0X55= (添加Carry)	0x9F 0X9F	0	1 1	0 0	0 0	1 1	1 1	1 1	1 1	1 1
+0X93= (添加Carry)	0X132 0X33	1	0 0	0 0	1 1	1 1	0 0	0 0	1 1	0 1
+0XE5= (添加Carry)	0X118 0X19	1	0 0	0 0	0 0	1 1	1 1	0 0	0 0	0 1
取向	0XE6		1	1	1	0	0	1	1	0
0x19+0xE6 =	0XFF		1	1	1	1	1	1	1	1

总和是0x19。取向后，产生的最后结果是：校验和=0xE6。

接收节点能够利用同样的加法原理，很容易地对接收到的帧的一致性进行检查。当收到的校验和 (0XE6) 被加到 (0x19) 上，得到的和即为0xFF。

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 7.4 本标准中使用的句法和数学符号

### 序列图

为了使本标准的含义形象化，我们在适当地时候使用了序列图。图表中的句法详见图7.1中的例子。

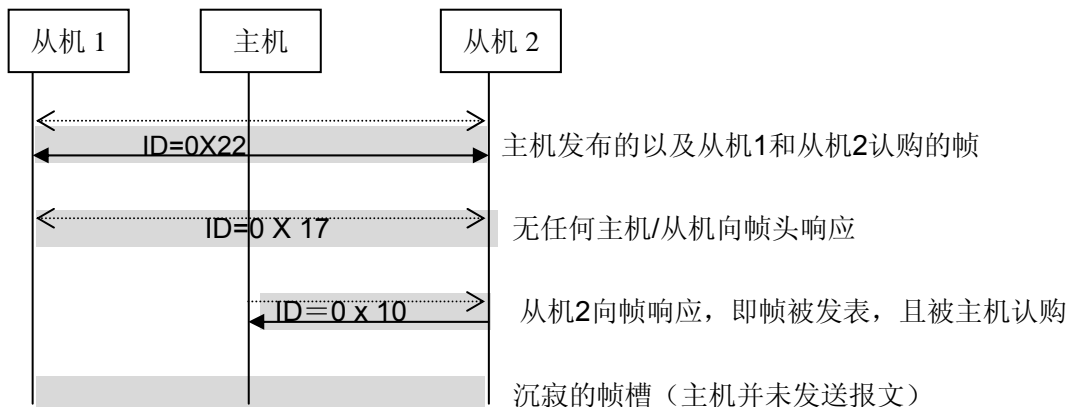


图7.1：帧的序列图。灰色部分代表了帧槽（帧槽之间有间隔，目的是使制图更加清爽）。虚线箭头代表报文，实心箭头代表响应。

### 数学符号

下文是本标准中包含的数学符号：

- $f \in S$  属于。如果F被包含在S组里面，则是真实的。
- $a \oplus b$  不包括，或者说，如果a和b里面有一个是正确的，则是真实的。
- $\neg a$  非。如果a是错的，则是真实的



## **LIN 诊断和配置规范**

### **版本 2.0**

本规范是在“ASIS”的基础上形成的，不可以作为任何索赔的依据。

© LIN 协会 2003.

版权所有。未经授权的复印、演示、或对本文件其他方面的使用，都是违反法律和知识产权的。

LIN是一个注册商标®.

本文件的任何分发都登记在案。



## 介绍

**LIN 诊断和配置规范**  
版本 2.0  
**2003年9月23日； P2**

### 1. 介绍

《LIN诊断和配置标准》定义了一个节点是如何被配置的（这对所有节点来说，都是强制实行的），并提出了3条用于执行诊断数据集的备选方法（所有方法都可以选择执行的）。

LIN协议会输送诊断和配置数据，具体见《LIN协议规范》。用于C编程语言的标准化AP，也在《LIN API规范》中被详细叙述。

联系方式: **H.-Chr. v. d. Wense**, 摩托罗拉有限公司, **Schatzbogen 7, D-81829 慕尼黑, 德国**  
电话: **+49 (89) 92103-882** E-Mail: **H.Wense@Motorola.com**



2. 节点配置

LIN节点的配置可以用来设置一个机群内部的LIN从机节点。这对一个不具有离架节点的机群来说，它可以使机群内部的从机节点避免相互间的冲突。

通过拥有大型的地址间隙，就可以完成配置。该间隙包括一个帧的信息标识符，一个从机节点的LIN产品ID，以及一个从机节点的初始NAD。通过对这些数字的使用，就可以让所有在机群内部被输送的帧，具有一个独特的标识符。

对LIN节点来说，支持节点配置是一个强执行命令。

2.1 节点模型

一个从机节点的记忆，可以表述如图2.1：

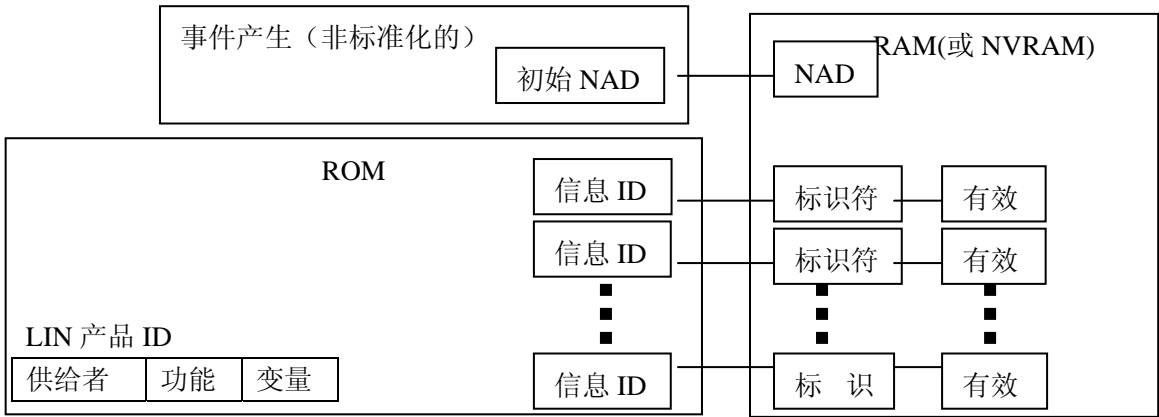


图2.1：从机节点的记忆模型

从机节点有一个固定的LIN产品ID，见第2.4章节；所有帧都有一个信息标识符。

重新设置<sup>1</sup>后，从机节点应处于下列状态中：

- 它有一个NAD单元，与动态决定的实体值（即intance value，它是初始NAD）一致。该实体值的第一个实体是1，第二个是2，如此类推。至于确定实体号码的方法并不属于LIN标准<sup>2</sup>的一部分。
- 它会对所有受保护的标识符打上标记，视为无效。

标识符为0x3c或0x3c以上的帧有固定的（有效）标识符，且无需具备任何信息标识符。

注1：如果在关闭电源的时候将配置保存在NVRAM里面，那么当电源打开时，系统不会重新设置。

注 2：有一种可能是利用跳线对实体进行配置。



## 节点配置

**LIN 诊断和配置规范**  
版本 2.0  
2003年9月23日； P4

### 通配符

为了能够删除一些未指明的信息，表2.1内的通配符可以用于节点配置请求：

表2.1：在所有请求中都可以使用的通配符

属性	通配符值
NAD	127
供给者ID	0 X 7FFF
功能ID	0 X FFFF
信息ID	0 X FFFF

供给者ID通配符、功能ID通配符以及信息ID通配符的实施是可以随意选择的。

### PDU结构

本章节的信息是章节3.3.1所提供的信息的一个子集。对LIN节点来说，子集是强制执行的，而第3.3.1章的扩展集是可选择的。

在LIN诊断帧内部输送的单元被称为PDU（分组数据单元）。用于节点配置的PDU是一个完整的信息。客户机程序（ISO:检测器，LIN：主机）发布的信息被称为请求；服务器（ISO:主机，LIN：从机）发布的信息，被称为响应。

LIN机群不是用流量控制（其定义见ISO[1]）。如果主干总线（back-bone bus）测试仪器需要流量控制PDU，PDU必须由主机节点生成。

#### 2.3.1 概述

为了简化ISO诊断帧[1]和LIN诊断帧之间的转化，我们定义了一个非常相似的结构，该结构可以支持PDU类型，见图2.2。

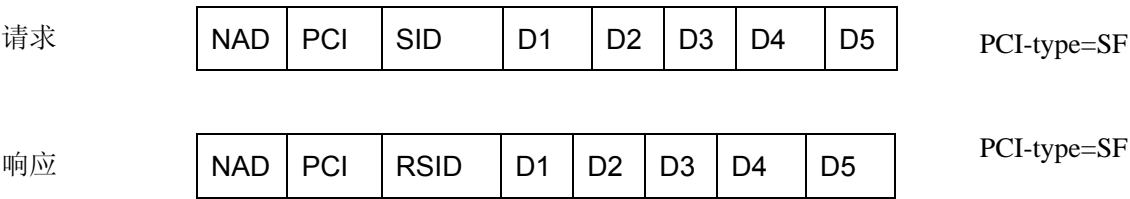


图2.2：由LIN配置支持的PDU。左边的字节（NAD）是第一个被发送的字节，右边的字节是最后一个被发送的字节（D5）。



## 节点配置

**LIN 诊断和配置规范**  
版本 2.0  
**2003年9月23日； P5**

请求通常是在主机请求帧内部发送的，而响应则是在从机响应帧内部发送的。PDU内部每个字节的定义详见下面几个章节。

### NAD

NAD是从机节点的地址。发送请求的时候，会给从机节点标注地址，也就是说，只有从机节点才有地址。NAD同时也可以标是响应的来源。

NAD值的范围是1-127。0以及128-255是为其它用途预留的。

0 是为进入休眠命令预留的，见《LIN协议规范》

1-126 (0x7E) 诊断从机节点的地址

127 (0x7F) 为无线广播预留的

128 (0x80) -255(0xFF) 自由使用。帧不会被判读为一个诊断帧。见第3.2章节

注意：物理节点和逻辑节点之间有一个一对一制图，它是使用NAD进行地址标注的。

### PCI

PCI（协议控制信息）包含运输层的流量控制信息。对节点配置来说，PCI的字节判读是存在的，见表2.2

表2.2：用于PDU配置的PCI字节结构

类型	PCI类型				附加信息			
	B7	B6	B5	B4	B3	B2	B1	B0
SF	0	0	0	0	长度			

PCI类型的单帧（SF）表示备输送的信息符合单PDU，也就是说，它最多包含了5个数据字节。长度等于所使用的数据字节的数量再加上1（这只针对SID或RSID）。

### SID

服务标识符（SID）具体规定了从机节点应该执行的请求。该从机节点有地址标注。0x b0到0x b4是用于配置的。SID的编号应该与ISO 15765-3一致，并将节点的配置放置到某个区域。该区域见“机动车生产商的定义”。

注 3：根据 LIN1.2 标准的规定，第一个字节范围在 128 (0x80) -255 (0 x ff) 的诊断帧，可以自由使用。

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com





## 节点配置

**LIN 诊断和配置规范**  
版本 2.0  
2003年9月23日； P6

### RSID

响应服务标识符（RSID）具体规定了响应的内容。用于积极响应的RSID一般都是SID+0x 40

### D1到D5

数据字节（每个字节配置PDU有5个数据字节）的判读取决于SID或RSID。

如果一个PDU无法完全被填满，剩余的字节应该用“1”来填满，也就是说，它们的值应该是255(0xff)。这是非常有必要的，因为一个诊断帧，其长度通常是8个字节。

## 2.4 LIN产品ID

每个LIN部件都有一个独特的号码，见表2.3：

表2.3： LIN产品ID

D1	D2	D3	D4	D5
供给者ID LSB	供给者ID MSB	功能ID LSB	功能ID MSB	变量ID

根据LIN协会，每个供给者都有一个供给者ID。它是一个16位的值，最重要的位，其值为0<sup>4</sup>。

每个供给者都有一个功能ID。如果两个产品，其功能不一样，也就是说，LIN通讯或物理世界交互，那么它们的功能ID应该会不一样。不过，对那些功能绝对一样的产品来说，其功能ID是不会变的。

最后，每当产品变化是，其变量ID也会发生变化的。不过，对这些产品的要求是它们的功能不能被改变。

将LIN产品ID纳入LIN节点内部，是强制性命令，必须要执行的。

## 2.5 强制请求

本章节列出的请求都需得到所有LIN从机节点的支持。

### 2.5.1 分配帧标示符

分配帧id（assign frame id）用于设置有效的受保护标示符，该标示符是信息标识符具体规定的帧的标示符。分配帧id的结构见表2.4。

需要值的注意的是，请求会提供一个受保护的标示符，也就是说，标识符和它的奇偶校验。进一步说，标识符为60（0x3c）以及以上的帧无法被改变（包括诊断帧、用户定义的帧以及保留帧）。

注 4：将大多数重要的位设置为 1，这是为以后扩展的编号系统预留的。

**联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国**  
**电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com**



## 节点配置

**LIN 诊断和配置规范**  
版本 2.0  
2003年9月23日; P7

表2.4: 分配帧id请求

NAD	PCI	SID	D1	D2	D3	D4	D5
NAD	0x06	0xb1	供给者ID LSB	供给者ID MSB	信息ID LSB	信息ID MSB	受保护的ID

只有当NAD和供给者ID匹配的时候, 才会发送响应。如果成功了, 那么表2.5内的信息就会作为响应被发送。响应的执行也是可选择的。

表2.5: 积极的分配帧ID响应

NAD	PCI	RSID	未被使用的				
NAD	0x01	0xf1	0xff	0xff	0xff	0xff	0xff

### 2.5.2 read by identifier request(由标识符请求读取)

通过表2.6内请求, 可以实现从从机节点处读取供给者的ID和其他属性。

只有当NAD、供给者ID以及功能ID匹配时才可以发送响应。积极的响应见表2.8。如果响应无效, 也就是说, 子功能不被支持, 那么表2.9内的消极响应就会被发送出去。

表2.6: Read by identifier request

NAD	PCI	SID	D1	D2	D3	D4	D5
NAD	0x06	0xb2	标识符	供给者ID LSB	供给者ID MSB	功能ID LSB	功能ID MSB

标识符的定义见表2.7。

表2.7: 使用read by identifier request读取标识符

标识符	判读	响应的长度
0	LIN协议ID	5+RSID
1	编号	4+RSID
2-15	预留	消极响应; 不被子功能支持
16-31	信息ID, 1...16	3个字节+RSID
32-63	用户定义的	用户定义的
64-255	预留的	预留的

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 节点配置

**LIN 诊断和配置规范**  
版本 2.0  
2003年9月23日; P8

表2.8: 可能的积极read by identifier响应。每行代表一个可能的响应

Id	NAD	PCI	RSID	D1	D2	D3	D4	D5
0	NAD	0x06	0xf2	供给者ID: LSB	供给者ID MSB	功能ID LSB	功能ID MSB	变量
1	NAD	0x05	0xf2	编码0, LSB	编号1	编号2	编号3, MSB	0xff
预留								
16	NAD	0x04	0xf2	信息ID1 LSB	信息ID1 MSB	受保护的ID (或FF)	0xff	0xff
17	NAD	0x04	0xf2	信息 ID2 LSB	信息 ID2 MSB	受保护的 ID(或 FF)	0xff	0xff
18	NAD	0x04	0xf2	信息 ID3 LSB	信息 ID3 MSB	受保护的 ID(或 FF)	0xff	0xff
19	NAD	0x04	0xf2	信息 ID4 LSB	信息 ID4 MSB	受保护的 ID(或 FF)	0xff	0xff
20	NAD	0x04	0xf2	信息 ID5 LSB	信息 ID5 MSB	受保护的 ID(或 FF)	0xff	0xff
21	NAD	0x04	0xf2	信息 ID6 LSB	信息 ID6 MSB	受保护的 ID(或 FF)	0xff	0xff
22	NAD	0x04	0xf2	信息 ID7 LSB	信息 ID7 MSB	受保护的 ID(或 FF)	0xff	0xff
23	NAD	0x04	0xf2	信息 ID8 LSB	信息 ID8 MSB	受保护的 ID(或 FF)	0xff	0xff
24	NAD	0x04	0xf2	信息 ID9 LSB	信息 ID9 MSB	受保护的 ID(或 FF)	0xff	0xff
25	NAD	0x04	0xf2	信息 ID10 LSB	信息 ID10 MSB	受保护的 ID(或 FF)	0xff	0xff
26	NAD	0x04	0xf2	信息 ID11 LSB	信息 ID11 MSB	受保护的 ID(或 FF)	0xff	0xff
27	NAD	0x04	0xf2	信息 ID12 LSB	信息 ID12 MSB	受保护的 ID(或 FF)	0xff	0xff
28	NAD	0x04	0xf2	信息 ID13 LSB	信息 ID13 MSB	受保护的 ID(或 FF)	0xff	0xff
29	NAD	0x04	0xf2	信息 ID14 LSB	信息 ID14 MSB	受保护的 ID(或 FF)	0xff	0xff
30	NAD	0x04	0xf2	信息 ID15 LSB	信息 ID15 MSB	受保护的 ID(或 FF)	0xff	0xff
31	NAD	0x04	0xf2	信息 ID16 LSB	信息 ID16 MSB	受保护的 ID(或 FF)	0xff	0xff

表2.9: 消极响应

NAD	PCI	RSID	D1	D2	未被使用		
NAD	0X03	0X7F	被请求的SID (=0XB2)	错误代码 (=0x12)	0 x ff	0xff	0xff

联系方式: **H.-Chr. v. d. Wense**, 摩托罗拉有限公司, **Schatzbogen 7, D-81829 慕尼黑, 德国**  
电话: **+49 (89) 92103-882** E-Mail: **H.Wense@Motorola.com**



## 节点配置

**LIN 诊断和配置规范**  
版本 2.0  
2003年9月23日； P9

注：

对标识符0（LIN产品ID）的支持是唯一的一个强制实行的标识符，也就是说，编号和信息ID是可供选择的。

范围在32-63的标识符是用户定义的，因此没有列在表2.8中。

### 2.6 选择要求

本章节列出的请求使可供选择的，可以在LIN从机节点内执行。这些请求是可供选择的，因为它们的成本效果并不总是会被激发。此外，为了鼓励形成统一的问题解决方案，也对这些请求做出了规范。也就是说，推荐系统使用下列不同类型的请求来解决同一个问题。

#### 2.6.1 Assign NAD（分配NAD）

Assign NAD用于解决有冲突的节点地址。其结构如表2.10所示：

只有当NAD、供给者ID和功能ID匹配时，才会发送响应。如果成功的话，表2.11内的信息就会作为响应被发送。响应的执行是可选的。

表2.10: Assign NAD请求

NAD	PCI	SID	D1	D2	D3	D4	D5
初始 NAD	0x06	0xb0	供给者ID LSB	供给者ID MSB	功能ID LSB	功能ID MSB	新的 NAD

表： 2.11： 积极的Assign NAD请求

NAD	PCI	RSID	未被使用的				
初始NAD	0x01	0xf0	0xff	0xff	0xff	0xff	0xff

注：

这个服务总会用到初始NAD；目的是希望避免节点地址的丢失。用于响应的NAD应该与请求中使用的NAD一致，即都是初始NAD。

#### 2.6.2 Conditional change NAD（有条件变化的NAD）

Conditional change NAD用于检测和分离机群中未知的从机节点。这些未知节点出现在机群里面的潜在原因是，举例来说，是当机群的时候有不正确的集合，或在服务阶段，发生不正确的节点替代。

Conditional change NAD可以检测到节点，并允许主机节点汇报描述问题的诊断信息。

请求的行为是：

联系方式： H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 节点配置

**LIN 诊断和配置规范**  
版本 2.0  
2003年9月23日; P10

1. 获得表2.8规定的标识符，并根据ID选择。
2. 截取根据字节选择的数据字节（字节=1对应第一个字节，D1）。
3. 先取反，再按位异或操作。
4. 先掩码，再按位和操作。
5. 如果最后的结果是0，将NAD变为新的NAD。

表2.12: Conditional change NAD请求

NAD	PCI	SID	D1	D2	D3	D4	D5
NAD	0x06	0xb3	ID	字节	MASK	INVERT	新的NAD

表2.13: 可选择的积极Conditional change NAD响应

NAD	PCI	RSID	未被使用的				
NAD	0x01	0xf3	0xff	0xff	0xff	0xff	0xff

注:

注意：用现有的NAD标注Conditional change NAD的地址，也就是说，它并不总是使用初始NAD，这与Assign NAD请求相反。

### 2.6.3 Data Dump (数据转储)

注:

SID=0xb4是节点供给者为节点的初始配置保留的。信息的模板是由供给者具体规定的。因此，SID不可以在运算LIN机群中使用。

表2.14: Data dump请求

NAD	PCI	SID	D1	D2	D3	D4	D5
NAD	0x06	0xb4	用户定义	用户定义	用户定义	用户定义	用户定义

表2.15: Data dump响应

NAD	PCI	SID	D1	D2	D3	D4	D5
NAD	0x06	0xf4	用户定义	用户定义	用户定义	用户定义	用户定义

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 诊断

**LIN 诊断和配置规范**  
版本 2.0  
2003年9月23日； P11

## 2. 诊断

诊断信息的收集有3种方法，具体见下文。这些方法可能会共存于同一个机群里面，甚至同一个节点里面。本文中所列出的任何方法的执行都是可选择的。

### 3.1 基于信号的诊断

关于收集诊断信息最简单的方法是在正常的、无条件的帧里面使用通用信号。这个解决方案的特点是：

- 从机节点的费用非常低
- 概念的标准化（因为它使用的是常规信号/帧）。
- 由于数据内容是固定在帧结构里面，因此不是很有弹性。

### 3.2 用户定义的诊断

诊断帧的自由范围是可以被使用的。该自由范围必须具有范围在128 (0x80) -255(0xff)之间所有第一个数据字节，见章节2.3.2。基于自由范围诊断的解决方案，其特点如下：

- 非标准化的，因此不可随意移动。
- 由于是为了满足特定需求而设计的，因此费用合理。

正是因为用户定义的诊断是非标准化的，因此基于诊断的信号就成为了受人欢迎的解决方案。

## 诊断运输层

LIN诊断运输层可以用于目标系统的锁定。在这些系统里面，ISO诊断是在（基于CAN）主干总线上被执行的，且系统的建立者希望能够在LIN子辅助总线机群上执行同样的诊断能力。事实上，这些信息与ISO信息是一样的，且PDU所携带的信息也非常相似，见第3.3.1章节的定义。典型的系统配置见图3.1所示。

LIN帧断层的目标如下：

- 主机低负荷。
- LIN从机能够直接提供全部（或部分）ISO诊断。
- 可以锁定有功效强大的节点建成的机群（非主流的低成本LIN）。

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 诊断

**LIN 诊断和配置规范**  
版本 2.0  
2003年9月23日；P12

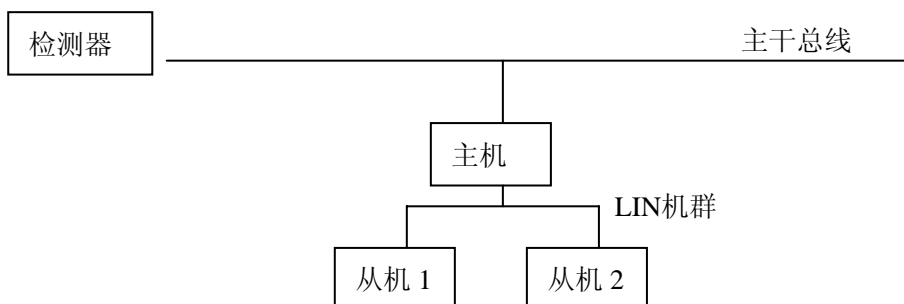


图3.1：使用运输层，对LIN机群进行系统设置（典型）

### 3.3.1 PDU结构

本章节的信息是章节2.3所提供的信息的一个子集。

在LIN诊断帧内部输送的单元被称为PDU（分组数据单元）。PDU可以是一个完整的信息，也可以是一个信息的一部分。在后一种情况下，多个被连接在一起的PDU形成一个完整的信息。

客户机程序（ISO:检测器，LIN：主机）发布的信息被称为请求；服务器（ISO:主机，LIN：从机）发布的信息，被称为响应。

LIN机群不是用流量控制（其定义见ISO[1]）。如果主干总线（back-bone bus）测试仪器需要流量控制PDUs，PDUs必须由主机节点生成。

#### 概述

为了简化ISO诊断帧[1]和LIN诊断帧之间的转化，我们定义了一个非常相似的结构，该结构可以支持PDU类型，见图3.2。

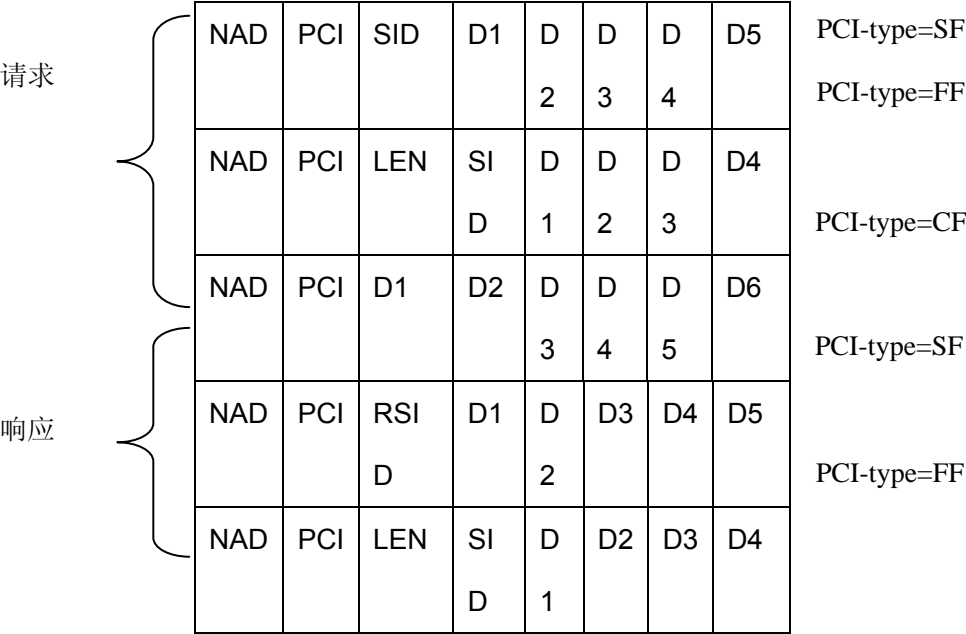


图3.2: 由LIN诊断运输层支持的PDUs。左边的字节（NAD）是第一个被发送的字节，右边的字节是最后一个被发送的字节（D4、D5或D6）。

请求通常是在主机请求帧内部发送的，而响应则是在从机响应帧内部发送的。PDUs内部每个字节的定义详见下面几个章节。

NAD

NAD的定义见章节2.3.2

PCI

PCI（协议控制信息）包含运输层的流量控制信息。对节点配置来说，PCI的字节判读有3种，见表3.1

表3.1: PCI字节的结构

类型	PCI类型				附加信息			
	B7	B6	B5	B4	B3	B2	B1	B0
SF	0	0	0	0	长度			
FF	0	0	0	1	长度/256			
CF	0	0	1	0	帧计数器			

PCI类型的单帧（SF）表示备输送的信息符合单PDU，也就是说，它最多包含了5个数据字节。长度等于所使用的数据字节的数量再加上1（这只针对SID或RSID）。

PCI 类型的第一个帧（FF）用于表示一个多 PDU 信息的起始；接下去的帧都是 CF 类型的，见下文。信息内的数据字节总数加上 1（只针对 SID 或 RSID），被视为运输长度。长度最重要的四个位是在 PCI 类型内输送的（8 个最不重要的位是在 LEN 上输送，见下文）。



联系方式: **H.-Chr. v. d. Wense**, 摩托罗拉有限公司, **Schatzbogen 7, D-81829 慕尼黑, 德国**  
电话: **+49 (89) 92103-882** E-Mail: **H.Wense@Motorola.com**



## 诊断

**LIN 诊断和配置规范**  
版本 2.0  
2003年9月23日； P14

一个多PDU信息有一系列持续帧（CF）。一个信息的第一个CF帧编号为1，第二个为2，诸如此类。如果运输一条完整的信息需要超过15个的CF PDU，帧计数器会抱合，从0,1.....开始。

### LEN

LEN字节只能在FF帧内使用；它信息长度有8个最不重要的位。因此，一个信息最大的长度是4095（0XFF）字节。

### SID

服务标识符（SID）具体规定了从机节点应该执行的请求。该从机节点有地址标注。0- 0xaf和0xb8-0xfe用于诊断，而0xb0-0xb7则用于节点配置。SID的编号应该与ISO 15765-3一致，并将节点的配置放置到某个区域。该区域见“机动车生产商的定义”。

### RSID

响应服务标识符（RSID）具体规定了响应的内容。

### D1-D6

数据字节（每个PDU有6个数据字节）的判读取决于SID或RSID。对多PDU信息来说，信息中所有PDU的字节应该在解析前连接形成完整的信息。

如果一个PDU无法完全被填满（仅仅适用于CF和SF PDU），剩余的字节应该用“1”来填满，也就是说，它们的值应该是255(0xff)。

#### 3.3.2 定义的请求

LIN运输层使用相同的诊断信息，作为ISO诊断标准[2]。之后，SID和RSID也应该负荷ISO标准。节点可以执行ISO标准所定义的服务的一个子集。

#### 3.3.3 ISO的定时限制

ISO[1][2]所使用的定时功能是根据几个属性，如P2、ST和T1，决定的。这些属性需符合特定的范围。由于LIN比CAN要慢，这些数值需做相应地调整。

这些属性的数值并不是 LIN 标准的一部分。通过时间表的选择就可以对这些数值进行控制。诊断帧之间的时间表有预期时间。用这种方式，机群开发者就可以完全控制这些值，且这些值也可以根据相关的协定进行设置。

**联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国**  
**电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com**



## 诊断

**LIN 诊断和配置规范**  
版本 2.0  
2003年9月23日；P15

### 3.3.4 序列图

有两个通讯盒；检测器希望向从机节点发送诊断请求，而从机节点希望向检测器发送诊断反响。下面的图3.3和图3.4显示了两个通讯盒之间的信息流。

组成通讯的单元（检测器或主机）应避免要求多个从机同时相应（因为这会造成总线碰撞），这是非常重要的。

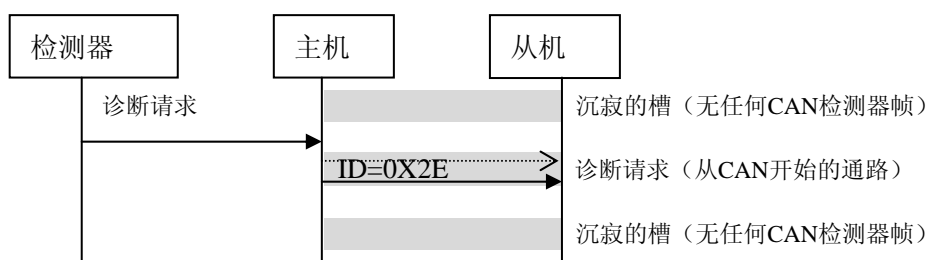


图3.3: CAN信息到LIN的通路

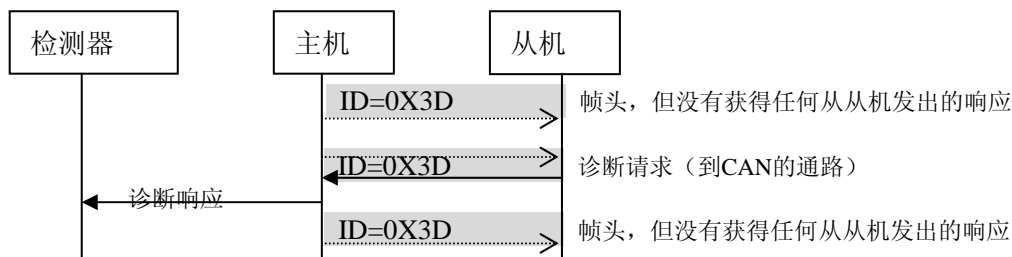


图3.4: CAN信息到LIN的通路

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 参考文献

**LIN 诊断和配置规范**  
版本 2.0  
2003年9月23日；P16

### 4 参考文献

- [1] 《道路车辆》——控制器局域网络（CAN）的诊断——第二部分：网络层服务》，国际标准ISO 15765-2.4, 第4期，2002年6月21日。
- [2] 《道路车辆》——控制器局域网络（CAN）的诊断——第三部分：诊断服务的执行》，国际标准ISO 15765-3.5, 第5期，2002年12月21日。

联系方式: **H.-Chr. v. d. Wense**, 摩托罗拉有限公司, **Schatzbogen 7, D-81829 慕尼黑, 德国**  
电话: **+49 (89) 92103-882** E-Mail: **H.Wense@Motorola.com**



## **LIN 物理层规范**

### **版本 2.0**

本规范是在“ASIS”的基础上形成的，不可以作为任何索赔的依据。

© LIN 协会 2003.

版权所有。未经授权的复印、演示、或对本文件其他方面的使用，都是违反法律和知识产权的。

LIN是一个注册商标®.

本文件的任何分发都登记在案。



## 振荡器偏差

**LIN 物理层规范**  
版本 2.0  
2003年9月23日; P2

### 1. 振荡器偏差

芯片集成时钟发生器使用内部校准, 能够获得比 $\pm 14\%$ 更精准的频率偏差。这个精准度足可以在信息流里面探测到同步间隔。使用同步场的精细校准可以确保更加适当地接收和发送信息。在考虑到任何可能会影响振荡器频率的情况下, 特别是操作过程中的温度和电压漂移情况, 芯片集成振荡器必须允许对剩余信息帧进行正确的比特率测量和生成。

编号	时钟偏差	名字	$\Delta. F / F_{Nom.}$
1.1.1	主机节点 (与标定时钟率的偏差。标定时钟率 <b>FNom</b> 的定义见LIN描述文件)	FTOL_RES_MASTER	$< \pm 0.5\%$
1.1.2	不需要同步的从机节点 (与标定时钟率的偏差。) 注: 对任何两个节点的通讯来说, 它们之间的比特率偏差不得超过 $\pm 2\%$ 。	FTOL_RES_SLAVE	$< \pm 1.5\%$
1.1.3	同步前, 从机节点时钟相对于标定时钟率的偏差; 与使用同步和直接 <b>SYNCH BREAK</b> (同步间隔) 检测的节点做比较	FTOL_UNSYNCH	$< \pm 14\%$

表1.1: 振荡器与标定时钟的偏差

编号	时钟偏差	名字	$\Delta. F / F_{Nom.}$
1.2.1	同步后, 从机节点时钟相对于主机节点时钟的偏差; 与使用同步的节点做比较; 对所有在 <b>SYNCH</b> (同步) 场之后的帧的所有场来说, 任何从机节点必须在偏差的范围内。 注: 对任何两个节点的通讯来说, 它们之间的比特率偏差不得超过 $\pm 2\%$ 。	FTOL_SYNCH	$< \pm 2\%$

表1.2: 从机振荡器相对于主机节点的偏差

联系方式: **H.-Chr. v. d. Wense**, 摩托罗拉有限公司, **Schatzbogen 7, D-81829 慕尼黑, 德国**  
电话: **+49 (89) 92103-882** E-Mail: **H.Wense@Motorola.com**



## 位定时要求

**LIN 物理层规范**  
版本 2.0  
2003年9月23日; P3

## 2. 位定时要求和同步程序

### 2.1 位定时要求

如无另外规定，本文件中的所有位时间都参考主机节点的位定时。

### 2.2 同步程序

同步场在其字节场里面有数据“0x55”。同步过程是基于模式下降沿之间的时间量度。现有的下降沿存在着2、4、6和8位时间，可以简单地计算基本位时间 $T_{\text{bit}}$ 。

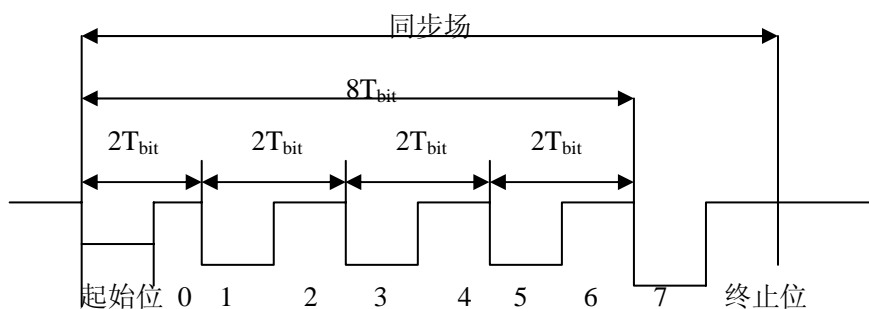


图2.1: 同步场

我们建议，测量两个下降沿之间的时间，即起始位和位7，并将得到的值除以8。将结果除8是将二进制的定时器值向LSB右移3位，将最低位四舍五入，校正即得到结果。

## 3. 总线驱动器/接收器

### 3.1 总体配置

总线驱动器/接收器是一个ISO 9141标准的增强设备。它包括双向LIN总线。这个双向总线连接每个节点的驱动器/接收器，并通过一个终端电阻和一个二极管连接到电池节点的正极 $V_{BAT}$ (见图3.1)。二极管可以在丢失电池、掉电的情况下，可以阻止ECU的电源供应不受总线的影响。ECU的电源供应是不受控制的。

要注意：LIN规范将电子控制单元（ECU）的外部电气连接电压作为参考电压，而不是将ECU 内部电压作为参考电压。当设计LIN 的收发器电路时，特别要考虑二极管的反向极性寄生电压降。

### 3.2 物理接口供应电压的定义

$V_{BAT}$ 表示控制单元连接器的供应电压。单元内的电子组件，其内部供应电压 $V_{SUP}$ 与 $V_{BAT}$ 不一样（见图3.1）。这可能是为了保护滤波器元件以及由于总线上的动态电压变化引起的。因此，需要考虑到LIN 适用的半导体产品。

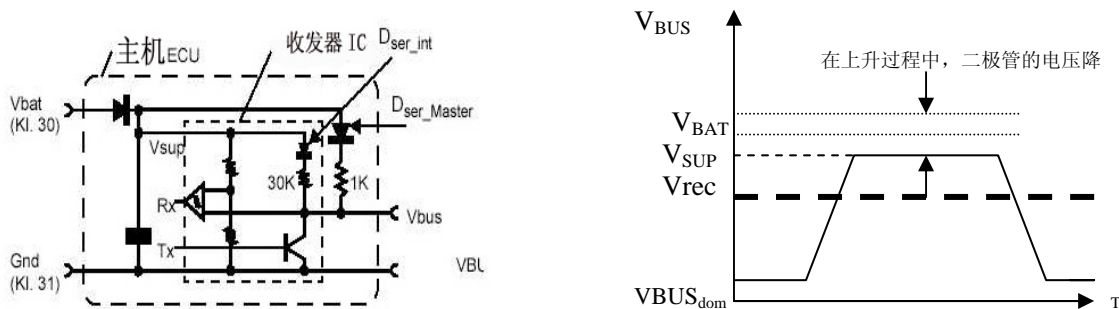


图3.1: 外部供应电压 $V_{BAT}$ 和内部供应电压 $V_{SUP}$ 的不同



## 3.3 信号规范

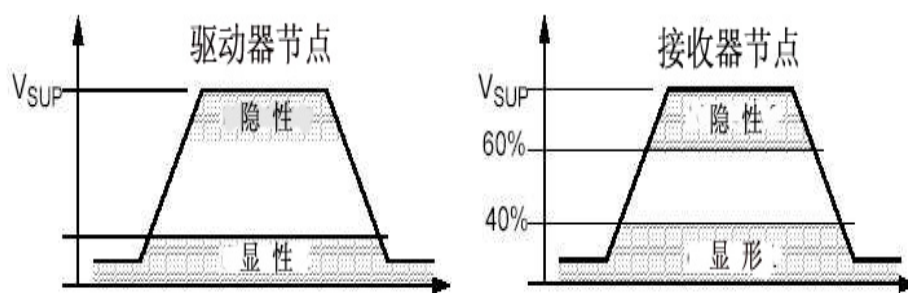


图3.2: 总线上的电压级

为了能够让比特正确输送和接收,需确保处于接收器位采样时间的信号有正确的电压级(显性或隐性)。

电压在供应过程中,需考虑到地面电压的转换和下降,以及在信号传播过程中因延误造成的不对称。

图3.3显示了会对LIN总线行为产生冲突的定时参数。

不同参数的最大和最小值见下文表格。



## 总线驱动器/接收器

LIN 物理层规范  
版本 2.0  
2003年9月23日; P6

定时图

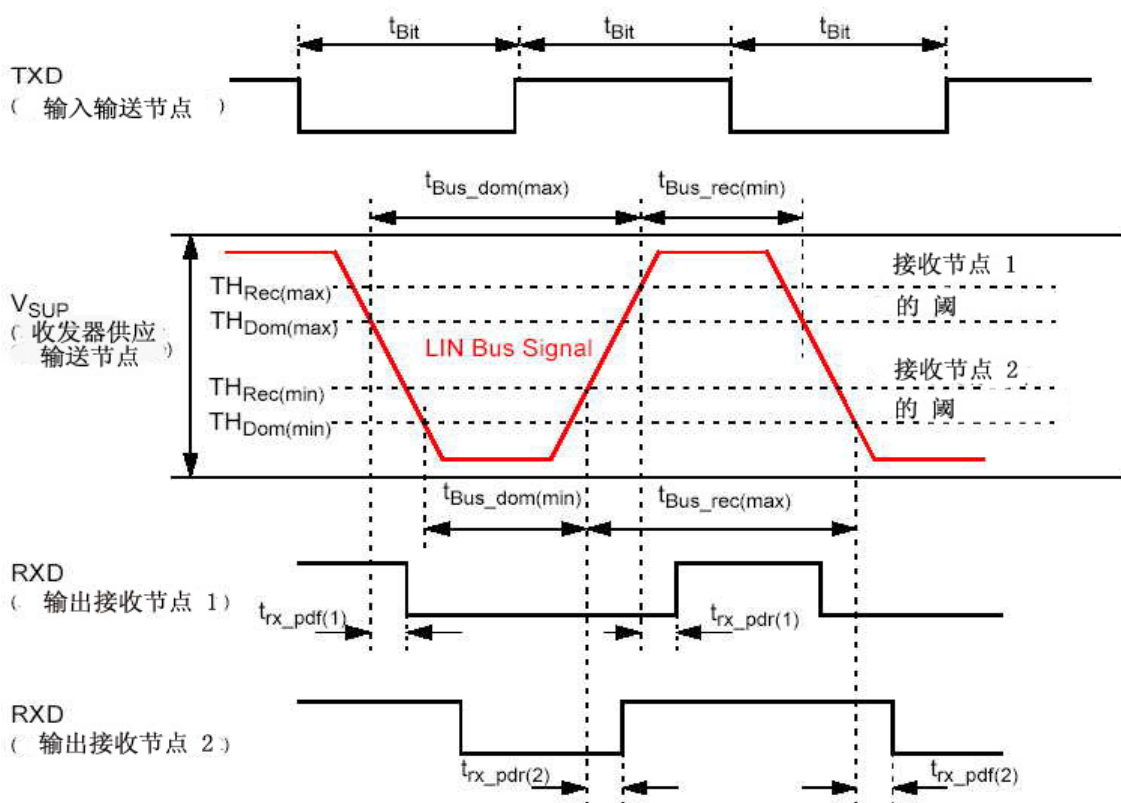


图3.3: 总线定时参数的定义



## 总线驱动器/接收器

**LIN 物理层规范**  
版本 2.0  
2003年9月23日; P7

LIN物理层的电气直流参数和终端电阻器分别见表3.1和表3.2。注意，由于在一个集成的电阻二极管网络中没有寄生的电流通路，所以要在总线和ECU内部电压（ $V_{SUP}$ ）之间形成一条寄生电流通道，比如通过ESD元件。

编号	参数	最小值	典型值	最大值	单位	备注/条件
3.1.1	$V_{BAT}^a$	8		18	V	工作电压范围
3.1.2	$V_{sup}^b$	7.0		18	V	供应电压范围
3.1.3	$V_{sup\_non-op}$	-0.3		40	V	器件不被破坏的电压范围，
3.1.4	$I_{BUS\_LIM}^c$	40		200	mA	驱动器显性状态的电流限制 启动驱动器 $V_{BUS} = V_{BAT\_max}^d$
3.1.5	$V_{BUS\_PAS\_dom}$	-1			mA	接收器的输入泄漏电流，包括 上拉电阻，具体见表3.2。 关闭驱动器 $V_{BUS} = 0V$ $V_{BAT} = 12V$
3.1.6	$V_{BUS\_NO\_rec}$				$\mu A$	关闭驱动器 $8V < V_{BAT} < 18V$ $8V < V_{BUS} < 18V$ $V_{BUS} \geq V_{BAT}$
3.1.7	$V_{BUS\_NO\_GND}$	-1			mA	控制单元不接地 $GND_{Device} = V_{SUP}$ $0V < V_{BUS} < 18V$ $V_{BAT} = 12V$ 控制单元不接地的情况不得 影响的其他网络的通讯。

表3.1: LIN物理层的电气直流电参数

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 总线驱动器/接收器

**LIN 物理层规范**

版本 2.0

2003年9月23日; P8

编号	参数	最小值	典型值	最大值	单位	备注/条件
3.1.8	IBUS			100	μA	VBAT已经断开 VSUP_Device = GND 0<VBUS<18V 节点需确保电流能够在该情况下流动,且总线仍然可以继续操作。
3.1.9	VBUSdom			0.4	Vsup	接收器的显性状态
3.1.10	VBUSrec	0.6			Vsup	接收器的隐性状态
3.1.11	VBUS_CNT	0.475	0.5	0.525	Vsup	$VBUS\_CNT = (V_{th\_dom} + V_{th\_rec})/2^e$
3.1.12	VHYS			0.175	Vsup	$VHYS = V_{th\_rec} - V_{th\_dom}$
3.1.13	VSerDiode	0.4	0.7	1.0	V	在上升通路上, 串行二极管 Dser_Master 和 Dser_int 的电压下降 (见图3.1)。
3.1.14	Vshift_BAT	0		10%	VBAT	VBAT-Shift
3.1.15	Vshift_GND	0		10%	VBAT	GND-Shift

表3.1: LIN物理层的电气直流电参数

- a. VBAT控制单元连接器的供应电源, 与电气组件来的内部供应电压VSUP, 可能会有所不同 (见章节3.2)。
- b. VSUP表示控制单元内部的收发器的供应电压, 可能会与控制单元的外部供应电压VBAT有所不同 (见章节3.2)。
- c. IBUS: 流入节点的电流。
- d. 收发器至少可以下拉40mA电流。流入节点的最大电流在直流电环境下不得超过200 mA, 以避免可能的损坏。
- e. Vth\_dom: 接收器从隐性LIN总线边到显性LIN总线边的阈。  
Vth\_rec: 接收器从显性LIN总线边到隐性LIN总线边的阈

编号	参数	最小值	典型值	最大值	单位	备注
3.2.1	Rmaster	900	1000	1100	Ω	串行二极管是必要的 (见图 3.1)
3.1.9	Rslave	20	30	600	KΩ	串行二极管是必要的

注:

所有定义参数是在-40℃—125℃的环境范围内得到的。

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 总线驱动器/接收器

**LIN 物理层规范**  
版本 2.0  
2003年9月23日; P9

### 3.5 电气交流电的参数

LIN物理层的电气交流电的参数见表3.3、表3.4和表3.5。参数的定义见图3.3。总线的电气交流电特性受到总线特性的强烈影响，见章节3.3。总线的时间常数  $\tau$ （以及总的电容）必须小心选择，以便能够在较糟的环境状况下执行正确的信号。

下列表格（表3.3）具体规定了在20.0 kBit/sec操作环境下的定时参数

编号	参数	最小值	典型值	最大值	单位	备注 / 条件
LIN 驱动器 总线负荷条件 $(C_{BUS}; R_{BUS}): 1nF; 1k\Omega / 6,8nF; 660\Omega / 10nF; 500\Omega$						
3.3.1	D1 (Duty Cycle 1)	0.396				$TH_{Rec(max)} = 0.744 \times V_{SUP};$ $TH_{Dom(max)} = 0.581 \times V_{SUP};$ $V_{SUP} = 7.0V \dots 18V; t_{Bit} = 50\mu s;$ $D1 = t_{Bus\_rec(min)} / (2 \times t_{Bit})$
3.3.2	D2 (Duty Cycle 2)			0.581		$TH_{Rec(min)} = 0.284 \times V_{SUP};$ $TH_{Dom(min)} = 0.422 \times V_{SUP};$ $V_{SUP} = 7.6V \dots 18V; t_{Bit} = 50\mu s;$ $D2 = t_{Bus\_rec(max)} / (2 \times t_{Bit})$

表3.3: LIN物理层驱动器电气交流电的参数（20.0 kBit/s）

为了改进EMC的性能，可以允许例外情况存在，如速度为10.4 kBit/sec，或速度为下文所列的值。具体见下表（表3.4）。该表格具体规定了在10.4 kBit/sec操作环境下的定时参数。

编号	参数	最小值	典型值	最大值	单位	备注 / 条件
LIN 驱动器, 总线负荷条件 $(C_{BUS}; R_{BUS}): 1nF; 1k\Omega / 6,8nF; 660\Omega / 10nF; 500\Omega$						
3.4.1	D3 (Duty Cycle 3)	0.417				$TH_{Rec(max)} = 0.778 \times V_{SUP};$ $TH_{Dom(max)} = 0.616 \times V_{SUP};$ $V_{SUP} = 7.0V \dots 18V; t_{Bit} = 96\mu s;$ $D3 = t_{Bus\_rec(min)} / (2 \times t_{Bit})$
3.4.2	D4 (Duty Cycle 4)			0.590		$TH_{Rec(min)} = 0.251 \times V_{SUP};$ $TH_{Dom(min)} = 0.389 \times V_{SUP};$ $V_{SUP} = 7.6V \dots 18V; t_{Bit} = 96\mu s;$ $D4 = t_{Bus\_rec(max)} / (2 \times t_{Bit})$

表3.4: LIN物理层的驱动器电气交流电参数（10.4 kBit/sec）

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 总线驱动器/接收器

**LIN 物理层规范**  
版本 2.0  
2003年9月23日; P10

专用集成电路（ASICs）应符合表3.3和/或表3.4的参数。如果两套表的参数都被执行了，那么应该根据总线比特率选择合适的模式。

编号	参数	最小值	典型值	最大值	单位	备注 / 条件
LIN 接收器, RXD 负荷条件 ( $C_{RXD}$ ): 20pF; (如果是开漏行为: $R_{pull-up} = 2.4k\Omega$ )						
3.5.1	$t_{rx\_pd}$			6	$\mu s$	接收器的传播延迟
3.5.2	$t_{rx\_sym}$	-2		2	$\mu s$	接收器传播延迟过程上边开和下降边的对称

表3.5: LIN

### 物理层的接收器电气交流电参数

LIN总线的EMC行为取决于信号形态。信号形态由回旋率和其他因素如 $di/dt$  和  $d^2V/dt^2$ 决定。请仔细选择信号的形态，因为一方面它可以降低信号发射，另一方面也能够允许将比特率提高至20 kBit/sec

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 总线驱动器/接收器

**LIN 物理层规范**  
版本 2.0  
2003年9月23日； P11

### 3.6 总线特性

总线信号上升和下降的最大旋转率在实际操作过程中，会受到典型总线收发器控制的旋转率限制。上升信号的最小旋转率由RC时间常数给定。因此，总线的电容应保持非常低，这样可以保持较低的波形非对称性。主机模块选择的电容要比从机模块大，因此，一旦网络变量存在多个节点编号时，大的电容就可以成为一个“缓冲器”。整个总线的电容 $C_{BUS}$ 可以用下面的方程（3.6.1）算出：

$$\text{方程 3.6.1: } C_{BUS} = C_{MASTER} + n \cdot C_{SLAVE} + C'_{LINE} \cdot LEN_{BUS}$$

RC 时间常数 “ $\tau$ ” 可以通过方程 (3.6.2)算出：

$$\text{方程 3.6.2: } \tau = C_{BUS} \cdot R_{BUS}$$

$$\text{方程 3.6.3: } R_{BUS} = R_{master} \parallel R_{SLAVE\ 1} \parallel R_{SLAVE\ 2} \parallel \dots \parallel R_{SLAVE\_n}$$

上述方程可以根据表3.6给定的参数得出结果。

			最小值	典型值	最大值	单位
3.6.1	总线总长度	$LEN_{BUS}$			40	m
3.6.2	总线总电容，包括从机电容和主机电容	$C_{BUS}$	1	4	10	nF
3.6.3	整个系统的时间常数	$\tau$	1		5	$\mu s$
3.6.4	主机节点的电容	$C_{master}$		220		pF
3.6.5	从机节点的电容	$C_{SLAVE}$		220	250	pF
3.6.6	线电容	$C'_{LINE}$		100	150	pF/ m

表3.6：总线特性和参数

$C_{master}$  和  $C_{SLAVE}$  定义了ECU连接器的总节点电容，包括物理总线驱动器（收发器）以及其他应用于LIN总线pin的所有元件，像电容器或保护电路。

LIN机群的节点数量不得超过16<sup>1</sup>。

注 1：在很糟糕的情况下，如果节点数量超过 16 个，网络全电阻可能会禁止无任何错误的通讯。每一个附加节点都会降低大约 3%的网络电阻（30 k $\Omega$   $\parallel$  ~1 k $\Omega$ 。）。

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 总线驱动器/接收器

**LIN 物理层规范**  
版本 2.0  
2003年9月23日； P12

### ESD/EMI的符合条件

半导体物理层设备必须遵守IEC 1000-4-2:1995的要求，预防因人体放电而产生的破坏。最少的放电电压级是 $\pm 2000V$ 。

#### 注：

在ECU连接器的汽车应用中，要求ESD的电压级可达到 $\pm 8000V$

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com





## **LIN 应用程序接口规范**

### **版本 2.0**

本规范是在“ASIS”的基础上形成的，不可以作为任何索赔的依据。

© LIN 协会 2003.

版权所有。未经授权的复印、演示、或对本文件其他方面的使用，都是违反法律和知识产权的。

LIN是一个注册商标®.

本文件的任何分发都登记在案。



## 介绍

**LIN 应用程序接口规范**  
版本 2.0  
2003年9月23日；P2

### 1. 介绍

LIN设备驱动程序可以在硬件或软件中执行。若在软件中执行程序，有多种语言可工选择。因此将LIN设备驱动程序直接与应用软件进行集成，是有可能的。

本文件定义了用C语言编程的软件——LIN设备驱动程序——的强制接口。因此，硬件设备和用其他语言编程的设备都不是标准化的。如果LIN设备驱动程序以其他语言形式出现，比如Ada，我们鼓励用户使用本文件介绍的概念，即使句法将会有所不同。

API被分成两部分：LIN内核API和LIN的诊断API；在LIN系统中，诊断的特性是可以选择的。

LIN的内核API覆盖的行为详见《LIN协议规范》内的定义；而LIN的诊断API覆盖的行为详见《LIN诊断和配置规范》。

### 操作概念

#### 1.1.1 系统生成

LDF文件（见《LIN配置语言规范》）通过工具解析，生成API和驱动程序模块。这就是系统的生成。

由于LDF文件只涉及LIN的机群，因此在系统生成过程中，需要更多的信息。不过这并不是LIN标准<sup>1</sup>的一部分。

#### 1.1.2 API

##### LIN的内核API

LIN的内核API使用的是应用程序和LIN内核之间的交互作用。该交互作用是基于信号的。这表示应用软件并不一定要干扰到帧和帧的输送。如果必要的话，现有的工具可以探测到某个具体帧的转移，见章节2.3。当然，控制LIN内核的API调用也是存在的。

两个版本包含了绝大多数的API调用；静态程序和动态程序。静态程序将信号的名称或接口嵌入程序名称里面，而动态程序则将此作为一个参数。至于该选择那个程序，这属个人口味问题。

注 1：LIN 开发工具销售商可以免费为您安装，以符合他们自己的工具链。



## 介绍

**LIN 应用程序接口规范**  
版本 2.0  
2003年9月23日；P3

### **LIN节点配置API**

LIN节点配置API是建立在报文帧的基础上的，也就是说，主机节点内的应用软件会调动一个API程序，该程序可以向指定的从机节点发送请求，并等待响应。从机节点LIN设备驱动程序会自动处理这个请求/响应。

### **LIN诊断运输层API**

LIN诊断运输层也是建立在报文帧的基础上的，但是，它的目的用途是作为报文帧的一个运输层，为LIN设备驱动程序以外的诊断报文剖析器提供服务。这是一个典型的ISO诊断模块。APIs有两个独特的方案，一个是未加工的API，它允许应用软件控制每一个被发送出去的帧的内容，另一个是已加工的API，它执行所有运输层的功能。

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 内核 API

**LIN 应用程序接口规范**  
版本 2.0  
2003年9月23日； P4

## 2 内核API

LIN的内核API有一套完整的功能，这些功能全部是在“赋予API一个独立的名称空间（name space）”的理论基础上成立的，其目的就是为了使现有软件上的冲突风险最小化。所有功能和类型将会有有一个前缀“l\_”（“L”的小写字母后面，再加一个下划线）。

LIN的内核可以定义下列几个类型：

. l\_bool  
. l\_ioctl\_op  
. l\_irqmask  
. l\_u8  
. l\_u16

为了获得效率，大多数功能都是静态功能（不需要参数，因为功能是通过信号、接口等等存在的）。

### 2.1 驱动器和机群管理

#### 2.1.1 l\_sys\_init

##### 原型

l\_bool l\_sys\_init (void);

##### 描述

l\_sys\_init 执行LIN内核的初始化

##### 返回

如果初始化成功，则结果为0

如果初始化不成功，则结果不为0

##### 注

用户在使用其他API功能前，l\_sys\_init 的调用是他在LIN内核内必须使用的第一个调用。

### 2.2 信号交互

#### 2.2.1 信号类型

信号有3中不同的类型：

l\_bool 代表一位信号；如果错误，则为0；如果不是错误，则不为0

l\_u8 代表大小在1-8个位的信号。

l\_u16 代表大小在9-16个位的信号。

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



内核 API

LIN 应用程序接口规范  
版本 2.0  
2003年9月23日； P5

2.2.2 零星信号的读取

动态原型

```
l_bool l_bool_rd (l_signal_handle sss);  
l_u8 l_u8_rd (l_signal_handle sss);  
l_u16 l_u16_rd (l_signal_handle sss);
```

静态执行

```
l_bool l_bool_rd_sss (void);  
l_u8 l_u8_rd_sss (void);  
l_u16 l_u16_rd_sss (void);  
方程中sss是信号的名称，比如，l_u8_rd_EngineSpeed ().
```

描述

读取并返回名称为sss的当前信号值。

2.2.3 零星信号的写入

动态原型

```
void l_bool_wr (l_signal_handle sss, l_bool v);  
void l_u8_wr (l_signal_handle sss, l_u8 v);  
void l_u16_wr (l_signal_handle sss, l_u16 v);
```

静态执行

```
void l_bool_wr_sss (l_bool v);  
void l_u8_wr_sss (l_u8 v);  
void l_u16_wr_sss (l_u16 v);  
方程中sss是信号的名称，比如，l_u8_wr_EngineSpeed (v).
```

描述

将sss指定的当前信号值设置为V。


2.2.4 字节数组读取

动态原型

```
void l_bytes_rd (l_signal_handle sss,  
                l_u8 start, /* first byte to read from */  
                l_u8 count, /* number of bytes to read */  
                l_u8* const data); /* where data will be written */
```

静态执行

```
void l_bytes_rd_sss (l_u8 start,  
                    l_u8 count,  
                    l_u8* const data);  
方程中 sss 是信号的名称，比如，l_bytes_rd_EngineSpeed (...).
```

	<h1>内核 API</h1>	<b>LIN 应用程序接口规范</b> 版本 2.0 2003年9月23日； P6
<p><u>描述</u></p> <p>读取并返回信号中被选择的字节的当前值，该信号的名称为sss。</p> <p>假设一个字节数组的长度是6个字节，编号是0-5。那么如果要读取该数组中的字节2和3，要求起始位是2（直接跳过字节0和1），并计数为2（读取字节2和3）。在这个例子中，字节2被读入数据[0]，字节3会被读入数据[1]。</p> <p>虽然可以选择让设备驱动程序不执行，但起始位和计数的总和不得大于整个字节数组的长度。</p> <p><b>2.2.5 字节数组写入</b></p> <p><u>动态原型</u></p> <pre>void l_bytes_wr (l_signal_handle sss,                 l_u8          start, /* first byte to write to */                 l_u8          count, /* number of bytes to write */                 const l_u8* const data); /* where data is read from */</pre> <p><u>静态执行</u></p> <pre>void l_bytes_wr_sss (l_u8          start,                     l_u8          count,                     const l_u8* const data);</pre> <p>方程中，SSS是信号的名称，如l_bytes_wr_EngineSpeed (..).</p> <p><u>描述</u></p> <p>将信号中被选择字节的当前值设置为指定的值，该信号的名称为SSS。</p> <p>假设一个字节数组的长度是7个字节，编号为0-6。如果要写入该数组中的字节3和4，那么要求起始位为3（跳过0、1和2），并计数为2（读取字节3和4）。在这个例子中，字节3从数据 [0]读出，字节4从数据[1]中读出。</p> <p><b>2.3 通知</b></p> <p>标记是节点内部的符号。它们可以使应用程序与LIN内核同步。标记使LIN内核自动设置的，只能被应用程序测试或清楚。</p> <p><b>2.3.1 l_flg_tst</b></p> <p><u>动态原型</u></p> <pre>l_bool l_flg_tst (l_flag_handle fff);</pre>		
<p>联系方式: <b>H.-Chr. v. d. Wense</b>, 摩托罗拉有限公司, <b>Schatzbogen 7, D-81829 慕尼黑, 德国</b> 电话: +49 (89) 92103-882 E-Mail: <b>H.Wense@Motorola.com</b></p>		

	<h1>内核 API</h1>	<b>LIN 应用程序接口规范</b> 版本 2.0 2003年9月23日； P7
<p><u>静态执行</u></p> <p><code>l_bool l_flg_tst_fff (void);</code> <code>fff</code>是标记的名称，如<code>l_flg_tst_RxEngineSpeed ()</code>.</p> <p><u>描述</u></p> <p>返回表明标记当前状态的C布尔值，该标记的名称为<code>fff</code>，即如果标记被清除后，返回0，如果没有被清除，则不为0。</p> <p><u>注：</u></p> <p>当关联物（信号或帧）被LIN模块更新后，会设置标记。</p> <p><b>2.3.2 l_flg_clr</b></p> <p><u>动态原型</u></p> <p><code>void l_flg_clr (l_flag_handle fff);</code></p> <p><u>静态执行</u></p> <p><code>void l_flg_clr_fff (void);</code></p> <p>信号的名称为<code>fff</code>，如<code>l_flg_clr_RxEngineSpeed ()</code>.</p> <p><u>描述</u></p> <p>将名称为<code>fff</code>的标记的当前值设置为0</p> <p><b>2.4 进度管理</b></p> <p><b>2.4.1 l_sch_tick</b></p> <p><u>动态原型</u></p> <p><code>l_u8 l_sch_tick (l_ifc_handle iii);</code></p> <p><u>静态执行</u></p> <p><code>l_u8 l_sch_tick_iii (void);</code></p> <p>是接口的名称，如 <code>l_sch_tick_MyLinIfc ()</code>.</p> <p><u>描述</u></p> <p><code>l_sch_tick</code>的功能是根据进度执行的。当根据进度表帧要被发送时，程序便会启动这个发送动作。当当前的进度表都被完成时，<code>l_sch_tick</code>又会从头开始执行。</p> <p>对每个有节点的接头来说，须内招网络配置文件规定的速度逐个调动 <code>l_sch_tick</code>。</p>		
<p>联系方式： <b>H.-Chr. v. d. Wense</b>, 摩托罗拉有限公司, <b>Schatzbogen 7, D-81829 慕尼黑, 德国</b> 电话： +49 (89) 92103-882 E-Mail: <b>H.Wense@Motorola.com</b></p>		



## 内核 API

**LIN 应用程序接口规范**  
版本 2.0  
2003年9月23日； P8

### 返回

如果l\_sch\_tick的下一个调动将在下一条进度条目里启动帧的发送，那么结果不为0。在这个例子中，返回值将是这个进度条目在整个进度表上的编号（从进度表的起始位置开始计数）。如果进度有N条条目，那么返回值的范围是1-N。如果l\_sch\_tick的下一个调动不会开始帧的发送，那么结果为0。

### 注

l\_sch\_tick可以在主机节点内使用。

调用l\_sch\_tick不仅仅可以启动下一个帧的发送，而且从之前l\_sch\_tick的调用开始，即在该接口的最后一个帧内，它还可以更新那些接收到的信号的信号值。

返回值的使用见l\_sch\_set 的注解。

### **l\_sch\_set**

#### 动态原型

```
void l_sch_set (l_ifc_handle      iii,  
               l_schedule_handle schedule,  
               l_u8               entry);
```

#### 静态执行

```
void l_sch_set_iii (l_schedule_handle schedule, l_u8 entry);
```

iii 是接口的名称，比如l\_sch\_set\_MyLinIfc (MySchedule1, 0).

#### 描述

设置下一个进度，进度后面是针对某个接口iii的l\_sch\_tick函数。一旦当前的进度达到下一个进度条目的入口点，新的进度就会被激活。

条目定义了新进度表上的起始入口点。如果进度表有N条条目，那么这个值的范围在0-N，如果条目是0或1，那么新的进度表将会从头开始。

#### 注：

l\_sch\_set 只可以在主机节点里面使用。

关于条目值的可能用途，它可以与l\_sch\_tick 的返回值联合，用一个进度表暂时中断另一个进度表。此外，条目值还可以跳回到中断前在被中断进度表上的位置。

预先定义的进度表，L\_NULL\_SCHEDULE，可以用来阻止 LIN 机群上的所有转移。

**联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国**  
**电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com**





2.5 接口管理

2.5.1 l\_ifc\_init

动态原型

void l\_ifc\_init (l\_ifc\_handle iii);

静态执行

void l\_ifc\_init\_iii (void);

iii 是接口的名称，如 l\_ifc\_init\_MyLinIfc ()。

描述

l\_ifc\_init 使名称为iii的控制器初始化，也就是说l\_ifc\_init设置了例如波特率之类的内核。调用l\_ifc\_init 时，其所设置的默认进度是 L\_NULL\_SCHEDULE；在这个程序里面，不发生任何帧的发送和接收。

注

所有接口按照它们的名称被列入了本地描述文件中。  
在使用其他与LIN API功能相关的接口——比如 l\_ifc\_connect () 或 l\_ifc\_rx ()——之前，l\_ifc\_init () 函数是用户必须操作的第一个调用。

2.5.2 l\_ifc\_connect

动态原型

l\_bool l\_ifc\_connect (l\_ifc\_handle iii);

静态执行

l\_bool l\_ifc\_connect\_iii (void);

iii是接口的名称，如如 l\_ifc\_connect\_MyLinIfc ()。

描述

调用l\_ifc\_connect会将接口iii 与LIN机群连接起来，并将帧头和数据输送到总线上。

返回

如果“连接操作”成功，则为0，如果“连接操作”失败，在不为0。

2.5.3 l\_ifc\_disconnect


动态原型



内核 API

LIN 应用程序接口规范  
版本 2.0  
2003年9月23日；P10

`l_bool l_ifc_disconnect (l_ifc_handle iii);`  
静态执行  
`l_bool l_ifc_disconnect_iii (void);`  
iii是接口的名称，如如 `l_ifc_disconnect_MyLinIfc ()`.  
描述  
调用`l_ifc_disconnect`将会断开接口iii 与LIN机群的连接，并终止接口与机群中其他节点的交互。  
返回  
如果“断开连接操作”成功，则为0，如果“断开连接操作”失败，在不为0。  
**2.5.4 l\_ifc\_goto\_sleep**  
动态原型  
`void l_ifc_goto_sleep (l_ifc_handle iii);`  
静态执行  
`void l_ifc_goto_sleep_iii (void);`  
iii是接口的名称，如如 `l_ifc_go_to_MyLinIfc ()`.  
描述  
通过启动特殊的go-to-sleep-mode-command （进入休眠模式命令），`l_ifc_goto_sleep` 会命令与接口相连接的机群上的所有从机节点进入休眠模式。见 2.5.10章节  
注  
`l_ifc_goto_sleep`只可以在主机节点中使用。  
**2.5.5 l\_ifc\_wake\_up**  
动态原型  
`void l_ifc_wake_up (l_ifc_handle iii);`  
静态执行  
`void l_ifc_wake_up_iii (void);`  
iii是接口的名称，如如 `l_ifc_wake_up_MyLinIfc ()`.  
描述  
该函数的调用可以输送LIN总线上一个 0xf0 的字节，即介于250 μs和 5 ms之间的主脉冲（取决于配置的比特率），见《LIN协议规范》第5.1章节

	<h1>内核 API</h1>	<b>LIN 应用程序接口规范</b> 版本 2.0 2003年9月23日； P11
<div><h2>2.5.6 l_ifc_ioctl</h2><p><u>动态原型</u></p><pre>l_u16 l_ifc_ioctl (l_ifc_handle iii, l_ioctl_op op, void *pv);</pre><p><u>静态执行</u></p><pre>l_u16 l_ifc_ioctl_iii (l_ioctl_op op, void *pv);</pre><p>iii是接口的名称， 如如 l_ifc_init_MyLinIfc (MyOp, &amp;MyPars).</p><p><u>描述</u></p><p>该函数控制协议和接口的特殊参数。请使用操作程序在op中所定义的名称为iii 的接口。指示器pv 指向一个可选择的参数块。</p><p>至于到底要支持哪个操作，这取决于接口的类型；程序员因此必须要参考目标绑定文件中关于特殊接口的文件。</p><p><u>注</u></p><p>参数块的判读取决于所选择的操作。有些操作程序并不需要这个参数块。在这种情况下，指示器pv可以被设置为NULL（空）。如果参数块与操作相关，那么它的模板则由接口决定，因此不必须要参考目标绑定文件上关于接口规格的信息。</p><h2>2.5.7 l_ifc_rx</h2><p><u>动态原型</u></p><pre>void l_ifc_rx (l_ifc_handle iii);</pre><p><u>静态执行</u></p><pre>void l_ifc_rx_iii (void);</pre><p>iii是接口的名称， 如l_ifc_rx_MyLinIfc ().</p><p><u>描述</u></p><p>当接口iii已经收到数据的一个字符的时候，函数便会被调用。</p><p>比如，当收到数据的一个字符时，该函数便可以从用户定义的由UART触发的中断处理器中调用出来。</p><p>在这个情况下，函数将会在UART控制寄存器上执行必要的操作</p><p><u>注</u></p><p>应用程序负责绑定中断并设置正确的接口操作（前提中断已经启用）。</p></div>		
<p><b>联系方式： H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国</b> <b>电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com</b></p>		



## 内核 API

**LIN 应用程序接口规范**  
版本 2.0  
2003年9月23日； P12

### 2.5.8 l\_ifc\_tx

#### 动态原型

void l\_ifc\_tx (l\_ifc\_handle iii);

#### 静态执行

void l\_ifc\_tx\_iii(void);

iii 是接口的名称，比如l\_ifc\_tx\_MyLinIfc ()。

#### 描述

当接口iii传输了一个字符的资料时，函数就会被调用。

比如，当已经输送了数据的一个字符时，该函数便可以从用户定义的由UART触发的中断处理器中调用出来。在这个情况下，函数将会在UART控制寄存器上执行必要的操作

#### 注意

本应用程序只负责绑定中断以及设置正确的接口操作器（如果使用了中断）。

如果输送与the l\_ifc\_rx函数调用连接时，函数在某些执行过程中可能为空。该信息在目标绑定文件中已经为用户进行描述了。

### 2.5.9 l\_ifc\_aux

#### 动态原型

void l\_ifc\_aux (l\_ifc\_handle iii);

#### 静态执行

void l\_ifc\_aux\_iii(void);

iii 是接口的名称，比如l\_ifc\_aux\_MyLinIfc ()。

#### 描述

该函数可以在从机节点中使用，以便能够与BREAK和SYNC字符同步。BREAK和SYNC字符是由iii指定接口上的主机发送的。

比如，该函数可以从用户定义的中断处理器中调用出来。该中断处理器是硬件pin（与接口iii连接）在侧边检测（flank detection）过程中产生的。

#### 注：

l\_sch\_set 只可以在从机节点里面使用。

该函数是与硬件紧密连接的；关于该函数的正确执行和使用，已经在目标绑定文件详细描述。

**联系方式： H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国**  
**电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com**



内核 API

LIN 应用程序接口规范  
版本 2.0  
2003年9月23日； P13

如果BREAK/SYNC的检测是在l\_ifc\_rx 函数中执行的，那么函数可能会空

2.5.10 l\_ifc\_read\_status

动态原型

l\_u16 l\_ifc\_read\_status (l\_ifc\_handle iii);

静态执行

l\_u16 l\_ifc\_read\_status\_iii(void);

iii是接口的名称，比如l\_ifc\_read\_status\_MyLinIfc ()。

描述

函数的调用可以返回一个16位的值，见表2.1。

表 2.1： l\_ifc\_read\_status 的返回值 (位 15 是 MSB, 位 0 是 LSB)。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
最后一个帧受保护标识符								0	0	0	0	进入休眠	溢出	成功转移	响应错误

如果从前一个l\_ifc\_read\_status的调用开始，由节点处理的一个（或多个）帧在帧响应阶段产生了错误，那么函数就会设置“响应错误”，比如校验和错误、帧错误，等等<sup>2</sup>。

如果从前一个l\_ifc\_read\_status的调用开始，一个（或多个）帧在处理过程无任何错误，那么函数就会设置“成功转移”。

如果从前一个l\_ifc\_read\_status的调用开始便有2个或2个以上的帧被处理，那么函数就会设置“溢出”。

如果从前一个l\_ifc\_read\_status的调用开始，收到了go-to-sleep-mode-command（进入休眠模式命令）的信息，那么函数就会设置“进入休眠”。

“最后一个帧的受保护标识符”指得是总线上最后一个被检测的受保护标识符，该标识符需在节点中被处理。如果函数设置了“溢出”，那么最后一个帧的受保护标识符的一个或多个值将会丢失；只有最新的值才可以被保留下来。

注 2：帧头中的一个错误会导致帧头无法被识别，从而导致帧被忽视。



## 内核 API

**LIN 应用程序接口规范**  
版本 2.0  
2003年9月23日；P14

### 注：

LIN从机节点在总线上沉寂4秒后，便会要求进入休眠模式。这可以通过应用程序监控状态位进行操作；如0读入持续1秒钟，这表示总线处于沉寂状态<sup>3</sup>。

### 案例

`l_ifc_read_status`的设计，可以使系统以比帧槽频率要低的频率读取，如每50个帧槽发送一次。在这例子中，最后一个帧的受保护标识符一点用处都没有。而“溢出”则可以检查总线交通是否按照正常状态运行，也就是说，“溢出”应该一直都处于设置状态。

不过，在每个帧槽时间之间，都可以调用`l_ifc_read_status`，并获得一个较为准确的错误统计表；如果您了解拓扑学，那么您可以看到转移失败的受保护标识符，并可以获得关于错误节点的一个较好的结论。

所提供的信息，特别是与`error_response`信号（见《LIN协议标准》第6.3章节）关联的信息，规定了非常详细的汽车OEM错误节点或配线的特殊记录。

### 操作注解

如果调用`l_ifc_read_status`两次后而无任何帧转移发生，那么函数将会在第二此调用时返回0。

在对帧的校验和进行处理和确认后，函数将会设置“成功转移”。

如果在帧响应处理过程中检测到了一个错误，那么函数将会设置“响应错误”。

如果成功转移或响应错误被设置时，那么函数就会同时设置“最后一个帧的受保护标识符”。

如果成功转移或响应错误中有一个已经被设置，且如果驱动器需要设置其中一个时，函数就会设置“溢出”。

当`go-to-sleep-mode-command`(进入休眠模式命令)已经被发送出去或尚未被发送出去的时候，主机节点操作便会在API调用被启动时可以设置“进入休眠”模式<sup>4</sup>。

注3：暗示了主机每秒必须与所有从机节点通讯一次；如果无任何其他要求，它至少要统计`receive_error`（接收错误）状态位。

注 4：为什么这么有弹性的原因是，主机节点操作不可被迫接收它自己输送的信号。

**联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国**  
**电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com**



## 内核 API

**LIN 应用程序接口规范**  
版本 2.0  
2003年9月23日； P15

### 2.6 用户提供的标注

用户必须提供一对函数，它们可以从LIN内核中调用出来，以便在某内核操作前，使所有控制器的中断操作终止，并在这些操作后，修复之前的状态。举例来说，这些功能可以在l\_sch\_tick 函数中使用。

#### 2.6.1 l\_sys\_irq\_disable

##### 动态原型

l\_irqmask l\_sys\_irq\_disable (void);

##### 描述

用户在操作函数时，必须获得一个状态；在该状态中，不会发生任何控制器中断操作。

#### 2.6.2 l\_sys\_irq\_restore

##### 动态原型

void l\_sys\_irq\_restore (l\_irqmask previous);

##### 描述

用户在操作函数时，必须重新回到由providedl\_irqmask previous 所确定的状态。



3. 节点配置

LIN 节点配置 API（应用程序接口）拥有一组函数；其目的是为 API 提供一个单独的名称空间，以便能够减少与现有软件发生冲突的风险。所有的函数和类型都有前缀 “ld-” (小写的 “LD” 和一个下划线)。

LIN 节点配置正确操作的一个要求是：激活的进度表需包含两个按先后次序排列的诊断帧（主机请求帧和从机响应帧）。如果主机并不担心响应问题（不推荐），那么在进度表中包含主机请求帧就足够了。

注释：  
LIN 节点配置 API 中的所有调动都只针对主机节点。从机节点自动控制节点配置。  
任何初始化需求都应由 LIN 内核模块 call l-sys-init 自动执行。

3.0.1 ld-is-ready

动态原型

l\_bool ld\_is\_ready (l\_ifc\_handle iii);

描述

如果指定接口的诊断模块已经为接受新命令做好了准备，那么该函数的程序返回则为真实的。这也意味着先前的命令已经完成，如：响应有效并可以被分析。

注释：  
只有在主机中才可以进行调用。

除非以前的调用已经完成，否则您无法发布下一个节点配置 API 调用，即：ld-is-ready 已经返回，且函数值是为真实的。

执行注释：  
如果API内部有强制调用，那么ld-is-ready则会被清除掉（除了ld-is-ready和ld-check-response之外的所有调用）。

当主机请求帧和从机响应帧完成之后，应该设置ld-is-ready（从机响应帧受到控制）。如果从机响应帧没有紧随在主机请求帧之后，或者如果从机不响应，当下一个帧槽正在被处理时，ld-is-ready也应该被设置。

在后一种情况下，模块应该记录请求失败的信息（以防应用程序调用ld-check-response）。





### 3.0.2 Id-check-response

#### 动态原型

```
l_u8 Id_check_response (l_ifc_handle iii,  
                        l_u8*      RSID,  
                        l_u8*      error_code);
```

#### 描述

本程序会返回最后一个节点配置调用的全部结果。RSID以及error\_code（错误码）也被返回，以便可以进行更加细致的分析。结果判读如下：

LD-SUCCESS	函数调用已经成功实行
LD-NEGATIVE	调用失败，可以通过错误码的解析获得更多信息。
LD-NO-RESPONSE	请求没有得到响应
LD-OVERWRITING	从机响应帧被另外的操作盖写，即：结果丢失 <sup>5</sup> 。

#### 注释：

只有在主机里面才能被调用。

#### 执行注释：

如果从机不响应，程序可能不会响应LD-SUCCESS。但是，RSID和error\_code的值可能会返回一个成功的回复。（这允许程序直接从响应RAM缓冲器中读取RSID和error\_code。）

### 3.03 Id-assign-NAD

#### 动态原型

```
void Id_assign_NAD (l_ifc_handle iii,  
                   l_u8      NAD,  
                   l_u16     supplier_id,  
                   l_u16     function_id,  
                   l_u8      new_NAD);
```

注释5：只有当机群同时使用接点配置以及诊断运输层时，才会发生此类情况，见章节4。



## 节点配置

**LIN API 规范**

版本 2.0

2003年9月23日； P18

### 描述

该函数的调用可以将所有与NAD匹配的从机节点的NAD（节点地址诊断）进行赋值。配合的所有从机节点，供应者识别码，以及操作识别码的NAD赋值。那些节点的新NAD将会是new-NAD。

### 注释：

只有主机才能进行调用。

在此调用中，可以使用LD-BROADCAST， LD-ANY-SUPPLIER 和/或 LD-ANY-FUNCTION（假使簇内所有的节点都有唯一的供应者/函数ID）

此调用的目的在于改变LIN簇内使用离架从节点或者再使用的接点构建的相冲突的NADS。

### **3.0.4 Id-assign-frame-id**

#### 动态原型

```
void ld_assign_frame_id (l_ifc_handle   iii,
                        l_u8           NAD,
                        l_u16          supplier_id,
                        l_u16          message_id,
                        l_u8           PID);
```

### 描述

此调用给在有地址 NAD 的从机节点内的帧的受保护的标识符，以及指定的供应者 ID 赋值。变化了的帧应该拥有指定的信息 ID 并且将在调用后象受保护的标识符一样获得 PID。

### 注释：

只有主机才能进行调用。

在此调用中，可以使用LD-BROADCAST， LD-ANY-SUPPLIER 和/或 LD-ANY-FUNCTION（假使簇内所有的节点都有唯一的供应者/函数ID）

### **3.0.5 Id-read-by-id**

#### 动态原型

```
void ld_read_by_id (l_ifc_handle   iii,
                   l_u8           NAD,
                   l_u16          supplier_id,
                   l_u16          function_id,
                   l_u8           id,
                   l_u8* const    data);
```



## 节点配置

**LIN API 规范**  
版本 2.0  
2003年9月23日; P19

### 描述

此调用要求与 NAD 一起选择的节点返回与 ld 参数相关的属性。当对 ld-is-ready 的下次调用返回正确时，根据要求，通过数据详细说明了的 RAM 区包含 1 到 5 字节的数据。

### 注释:

只有主机才能进行调用。

表3.1 展示了ld的可能值。

表3.1可以通过使用ld-read-by-id进行阅读

ld	解释
0	LIN 产品标识
1	序列号
2-15	保留
16-31	信息 ID 1.. 16
32-63	自定义
64-255	保留

### 执行注释

结果以 big-endian 方式（高位字节(序列中最重要值)先存放在低地址处的顺序）返回。little-endian 中央处理器交换字节，而不是 LIN 诊断驱动程序。（使用 big-endian 数据的原因是为了简化信息行程安排到一个 CAN 骨干网。

### 3.0.6 ld-conditional-change-NAD

#### 动态原型

```
void ld_conditional_change_NAD (l_ifc_handle iii,  
                                l_u8          NAD,  
                                l_u8          id,  
                                l_u8          byte,  
                                l_u8          mask,  
                                l_u8          invert,  
                                l_u8          new_NAD);
```

### 描述

如果节点属性能达到id，字节，电路模板，以及倒置指定的检验标准，调用就会改变NAD，请参考 LIN 诊断规范。

应该在0到31的范围内，请参考表3.1，并且字节在1到5的范围内（详细说明id中使用的字节）。电路模板，以及倒置的值应该在0到255之间。

注释: 只有主机才能进行调用。

**联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国**  
**电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com**



## 诊断传送层

**LIN API 规范**

版本 2.0

2003年9月23日；P20

### 4. 诊断传送层

LIN传送层API（应用程序接口）拥有一组基于为API提供一个单独的名字空间的函数，以减少与现有的软件发生冲突的风险。所有的函数和种类都将加上前缀“ld-”（紧接着一一条“底线”的小写字母“LD”）

要使用LIN诊断传送层API，就要求了解下面的协议。相关的信息可以在**LIN诊断和配置规范**中找到。

LIN诊断传送层旨在通过主机节点在CAN骨干网中的测试设备和LIN次机节点之间传递ISO诊断请求/响应。

由于CAN骨干网中的ISO PDUs与LIN诊断帧相当相似，于是提供了Raw API。Raw API是基于帧/PDU上的，需要依靠应用程序控制PCI信息。特点是：当源点是在基于CAN的ISO PDUs时，或者API需要很少的源点(RAM和CPU周期)时，这就相当的简单。

选择性的API是以信息为基础的。为指示器提供一个信息缓冲器，开始传递，LIN诊断驱动器就会进行包装和拆包，即：起作一个传送层的作用。这在从机节点中很有用，因为它们会对信息进行分解。

#### 注释

系统的行为在病理状况下是不明确的，而此时应用程序试图通过Raw API 和cooked API处理一个单帧。

#### 4.1 RAW API

##### 执行注释

Raw API 是基于传送 PDUs 的，它被用于 CAN 和 LIN 之间的网关 PDUs。如果网络速度不同，一个关于传送以及接受的 FIFO（先入先出(法)）函数是非常有用的。LIN 诊断模块的所有执行时，鼓励结合 FIFO 法以使得它们的大小在系统生成时可进行配置。

##### 4.1.1 ld-put-raw

###### 动态原型

```
void ld_put_raw (l_ifc_handle          iii,  
                 const l_u8* const    data);
```

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 诊断传送层

**LIN API 规范**

版本 2.0

2003年9月23日；P21

### 描述

调用排列字节数据规定的 8 字节传送

### 注释

数据在下一个适当的帧中被传送。（用于主机节点的主机请求帧和用于从机节点的从机响应帧）

### 执行注释

调用时，必须拷贝数据区，不允许只记住指示器。

## 4.1.2 Id-get-raw

### 动态原型

```
void Id_get_raw (l_ifc_handle iii,  
                l_u8* const data);
```

### 描述：

调用将最老的，被认为标准的诊断帧拷贝到数据规定的存储器中。

### 注释：

返回的数据是从适当的帧处接收到的（用于从机节点的主机请求帧和用于主机节点的从机响应帧）

### 执行注释：

如果接收程序是空的，不要采取任何行动。

## 4.1.3 Id-raw-tx-status

### 动态原型

```
l_u8 Id_raw_tx_status (l_ifc_handle iii);
```

### 描述：

调用返回 Raw frame 传送函数的地位。

LD-QUEUE- FULL            传送排列完全并且不能再接受更多的帧了。

LD-QUEUE- EMPTY        传送排列空间空闲

LD-TRANSFER-ERROR    传送过程中发生 LIN 协议错误，中断并且重新传送



## 诊断传送层

**LIN API 规范**  
版本 2.0  
2003年9月23日: P22

注释：

怎样终止传送不属于标准的一部分。

#### 4.1.4 Id-raw-rx-status

## 动态原型

```
1 u8 ld raw rx status (1 ifc handle iii);
```

描述:

调用返回 Raw frame 传送函数的地位。

LD-DATA-AVAILABLE 接收排列包含可读的数据

**LD-TRANSFER-ERROR** LIN 协议错误在传送过程中产生，终止，以及重新传送。

注释:

怎样终止传送不属于标准的一部分。

## 4.2 COOKED API

执行注释:

诊断信息的计划处理一次只能处理一条信息。因此，不需要执行信息 FIFO，也不需在应用程序缓冲器和要诊断模块的缓冲器中复制信息。

#### 4.2.1 Id-send-message

## 动态原型

```
void ld_send_message (l_ifc_handle      iii,  
                    l_u16              length,  
                    l_u8               NAD,  
                    const l_u8*        const data);
```

描述:

调用把数据和长度所规定的信息压缩在一个或多个诊断帧中。如果调用发生在一个主机节点中，帧就被发送到有地址的节点 **NAD**（从机节点把它们发送到主机）

注释:

在数据区，SID（或 RSID）应该是第一字节并且它应该包含在长度中。长度的范围必须是 1 到 4095 字节。

参数 **NAD** 不被使用于从机节点，而是内置的，以产生一个公用 API。

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 诊断传送层

### LIN API 规范

版本 2.0

2003年9月23日；P23

调用应该立刻返回调用，即：不要延缓到信息已经被传送出去，以及在对 **ld-tx-status** 的调用不返回 **LD-IN-PROGRESS** 的情况下，缓冲器不会发生改变。

当帧对随后的适当的帧页首产生响应时，数据就被传出（用于主机节点的主机请求帧和用于从机节点的从机响应帧）。

如果先前的传输仍然在进行时，调用是非法的。

#### 4.2.2 ld-receive-message

##### 动态原型

```
void ld_receive_message (l_ifc_handle    iii,
                        l_u16*          length,
                        l_u8*            NAD,
                        l_u8* const      data);
```

##### 描述

调用使得 **LIN** 诊断模块准备好接收一条信息并且把它储存在数据指向的缓冲器中。所有的调用长度应该详细说明允许的最大长度值。当接收完成后，长度就变为实际长度，**NAD** 变成信息内的 **NAD**（只适用于主机节点）

##### 注释

在数据区，**SID**（或 **RSID**）应该是第一字节并且它应该包含在长度中。长度的范围必须是 1 到 4095 字节。但是任何时候都不能超过调用原始设定值。

参数 **NAD** 不被使用于从机节点，而是内置的，以产生一个公用 **API**。

调用应该立刻返回调用，即：不要延缓到信息已经被传送出去，以及在对 **ld-rx-status** 的调用不返回 **LD-IN-PROGRESS** 的情况下，缓冲器不会发生改变。如果调用“太迟”。即：在信息传送开始后，调用应该等待下一条信息。

从随后的适当的帧处接收数据，（用于用于从机节点的主机请求帧和主机节点的从机响应帧）。

如果先前的传输仍然在进行时，调用是非法的。即：**ld-rx-status** 的调用返回 **LD-IN-PROGRESS**。

#### 4.2.3 ld-tx-status

##### 动态原型

```
l_u8 ld_tx_status (l_ifc_handle iii);
```

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国

电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 诊断传送层

**LIN API 规范**

版本 2.0

2003年9月23日；P24

### 描述：

调用返回对 Id-send-message 的最后调用的状态。可以返回如下值：

LD-IN-PROGRESS	传送还没有完成
LD-COMPLETED	传送已经成功完成（你可以进行新的 Id-send-message 调用）
LD-FAILED	传送以出错误告终。数据只传送了一部分（你可以进行新的 Id-send-message 调用）

### 注释

为了弄明白为什么传送失败，检查 LIN 核心中的状态管理函数。

#### **4.2.4 Id-rx-status**

##### 动态原型

```
l_u8 Id_rx_status (l_ifc_handle iii);
```

### 描述：

调用返回对 Id-receive-message 的最后调用的状态。可以返回如下值：

LD-IN-PROGRESS	接收还没有完成
LD-COMPLETED	接收已经成功完成，获得了所有的信息（长度，NAD，数据）（你可以进行新的 Id-receive-message 调用）
LD-FAILED	接收以出错误告终。数据只接收了一部分并且应该是不可信任的（你可以进行新的 Id-receive-message 调用）

### 注释：

为了弄明白为什么接收失败，检查 LIN 核心中的状态管理函数。





## 示例文件

**LIN API 规范**

版本 2.0

2003年9月23日；P25

### 5. 示例

在接下来的章节里，提供了一个示例，以展示 API 是怎样被使用的。除了 LIN 描写文外，还展示了 C 应用码。

#### 5.1 LIN 核心 API 用法

```
/******  
*           File:      hello.c  
*           Author:    Christian Bondesson  
*           Description: Example code for using the LIN API in a LIN master node  
*                       NOTE! This example uses the static API  
*/  
#include "lin.h"  
/******  
*   PROCEDURE :   l_sys_irq_restore  
*   DESCRIPTION :   Restores the interrupt mask to the one before the call to  
*                   l_sys_irq_disable was made  
*                   IN : previous - the old interrupt level  
*/  
void l_sys_irq_restore (l_imask previous)  
{  
/* Some controller specific things... */  
} /* l_sys_irq_restore */  
/******  
*   PROCEDURE :   l_sys_irq_disable  
*   DESCRIPTION :   Disable all interrupts of the controller and returns the  
*                   interrupt level to be able to restore it later  
*/  
l_imask l_sys_irq_disable (void)  
{  
/* Some controller specific things... */  
} /* l_sys_irq_disable */  
/******  
*   INTERRUPT :   lin_char_rx_handler  
*   DESCRIPTION :   LIN recieve character interrupt handler for the  
*                   interface named LIN_ifc  
*/  
void INTERRUPT lin_char_rx_handler (void)  
{  
/* Just call the LIN API provided function to do the actual work */
```

联系方式: **H.-Chr. v. d. Wense**, 摩托罗拉有限公司, **Schatzbogen 7, D-81829 慕尼黑, 德国**  
电话: **+49 (89) 92103-882** E-Mail: **H.Wense@Motorola.com**



## 示例文件

**LIN API 规范**  
版本 2.0  
2003年9月23日； P26

```
l_ifc_rx_MyLinIfc ();
} /* lin_char_rx_handler */
/*****
*   INTERRUPT :    lin_char_tx_handler
* DESCRIPTION :    LIN transmit character interrupt handler for the
*                  interface named LIN_ifc
*/
void INTERRUPT lin_char_tx_handler (void)
{
    /* Just call the LIN API provided function to do the actual work */
    l_ifc_tx_MyLinIfc ();
} /* lin_char_tx_handler */
/*****
*   PROCEDURE :    main
* DESCRIPTION :    Main program... initialization part
*/
void main (void)
{
    /* Initialize the LIN interface */
    if (l_sys_init ())
    {
        /* The init of the LIN software failed */
    }
    else
    {
        l_ifc_init_MyLinIfc (); /* Initialize the interface */
        if (l_ifc_connect_MyLinIfc ())
        {
            /* Connection of the LIN interface failed */
        }
        else
        {
            /* Connected, now ready to send/receive set the normal
             * schedule to run from beginning for this specific interface */
            l_sch_set_MyLinIfc (MySchedule1, 0);
        }
    }
    start_main_application (); /* Ready with init, start actual applic */
} /* main */
/* 10 ms based on the minimum LIN tick time, in LIN description file... */
void main_application_10ms (void)
```

```
{  
  /* Do some application specific stuff... */
```

联系方式: **H.-Chr. v. d. Wense**, 摩托罗拉有限公司, **Schatzbogen 7, D-81829 慕尼黑, 德国**  
电话: **+49 (89) 92103-882** E-Mail: **H.Wense@Motorola.com**



## 示例文件

**LIN API 规范**  
版本 2.0  
2003年9月23日； P27

```
/* Just a small example of signal reading and writing */
if (l_flg_tst_RxInternalLightsSwitch ())
{
    l_u8_wr_InternalLightsRequest (l_u8_rd_InternalLightsSwitch());
    l_flg_clr_RxInternalLightsSwitch ();
}
/* In-/output of signals, do not care about the return value, as we
   * will never switch schedule anyway... */
(void) l_sch_tick_MyLinIfc();
} /* main_application_10ms */
```

### 5.2 LIN 描写文件

```
/* *****
 *          File:  hello.ldf
 *          Author:  Christian Bondesson
 *          Description:  The LIN description file for the example program
 */

LIN_description_file ;
LIN_protocol_version = "2.0";
LIN_language_version = "2.0";
LIN_speed = 19.2 kbps;
Nodes {
    Master: CEM, 5 ms, 0.1 ms;
    Slaves: LSM;
}
Signals {
    InternalLightsRequest: 2, 0, CEM, LSM;
    InternalLightsSwitch: 2, 0, LSM, CEM;
}
Frames {
VL1_CEM_Frm1: 1, CEM {
    InternalLightsRequest, 0;
}
VL1_LSM_Frm1: 2, LSM {
    InternalLightsSwitch, 0;
}
}
Schedule_tables {
    MySchedule1 {
        VL1_CEM_Frm1 delay 15 ms;
        VL1_LSM_Frm1 delay 15 ms;
    }
}
```

```
}  
Signal_encoding_types {  
    Dig2Bit {  
        logical_value, 0, "off";  
        logical_value, 1, "on";  
        logical_value, 2, "error";  
        logical_value, 3, "void";  
    }  
}  
Signal_representations {  
    Dig2Bit: InternalLightsRequest, InternalLightsSwitch;  
}
```



# **LIN**

## **节点能力语言规范**

版本2.0

本规范是在“ASIS”的基础上形成的，不可以作为任何索赔的依据。

© LIN 协会 2003.

版权所有。未经授权的复印、演示、或对本文件其他方面的使用，都是违反法律和知识产权的。

LIN是一个注册商标®.

本文件的任何分发都登记在案。

## 1 引言

LIN 节点能力语言旨在能够描述从机节点在机器能识别的标准化的句法情况下的可能性。在未来的几年里，预先完成的离架从节点的实用性有望增加。如果它们都附加有节点能力文件，就有可能生成 LIN 规范包(请参考 LIN 配置语言规范)和主机节点的初始化代码 1

如果任何的 LIN 簇的设置和配置都完全是自动的，那么将朝 LIN 即插即用的功能发展迈出一大步。换言之，在 LIN 簇中使用分布式节点就好比将带有物理设备的中央处理器单节点直接连接到节点上一样简单。

### 即插即用 workflow

图 1.1 显示了 LIN 簇分裂在三个区域的发展：设计，调试，和 LIN 物理体系。此说明集中在设计阶段。

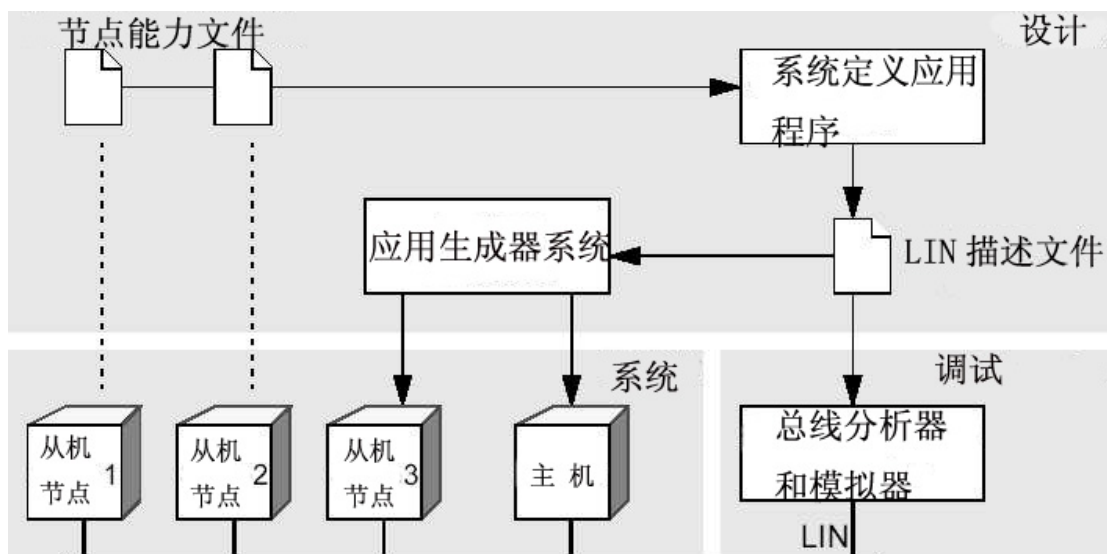


图 1.1: LIN 簇的发展

### 1.1.1 系统生成

LIN 簇的核心描述文件是指节点能力文件 (LDF)。在此文件基础上，就有可能生成簇内所有节点的通信驱动器。此过程被称为系统生成。在此簇类的所有帧以及信号都会被显示。

注释 1: 代码应该设置簇，比如：重新设置相冲突的标识符。



## 节点能力文件 定义

**LIN 节点能力**  
版本 2.0  
2003年9月23日；P3

### 1.1.2 系统定义

创建 LDF 文件的过程被称为系统定义。当你在设置一个全新的簇的时候，编写 LDF 文件（手写或利用计算机帮助）是给你的簇的通讯下定义的一种有效方法。

但是，当你拥有现有的从机节点并且想从头开始给它们创建一个簇的时候，就不那么方便了。如果自定义系统包含了节点地址冲突或者帧标识符冲突，这就情况得更加明显了。

通过接受节点能力文件（NCF），包括现有的从机节点，系统定义步骤会是自动的：向你系统定义应用程序内的方案添加 NCF 文件，就会生成 LDF 文件。

如果你也想创造新的从机节点，（图 1 中的从机节点 3），此过程将变得更加复杂。操作步骤依赖于使用的系统定义应用程序，而系统定义应用程序不属于 LIN 规范。在生成 LDF 前，需要一种有用的应用程序以进行额外信息的输入。（在任何情况下，都可能为不存在的从机节点编写虚拟的 NCF 文件，因此它被包括在内）。

值得注意的是，生成的 LDF 文件反映了设置的网络。在进行簇通信之前，最初位于节点或者帧之间的任何冲突都必须得到解决。

### 1.1.3 调试

调试和节点模拟是在通过系统定义而生成的 LDF 文件的基础上实现的。因此，应该进行监测，正如 LIN 规范早期版本一样。

主机模拟增加了要求，即：簇的设置必须是没有冲突存在的。因此，模拟应用程序必须能够读取通过系统定义应用程序生成的重置数据。





# 节点能力文件 定义

**LIN 节点能力**  
版本 2.0  
2003年9月23日； P4

## 2. 节点能力文件定义

node\_capability\_file ;

<language\_version>

[<node\_definition>]

### 2.1 全局定义

全局定义数据定义此文件的一般属性。

#### 2.1.1 节点能力语言版本号定义

<language\_version> ::=

**LIN\_language\_version** = char\_string ;

应该在“0.01”到“99.99”范围内。此规范对 2.0 版本进行了描述。

### 2.2 节点定义

<node\_definition> ::=

```
node <node_name> {  
    <general_definition>  
    <diagnostic_definition>  
    <frame_definition>  
    <status_management>  
    (<free_text_definition>)  
}
```

<node\_name> ::= identifier

如果一节点能力文件包含不止一个节点，文件内节点的名称应该是唯一的。标明的节点应该被看作物理节点实例的组（模板）。节点定义的属性将在如下的进行详细说明。

### 2.3 一般定义

<general\_definition> ::=

**general** {

**LIN\_language\_version** = <protocol\_version> ;

**supplier** = <supplier\_id> ;

**function** = <function\_id> ;

**variant** = <variant\_id> ;

**bitrate** = <bitrate\_definition> ;

(**volt\_range** = real\_or\_integer, real\_or\_integer ;)

(**temp\_range** = real\_or\_integer, real\_or\_integer ;)

(**conformance** = char\_string ;)

一般定义显示那些详细说明了簇的一般兼容性的属性。

所有任选子句只适用于**LIN2.0**，也就是说，它们不能用于**LIN 1.2/1.3**。

联系方式：**H.-Chr. v. d. Wense**，摩托罗拉有限公司，**Schatzbogen 7, D-81829 慕尼黑，德国**  
电话： +49 (89) 92103-882 E-Mail: **H.Wense@Motorola.com**



## 节点能力文件 定义

**LIN 节点能力**  
版本 2.0  
2003年9月23日; P5

### 2.3.1 LIN 协议版本号的定义

`<protocol_version> ::= char_string ;`

此定义详细说明了节点使用的协议。它应该在“0.01”和“99.99”的范围内。发布时，版本属于2.0版本。

### 2.3.2 LIN 产品标识

`<supplier_id> ::= integer`

`<function_id> ::= integer`

`<variant_id> ::= integer`

supplier-id 为 16 位元，被分配到 LIN 协会成员。Function-id 是通过厂商分配从而使其唯一的 16 位元。最后，Variant-id 是详细说明变量的 8 位值（请参考<LIN 诊断和配置规范>2.4 部分）。

### 2.3.3 比特率

`<bitrate_definition> ::=`

`automatic (min <bitrate>) (max <bitrate>) |`

`select {<bitrate> [, <bitrate>]} |`

`<bitrate>`

可能存在三种比特率：

- 自动比特率 节点可以采用在数据传送总线上使用的任何合法的比特率。如果添加了 words min 和/或者 words max, 可以使用源于/等于所提供的比特率的任意比特率。
- 选择比特率 如果列出的比特率中的比特率被使用，节点就能够探测到，否则就会彻底中止运转。
- 固定比特率 只能使用一种比特率

标准化的离架节点的制造商被鼓励建造自动节点，因为这给簇建造者很大的机动性。

`<bitrate> ::= integer`

固定比特率作为整数被列入清单，范围是 1000 到 2000 字节/秒。

### 2.3.4 非网络参数

Volt-range 和 temp-range 的选择规范详细说明了节点 i Volt 和 celius 所分别允许的最小值和最大值。最后，一致性详细说明了，如果从机节点通过了一致性测试程序，就应该是“LIN2.0”或“没有”。

## 2.4 诊断定义

`<diagnostic_definition> ::=`

`diagnostic {`

`NAD = integer (, <integer>) ;`

`[ P2_min = integer ms; ]`

`[ ST_min = integer ms; ]`

`[ support_sid {<sid> ([, <sid>]) } ; ]`

`[ max_message_length = integer ; ]`

`}`

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



# 节点能力文件 定义

**LIN 节点能力**  
版本 2.0  
2003年9月23日； P6

诊断定义详细说明了诊断和设置的属性。**NAD** 属性详细说明了最初的节点地址，它应该与 2.3.2 节的 **LIN 诊断和配置规范** 相一致。如果给定两个值，从机将动态的选择一个基于物理属性范围内的值。

**P2-min** 详细说明了将准备好回应的节点的主机请求帧和紧随的从机回应帧之间的最短时间。

**ST-min** 详细说明了在 **multi-PDU** 回应情况下，两从机回应帧之间的最短时间。换言之，它只能适用于诊断传输层。默认为 0ms.

**PDU** 回应，换言之，它只适用于诊断传输层。默认为 0ms.

最大信息长度属性仅适用于诊断传输层。它消息说明了诊断信息的最大信息长度。默认值：4095

**Support-sid** 罗列了节点支持的所有 **SID** 值。默认值：{0xb1, 0xb2}.

## 执行注释

未来的此规范版本可能会给诊断定义添加新的属性。目的在于遵循结构 〈属性〉=值或者 〈属性〉{〈实体目录〉}

## 2.5 帧的定义

<frame\_definition> ::=


```
frames {  
  [ <single_frame> ]  
}
```

所有公布的帧应该在此声明中列出

```
<single_frame> ::=  
<frame_kind> <frame_name> {  
  <frame_properties>  
  (<signal_definition>)  
}  
<frame_kind> ::= publish | subscribe  
<frame_name> ::= identifier
```

注释2： 诊断帧和自定义的帧除外

联系方式： **H.-Chr. v. d. Wense**, 摩托罗拉有限公司, **Schatzbogen 7, D-81829 慕尼黑, 德国**  
电话： +49 (89) 92103-882 E-Mail: **H.Wense@Motorola.com**

	<div>节点能力文件</div> <div>定义</div>	<div>LIN 节点能力</div> <div>版本 2.0</div> <div>2003年9月23日； P7</div>
<p>根据如上说明，对每个发表了的帧进行了公布。帧名称是帧的象征<sup>3</sup>名字。当然，公布了的帧应该从帧种类公布开始。</p> <p><b>2.5.1 帧属性</b></p> <pre>&lt;frame_properties&gt; ::= message_ID = integer ; length = integer ; ( min_period = integer ms ; ) ( max_period = integer ms ; ) ( event_triggered_message_ID = integer ; )</pre> <p>Message-Id 详细说明了帧的信息标识符（0 到 0xFFFF）以及帧长的长度（1 到 8）。 Min-period 和 max-period 的可选择值被用来指导生成进度表的应用程序。二者的值以毫秒计算。 event-triggered-message-ID 也是可选择的，它为公布事件触发帧提供了二级信息的 ID。</p> <p>注释：当一个帧也被事件触发时，会存在几种限制情况，请参考 <b>LIN 协议规范</b></p> <p><b>2.5.2 信号定义</b></p> <pre>&lt;signal_definition&gt; ::= signals {     [&lt;signal_name&gt; { &lt;signal_properties&gt; } ] } &lt;signal_name&gt; ::= identifier</pre> <p>所有的帧（除诊断帧外）都传送信号，这已经根据信号的定义作了说明。</p> <pre>&lt;signal_properties&gt; ::= &lt;init_value&gt; ; size = integer ; offset = integer ; (&lt;encoding&gt; ; ) &lt;init_value&gt; ::= init_value = integer   init_value = { integer ([, integer ]) }</pre> <p>Init-value 详细说明了从伏特量开始到出版应用程序的第一个集合为止的用于信号的值。其大小是为信号预定的字节数。偏移量详细说明了信号在帧内的位置（从帧内中的第一字节开始的偏移量中的字节量）。</p> <p>注释 3：系统定义应用程序可以直接使用此名字代替物理帧，除非簇内使用了不止一个阐明了的节点程序。</p> <div>}</div>		
<div>联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国</div> <div>电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com</div>		



# 节点能力文件 定义

**LIN 节点能力**  
版本 2.0  
2003年9月23日; P8

对于字节排列，大小和偏移量二者都必须是 8 的倍数。编码不用于字节排列。

注释：

如果一个大小为 8 或者 16 的信号是有一个或两个元件的字节排列或一个标量信号，描述的唯一方法就是通过分析 init-value。换言之，paranthesis 对区别排列和标量值至关重要。

## 2.5.3 信号编码类型的定义

编码的目的旨在提供信号的表示法和缩放比例属性。

```
<encoding> ::=  
    encoding <encoding_name> {  
        [<logical_value> |  
         <physical_range> |  
         <bcd_value> |  
         <ascii_value>]  
    }  
<encoding_name> ::= identifier  
<logical_value>  ::= logical_value, <signal_value> (<text_info>);  
<physical_range> ::= physical_value, <min_value>, <max_value>, <scale>,  
<offset> (<text_info>);  
<bcd_value>      ::= bcd_value ;  
<ascii_value>    ::= ascii_value ;  
<signal_value>   ::= integer  
<min_value>      ::= integer  
<max_value>      ::= integer  
<scale>          ::= real_or_integer  
<offset>         ::= real_or_integer  
<text_info>      ::= char_string
```

信号值，最小值和最大值应该在 0 到 65535 的范围内。最大值应该大于或者至少等于最小值。如果原始的值在最大值和最小值定义的范围，物理值应该按如下 (1) 方式计算。

$$\text{物理值} = \text{数值范围} * \text{原始的值} + \text{偏移量} \quad (1)$$

## 2.6 状态管理

```
<status_management> ::=  
    status_management {  
        error_response = <published_signal>;  
    }  
<published_signal> ::= identifier
```

状态管理部分详细说明了主机节点应该监测哪个公布的信号，以了解从机节点是否向预期的方式运行。

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 节点能力文件 定义

**LIN 节点能力**  
版本 2.0  
2003年9月23日； P9

注释：

状态管理与一般定义是分开的，以永远保持公布先于使用。换言之，信号在被指明为误差的回应信号前必须进行公布。

### 2.7 纯文字的定义

`<free_text_definition> ::=`

```
free_text {  
    <anything_but_right_curly_paranthesis>  
}
```

纯文字是在必要情况时，在系统定义应用程序下生成帮助文字，和限制规定等的。此内容的唯一局限在于 “}” 字符可能不会被使用到。

注释：

在纯文字定义时提供的推荐信息是：

- 节点用途以及实体世界交互，比如：电动机转速，能量消耗等。
- 节点可用性
- 背离 LIN 标准，即：未实现的功能性

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 句法总览

**LIN 节点能力**  
版本 2.0  
2003年9月23日；P10

### 3. 句法总览

使用修订的 BNF（巴克斯范式）对句法进行描述。总结请参看如下表格 3.1

表 3.1：此文件中使用的 BNF 句法

符号	含义
::=	在::=左边的名称是通过使用其右边的句法进行表达的
<>	用来表示后面将详细说明了的物体
	垂直条表示选择。垂直条的左边或者右边将出现
黑体	黑体的文本的保留是由于它是一个预定字，或者由于它是强制的标点
[]	方括号内的文本应该一次或多次出现
()	圆括号内的文本是可选择的，即：出现一次或者不出现
字符串	任何的字符串用引号围住，“象这样”
标识符	标识符典型的用途是个物体命名。标识符应该遵循正常的 C 规则
整数	整数可以是十进制的（第一位范围是 1 到 9）和十六进制（前面加上 0x）
实数或整数	实数或整数。实数经常以十进制为基础并且有一个内含的小数点

在使用这种句法的文件内，可以对任何部分进行注释。注释句法与 C++的注释句法一样。其中，从//到行末的任何部分以及用分隔符包围的部分应该被忽略。

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



## 示例文件

**LIN 节点能力**  
版本 2.0  
2003年9月23日; P11

### 4. 示例文件

```
node_capability_file;
LIN_language_version = 2.0;
node step_motor {
    general {
        LIN_protocol_version = 2.0;
        supplier = 0x0011;
        function = 0x1234;
        variant = 1;
        bitrate = automatic max 10400;
        volt_range = 8.0, 15.0;
    }
    diagnostic {
        NAD = 1, 3;
        P2_min = 40 ms;
        support_sid {0xb0, 0xb1, 0xb2};
    }
    frames {
        publish node_status {
            message_ID = 0x1001;
            length = 2;
            min_period = 10 ms;
            max_period = 100 ms;
            signals {
                state {init_value = 0; size = 8; offset = 0;}
                error_bit {init_value = 0; size = 1; offset = 8;}
            }
        }
    }
    subscribe control {
        message_ID = 0x1002;
        length = 1;
        max_period = 100 ms;
    }
    signals {
        command {init_value = 0; size = 8; offset = 0;
            encode position {physical_value 0, 199, 1.8, 0, deg;}; }
    }
}
status_management { error_response = error_bit; }
free_text {
```



The step\_motor command signal shall be in the range 0 to 199, or  
the command will be ignored.

}  
}



# **LIN**

## **配置语言规范**

### **版本2.0**

本规范是在“ASIS”的基础上形成的，不可以作为任何索赔的依据。

© LIN 协会 2003.

版权所有。未经授权的复印、演示、或对本文件其他方面的使用，都是违反法律和知识产权的。

LIN是一个注册商标®.

本文件的任何分发都登记在案。



## 引言

**LIN 配置语言**  
版本 2.0  
2003年9月23日；P2

# 1 引言

使用本文件中所描述的配置语言是为了生成一个LIN描述文件。该LIN描述文件描述了完整的LIN通讯网，同时包含了监控该网络所必需的所有信息。如果某个或某些节点无法获得，这些信息足以对这个进行有限仿真。

LIN描述文件可作为一个组件使用，为组成LIN通讯网的电子控制单元编写软件。该文件已经定义了一个应用程序接口，具体请参阅《**LIN API规范**》，其目的是为不同的应用程序提供一个统一的LIN通讯网访问端口。但是LIN描述文件中未涉及应用程序的功能特性。

LIN描述文件的语句十分简单，可以通过手工输入，但是我们鼓励用户使用基于计算机应用的工具来输入。正如《**LIN节点能力语言规范**》中所描述的那样，节点能力文件向我们提供一个（几乎）可以自动生成LIN描述文件的方法。我们同时也提供了在LIN机群开发过程中的一个工作流程实例。



## 2 LIN 描述文件的定义

<LIN\_description\_file> ::=  
LIN\_description\_file ;  
<LIN\_protocol\_version\_def>  
<LIN\_language\_version\_def>  
<LIN\_speed\_def>  
<Node\_def>  
(<Node\_composition\_def>)  
<Signal\_def>  
(<Diag\_signal\_def>)  
(<Dynamic\_frame\_def>)  
<Frame\_def>  
(<Sporadic\_frame\_def>)  
(<Event\_triggered\_frame\_def>)  
(<Diag\_frame\_def>)  
<Node\_attributes\_def>  
<Schedule\_table\_def>  
(<Signal\_groups\_def>)  
(<Signal\_encoding\_type\_def>)  
(<Signal\_representation\_def>)

LIN描述文件的总体语法如上所示。

### 2.1 全局定义

全局定义数据定义了LIN机群的一般特性。

#### 2.1.1 LIN协议版本号的定义

<LIN\_protocol\_version\_def> ::=

**LIN\_protocol\_version** = char\_string ;

必须在“0.01”到“99.99”的范围内。截至发布时，这个版本号是2.0。

#### 2.1.2 LIN语言版本号的定义

<LIN\_language\_version\_def> ::=

**LIN\_language\_version** = char\_string ;

必须在“0.01”到“99.99”的范围内。本规范描述了版本2.0。

#### 2.1.3 LIN速度的定义

<LIN\_speed\_def> ::=

**LIN\_speed** = real\_or\_integer kbps ;

必须在1.00到20.00千比特/秒的范围内

### 2.2 节点的定义

节点的定义章节为主机节点标识了所有参与节点的名称，并指明了时基和时基误差。本模块中的所有定义构成了一个节点标识符组，而本组中的所有标识符都是唯一的。



# LIN 描述文件 定义

LIN 配置语言  
版本 2.0  
2003年9月23日； P4

## 2.2.1 参与节点

<Node\_def> ::=

Nodes {

Master:<node\_name>, <time\_base> ms, <jitter> ms ;

Slaves:<node\_name>([, <node\_name>]) ;

}

<node\_name> ::= identifier

在节点标识符组中，所有的**node\_name**标识符必须是唯一的。

**Master**后面的**node\_name**标识符规定了主机节点的名称。

<time\_base> ::= real\_or\_integer

**time\_base**值规定了主机节点所产生的最大帧转移时间所需的时基。时基必须以毫秒来计。

<jitter> ::= real\_or\_integer

时基误差值规定了从时基起点到帧头输送起点（暂停信号的递降限界）之间，最大与最小延迟的差异。时基误差值需以毫秒来计。（时基和时基误差的更多使用说明，请查阅**Schedule\_tables**的子集定义。

## 2.2.2 节点的属性

节点属性提供了关于某个单节点行为的所有信息

<Node\_attributes\_def> ::=

Node\_attributes {

[<node\_name> {

LIN\_protocol = <protocol\_version> ;

configured\_NAD = <diag\_address> ;

(product\_id = <supplier\_id>, <function\_id>, <variant> ; )

(response\_error = <signal\_name> ; )

(P2\_min = <real\_or\_integer> ms ; )

(ST\_min = <real\_or\_integer> ms ; )

(configurable\_frames {

[ <frame\_name> = <message\_id> ; ]

})

}}

}

<node\_name> ::= identifier

<protocol\_version> ::= 1.2 | 1.3 | 2.0

<supplier\_id> ::= integer

<function\_id> ::= integer

<variant> ::= integer

<message\_id> ::= integer

所有任选子句只适用于**LIN2.0**，也就是说，它们不能用于**LIN 1.2/1.3**。

联系方式：H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话： +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



# LIN 描述文件 定义

LIN 配置语言  
版本 2.0  
2003年9月23日；P5

所有的`node_name`标识符都必须存在于节点标识符组中，表示一个从机节点。`supplier_id`必须在0-0x7FFE的范围内，`function_id`在0-0xFFFE的范围内，变量在0-255的范围内，`message_id`在0-0xFFFE的范围内。

`<signal_name> ::= identifier`

所有的`signal_name`标识符都必须存在于信号标识符组中，表示一个1位标准信号，请参阅2.3.1节。信号应由指定的节点发布。详情请参阅《LIN协议规范》中的状态管理。

`<diag_addr> ::= integer`

`diag_addr`为指定的节点规定了诊断地址，范围在1至127之间，详情请查阅《LIN诊断与配置规范》。在解决所有的机群冲突后，它将规定一个唯一的节点诊断地址（NAD），以供节点使用。也就是说，每个节点诊断地址在机群内都是唯一的。

可配置的帧需罗列出所有由节点和他们关联信息ID所处理过的帧，本节只适用于LIN2.0版本中的节点（不适用于LIN1.3）

## 2.2.3 节点结构定义

LDF文件从通讯的角度描述了节点的功能性；作为缺省值，这种“功能性节点”最好是实体（真实的）节点。但是，可以将这些实体节点表述为若干个功能性节点的合成体。这一子句的目的是使一个单主机节点软件可以在不进行任何变动的情况下操作多节点配置。

`<Node_composition_def> ::=`

```
composite {  
  [ configuration <configuration_name> {  
    [<composite_node> {  
      <functional_node> ([ , <functional_node> ])  
    }]  
  }]  
}
```

`<configuration_name> ::= identifier`

`<composite_node> ::= identifier`

`<functional_node> ::= identifier`

如果要使用本子句，必须将所有合成节点列入2.2.1节和2.2.2章节中；同时，只有功能节点才可以发布或认购这些信号和帧（2.3.1节和2.4.2节）。

在节点标识符组中，所有的`composite_node`标识符和`functional_node`标识符都必须是唯一的。

实体机群是根据某个`configuration_name`用数据建立起来的。



## 2.3 信号定义

信号定义模块定义了机群中所有信号的名称和特性。本模块中的所有定义组成了一个信号标识符组，而本组中的所有标识符必须是唯一的。

### 2.3.1 标准信号

`<Signal_def> ::=`

**Signals {**

`[<signal_name>: <signal_size>, <init_value>, <published_by>`

`[, <subscribed_by>];]`

**}**

`<signal_name> ::= identifier`

信号标识符组中所有的`signal_name`都必须是唯一的。

`<signal_size> ::= integer`

`signal_size`标识符规定了信号的尺寸。标量信号的范围在1至16位之间，而字节数组信号的值为8, 16, 24, 32, 40, 48, 56或64。

`<init_value> ::= integer | { integer ([ , integer ] ) }`

`init_value`规定了信号值，所有的认购器节点在没有收到载波帧之前都需要使用这个信号值。认购节点将发出同一个初始信号值，直到应用程序更新了信号。当然，字节数组信号采用的是向量再现法。

注：

如果信号是一个尺寸为8或16的由一至两个元件组成的字节数组或是一个标量信号，认购这一信号的唯一办法是对`init_value`进行缝隙，也就是说括弧对于区分数组信号和标量信号是非常重要的。

`<published_by> ::= identifier`

`<subscribed_by> ::= identifier`

`published_by`标识符和`subscribed_by`标识符必须都存在于节点标识符组中。

### 2.3.2 诊断信号

`<Diagnostic_signal_def> ::=`

**Diagnostic\_signals {**

`MasterReqB0:8,0;`

`MasterReqB1:8,0;`

`MasterReqB2:8,0;`

`MasterReqB3:8,0;`

`MasterReqB4:8,0;`

`MasterReqB5:8,0;`

`MasterReqB6:8,0;`

`MasterReqB7:8,0;`

`SlaveRespB0:8,0;`



# LIN 描述文件 定义

LIN 配置语言  
版本 2.0  
2003年9月23日； P7

```
SlaveRespB1:8,0;  
SlaveRespB2:8,0;  
SlaveRespB3:8,0;  
SlaveRespB4:8,0;  
SlaveRespB5:8,0;  
SlaveRespB6:8,0;  
SlaveRespB7:8,0;  
}
```

由于发布者和认购者的信息在此处不适用，因此诊断信号在LIN描述文件中有一个独立的模块。

### 2.3.3 信号组

组的定义是LIN1.3的一个特点。不赞成使用信号组；下列句法定义的使用不会影响LIN2.0机群。

```
<Signal_groups_def> ::=  
Signal_groups {  
    [<signal_group_name>:<group_size> {  
        [<signal_name>,<group_offset> ;]  
    }]  
}  
<signal_group_name> ::= identifier  
<group_size> ::= integer  
<signal_name> ::= identifier  
<group_offset> ::= integer
```

### 2.4 帧定义

帧定义模块定义了机群中所有帧的名称及其特性。这一模块中的定义构成了一个帧标识符组（它们的符号名称）和相关的帧标识符组（LIN帧标识符）。这两个标识符组中的标识符都必须是唯一的。

#### 2.4.1 动力帧id

```
<Dynamic_frame_def> ::=  
dynamic_frames { <frame_id> [, <frame_id> ] }
```

动力帧定义用于定义那些不具有唯一性的帧标识符，也就是说多个帧可以使用同一个帧标识符，参阅2.4.2节。

#### 2.4.2 无条件帧

```
<Frame_def> ::=  
Frames {  
    [<frame_name>:<frame_id>,<published_by>(<frame_size> ) {  
        [<signal_name>,<signal_offset>;]  
    }]  
}
```





# LIN 描述文件 定义

LIN 配置语言  
版本 2.0  
2003年9月23日； P8

<frame\_name> ::= identifier

帧定义组中的frame\_name 标识符都必须是唯一的。

<frame\_id> ::= integer

frame\_id标识符规定了标识符的号码，其范围在0至59或62。帧标识符组中的所有帧标识符都必须是唯一的，除非是那些在动力帧定义模块中列出的标识符，请参阅章节2.4.1

<published\_by> ::= identifier

published\_by标识符必须存在于节点标识符组中。

<frame\_size> ::= integer

frame\_size 是一个可选项，它将帧的尺寸规定在1至8个位之间。如果frame\_size 规格不存在，帧的尺寸将根据表2.1的帧标识符来确定。尽管frame\_size具有向后兼容的特性，但是我们鼓励您提供帧的尺寸值，即使这个值与默认值相匹配。

表2.1: 默认帧长

标识符范围	帧长
0 - 31 (0x1f)	2
32 (0x20) - 47 (0x2f)	4
48 (0x30) - 63 (0x3f)	8

<signal\_name> ::= identifier

signal\_name标识符必须存在于信号标识符组中。

根据为某一个帧设定的published\_by标识符的规定，这个帧定义里的所有信号必须由同一个节点来发布。

<signal\_offset> ::= integer

signal\_offset值定义了信号在帧内部最不重要的比特位置。这个值的变化范围在1至(8 \* frame\_size - 1)之间。信号中最不重要的比特是最先被传输的。

## 实例

下表 2.2向我们演示了，在一个有4字节数据场的帧中，一个10比特的信号被打包的过程。

S中的LSB在偏移量16处，MSB在偏移量15处。注意，随着字节的传输，偏移量的数值不断移动（首先移动的是LSB）

表 2.2: 信号打包

字节0								字节1								字节2								字节3							
																S	S	S	S	S	S	S	S	S	S						
0								7	8							15	16							23	24						31
最早传输																最晚传输															

联系方式: H.-Chr. v. d. Wense, 摩托罗拉有限公司, Schatzbogen 7, D-81829 慕尼黑, 德国  
电话: +49 (89) 92103-882 E-Mail: H.Wense@Motorola.com



在帧内进行信号打包的唯一一个准则就是让信号穿越最大一个字节的边界。

### 2.4.3 零星帧

```
<Sporadic_frame_def> ::=  
Sporadic_frames {  
    [<sporadic_frm_name>:<frame_name>(<frame_name>);]  
}
```

<sporadic\_frm\_name> ::= identifier

在帧标识符组中，所有的sporadic\_frm\_name标识符必须具有唯一性。

<frame\_name> ::= identifier

所有的frame\_name标识符必须存在于帧标识符组中，并指向无条件帧。当有多个发布过的帧需要被传输时，用户应选择位于列表中的第一个帧。

所有的frame\_name标识符都必须由主机节点发布或与一个event\_trig\_frm\_name相联系。此外，他们不能跟sporadic\_frm\_name一样被直接包含在同一个进度表里。

### 2.4.4 事件触发帧

```
<Event_triggered_frame_def> ::=  
Event_triggered_frames {  
    [<event_trig_frm_name>:<frame_id>(<frame_name>);]  
}
```

<event\_trig\_frm\_name> ::= identifier

帧标识符组中所有event\_trig\_frm\_name 标识符必须具有唯一性。

<frame\_id> ::= integer

frame\_id的规定了帧标识符数值的变化范围为0至59，在帧标识符组中，每个帧的标识符必须具有唯一性。

<frame\_name> ::= identifier

所有的frame\_name标识符必须存在于帧标识符组中，并指向无条件帧。如果产生碰撞，应传输frame\_name列表将会根据发布表的循序，取代event\_trig\_frm\_name被率先输送。

注1：如果信号是根据字节排列和/或者他们没有超越字节的边界，那么用基于软件的节点对信号进行打包/解包都会更为有效。



## LIN 描述文件 定义

**LIN 配置语言**  
版本 2.0  
2003年9月23日； P10

使用一个event\_trig\_frm\_name来发布frame\_name标识符，必须遵循下列各条：

- 他们必须使用同一个求校验和模型，也就是同一个LIN协议标准（LIN2.0或LIN1.3）
- 他们必须由机群中的不同从机节点来发布。
- 他们的长度必须相同，范围为1至8个字节。
- 帧的第一个比特不能运载任何信号。
- 他们不能和event\_trig\_frm\_name被直接包含在同一个进度表里。

### 注

帧的第一个字节携带关联帧的受保护标识符，因此，不能移作他用。

### 2.4.5 诊断帧

<Diag\_frame\_def> ::=

Diagnostic\_frames {

MasterReq : 60 {

MasterReqB0,0;

MasterReqB1,8;

MasterReqB2,16;

MasterReqB3,24;

MasterReqB4,32;

MasterReqB5,40;

MasterReqB6,48;

MasterReqB7,56;

}

SlaveResp : 61 {

SlaveRespB0,0;

SlaveRespB1,8;

SlaveRespB2,16;

SlaveRespB3,24;

SlaveRespB4,32;

SlaveRespB5,40;

SlaveRespB6,48;

SlaveRespB7,56;

}

}

MasterReq和SlaveResp保留帧的名称对诊断帧进行标识；他们在帧标识符组中必须具有唯一性。



# LIN 描述文件 定义

**LIN 配置语言**  
版本 2.0  
2003年9月23日； P11

按照《LIN协议规范》的规定，MasterReq帧有固定的ID，60 (0x3c)，以及一个固定的尺寸（8字节）。MasterReq帧只能由主动节点发出。

按照《LIN协议规范》的规定，SlaveResp帧有固定的ID，61 (0x3d)以及一个固定的尺寸（8字节）。SlaveResp帧只能由指定的从机节点发出，请查阅《LIN诊断和配置规范》。从机节点的选择是根据章节2.2.2中所规定的诊断地址来决定的。

## 2.5 进度表定义

与之前的版本相比，进度表略有改动：改动的原因是为了在不需要主机应用程序干涉的情况下可以认购和操作更为复杂，变动更为频繁的机群。

<Schedule\_table\_def> ::=

**Schedule\_tables** {

  [<schedule\_table\_name> {

    [<command> **delay** <frame\_time> **ms** ;]

  }]

}

<schedule\_table\_name> ::= identifier

所有进度表标识符组中的schedule\_table\_name标识符都必须具有唯一性。

<command> ::=

<frame\_name> |

**MasterReq** |

**SlaveResp** |

**AssignFrameId** { <node\_name>, <frame\_name> } |

**UnassignFrameId** { <node\_name>, <frame\_name> } |

**AssignNAD** { <old\_NAD>, <new\_NAD>, <supplier\_id>, <function\_id> } |

**FreeFormat** { <D1>, <D2>, <D3>, <D4>, <D5>, <D6>, <D7>, <D8> }

命令语句定义了帧槽中所做的变动。提供一个帧名可以转移指定的帧。

MasterReq 和SlaveResp可以像2.4.5章节中的帧一样被定义，或者，如果省略了这个定义子句，就会自动被定义。这些帧的内容是通过诊断和配置API来提供的，具体请参阅《LIN应用程序借口规范》。

**AssignFrameId**产生了一个Assign\_frame\_id主请求帧，其内容取决于下列参数：NAD（节点诊断地址）、supplier\_id和message\_id是从<node\_name>的属性中获得的，参阅章节2.2.2，而protected\_id是从<frame\_name>的帧定义中获得的，参阅章节2.4。本帧中的所有数据都是固定的，是在处理LDF文件过程中确定的。



LIN 描述文件  
定义

LIN 配置语言  
版本 2.0  
2003年9月23日；P12

如果事件触发帧的报文标识符被赋予一个帧受保护标识符（PID），那么与之相关的无条件帧必须先被赋予一个受保护的标识符。

**UnassignFrameId** 产生了一个**Assign\_frame\_id**主机请求帧，其内容像**AssignFrameId**一样，也是取决于参数的，但是不同的是，受保护的标识符是**0x40**，也就是说，标识符**0**的两个奇偶校验位无效。这可用于终止动态赋值的标识符的接受/传输（将标识符用于其他帧的情况下是十分必要的，见2.4.1章节）

**AssignNAD**产生一个**Assign\_NAD**主机请求帧，其内容直接源自参数。这个帧的所有数据都是固定的，这些数据是在处理LIN描述文件的过程中确定的。

最后，**FreeFormat**发送一个有8个数据字节的固定主机请求帧。这个帧可用于发布用户明确的固定的帧。

<frame\_name> ::= identifier

**frame\_name**标识符必须存在于帧标识符组中。如果这个**frame\_name**指向一个事件触发帧或零星帧，那么相关的无条件帧可能就无法被用于同一个进度表中。

<frame\_time> ::= real\_or\_integer

**frame\_time**规定了帧槽的延续时间。它必须比最大可允许的帧传输时间要长，同时，根据2.2.1节的要求，它必须是主机节点时基值的整数倍。**frame\_time** 值必须以毫秒来计。

进度表的选择必须由主机应用程序来控制。进度表之间的转换必须正好赶在帧时间（当前正在传输的帧）完成后进行，具体参阅《LIN应用程序借口规范》。

实例

图 2.1 是与进度表VL1\_ST1相对应的时间线。在此图中， **time\_base** 被设定为**5ms**（参阅2.2.1节）

```
Schedule_tables {
  VL1_ST1 {
    VL1_CEM_Frm1 delay 15 ms;
    VL1_LSM_Frm1 delay 15 ms;
    VL1_CPM_Frm1 delay 15 ms;
    VL1_CPM_Frm2 delay 20 ms;
  }
}
```



# LIN 描述文件 定义

LIN 配置语言  
版本 2.0  
2003年9月23日; P13

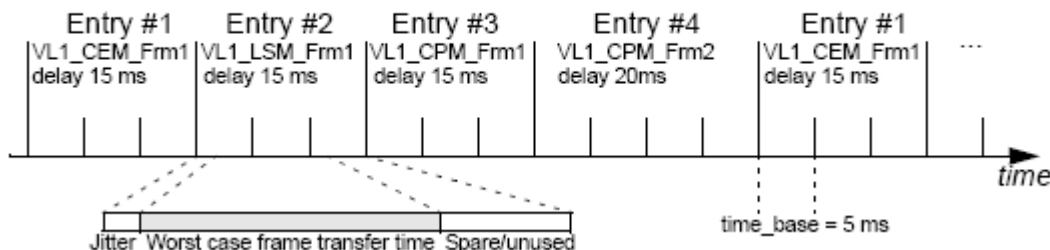


图2.1: VL1\_ST1进度表的时间线

为每个进度条目所规定的延迟时间必须长于时基误差和传输最慢的帧传输时间。

## 2.6 附加信息

以下各分节提供的附加信息，尽管无法改变LIN机群的行为，但是通过使用总线探测工具，可以提供通讯流量再现的线索。所有定义段都是可选项。

### 2.6.1 信号编码类型定义

本信号编码类型为了再现信号并测定它们的特性。尽管本信息可以在LIN节点中自动生成API校准程序，但是这种API程序需要很强的节点来支持。这个信号编码类型定义的作用主要体现在总线流量探测工具中，它可以通过一种很容易的方法再现所记录的流量。

```
<signal_encoding_type_def> ::=
```

```
Signal_encoding_types {
```

```
  [<signal_encoding_type_name> {
```

```
    [<logical_value> |
```

```
    <physical_range> |
```

```
    <bcd_value> |
```

```
    <ascii_value>]
```

```
  }
```

```
}
```

```
<signal_encoding_type_name> ::= identifier
```

信号编码类型标识符组中的所有Signal\_encoding\_type\_name都必须具有唯一性。

```
<logical_value> ::= logical_value,<signal_value>(<text_info>) ;
```

```
<physical_range> ::= physical_value,<min_value>,<max_value>,<scale>,<offset>(<text_info>) ;
```

```
<bcd_value> ::= bcd_value ;
```

```
<ascii_value> ::= ascii_value ;
```

```
<signal_value> ::= integer
```



# LIN 描述文件 定义

LIN 配置语言  
版本 2.0  
2003年9月23日; P14

<min\_value> ::= integer  
<max\_value> ::= integer  
<scale> ::= real\_or\_integer  
<offset> ::= real\_or\_integer  
<text\_info> ::= char\_string

signal\_value, min\_value和max\_value都必须在0至65535的范围内。

max\_value必须大于或等于min\_value, 如果原始值在最小和最大值限定的范围内, 那么常规值就需根据等式(1)来计算。

常规值=数值范围\*原始值+偏移 (1)

## 实例

V\_battery信号是一个八位信号, 具体参看图2.2, 也就是说在12V附近, 分辨率高, 同时有3个超出范围的特殊值。

```
Signal_encoding_types {  
    power_state {  
        logical_value, 0, "off";  
        logical_value, 1, "on";  
    }  
    V_battery {  
        logical_value, 0, "under voltage";  
        physical_value, 1, 63, 0.0625, 7.0, "Volt";  
        physical_value, 64, 191, 0.0104, 11.0, "Volt";  
        physical_value, 192, 253, 0.0625, 13.0, "Volt";  
        logical_value, 254, "over voltage";  
        logical_value, 255, "invalid";  
    }  
}
```

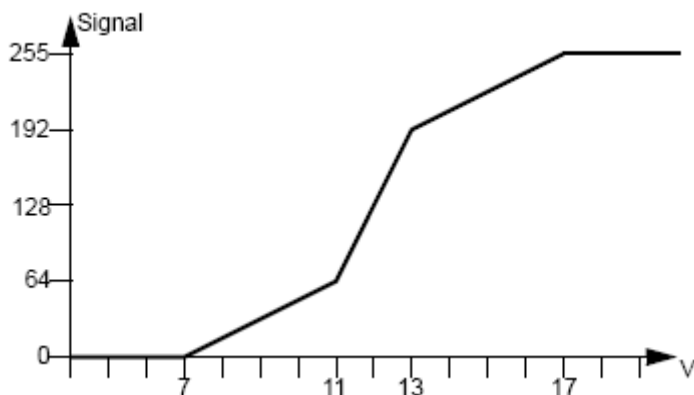


图2.2: V\_battery再现



## LIN 描述文件 定义

**LIN 配置语言**  
版本 2.0  
2003年9月23日； P15

### 2.6.2 信号再定义

这个信号再定义可以将信号与相对应的信号编码类型联系起来。

```
<Signal_representation_def> ::=
```

```
Signal_representation {
```

```
    [<signal_encoding_type_name>:<signal_name> ([,<signal_name>]);]
```

```
}
```

```
<signal_encoding_type_name> ::= identifier
```

**signal\_encoding\_type\_name**标识符必须存在于信号编码类型标识符组中。

```
<signal_name> ::= identifier
```

**signal\_name**标识符必须存在于信号标识符组中，每个信号可能只与一个

**signal\_encoding\_type\_name**相联系，而且可能无法套入一个**signal\_group\_name**中。





LIN 描述文件  
定义

LIN 配置语言  
版本 2.0  
2003年9月23日；P16

3 句法概述

本句法是使用修订过的BNF（巴科斯-诺尔范式）来描述的，下表3.1对此作了概述。

表3.1：本文件中所使用的巴科斯-诺尔范式语句

符号	含义
::=	::=符号表示，其左侧的名称由其右侧的语句来表述。
<>	对后续语句中所指定的词进行标注。
	竖线表示可选择。竖线左侧或右侧的语句可以选择使用。
<b>粗体</b>	粗体表示限定——加粗体的部分本身是限定词，或者是标点符号。
[]	方括号中的语句将出现一次或多次。
()	圆括号中的内容为可选项，也就是说，将会出现一次，或从不出现。
Char-string	用“像这样”标出来的字符串。
identifier	一个标识符。典型的用法是为事物命名。标识符必须遵循“用于变量定义的常规 C 标准”。
integer	一个整数。整数可以是十进制的（第一个数字在 1 至 9 之间），也可以是十六进制的（带前缀 0x）。
Real_or_integer	一个实数或整数。实数通常是十进制的，并带有小数点。

文件中使用本句法的地方都允许加入注释。注释符的句法同C++，从//到一个语句的结尾，以及包含在/\*中的语句，\*/分隔符应省略。