国际 ISO
标准 17987-2

第一版
2016-08-15

道路车辆 — 本地互连网络（LIN）—

第 2 部分：
传输协议和网络层服务

道路车辆 — 本地互联网（LIN）— 第 2 部分：传输协议和网络服务层

Reference number
ISO 17987-2:2016(E)

受版权保护的文档

ISO 版权局
章。德布兰多内特 8 • CP 401
CH-1214 瑞士日内瓦游标卡尺
电话。 +41 22 749 01
11 传真 +41 22 749 09
47 版权所有@iso.org w
ww.iso.org

前言

ISO（国际标准化组织）是各国标准机构（ISO成员机构）的全球性联合会。制定国际标准的工作通常由ISO技术委员会进行。每个对已成立技术委员会的主题感兴趣的成员机构都有权在该委员会中派代表。与ISO联络的政府和非政府国际组织也参与这项工作。ISO与国际电工委员会（IEC）在所有电工标准化事务上密切合作。

ISO/IEC 指令第 1 部分描述了制定本文档所用的程序以及其进一步维护的程序。特别要注意不同类型的 ISO 文档所需的不同批准标准。本文档是根据 ISO/IEC 指令第 2 部分的编辑规则起草的（请参阅www.iso.org/directives ） 。

请注意，本文档的某些元素可能属于专利权的主题。ISO 不负责识别任何或所有此类专利权。在文档开发过程中识别的任何专利权的详细信息将在简介中和/或 ISO 收到的专利声明列表中列出（请参阅www.iso.org/patents ） 。

本文件中使用的任何商品名称仅为方便用户而提供的信息，并不构成认可。

有关 ISO 特定术语和与合格评定相关的表达含义的解释，以及有关 ISO 在技术性贸易壁垒 (TBT) 中遵守世界贸易组织 (WTO) 原则的信息，请访问以下 URL： www.iso.org/iso/foreword.html 。

负责该文件的委员会是 ISO/TC 22、*道路车辆*、SC 31、*数据通信*。

ISO 17987 系列中所有部件的列表可在 ISO 网站上找到。

介绍

ISO 17987（所有部分）规定了车载通信网络（称为本地互连网络 (LIN)）的用例、通信协议和物理层要求。

所提议的 LIN 协议是一种以汽车为重点的低速通用异步接收器发射器 (UART) 网络。LIN 协议的一些关键特性包括基于信号的通信、基于调度表的帧传输、带错误检测的主/从通信、节点配置和诊断服务传输。

LIN 协议适用于低成本汽车控制应用，例如车门模块和空调系统。它通过提供以下功能，充当车辆低速控制应用的通信基础设施：

— 基于信号的通信，用于不同节点上的应用程序之间交换信息；

— 比特率支持从 1 kbit/s 到 20 kbit/s；

— 基于确定性调度表的帧通信；

— 网络管理以受控方式唤醒 LIN 集群并将其置于睡眠模式；

— 提供错误处理和错误信号的状态管理；

— 允许传输大量数据（例如诊断服务）的传输层；

— 如何处理诊断服务的规范；

— 电气物理层规范；

— 描述从属节点属性的节点描述语言； — 描述通信行为的网络描

述文件； — 应用程序员的接口。

ISO 17987（所有部分）基于 ISO/IEC 7498-1 中指定的开放系统互连 (OSI) 基本参考模型，该模型将通信系统分为七层。

OSI 模型将数据通信分为七层，即（自上而下）*应用层*（第 7 层）、*表示层*、*会话层*、*传输层*、*网络层*、*数据链路层*和*物理层*（第 1 层）。ISO 17987（所有部分）使用了这些层的子集。

ISO 17987（所有部分）区分了某一层向其上层提供的服务以及该层用于在该层的对等实体之间发送消息的协议。进行这种区分的原因是使服务（尤其是应用层服务和传输层服务）也可重复用于 LIN 以外的其他类型的网络。这样，协议对服务用户来说是隐藏的，并且可以根据特殊系统要求更改协议。

ISO 17987（所有部分）提供了支持实施下列相关要求所需的所有文件和参考资料。

— ISO 17987-1：本部分概述了 ISO 17987（所有部分）和结构以及用例定义和供所有后续部分使用的一组通用资源（定义、参考）。

— ISO 17987-2：本部分规定了在 LIN 节点之间传输消息的 PDU 的传输协议相关要求和网络层要求。

— ISO 17987-3：本部分规定了逻辑抽象层面上 LIN 协议实现的要求。硬件相关属性隐藏在定义的约束中。

— ISO 17987-4：本部分规定了互连协议实现所必需的主动硬件组件的实现要求。

— ISO/TR 17987-5：本部分规定了 LIN 应用程序编程器接口 (API) 以及节点配置和识别服务。节点配置和识别服务在 API 中规定，并定义了如何配置从属节点以及从属节点如何使用识别服务。

— ISO 17987-6：本部分规定了根据 ISO 17987-2 和 ISO 17987-3 检查 LIN 协议实施一致性的测试。其中包括数据链路层、网络层和传输层的测试。

— ISO 17987-7：本部分规定了根据 ISO 17987-4 检查 LIN 电气物理层实现（逻辑抽象级别）一致性的测试。

道路车辆 — 本地互连网络（LIN）—

第 2 部分：
传输协议和网络层服务

1 范围

本文件规定了传输协议和网络层服务，以满足本地互连网络上基于 LIN 的车辆网络系统的要求。该协议规定了未确认通信。

LIN 协议支持 ISO 14229-2 中规定的标准化服务原语接口。

该文档提供传输协议和网络层服务以支持不同的应用层实现，如正常通信消息和诊断通信消息。

传输层定义包含在一个或多个帧中的数据的传输。传输层消息通过诊断帧传输。为传输层指定了标准化 API。

传输层的使用针对的是诊断在主干总线（例如 CAN）上执行的系统，并且系统构建者希望在 LIN 子总线集群上使用相同的诊断功能。这些消息实际上与 ISO 15765-2 相同，并且携带这些消息的 PDU 非常相似。

传输层的目标是

— LIN 主节点负载低，

— 直接在 LIN 从属节点上提供完整（或其子集）诊断，以及

— 针对使用强大的 LIN 节点构建的集群（而非主流低成本）。

图1显示了典型的系统配置。

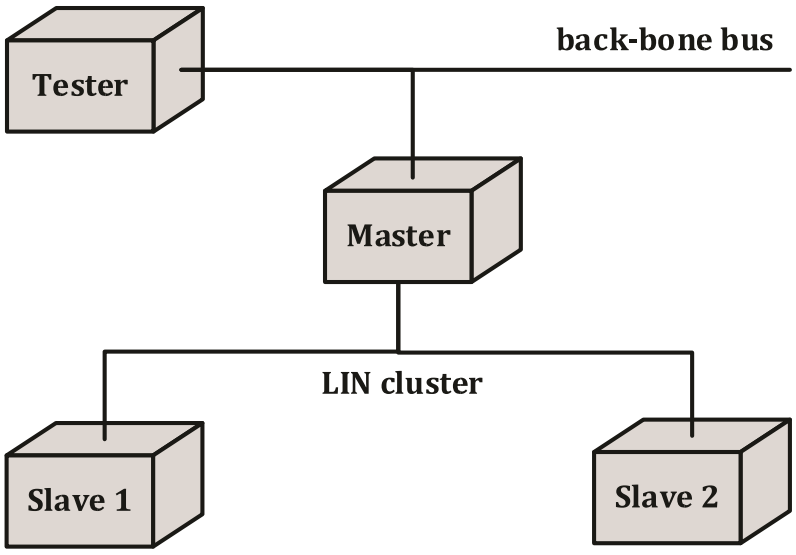


图 1 — 使用传输层的 LIN 集群的典型系统设置

2规范性引用文件

下列文件在文中引用，其部分或全部内容构成本文件的要求。对于注明日期的引用，仅引用的版本适用。对于未注明日期的引用，引用文件的最新版本（包括任何修订）适用。

ISO 14229-1，*道路车辆 — 统一诊断服务（UDS）— 第 1 部分：规范和要求*

ISO 14229-2，*道路车辆 — 统一诊断服务（UDS）— 第 2 部分：会话层服务*

ISO 14229-7:2015，*道路车辆 — 统一诊断服务（UDS）— 第 7 部分：本地互联网络上的 UDS（UDS onLIN）*

ISO 17987-3:2016，*道路车辆 — 本地互连网络（LIN）— 第 3 部分：协议规范*

3术语、定义、符号和缩写

3.1 术语和定义

就本文件的目的而言，ISO/IEC 7498-1 中给出的以及下列的术语和定义适用。

ISO 和 IEC 在以下地址维护用于标准化的术语数据库：

— IEC Electropedia：网址：http: //www.electropedia.org/

— ISO 在线浏览平台：http : //www.iso.org/obp

3.1.1 广播 NAD从属节点接收到NAD等于广播NAD的消息 7F $_{16}$消息被接收并处理

3.1.2 配置 N A D

分配给每个从节点的（01 $_{16}$至 7D $_{16}$）范围内的值

注 1：配置的 NAD 分配给每个从属节点在 LDF 中定义。配置的 NAD 用于节点配置和识别服务，以及根据 ISO 14229-7 的 UDS 服务。

注 2：通信初始化时，从节点配置的 NAD 可能相同。主节点应在诊断通信开始前为所有从节点分配唯一的配置 NAD。

注 3：可以通过以下方式设置或改变从属节点中配置的NAD：

— 主节点将新配置的NAD分配给支持“分配NAD”服务的从属节点；

— 从属节点中的 API 调用分配配置的 NAD；

— 已配置的 NAD 被分配静态配置。

3.1.3 功能性NAD

（7E $_{16}$）用于广播诊断请求

3.1.4 初始 NAD常数/静态值，范围为（01 $_{16}$至 7D $_{16}$）

注1：在进入操作状态（常规 LIN 通信）之前，初始 NAD 值可能来自引脚配置、EEPROM 或从属节点位置检测算法。

注2：在“分配NAD”命令中使用每个从属节点唯一的初始NAD、供应商ID和功能ID的组合，从而允许明确的配置NAD分配。

注3：如果没有为从属节点（LDF、NCF）定义初始 NAD，则该值与配置的 NAD 相同。

3.1.5 P2时序参数
ECU 和外部测试设备的应用时序参数

3.1.6 P2*时序参数
响应待处理帧传输后，ECU 应用程序的增强响应时序参数

3.1.7 P4时序参数
ECU 应用程序的定时参数，定义接收请求和最终响应之间的时间

3.1.8 专有NAD
80_{16} – FF_{16}]范围内，用于非标准化通信目的，例如 Tier-1 从属节点诊断

3.2符号

% 百分比 μs 微秒 ms

毫秒

| 竖线表示选择。竖线左侧或右侧应出现

3.3 缩写术语

4 公约

ISO 17987（所有部分）和 ISO 14229-7 基于 OSI 服务约定（ISO/IEC 10731）中指定的约定，适用于物理层、协议、传输协议和网络层服务以及诊断服务。

5网络管理

5.1网络管理一般信息

LIN 集群中的网络管理仅指集群唤醒和进入睡眠状态。其他网络管理功能（例如配置检测和跛行回家管理）则留给应用程序。

5.2 LIN节点通信状态图

图 2中的状态图显示了 LIN 通信状态的行为模型。

— Bus Sleep

首次连接电源、系统初始化、重置或主节点发送或从节点接收到进入睡眠状态的命令后，将进入总线睡眠状态。总线上的电平设置为隐性。集群上只能发送唤醒信号。

— Operational

本文档中指定的协议行为（发送和接收帧）仅适用于操作状态。

注意 LIN“总线休眠”状态不一定与节点的电源状态相关。

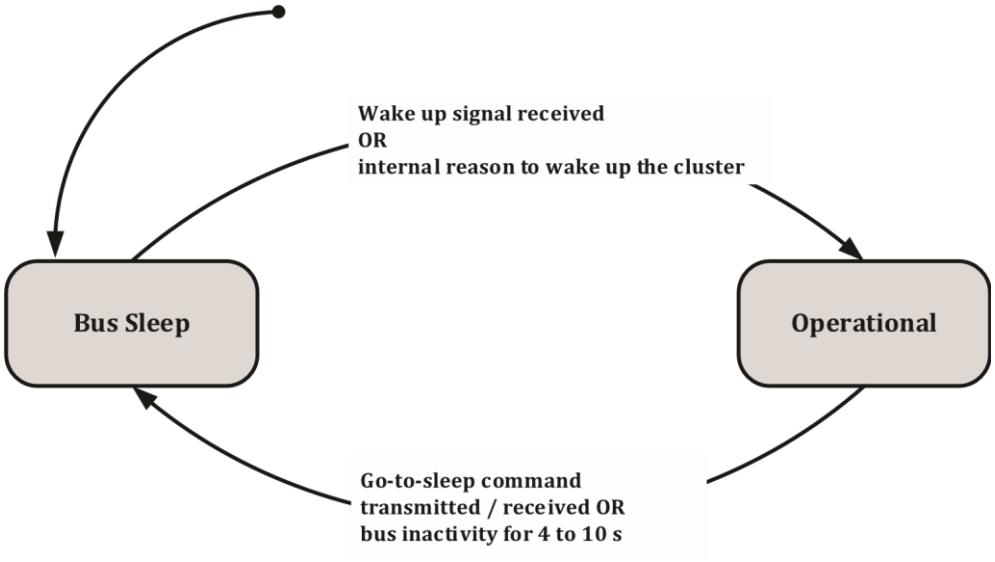


图 2 — LIN 节点通信状态图

5.3唤醒

5.3.1 唤醒通用信息

任何处于休眠状态的 LIN 集群中的节点都可以通过发送唤醒信号来请求唤醒。唤醒信号通过强制总线进入显性状态持续 250 μs 至 5 ms 来启动，并在总线信号返回隐性状态时有效。

5.3.2 主机产生唤醒

主节点可以发出中断字段，例如通过发出普通标头，因为中断充当唤醒信号（在这种情况下，主节点应意识到该帧可能不会被从属节点处理，因为它们可能尚未醒来并准备好监听标头）。

每个从属节点（连接到电源）应检测唤醒信号（一个长于 150 μs 的主脉冲，后跟总线信号的上升沿），并准备在 100 ms 内监听总线命令，从主脉冲的结束沿开始测量（见图3 ）。上升沿的检查应由收发器完成，也可以由微控制器 LIN 接口完成。

150 μs 的检测阈值与 250 μs 的脉冲生成相结合，为未校准的从节点提供了足够的检测裕度。检测到唤醒脉冲后，从节点任务机(ISO 17987-3:2016, 7.5.3) 应启动并进入空闲状态。在空闲状态下，从节点不得在总线上发出显性电平脉冲，直到状态机进入活动状态。

5.3.3 从机产生唤醒

如果发送唤醒信号的节点是从节点，则它应立即准备好接收或发送帧。主节点也应唤醒，当从节点准备就绪（>100 毫秒）时，开始发送报头以找出唤醒的原因（使用信号）。

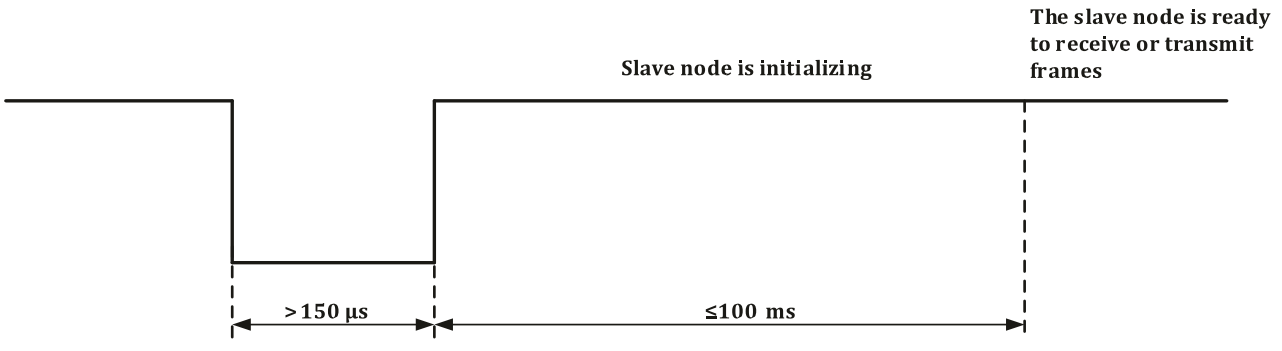


图 3 — 从属节点中的唤醒信号接收

主节点应检测唤醒信号（一个长于 150 μs 的显性脉冲，随后是总线信号的上升沿），并准备在集群设计者或特定应用决定的时间内开始通信。上升沿的检查应由收发器完成，也可以由微控制器 LIN 接口完成。

如果主节点未发送中断字段（即开始发送帧），或者发出唤醒信号的节点在唤醒信号发出后的 150 毫秒至 250 毫秒内未收到唤醒信号（来自另一个节点），则发出唤醒信号的节点应发送新的唤醒信号（参见图 4 ）。如果从属节点在主节点发送中断字段的同时发送唤醒信号，则从属节点应接收并识别该中断字段。

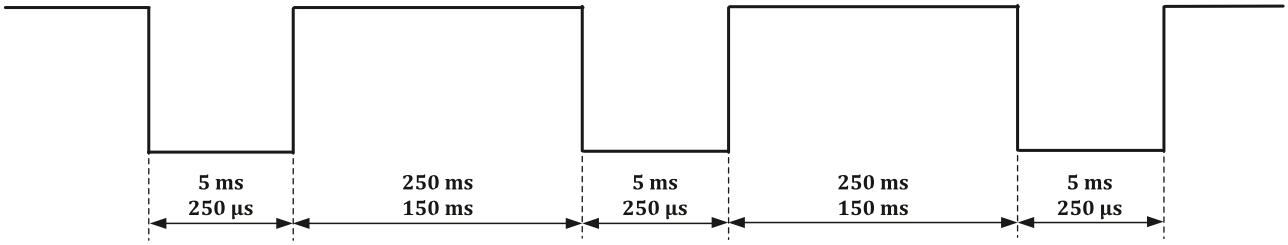


图 4—一组唤醒信号

三次请求（失败）后，节点应至少等待 1.5 秒，然后再发出第四次唤醒信号。持续时间较长的原因是，如果唤醒的从属节点出现问题，集群可以进行通信，例如，如果从属节点在读取总线时出现问题，它可能会无限次地重新发送唤醒信号。从属节点可以发送唤醒信号的次数没有限制。但是，建议从属节点在每次唤醒条件下发送的唤醒信号块不超过三个。图 5显示了唤醒信号在较长时间内的传输方式

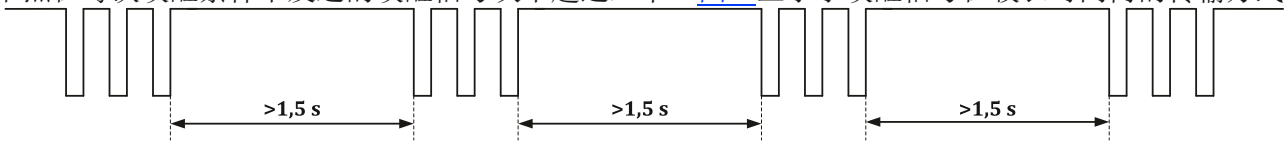


Figure 5 — Wake up signals over long time

5.4进入睡眠状态

主节点通过发送进入休眠命令，将集群中的每个节点立即设置为总线休眠状态。该请求不一定会强制从节点进入低功耗模式。在收到进入休眠命令后，从节点应用程序可能仍处于活动状态。此行为特定于应用程序。进入休眠命令是一个主节点请求帧，其中第一个数据字段（字节 1）设置为 00_{16}，其余设置为 FF_{16}（参见表1 ）。从节点应忽略数据字段字节 2 至字节 8，仅解释第一个数据字段（字节 1）。

表 1—进入睡眠状态命令

LIN 帧							
字节 1	字节 2	字节 3	字节 4	字节 5	字节 6	字节 7	字节 8
00_{16}	FF_{16}	FF_{16}	FF_{16}	FF_{16}	FF_{16}	FF_{16}	FF_{16}

将集群设置为休眠的正常方式是主节点传输进入休眠命令。在总线不活动的情况下，从属节点应能够接收/传输帧 4 秒。从属节点应在总线不活动最早 4 秒和最晚 10 秒内自动进入总线休眠模式。总线不活动定义为隐性位值和显性位值之间没有转换。总线活动则相反。

注意： LIN 收发器通常具有滤波器，用于消除总线上的短脉冲。此处的转换是指经过此滤波器的信号。

6 网络层概述

6.1 一般规定

本文件规定了未经证实的网络层通信协议，用于网络节点之间的数据交换，例如从 ECU 到 ECU，或外部测试设备与 ECU 之间的数据交换。如果要传输的数据不适合单个 LIN 帧，则提供分段方法。

为了描述网络层的功能，应考虑向更高层提供的服务和网络层的内部操作。

所有网络层服务都具有相同的通用结构。为了定义服务，指定了三种类型的服务原语：

— 服务请求原语，由较高的通信层或应用程序用来传递需要传输到网络层的控制信息和数据；

— 服务指示原语，网络层使用它将状态信息和接收的数据传递给上层通信层或应用程序；

— 服务确认原语，网络层使用它将状态信息传递给更高的通信层或应用程序。

该服务规范没有指定应用程序编程接口，而只是指定了一组独立于任何实现的服务原语。

6.2网络层服务的格式描述

所有网络层服务都具有相同的通用格式。服务原语的书写形式如下：

service_name.type(参数A,参数B[,参数C,...]
)

其中service_name为服务的名称，例如N_USData，type表示服务原语的类型，“参数A，参数B[参数C，…]”是服务原语传递的N_SDU作为值列表。括号表示这部分参数列表可以为空。

服务原语定义服务用户（例如诊断应用程序）如何与服务提供者（例如网络层）合作。本文档指定了以下服务原语：请求、指示和确认。

— 使用服务原语请求（ service_name.request ），服务用户向服务提供者请求服务。

— 服务提供者利用服务原语指示（ service_name.indication ）将网络层的内部事件或对等协议层实体服务用户的服务请求告知服务使用者。

— 服务提供者通过服务原语确认（ service_name.confirm ）告知服务使用者其先前服务请求的结果。

6.3网络层的内部操作

网络层的内部操作提供了分段和重组的方法。网络层的主要目的是传输可能适合或不适合单个 LIN 帧的消息。不适合单个 LIN 帧的消息被分段成多个部分，每个部分都可以在 LIN 帧中传输。

图6和图7分别示出了非分段消息传输和分段消息传输的示例。

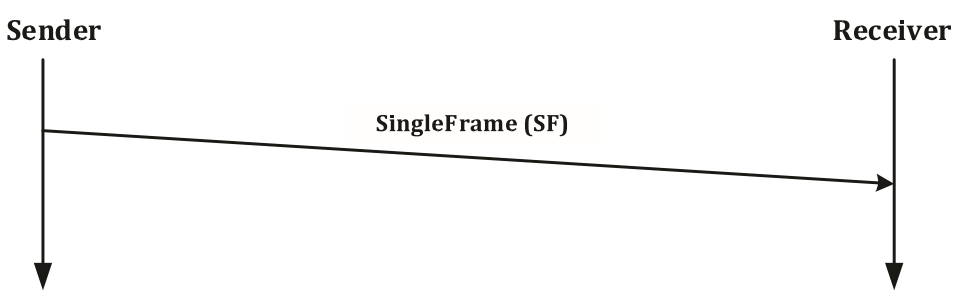


图 6 — 未分段消息的示例

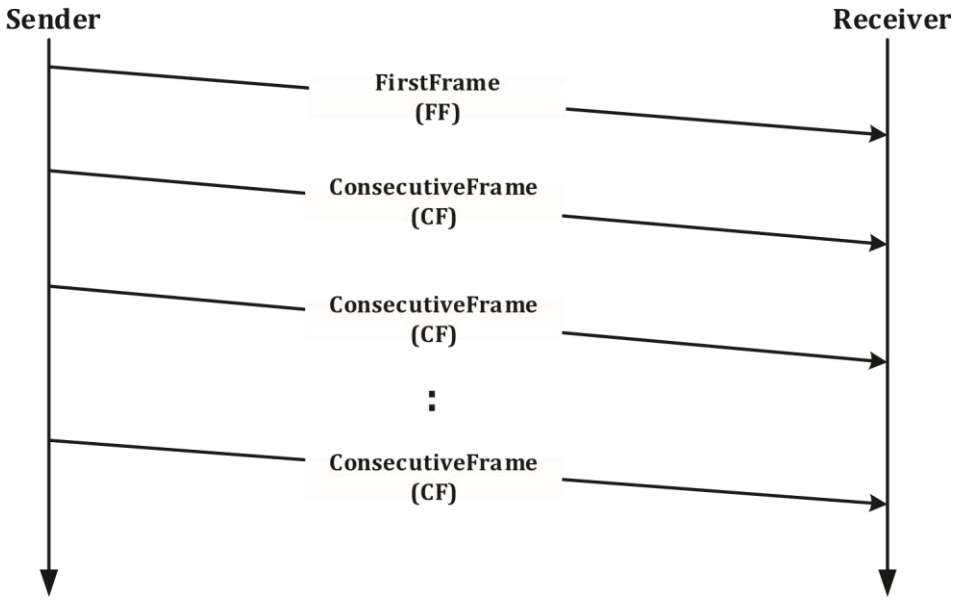


Figure 7 — Example of a segmented message

6.4 服务数据单元规范

在以下子条款中，描述了 LIN 网络层服务使用的所有网络层服务参数。

6.4.1 N_AI，地址信息

6.4.1.1 N_AI 描述

N_AI用于识别网络层的通信对等实体。N_AI由 LIN ID 和 N_NAD 组成。任何传输层协议通信都使用两个专用的 LIN ID 。

ID $3C_{16}$（MasterReq）被分配给主节点发送的消息传输。

ID $3D_{16}$（SlaveResp）分配给任何从属节点发送的消息传输。

因为节点类型（主节点或从节点）对于 LIN 集群中的所有节点始终是固定的，所以使用的 LIN ID 不受服务原语参数（ 6.5.1 ）的影响，而是由本文档分配给每个节点。

此外， N_NAD包括请求/响应寻址的从属节点以及寻址类型（物理、功能和广播）。

这些参数指的是寻址信息。总体而言， N_AI参数用于识别消息发送者和接收者以及消息的通信模型（ N_TAtype ）。

6.4.1.2 N_NAD，网络NAD

类型：1 字节无符号整数值

范围：01 $_{16}$ - FF $_{16}$

描述：指定了两种通信模型

— 1对1通信，称为物理寻址，以及

— 1 对 n 通信称为功能寻址或广播寻址。

根据N_NAD值给出通信模型

— 00 $_{16}$：不适用，保留用于睡眠命令帧，

— 01 $_{16}$ - 7D $_{16}$：物理寻址类型，

— 7E $_{16}$：功能寻址类型，

— 7F $_{16}$：广播寻址类型，以及

— 80 $_{16}$ - FF $_{16}$：专有寻址类型。

所有类型的网络层消息都应支持物理寻址。请求应得到响应。N_NAD值应始终根据从属节点中配置的 NAD 进行验证。

功能寻址仅支持单帧通信。功能请求不传输任何响应。主节点是传输功能消息的唯一节点。

所有类型的网络层消息都应支持广播寻址。响应是意外的，但支持，即使它们可能引起冲突。主节点应是唯一传输广播消息的节点。

$_{16}$ - FF $_{16}$范围内的 NAD 值的寻址类型，当使用此范围的 NAD 时，应由用户分配。

6.4.2 〈长度〉

类型： 12 位

范围：001 $_{16}$至 FFF $_{16}$

描述：此参数包括要发送/接收的数据的长度。

6.4.3 〈消息数据〉

类型：字节字符串

范围：不适用

描述：此参数包括上层实体交换的所有数据。

6.4.4 <N_Result>

类型：枚举

范围：N_OK、N_TIMEOUT_As、N_TIMEOUT_Cs、N_TIMEOUT_Cr、N_WRONG_SN、N_UNEXP_PDU、N_ ERROR、N_UNEXP_NEW_REQ

描述：此参数包含与服务执行结果相关的状态。如果同时发现两个或多个错误，则网络层实体应使用此列表中最先找到的参数值来向更高层发出错误指示。

— N_OK

该值表示服务执行已成功完成，可以发送给发送方和接收方的服务用户。

— N_TIMEOUT_As

当计时器 N_As 超过了其超时值 N_As max时，该值被发给发送方的服务用户。

— N_TIMEOUT_Cs

max时，该值被发送给发送方的协议用户。

— N_TIMEOUT_Cr

当计时器 N_Cr 超过了其超时值 N_Cr max时，该值被发给发送方的服务用户。

— N_WRONG_SN

在接收到意外的 SequenceNumber（N_PCI.SN）值时，该值发给服务使用者，只能在接收方发给服务使用者。

— N_UNEXP_PDU

在接收到意外的协议数据单元时，该值发给服务用户，可以发给接收方或发送方的服务用户。

— N_ERROR

这是通用错误值。当网络层检测到错误并且没有其他参数值可以更好地描述错误时，应将其发送给服务用户。它可以在发送方和接收方发送给服务用户。

— N_UNEXP_NEW_REQ

此值已发布

a) 在新的更高优先级的 N_USData.request 调用服务时，使用 N_USData.indication() 的参数以主动诊断通信的方式发送给优先级较低的服务实例，

b) 由于当前处于活动状态且优先级更高的诊断通信，无法接受 N_USData.request，并且

c) 在以相同优先级调用新的 N_USData.request 服务时，该值会发给服务用户。

6.5 网络层向上层提供的服务

6.5.1 网络层服务原语的规范

服务接口定义了访问网络层提供的功能所需的一组服务，即数据的发送/接收和协议参数的设置。

下面定义的通信服务可以传输最多 4,095 个字节的数据。

a) N_USData.request

此服务用于请求传输数据。如有必要，网络层将数据分段。b） N_USData_FF.indication

这个服务用来向上层发出分段消息接收的开始信号。c） N_USData.indication

该服务用于向更高层提供接收到的数据。

d） N_USData.confirm

该服务向上层确认所请求的服务已执行（成功或失败）。

6.5.2 N_USData.请求

该服务原语请求从发送方传输带有<Length>字节的<MessageData>到由N_NAD中的地址信息标识的接收方对等实体（参数定义见6.4.1.2 ）。

每次调用N_USData.request服务时，网络层应通过发出N_USData.confirm服务调用向服务用户发出消息传输完成（或失败）的信号

```
N_USData.request (
    N_NAD
    <MessageData>
    <Length>
)
```

6.5.3 N_USData. confirm

N_USData.confirm服务由网络层发出。该服务原语确认由N_NAD中的地址信息标识的N_USData.request服务的完成。参数<N_Result>提供服务请求的状态（参数定义见6.4.4 ）。

```
N_USData.request (
    N_NAD
    <N_Result>
)
```

6.5.4 N_USData_FF.indication

N_USData_FF.indication服务由网络层发出。该服务原语向相邻的上层指示从对等协议实体接收到的分段消息的首帧（FF）已到达，该对等协议实体由N_NAD中的地址信息标识（参数定义见6.4.1.2 ）。该指示应在收到分段消息的首帧（FF）时发生。

```
N_USData.request (
    N_NAD
    <Length>
)
```

N_USData_FF.indication服务应始终跟随来自网络层的N_USData.indication服务调用，指示消息接收完成（或失败）。

仅当接收到正确的 FirstFrame（FF）消息段时，网络层才会发出N_USData_FF.indication服务调用。

如果网络层检测到第一帧（FF）中存在任何类型的错误，则网络层应忽略该消息，并且不应向相邻的上层发出N_USData_FF.indication 。

如果网络层接收到的第一帧（FF），其数据长度值（FF_DL）大于可用的接收缓冲区大小，则应将其视为错误情况，并且不应向相邻的上层发出N_USData_FF.indication 。

6.5.5 N_USData. indication

N_USData.indication服务由网络层发出，该服务原语指示<N_Result>事件，并将从N_NAD中的地址信息标识的对等协议实体收到的带有<Length>字节的<MessageData>传送给相邻的上层（参数定义见6.4.1.2 ）。

参数<MessageData>和<Length>仅当<N_Result>等于N_OK时才有效。

```
N_USData.request (
     N_NAD
    <MessageData>
    <Length>
    <N_Result>
)
```

在接收到单帧 (SF) 消息后或作为分段消息接收完成（或失败）的指示，发出N_USData.indication服务调用。

如果网络层检测到单帧（SF）中任何类型的错误，则网络层应忽略该消息，并且不应向相邻的上层发出N_USData.indication 。

7 传输层协议

7.1 协议功能

传输层协议完成以下功能：

— 发送/接收最多 4,095 个数据字节的消息；

— 报告传输/接收完成/失败发生。

7.2 单帧传输

图 8显示了单帧 (SF)（称为 LIN 帧）的传输，该帧由 8 个字节（字节 1 ... 字节 8）组成，其中字节 1 = N_ NAD、字节 2 = N_PCI 和字节 3 ... 字节 8 = N_Data。SF 的实际消息有效载荷（参见7.4.2 ）最多为 6 个数据字节（另请参见8.3 ）。

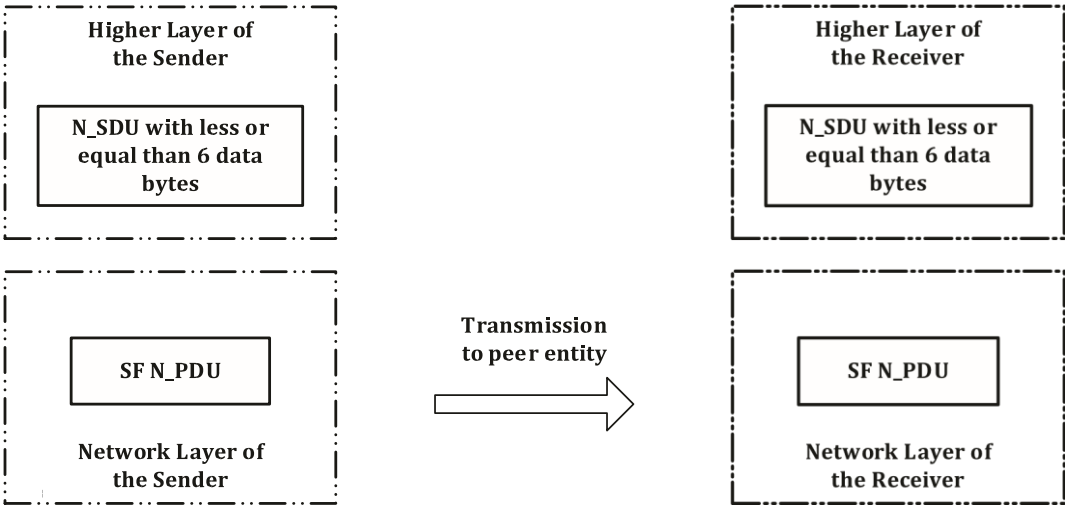


图 8 — 单帧（SF）传输示例

7.3 多帧传输

较长消息的传输是通过将消息分段为 LIN 帧（传输多个 N_PDU）来完成的。较长消息的接收是通过接收多个 LIN 帧（N_PDU）并将接收到的数据字节（串联）重新组装成一条消息来完成的。多个 N_PDU 称为 FirstFrame（针对消息的第一个 N_PDU）和 ConsecutiveFrame（针对所有后续 N_PDU）。

长度超过 SF N_PDU 所允许的数据字节数的消息将被分割为

— 第一帧协议数据单元 (FF N_PDU)，包含字节 1 = N_ NAD、字节 2 + 字节 3 = N_PCI 和字节 4 .. 字节 8 = N_Data。FF 的实际有效载荷（见7.4.3和8.3 ）为 5 个数据字节，并且

— 一个或多个连续帧协议数据单元 (CF N_PDU)，包含字节 1 = N_NAD、字节 2 = N_PCI 和字节 3 .. 字节 8 = N_Data。CF 的实际有效载荷（见7.4.4和8.3 ）为 6 个数据字节。最后一个 CF N_PDU 包含剩余的有效载荷数据字节 N_Data。最后一个 CF N_PDU 中未使用的数据字节填充了值 FF_{16}。

图 9显示了发送方的分段和接收方的重组。

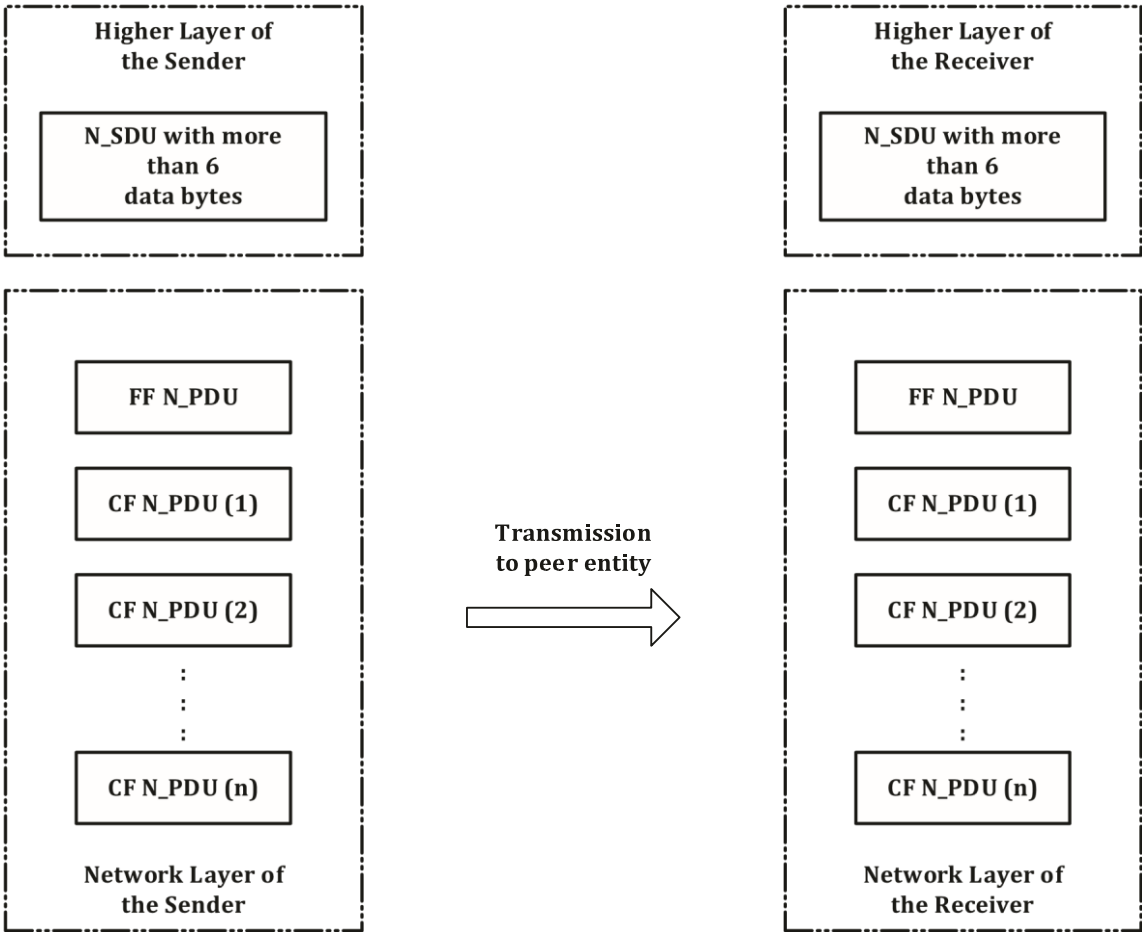


图 9—多帧传输示例（分段和重组）

消息长度在 FF N_PDU 中传输。所有 CF N_PDU 都由发送方编号，以支持验证传输顺序的正确性。

主节点应用层应符合从节点的功能。这可以通过调度表设计、调度模式（仅诊断模式与交错模式）或特定于应用程序的调度控制来实现。接收从节点的传输层不遵守 ST_{min}。

这些从属节点功能定义如下。

分离时间最小值（ST_{min}）是主节点在传输 FF 和其第一个 CF 以及两个 CF 之间应等待的最短时间。当支持传输层通信时，每个从属节点（LDF/NCF）的分离时间都是静态指定的。$ST_{min的测量}$在连续帧（CF）传输完成后开始，并在请求传输下一个 CF 时结束。

从属节点不必关心主节点（ST_{min}）的功能，因为应用适当的调度是主节点应用层的职责。

所有节点的功能是，每次发送方/接收方等待来自接收方/发送方的 N_PDU 时，超时机制允许检测传输失败（参见7.6.2）。

7.4 传输层协议数据单元

7.4.1协议数据单元类型

不同节点上的网络层对等协议实体之间的通信是通过交换N_PDU的方式来完成的。

本文档指定了三种不同类型的传输层协议数据单元：

— 单帧（SF N_PDU）；

— 第一帧（FF N_PDU）；

— 连续帧（CF N_PDU）；

用于在对等网络层实体之间建立通信路径、交换通信参数、传输用户数据以及释放通信资源。

7.4.2 SF N_PDU

SF N_PDU 由单帧协议控制信息（SF N_PCI）标识。SF N_PDU 应由发送网络实体发出，并由所有网络实体接收。只有由 N_AI 寻址的实体才能在分段消息传输期间继续执行此请求。其他未寻址的节点应拒绝已挂起的连接。应发送它以传输可通过单个服务请求传输到数据链路层的服务数据单元，并传输未分段的消息。

7.4.3 FF N_PDU

FF N_PDU 由 FirstFrame 协议控制信息（FF N_PCI）标识。FF N_ PDU 应由发送网络实体发出并由所有网络实体接收。只有由 N_AI 寻址的实体应在分段消息传输期间继续执行此请求。未寻址的其他节点应拒绝已挂起的连接。它标识由网络发送实体发送并由接收网络实体接收的分段消息的第一个 N_PDU。接收网络层实体应在收到 FF N_PDU 后开始组装分段消息。

7.4.4 CF N_PDU

CF N_PDU 由 ConsecutiveFrame 协议控制信息（CF N_PCI）标识。CF N_ PDU 传输服务数据单元消息数据（<MessageData>）的分段（N_Data）。发送实体在 FF N_PDU 之后发送的所有 N_PDU 都应编码为 CF N_PDU。接收实体在收到最后一个 CF N_PDU 后，应将组装的消息传递给网络接收实体的服务用户。CF N_PDU 应由发送网络实体发出，并在分段消息传输期间由唯一的接收网络实体接收。

7.4.5 协议数据单元字段描述

7.4.5.1 N_PDU 格式

协议数据单元（N_PDU）允许在一个节点的网络层和一个或多个其他节点（对等协议实体）的网络层之间传输数据。所有 N_PDU 均由三个字段组成，如表 2所示。

表 2—N_PDU 格式

地址信息	协议控制信息	数据字段
N_AI	N_PCI	N_Data

7.4.5.2 地址信息（N_AI）

N_AI 用于识别网络层的通信对等实体。应复制 N_SDU 一 N_NAD 中收到的 N_AI 信息并将其包含在 N_PDU 中。如果 N_SDU 中收到的消息数据（<MessageData>和<Length>）太长，网络层需要分段才能传输完整消息，则应复制 N_NAD 并将其包含（重复）在传输的每个 N_PDU 中。

该字段包含地址信息，用于标识交换的消息类型以及进行数据交换的收件人和发送者。

注：有关地址信息的详细说明，请参阅6.4.1 。

7.4.5.3 协议控制信息（N_PCI）

此字段标识所交换的N_PDU的类型。它还用于在通信的网络层实体之间交换其他控制参数。

注意：有关所有 N_PCI 参数的详细规范，请参阅7.5 。

7.4.5.4 数据字段(N_Data)

N_PDU 中的 N_Data 用于传输N_USData.request服务调用中 < MessageData > 参数中收到的服务用户数据。如果需要，< MessageData > 会被分割成更小的部分，每个部分都适合 N_PDU 数据字段，然后再通过网络传输。

在 SF 或属于消息的最后一个 CF 的情况下剩余的MessageData的数量。

7.5 协议控制信息规范

7.5.1 N_PCI

每个N_PDU通过N_PCI来识别（见表3和表4 ）。

Table 3 — Summary of N_PCI bytes

N_PDU name	N_PCI bytes		
	Byte #1		Byte #2
	Bits 7 – 4	Bits 3 – 0	Bits 7 – 0
SingleFrame (SF)	N_PCItype = 0	SingleFrame_DataLength (SF_DL)	N/A
FirstFrame (FF)	N_PCItype = 1	FirstFrame_DataLength (FF_DL)	
ConsecutiveFrame (CF)	N_PCItype = 2	SequenceNumber (SN)	N/A

表 4—N_PCI 类型值定义

价值	描述
0 16	**单帧 (SF)** 对于未分段的消息，网络层协议提供了优化的网络协议实现，消息长度仅嵌入在 N_PCI 字节中。应使用 SF 来支持可容纳单个 LIN 帧的消息传输。
1 16	**第一帧 (FF)** FF 仅用于支持无法容纳在单个 LIN 帧中的消息（即分段消息）的传输。分段消息的 FF 被编码为 FF。在收到 FF 后，接收网络层实体应开始组装分段消息。
2 16	**连续帧 (CF)** 发送分段数据时，FF 后面的所有 CF 均编码为 CF。在收到 CF 后，接收网络层实体应组装收到的数据字节，直到收到整个消息。在无误地收到消息的最后一帧后，接收实体应将组装的消息传递给相邻的上层协议层。
3 16	**预订的** 本文档保留此范围的值。LIN 不支持 FlowControl。
4 16 – F 16	**预订的** 此值范围由 ISO 15765-2 保留。

7.5.2 SingleFrameN_PCI参数定义

7.5.2.1 SFN_PCI字节

表 5概述了 SF N_PCI 字节。

表 5—SFN_PCIbyte 概述

N_PDU名称	SF N_PCI字节							
	字节 #1							
	7	6	5	4	3	2	1	0
单帧	0	0	0	0	单帧数据长度 (SF_DL)			

7.5.2.2 SingleFrame_DataLength(SF_DL)参数定义

SF N_PDU中单帧数据长度（SF_DL）参数用于指定服务用户数据字节数，见表6 。

表 6—SF_DL 值的定义

价值	描述
0_{16}	**无效的** 该值无效。
1_{16} - 6_{16}	**单帧数据长度 (SF_DL)** SF_DL 编码在 N_PCI 字节 #1 值的低半字节中。它应被分配服务参数<Length>的值。
7_{16} - F_{16}	**无效的** 此值范围无效。

7.5.3 FirstFrameN_PCI参数定义

7.5.3.1 FFN_PCI字节

表 7概述了 FF N_PCI 字节。

表 7—FFN_PCI字节概述

N_PDU名称	FF N_PCI字节								
	字节 #1								字节 #2
	7	6	5	4	3	2	1	0	7 - 0
第一帧	0	0	0	1	第一帧数据长度 (FF_DL)				

7.5.3.2 FirstFrame_DataLength(FF_DL)参数定义

参数FF_DL用于FF N_PDU中指定服务用户数据字节数，见表8 。

表 8—FF_DL值的定义

价值	描述
0_{16} - 6_{16}	**无效的** 此值范围无效。
7_{16} - FFF_{16}	**第一帧数据长度 (FF_DL)** 分段消息长度的编码产生一个 12 位长度值 (FF_DL)，其中最低有效位 (LSB) 指定为 N_PCI 字节 #2 的位“0”，最高有效位 (MSB) 为 N_PCI 字节 #1 的位 3。支持的最大分段消息长度等于 4095 字节的用户数据。应为其分配服务参数 <Length> 的值。

7.5.4 ConsecutiveFrameN_PCI参数定义

7.5.4.1 CFN_PCI字节

表 9提供了 CF N_PCI 字节的概述。

表 9—CFN_PCIbyte 概述

N_PDU名称	CF N_PCI字节							
	字节 #1							
	7	6	5	4	3	2	1	0
连续帧	0	0	1	0	序列号（S否）			

7.5.4.2序列号(SN)参数定义

参数SN用于连续帧（CF）N_PDU中指定以下内容。

— 连续帧（CF）的数字升序。

— 所有分段消息的 SN 都应以零开头。FirstFrame (FF) 应被赋值为零。它不包括 N_PCI 字段中的显式 SequenceNumber (SN)，但应被视为分段号零。

— 紧跟第一帧（FF）之后的第一个连续帧（CF）的 SN 应设置为 1。

— 在分段消息传输期间，每个新的连续帧（CF）都应将 SN 加一。

$_{16}$的值时，它应绕回并将下一个连续帧（CF）设置为 0_{16}。

这将导致表 10中给出的顺序。

表 10—SN 定义摘要

协议数据单元	FF	CF	CF	CF	CF	CF	CF	CF
序号	0_{16}	1_{16}	…	16号	F_{16}	0_{16}	1_{16}	…

SN值的定义见表11 。

表 11—SN 值的定义

价值	描述
0_{16}至 F_{16}	**序列号 (SN)** SN 应编码在 N_PCI 字节 #1 的低半字节中。SN 应设置为 0_{16} - F_{16}范围内的值。

7.6 网络层时序

7.6.1 时间限制

表 12定义了传输层的时序约束（基于 ISO 15765-2） 。属性应在定义的范围内。由于 LIN 比 CAN 慢，因此应相应调整值。这些属性是传输层的一部分，不应对节点配置有任何约束。

性能要求值是每个通信对等体必须满足的约束性通信要求，以便符合规范。由于 LIN 调度因具体用例而异，因此某个应用程序可能会在表 12定义的范围内定义特定的性能要求。传输层 N_SDUData.request 和数据链路层开始帧传输之间的时间取决于调度表定义和调度模式（有关更多信息，请参阅9.6.4 ）。

超时值定义为高于性能要求值，以确保系统正常工作并克服绝对不能满足性能要求的通信条件（例如高总线负载）。表 12中指定的超时值或节点（即 NCF）给出的值应被视为任何给定实现的上限。

一旦检测到错误情况，网络层应向网络层服务用户发出适当的服务原语。

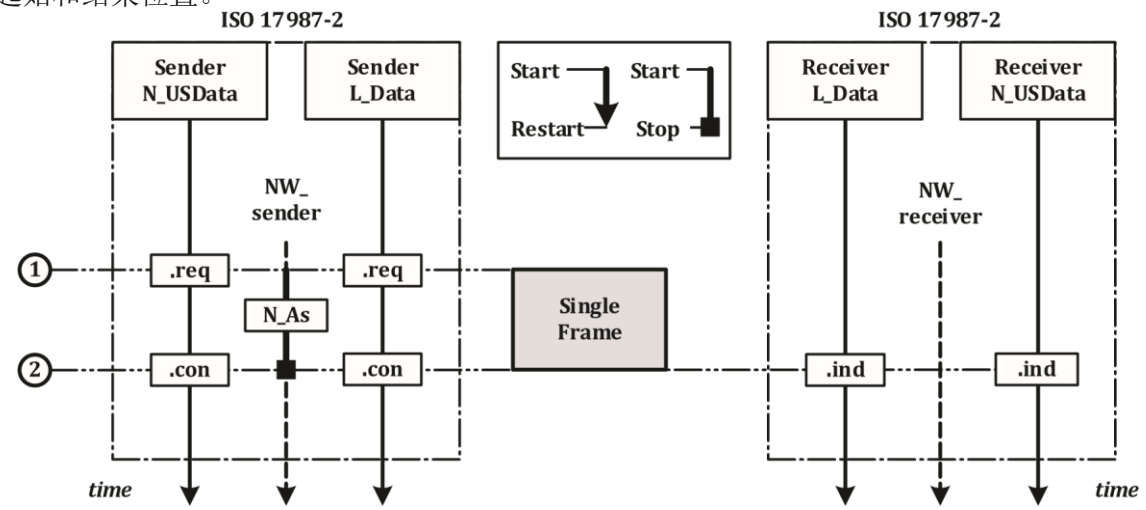
如果在对等协议实体之间建立了通信路径，由 N_AI 标识（有关详细信息，请参阅6.4.1和7.4.5.2 ），则将静态地为该通信路径分配一组网络层定时参数。在选择网络层定时参数时，除了 N_AI 之外，不使用任何其他信息。

表 12—网络层时序参数定义

定时 范围	描述	数据链路层服务		超时毫秒	表现 要求ms
		开始	结尾		
N_AS	LIN 帧的传输时间（MRF 或 发射机侧的 SRF	当传输层请求传输诊断帧	当诊断帧已确认传输时	1 000	不适用
N_Cs	请求传输的时间 下一连续帧（CF）	当前一个诊断帧同一信息已被确认已发送。	当传输层请求CF被传达	不适用	（N_Cs+N_As）< （0.9*N_Cr_{Max}）
N_Cr	收到 下一个连续-帧（CF）	当前一个诊断帧该消息已被指示为已收到。	当消息中的下一个诊断帧已被指示为已接收时。	1 000	—

N_Cs 参数不需要在传输节点中监控超时，因为 N_As 可确保正确的超时行为。但是，系统设计（调度和发射器软件设计）中应考虑 N_Cs，以避免接收器端（N_Cr）出现超时。

图10给出了非分段报文的网络层定时参数，表12根据数据链路层服务定义了网络层定时参数值及其对应的起始和结束位置。



钥匙

1 发送方N_USData.req：会话层向传输层发出未分段的消息。
发送方 L_Data.req：传输层向数据链路层请求单帧传输，并启动 N_As 定时器。此图显示了数据链路层和传输层之间的时序关系，并不表示实际的总线消息调度。

2 接收器 L_Data.ind：数据链路层向传输层发出 LIN 帧的接收。
接收方 N_USData.ind：传输层向会话层发出未分段消息的完成。
发送方 L_Data.con：数据链路层向传输层确认传输层 LIN 帧已成功传输。发送方停止 N_As 计时器。
发送方N_USData.con：传输层向会话层发出未分段消息的完成。

图 10 — 网络层定时参数的放置 — 未分段的消息

图 11和12说明了时域中的参数。这些图的目的是显示传输层时序参数，而不是要求特定的实现。主节点和从节点在较低层的行为是概括性的。

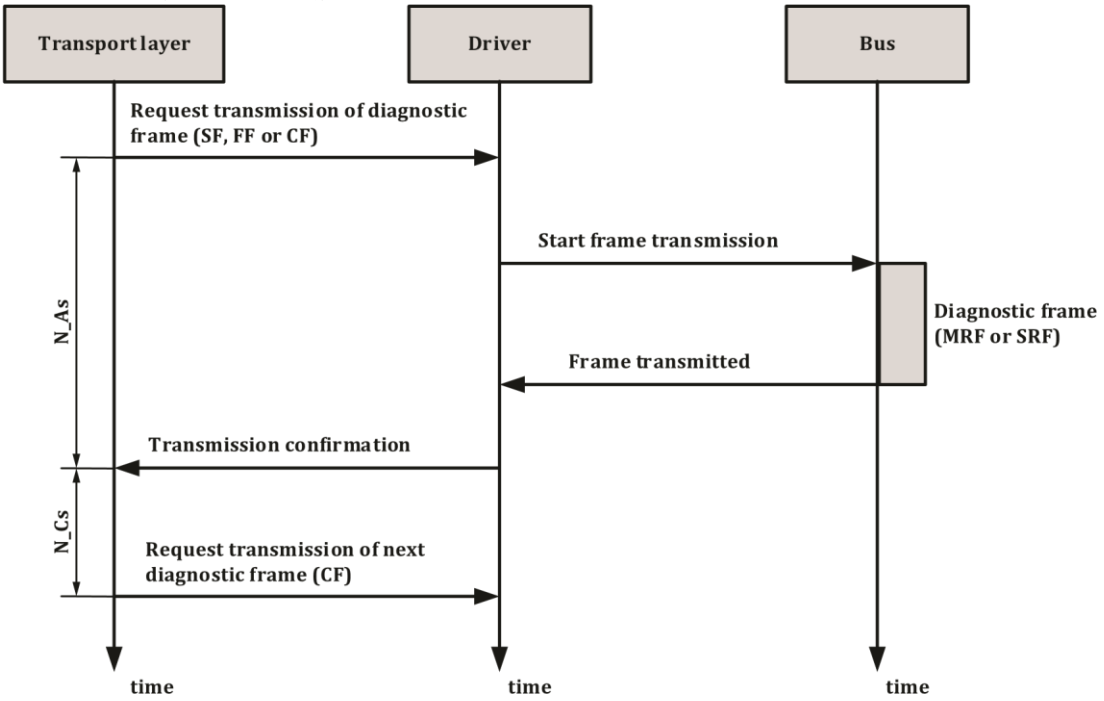


图 11 — 发送器侧的传输层时序

图12显示了接收端的传输层时序。

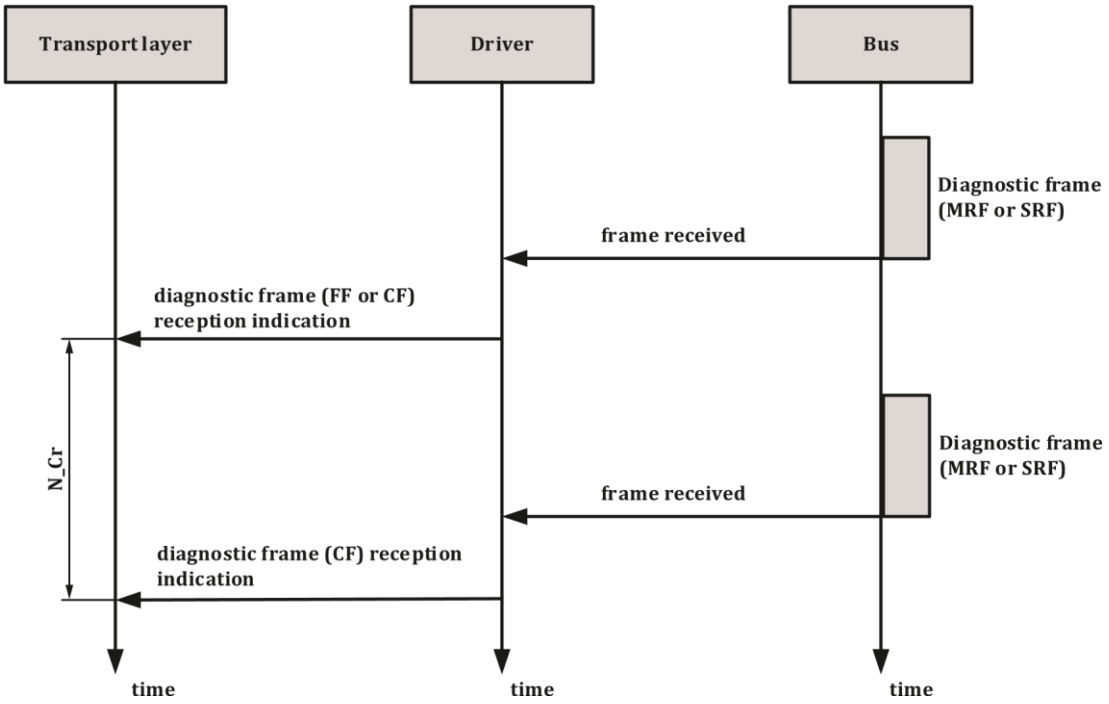
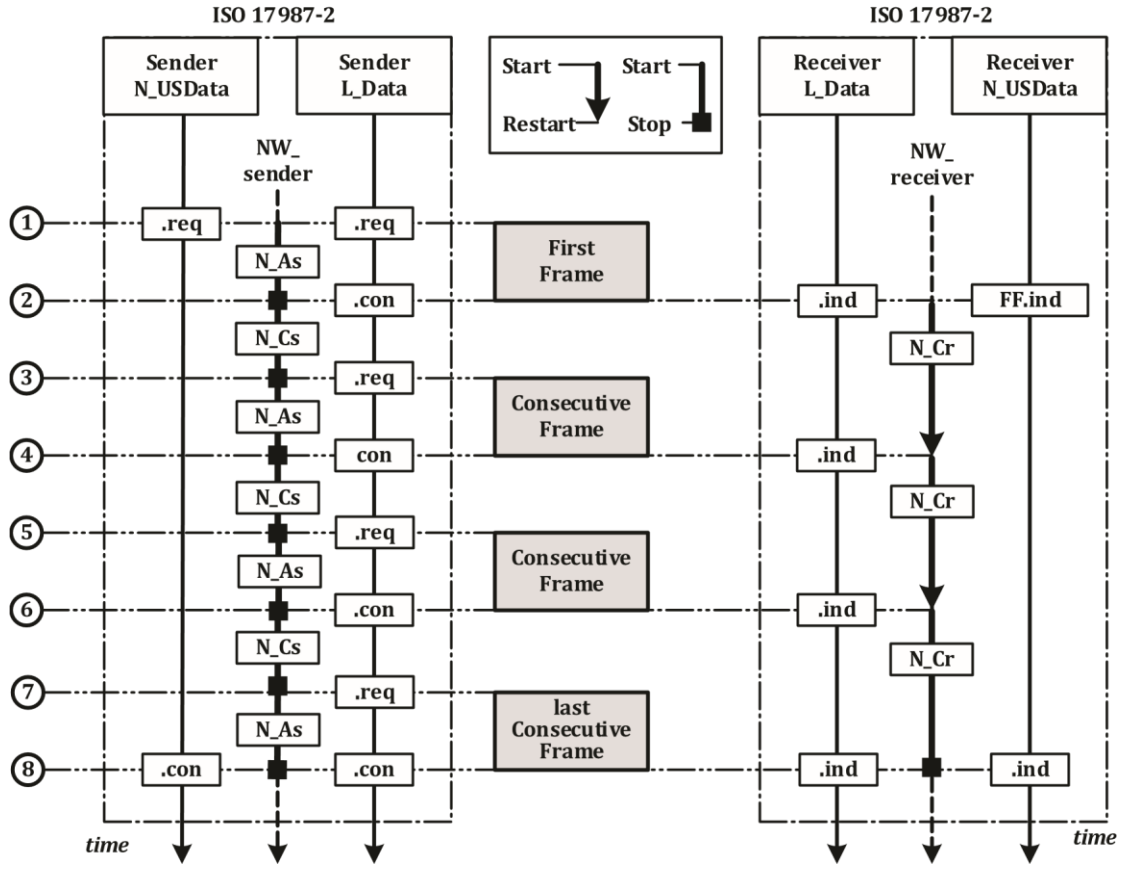


图 12 — 接收端传输层时序

图13给出了分段报文的网络层时序参数，表12根据数据链路层服务定义了网络层时序参数值及其对应的起始和结束位置。



钥匙

1 发送方N_USData.req：会话层向传输层发出分段消息。

发送方 L_Data.req：传输层向数据链路层请求 FirstFrame 传输并启动 N_As 计时器。此图显示了数据链路层和传输层之间的时序关系，并不表示实际的总线消息调度。

2 接收器 L_Data.ind：数据链路层向传输层发出 LIN 传输层帧的接收通知。接收器启动 N_Cr 计时器。

接收器 N_USData_FF.ind：传输层向会话层发出分段消息的第一帧的接收。

发送方 L_Data.con：数据链路层向传输层确认 LIN 帧已成功传输。发送方停止 N_As 计时器并启动 N_Cs 计时器。

3 发送方L_Data.req：传输层向数据链路层请求第一个连续帧传输，并启动N_As计时器。

4, 6 接收器 L_Data.ind：数据链路层向传输层发出 LIN 帧的接收通知。接收器重新启动 N_Cr 计时器。

发送方 L_Data.con：数据链路层向传输层确认 LIN 帧已成功传输。发送方根据分离时间值 (ST_{min}) 停止 N_As 计时器并启动 N_Cs 计时器。

5, 7 发送方 L_Data.req：当 N_Cs 计时器到期（ST_{min}）时，传输层请求向数据链路层进行下一个连续帧传输，并启动 N_As 计时器。

8 接收器 L_Data.ind：数据链路层向传输层发出 LIN 帧接收通知。接收器停止 N_Cr 计时器。

发送方 L_Data.con：数据链路层向传输层确认 LIN 帧已成功传输。发送方停止 N_As 计时器。

接收方N_USData.ind：传输层向会话层发出分段消息的完成。

发送方N_USData.con：传输层向会话层发出分段消息的完成。

图 13 — 网络层时序参数的放置 — 分段消息

7.6.2 网络层超时

表 13定义了网络层超时的原因和操作。

表 13—网络层超时错误处理

暂停	原因	行动
N_As	发送方未及时传输的任何 N_PDU。	中止消息传输并发出 N_USData.确认 <N_Result> = N_TIMEOUT_As。
N_Cs	发送方未及时传输连续帧 N_PDU。	中止消息传输并发出 N_USData.confirm <N_Result> = N_TIMEOUT_Cs。
N_Cr	接收方未接收到连续帧 N_PDU（受到干扰或未传输）。	中止消息接收并发出 N_USData.indication <N_结果> = N_TIMEOUT_Cr。

7.6.3 网络层错误处理

适用以下错误处理。

— 接收器应忽略数据长度（DL）值大于六个字节的单帧 PDU（SF）。

— 接收器应忽略数据长度（DL）值小于七个字节的 FirstFrame PDU（FF）。

— 接收器应忽略 DataLength（DL）值大于从属节点的最大可用接收缓冲区大小的 FirstFrame PDU（FF），并且接收器不应开始接收分段消息。这当然意味着接收节点正在接收完整消息（目标节点），而不是分段消息（网关的情况）。

— 如果网络层收到一个 SF_DL 等于零的 SF，那么网络层应忽略收到的 SF N_PDU。

— 除单帧（SF）和第一帧 PDU（FF）外，任何节点的具有意外 N_PCI 类型的 PDU 都应被忽略。

— 当从属节点接收到单帧（SF）或首帧（FF）PDU 时，如果 NAD 不等于功能 NAD，而消息接收（请求）或传输（响应）正在进行，则应中止当前接收或传输。如果 NAD 等于节点配置的 NAD 或广播 NAD，则应开始接收新消息。(忽略)

— 当主节点接收到单帧（SF）或第一帧（FF）PDU 时，如果消息接收（响应）正在进行中，则应中止当前接收。

— 仅当 NAD 等于相应请求中指定的 NAD 时，才开始接收新消息。

— 如果收到的 CF N_PDU 消息的序列号（SN）不符合7.5.4.2中的定义，则应中止消息接收，并且网络层应向相邻上层发出N_USData. 指示服务调用，参数为<N_Result> = N_WRONG_SN 。

最大超时后，接收器应中止消息接收。

— 发生 N_A 超时后，发送器应中止消息传输（见表13 ）。

— 如果分段数据传输的两个后续 CF 之间的时间（N_As + N_Cs）小于接收节点的 ST_{min}值，则无法保证分段数据传输的接收器正确接收和处理所有帧。无论如何，分段数据传输的接收器都不需要监控 ST_{min}值的遵守情况。

7.6.4 N_PDU意外到达

意外 N_PDU 定义为节点以非预期顺序接收的 N_PDU。它可能是本文档定义的 N_PDU（SF N_PDU、FF N_PDU、CF N_PDU），接收顺序与正常预期顺序不符，或者 NAD 与当前使用的 NAD 不同。如7.6.3中所述，PCI 类型未知的 N_PDU 将被忽略。

表 14和表15定义了主节点和从节点在收到意外 N_PDU 时的网络层行为。它考虑了当前的内部网络层状态（NWL 状态）和收到的不同可能 NAD 值。

当指定的操作是忽略意外的 N_PDU 时，这意味着网络层不得通知上层其到达。

表 14—主节点对意外 N_PDU 的处理

NWL 状态	接待 SF N_PDU	接待 FF N_PDU	接待 CF N_PDU
分段传输正在进行中	忽略	忽略	忽略
分段接收正在进行中	**新的 NAD 等于当前的 NAD ：** 终止当前接收，报告 N_USData.indication，并使用<N_Result>设置为N_UNEXP_ PDU，发送给上层，并将SF N_PDU处理为新接收的开始。 **其他 NAD ：** 被忽略	**新的 NAD 等于当前的 NAD ：** 终止当前接收，报告 N_USData.indication，并使用<N_Result>设置为 N_UNEXP_ PDU，发送给上层，并将 FF N_PDU 处理为新接收的开始。 **其他 NAD ：** 被忽略	**新的 NAD 等于当前的 NAD ：** 终止当前接收，报告 N_USData.indication，并使用<N_Result>设置为N_WRONG_SN，向上层报告。（只有错误的SequenceNumber才会出乎意料）**其他NAD ：** 被忽略

表 15—从节点对意外 N_PDU 的处理

NWL 状态	接待 SF N_PDU	接待 FF N_PDU	接待 CF N_PDU
分段传输正在进行中	**新的NADequalsconfiguredNADorbroadcastNAD ：** 终止当前传输，向上层报告 N_USData.confirm（其中 <N_Result> 设置为 N_UNEXP_PDU），并处理 SF N_PDU 作为新接收的开始。 **功能性NAD ：** 忽略 **其他 NAD：** 终止当前传输，向上层报告 N_USData.confirm，其中 <N_Result> 设置为 N_UNEXP_PDU。寻址不同的从站。	**新的NADequalsconfiguredNADorbroadcastNAD ：** 终止当前传输，向上层报告 N_USData.confirm，其中 <N_Result> 设置为 N_UNEXP_PDU，并处理 FF N_PDU 作为新接收的开始。 **功能性NAD ：** 忽略（无效的 N_PDU 格式） **其他 NAD ：** 终止当前传输，向上层报告 N_USData.confirm，其中 <N_Result> 设置为 N_UNEXP_PDU。寻址不同的从站。	忽略
分段接收正在进行中	**新的NADequalsconfiguredNADorbroadcastNAD ：** 终止当前接收，向上层报告 N_USData. 指示，其中 <N_Result> 设置为 N_UNEXP_PDU，并处理 SF N_PDU 作为新接收的开始。 **功能性NAD ：** 忽略 **其他 NAD ：** 终止当前接收，向上层报告 N_USData. 指示，并将 <N_Result> 设置为 N_UNEXP_PDU。寻址不同的从属设备。	**新的NADequalsconfiguredNADorbroadcastNAD ：** 终止当前接收，向上层报告 N_USData. 指示，其中 <N_Result> 设置为 N_UNEXP_PDU，并处理 FF N_PDU 作为新接收的开始。 **功能性NAD ：** 忽略（无效的 N_PDU 格式） **其他 NAD ：** 终止当前接收，向上层报告 N_USData. 指示，并将 <N_Result> 设置为 N_UNEXP_PDU。寻址不同的从属设备。	**新的NADequalsconfiguredNAD ：** 终止当前接收，向上层报告 N_USData. 指示，并将 <N_Result> 设置为 N_WRONG_SN。（只有错误的 SequenceNumber 才会是意外的） **其他 NAD ：** 被忽略

8 数据链路层的使用

本文档中定义的数据链路层服务仅为传输/网络层服务提供接口。LIN 上的任何数据交换均由主节点中的调度表控制。因此，定义的服务始终需要一个满足当前通信需求的 LIN 主节点应用程序。如果使用传输层服务进行诊断通信，主节点应用程序应激活包含 MRF 和 SRF 的适当调度表。

8.1 数据链路层服务参数

ISO 17987-3 中定义了以下数据链路层服务参数。

— <数据>: LIN 帧数据

— <Transfer_Status>：传输状态

8.2 数据链路层接口服务

8.2.1 L_Data.请求

服务原语请求传输应映射到 MRF/SRF的<Data> 。 <Data>由 N_NAD、N_PCI 和 N_Data 字段组成。如果是 SF 或最后一个 CF，还可以提供额外的填充数据。

L_Data.请求（
<数据>）

8.2.2 L_Data.确认

服务原语确认L_Data.request服务的完成。

参数<Transfer_Status>提供服务请求的状态：

L_Data.确认（
<传输状态>
）

8.2.3 L_Data.指示

服务原语向相邻的上层指示数据链路层事件，并传送<Data>标识的：

L_Data.指示（
<数据>）

8.3 N_PDU字段的映射

N_PCI 和 N_Data 放置在 LIN 帧数据字段中（见表16 ）。如果要在 SF 或最后一个 CF 中传输的 N_PDU 短于 SF N_PDU/CF N_PDU 字段，则发送方应使用 FF 16填充帧中任何未使用的字节。

Table 16 — Mapping of N_PDU parameters into LIN frame

	LIN frame							
	Byte 1	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
N_PDU type	**LIN frame address field (N_NAD)**	**LIN frame data field**						
		N_PCI	**LIN SF N_PDU field**					
SingleFrame (SF)	NAD	SF	N_Data/Byte Padding					
		N_PCI	**LIN FF N_PDU field**					
FirstFrame (FF)	NAD	FF	N_Data					
		N_PCI	**LIN CF N_PDU field**					
ConsecutiveFrame (CF)	NAD	CF	N_Data					
		N_PCI	**LIN CF N_PDU field**					
last CF	NAD	CF	N_Data/Byte Padding					

8.4 传输层PDU结构及通信

8.4.1 PDU结构

8.4.1.1 CAN 到 LIN 和 LIN 到 CAN帧数据字段的映射

在传输层帧中传输的单元称为 PDU。PDU 可以是完整的消息，也可以是消息的一部分；在后一种情况下，多个串联的 PDU 构成完整的消息。

客户端（测试器、主节点）发出的消息称为请求，服务器（主节点、从节点）发出的消息称为响应。

有效载荷中的第一个字节用作节点地址 (NAD)。由于使用诊断帧，因此传输层帧具有固定的帧 ID。这意味着使用 NAD 来寻址节点（或功能）。

重要 提示 — LIN 集群不支持流量控制帧。如果主干总线（例如 CAN 测试设备）需要流量控制 PDU，则这些 PDU 应由主干（CAN）侧的主节点生成。

当 N_PDU 字段的映射与 LIN 类似时，在数据链路层上就可以将主干总线传输层帧（如 ISO 15765-2 中定义的帧）路由到 LIN 集群。这意味着主干总线 N_PDU 的字节 1 用于识别 LIN 用户，并且可以由 LIN 主站转换为 N_NAD。

对于 CAN，满足此条件的是

— 11 位 CAN ID 混合寻址 Tp，

— 29 位 CAN ID 混合寻址 Tp，以及

— 扩展寻址。

注：扩展寻址使用目标地址 N_TA 定义。只有当每个从节点在主节点中代表自己的明确 N_TA 值时，才有可能转换为 N_NAD。否则，混合寻址不支持数据链路层路由。

如图 14所示，称为 Tp raw。

如果主干总线的数据映射与 LIN 格式不匹配，则支持在应用/会话层级别进行路由。消息数据接收应由主干总线网络层发出的 N_USData.indication 服务调用确认。主节点应用/会话层应将此消息作为新的 N_USData.request 服务请求转发到 LIN 网络层。接收将按相应方式处理。

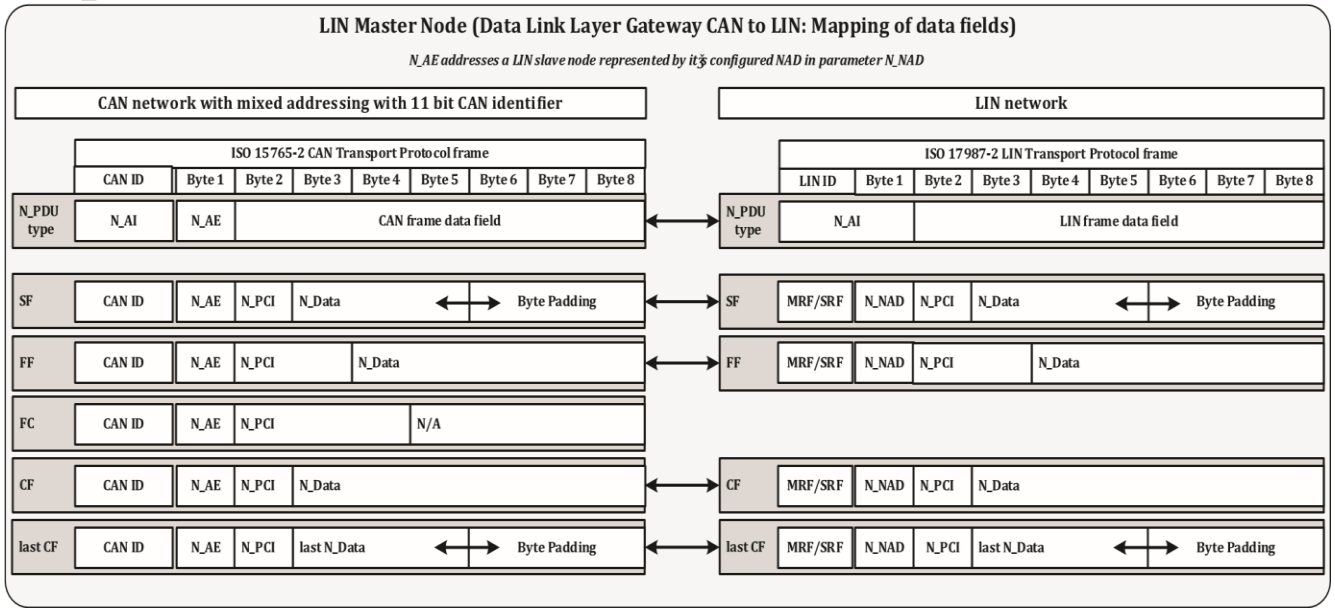


图 14—11位CANID混合寻址数据字段映射

图 15所示的 PDU 类型。

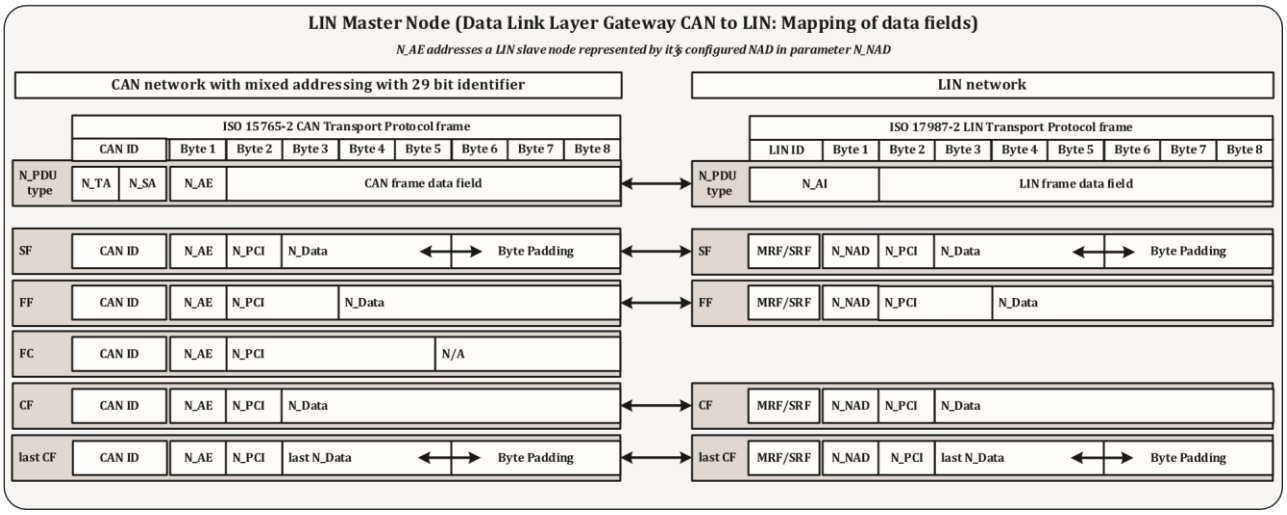


图 15—29 位 CANID 混合寻址数据字段映射

请求始终在主请求帧中发送，而响应始终在从属响应帧中发送。

8.4.1.2服务标识符(SID)

服务标识符（SID）指定应由所寻址的从属节点执行的请求。
00_{16} .. AF_{16}和 $C0_{16}$ .. FE_{16}用于诊断，而 $B0_{16}$ .. BF_{16}用于节点配置和节点识别（参见 ISO 17987-3）。响应服务标识符（RSID），SID+40_{16}，指定响应属于哪个请求的服务。

响应服务标识符（RSID），SID + 40_{16}，指定响应消息的内容。

8.4.1.3 数据字段(N_Data)

数据字节的解释取决于SID或RSID。在多PDU消息中，消息的所有PDU中的所有数据字节应连接成一个完整的消息，然后才能进行解析。

如果 PDU 未完全填充有效载荷数据（仅适用于最后一个 CF 和 SF PDU），则 PDU 的发布者应使用 FF_{16}填充未使用的字节。

8.4.2沟通

8.4.2.1 一般规定

要求传输层消息在一条总线上是独占的。这意味着在多帧传输消息期间，除功能请求外，不允许从设备的“请求接收”和“响应传输”交叉帧。

9诊断通信要求

9.1诊断类别的定义

9.1.1 一般规定

通过将诊断服务功能划分为三个诊断类别，可以满足从属节点的架构、诊断通信性能和传输协议需求。

因此，根据每个从属节点的诊断功能和复杂性级别为其分配一个诊断类别。

9.1.2诊断等级 I

智能和简单的设备，如智能传感器和执行器，不需要或只需要很少量的诊断功能。执行器控制、传感器读取和故障内存处理由主节点使用信号承载帧完成。因此，这些任务不需要特定的诊断支持。故障指示始终基于信号。

9.1.3诊断等级 II

诊断 II 类从属节点与诊断 I 类从属节点类似，但它提供节点识别支持。汽车制造商通常需要扩展节点识别。测试器或主节点使用 ISO 14229-1 诊断服务来请求扩展节点识别信息。执行器控制、传感器读取和故障内存处理由主节点使用信号承载帧完成。因此，这些任务不需要特定的诊断支持。故障指示始终基于信号。

9.1.4诊断级别 III

诊断 III 类从属节点是具有增强应用功能的设备，通常执行自己的本地信息处理（例如功能控制器、本地传感器/执行器环路）。从属节点执行超出基本传感器/执行器功能的任务，因此需要扩展诊断支持。直接执行器控制和原始传感器数据通常不与主节点交换，因此不包含在信号承载帧中。需要用于 I/O 控制、传感器值读取和参数配置（超出节点配置）的 ISO 14229-1 诊断服务。

诊断类 III 从属节点具有内部故障存储器以及相关的读取和清除服务。可选地，可以从属节点重新编程（闪存/NVRAM 重新编程）。这需要实施引导加载程序和必要的诊断服务来解锁设备、启动下载和传输数据等。

诊断级别 II 和诊断级别 III 之间的主要区别在于，对于诊断级别 II，LIN 主节点和 LIN 从属节点之间的诊断能力分配不同；而对于诊断级别 III LIN 从属节点，LIN 主节点中未实现 LIN 从属节点的诊断应用功能。

9.1.5 从属节点诊断类别摘要

表 17显示了不同诊断类支持的诊断服务列表。从属节点的所有支持配置和诊断服务均列在节点功能文件中，请参阅节点功能语言规范。有关节点配置和识别服务的详细描述，请参阅 ISO 17987-3:2016，6.3.4.4。ISO 14229-7 提供了基于 UDS 的 LIN 服务列表。

表 17—从属节点诊断属性

Slave diagnostic class	I	II	III	服务 标识符
诊断传输协议要求				
SingleFrame (SF) transport only	强制的	不适用	不适用	不适用
Full function transport protocol (multi-segment)	不适用	强制的	强制的	不适用
所需配置服务				
AssignNAD	选修的	选修的	选修的	$B0_{16}$
AssignFrameIdentifier (legacy)	选修的	选修的	选修的	$B1_{16}$
ReadByIdentifier (00_{16} = Product ID)	强制的	强制的	强制的	$B2_{16}$ 00_{16}
ReadByIdentifier (all except Product ID)	选修的	选修的	强制的	$B2_{16}$ XX_{16}
DataDump	选修的	选修的	选修的	$B4_{16}$
AutoAddressingSlave (legacy SID)	选修的	选修的	选修的	$B5_{16}$
Targeted Reset (SAE J2602 nodes only)	强制的	强制的	强制的	$B5_{16}$

SaveConfiguration	选修的	选修的	选修的	B6 16
AssignFrameIdentifierRange	强制的	强制的	强制的	B7 16
AutoAddressingSlave	选修的	选修的	选修的	B8 16

9.2 诊断消息

LIN 传输层使用与 ISO 14229-7 中定义的相同的诊断消息。SID 和 RSID 应符合 ISO 14229-7。节点可以实现 ISO 14229-7 中定义的服务子集。

一些非常简单的 LIN 从属设备（基于 ASIC）可以使用数据字段中的自定义参数来实现故障报告形式的诊断。由于这些参数可能是汽车制造商或特定于应用程序的，因此它们应该由 LIN 节点应用程序处理。

9.3 使用传输层

使用传输层存在两种通信情况，主节点想要向从节点传输诊断请求，或者从节点想要传输诊断响应。主节点有几种用例，为什么发送请求消息

— 来自主干网的诊断请求，

— 从属节点自我诊断，以及

— 从属节点配置和识别。

图 16和17显示了这两种情况下的消息流。

重要的是，控制通信的单元（测试仪或主站）避免请求多个从站同时响应，因为每个新请求都会拒绝之前寻址的从站节点的任何待处理响应。

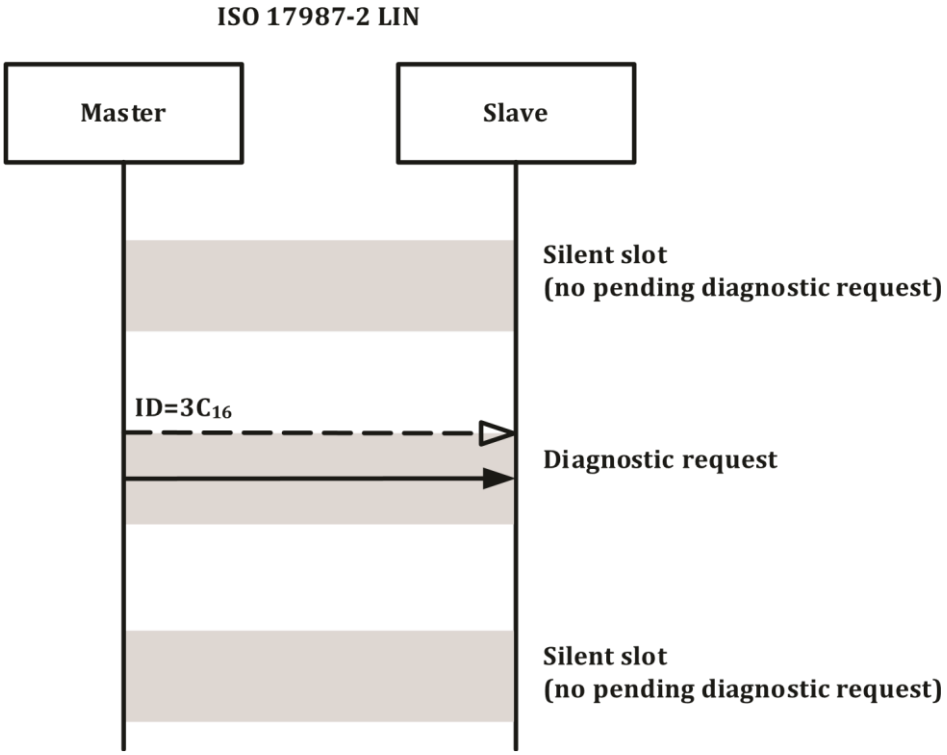


Figure 16 — Diagnostic request

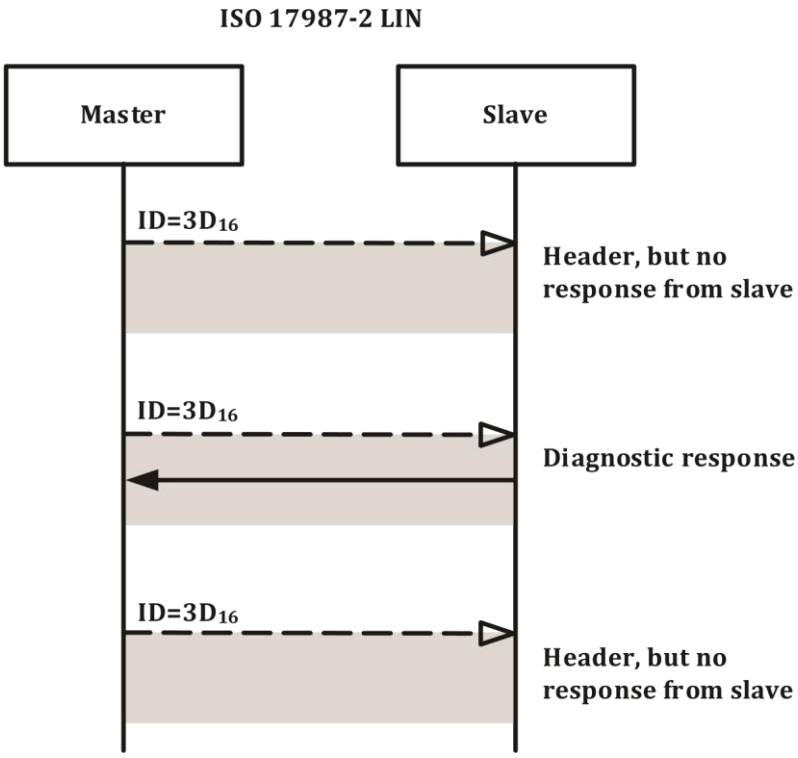


图 17 — 诊断响应

9.4从节点诊断时序要求

本款包含设计 LIN 集群时应考虑的时序参数要求。诊断 II 级和 III 级从属节点的监控应由从属节点本身实施。

表 18定义了诊断通信时序。

表 18—诊断通信时序

范围	做作的 设备	描述	最小值 /表现 要求	最大限度 价值/ 暂停
P2	主节点	LIN 总线上接收到诊断请求的最后一帧与从属节点能够提供响应数据之间的时间。 最大值定义了从属节点在丢弃其响应之前必须收到从属响应头的时间。 每个从属节点在 NCF 或 LDF 中定义此最小值，参见11.3.2和12.3.4 。	50 毫秒	500 毫秒
STmin	主节点	从属节点准备接收诊断请求的下一帧或传输诊断响应的下一帧所需的最短时间。 每个从属节点在 NCF 或 LDF 中定义此最小值，参见11.3.2和12.3.4 。	0 毫秒	无
P2*	主节点	发送 NRC 78 $_{16}$和 LIN 从站能够提供响应数据之间的时间。	P2	2 000 毫秒

诊断通信的时序图如图18所示。图中所示的外部测试设备仅作为示例。

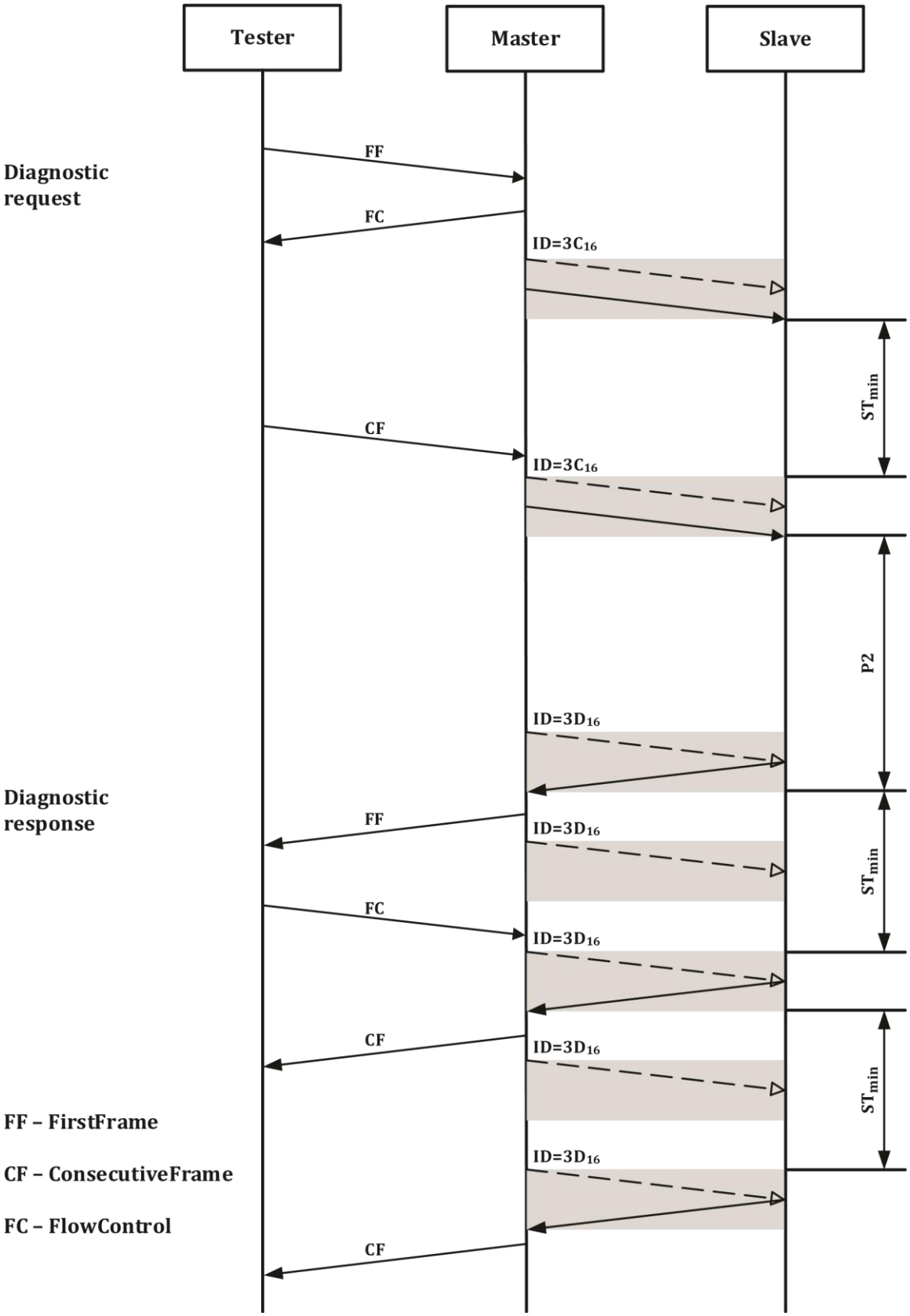


图 18 — 通过 LIN 进行测试仪诊断通信的时序图

主干总线

9.5 等待答复

诊断等级 II 和 III 的从属节点支持基于 ISO 14229-7 UDSonLIN 的诊断服务。如果服务需要比 P2 更长的时间进行处理，则从属节点将根据 ISO 14229-1 和 ISO 14229-2 传输响应待处理帧，以延长最大响应时间。响应待处理帧可以在 P4 时序内重复。每当使用响应待处理帧时，此请求都必须有最终的肯定或否定响应。接收响应待处理帧的主节点将响应超时延长至 P2*，并继续等待最终的从属响应。

响应待处理帧定义为具有 NRC 78 $_{16}$ 的单帧负响应。

表 19定义了响应待处理帧格式。

表 19—响应待处理帧格式

NAD	PCI	RSID	D1	D2	D3	D4	D5
NAD	03_{16}	$7F_{16}$	SID	$7F_{16}$	FF_{16}	FF_{16}	FF_{16}

9.6 LIN 主站中的传输协议处理

9.6.1 一般规定

LIN 主节点负责根据当前活动的诊断传输处理调度。本小节定义了调度和调度处理的要求，这些要求必须得到实施才能与任何从节点进行诊断通信。主节点充当主干总线和 LIN 集群之间的网络层路由器，这意味着主干总线和 LIN 集群上的传输协议由主节点处理。

在接下来的章节中，将使用多个通信序列图。请参阅图 19了解通信图的说明。为了清晰起见，交错的正常通信计划的大小以及诊断计划的大小并不代表计划中的正确延迟，而只是示意性的图形表示。

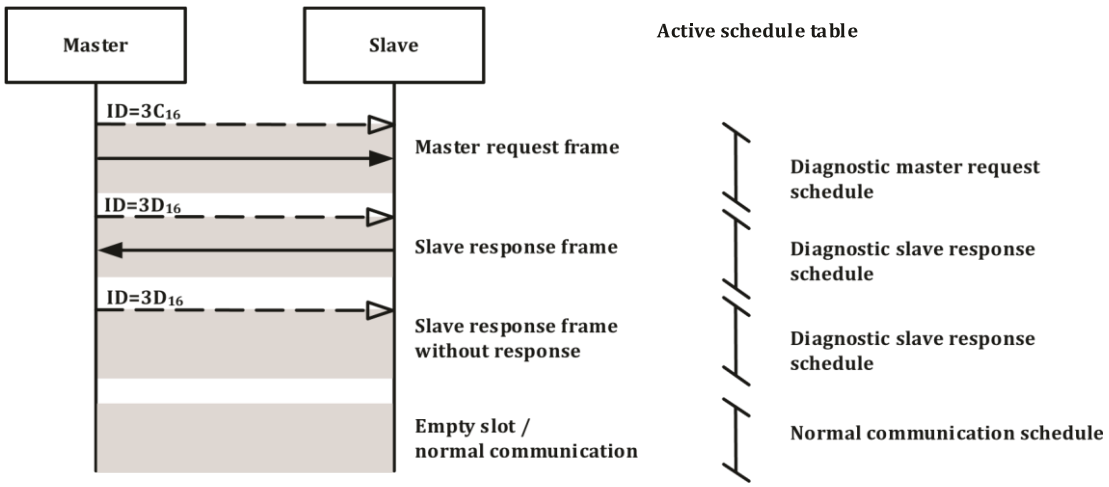


图 19 — 通信序列图图例

9.6.2 诊断主请求计划

主节点应支持包含单个主请求帧的诊断主请求调度表。LDF 设计人员可自行添加此特定调度表。

每当传输主请求帧时，都应执行诊断主请求调度表（见图20 ）。

注意：通过插入诊断主请求计划表的执行，正常通信计划的总体时间会受到影响。

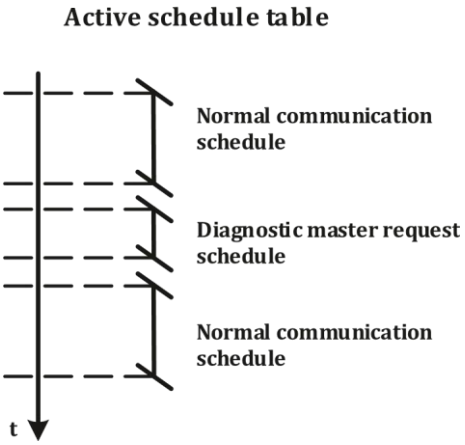


图 20—正常通信交错时间表

9.6.3诊断从站响应时间表

主节点应支持包含单个从节点响应帧的诊断从节点响应计划表。LDF 设计人员可自行添加此特定计划表。

每当需要传输从站响应帧时，应在正常通信计划的执行之间插入诊断从站响应计划表（见图21 ）。

注意，插入额外执行诊断从站响应计划表将会影响正常通信的总体时间。

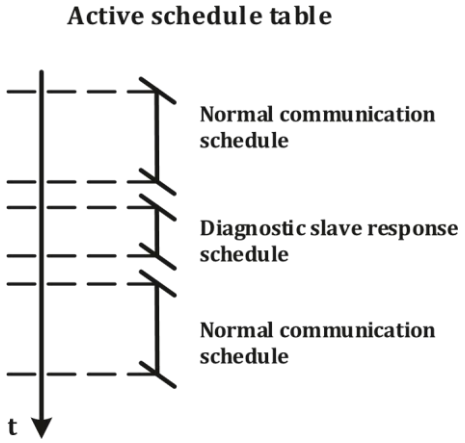


图 21—诊断从站响应时间表的交错

9.6.4 诊断计划执行

9.6.4.1 一般规定

当没有诊断通信处于活动状态时，主节点不得执行诊断计划表（见图22 ）。这应为主节点的默认行为。

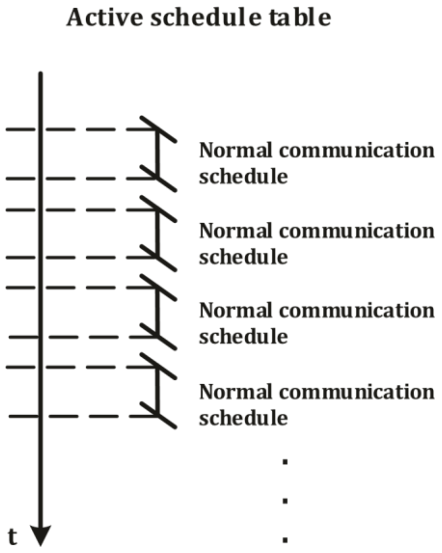


图 22 — 无诊断通信

主节点支持以下不同的调度模式：

— 交叉诊断模式（强制）；

— 仅诊断模式（可选）。

9.6.4.2和9.6.4.3对这两种模式有更详细的定义。主节点应支持根据外部诊断测试工具的请求以其中一种模式或另一种模式操作其连接的每个 LIN 集群。

9.6.4.2 诊断交错模式

当需要执行诊断计划时，主节点应使用最后定义的帧条目完成当前正在运行的正常通信计划，然后切换到所需的诊断计划以执行传输（参见图 20和21 ）。执行诊断计划后，主节点从头开始重新启动上一个正常通信表，或者如果在此期间已请求另一个计划表，则更改为另一个计划表和帧索引。直到交错的正常通信计划表结束时，才会执行新的诊断计划表。

使用诊断交错模式时，应确保（通过正常的通信时间表设计）两个连续诊断时间表之间的时间满足车辆制造商特定的诊断要求。

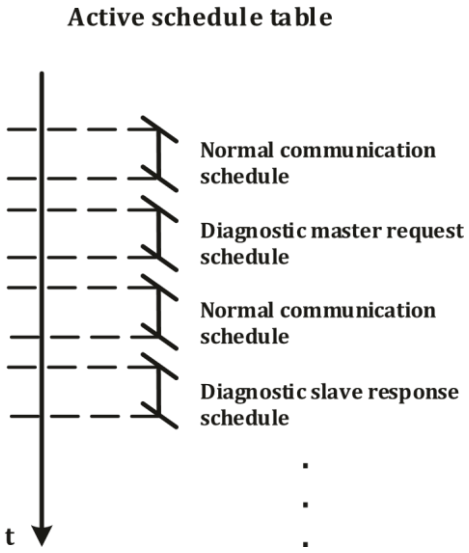


图 23 — 正常诊断通信模式

诊断主请求调度表的执行次数取决于需要传输的数据量，并应由主节点考虑 LIN 传输协议确定（例如，使用 LIN 传输协议传输 10 个用户数据字节的调度执行两次）。

诊断从站响应计划表的后续交叉执行取决于要传输的数据量，因此应由主节点执行，直到传输成功完成或发生传输协议超时。

如果已启动从从节点到主节点的诊断传输，则主节点应继续执行诊断从属响应计划，即使一个或多个从属响应帧头尚未得到答复（参见图 24 ），直到

— 发生 P2 $_{max}$ /P2* $_{max}$超时（参见9.4 ），并且

— 发生传输协议超时（见表12 ）。

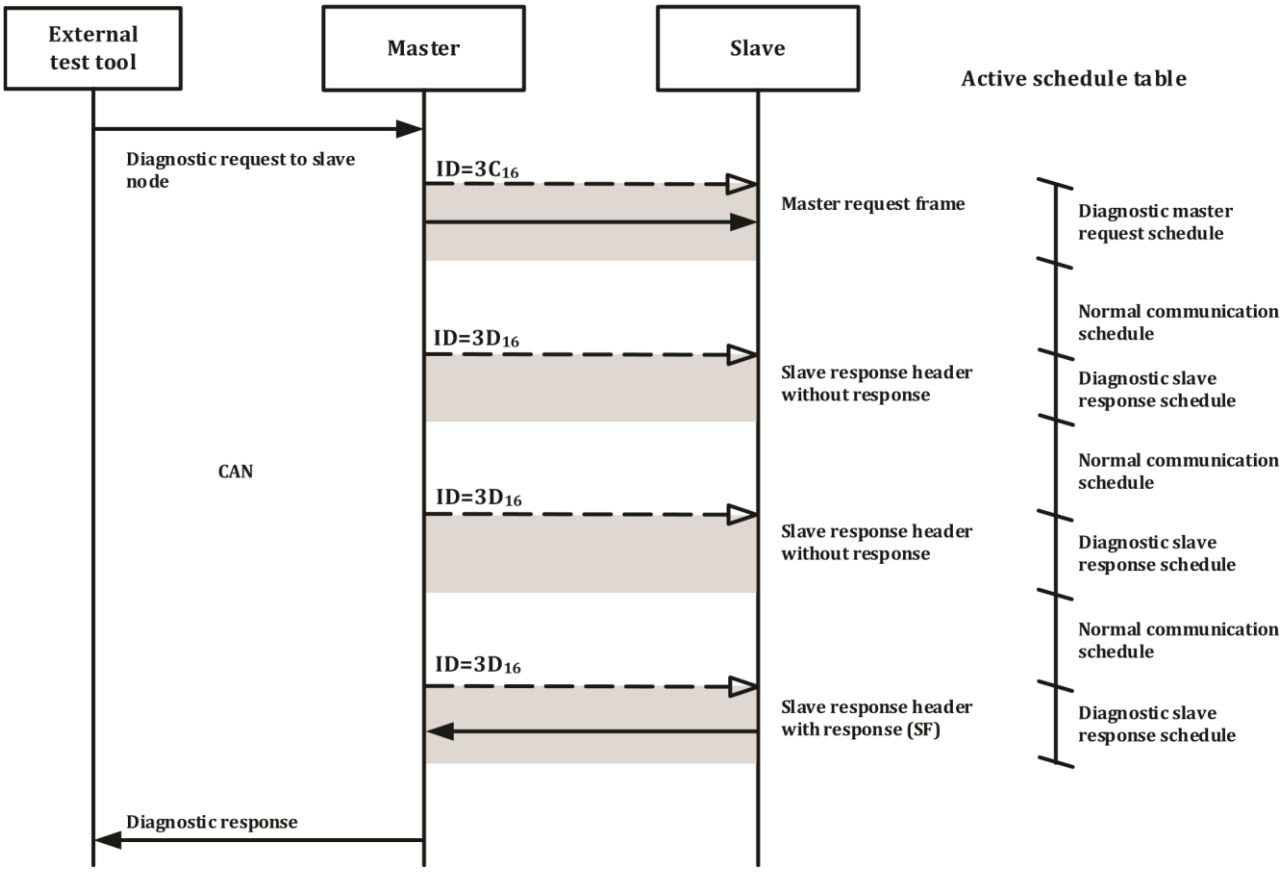
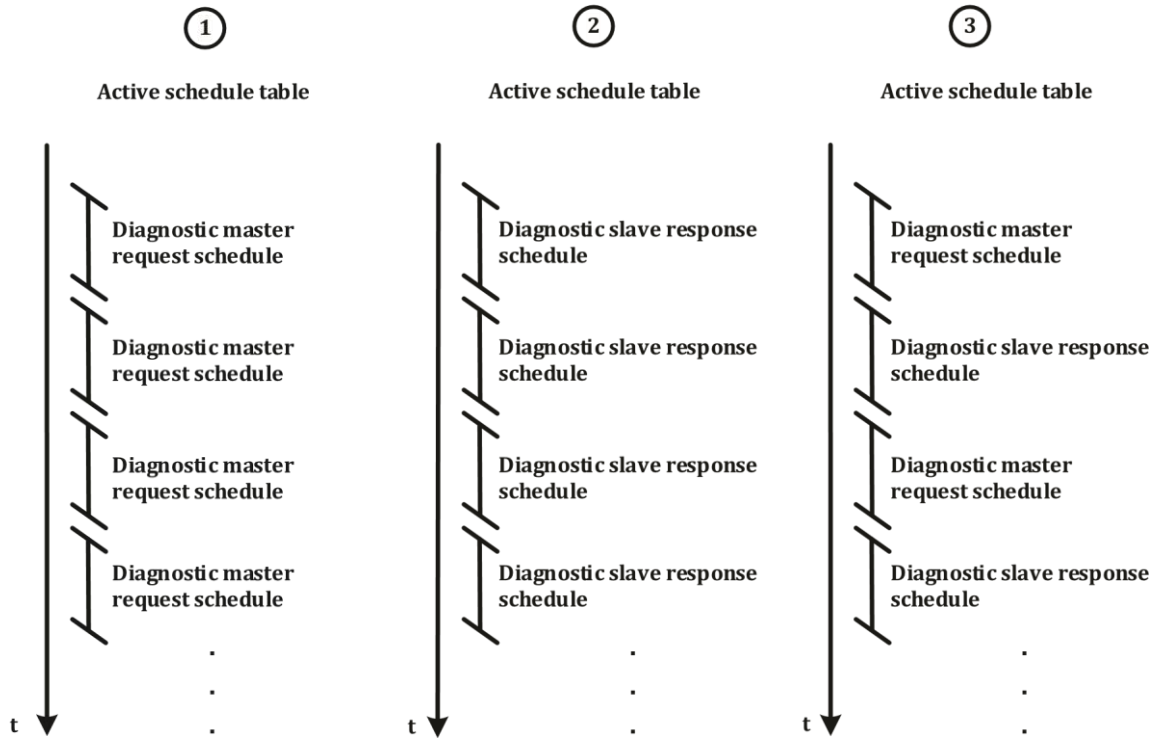


Figure 24 — Continued execution of diagnostic slave response schedule table until response is 已收到

9.6.4.3仅诊断模式

主节点可以选择性地实施“仅诊断模式”，在该模式下，只执行诊断计划而不执行正常通信计划。使用主请求帧计划表和从属响应帧计划表的基本原理与诊断交错模式相同，只是诊断计划表之间不交错正常通信计划。

这是为了实现优化的诊断数据传输（例如，在读取从属节点标识或在闪存重新编程期间，请参见图 25了解不同的用例）。



钥匙

1 长传输至诊断从站

2 诊断从站的长传输

3 诊断从属节点的后续请求和响应

图 25 — 仅诊断模式的用例

仅诊断模式应通过外部测试工具的诊断服务请求启用和禁用（例如，ISO 14229-7 中的“通信控制”服务禁用 LIN 集群上的正常通信会导致激活“仅诊断模式”）。在仅诊断模式下运行时，如果没有任何主动传输，主节点应执行诊断从属响应计划表，以防止从属节点进入睡眠模式（见图26 ）。

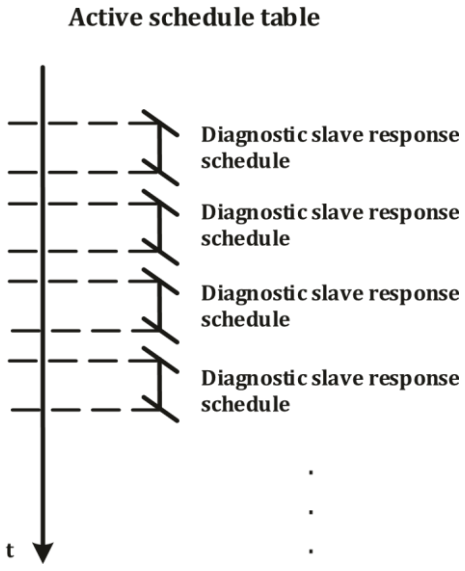


图 26 — 仅诊断模式下的默认计划

9.7 传输处理器要求

9.7.1 一般规定

9.7.2中指定的传输处理器的一个实例。传输处理器应能够在交叉诊断模式或仅诊断模式下运行。

每个集群至少可以处理一个活动主节点到从节点的物理传输加上一个功能传输。

无论当前活动的连接如何，始终可以向主节点的所有 LIN 集群进行广播。

禁止在没有任何事先请求的情况下从从节点发出异步响应。但是，具有引导加载程序功能的从节点可以确认传输层/诊断层在同一运行时上下文中尚未收到的编程请求。进入编程会话时，在从节点执行物理重置以进入引导加载程序后，将发送肯定响应。

9.7.2 主节点传输处理程序

主节点应按照图 27实现传输处理程序。调度完全由应用程序控制。两种模式均定义了以下状态：

- 闲置的：

  在此状态下，主节点既不接收也不发送 LIN 集群上的任何传输。它可连续接收任何新的传输请求。

— Tx功能活跃：

  在此状态下，主节点将功能寻址请求从骨干总线路由到 LIN 集群。根据 LIN 传输协议的限制，这只能是单帧 (SF)。

— Tx 物理活动：

  在此状态下，主节点当前正在将数据从主干总线路由到 LIN 集群中的一个从属节点。因此，主节点被占用，无法将任何其他物理传输从主干总线路由到 LIN 集群。此外，从属节点的响应无法路由到主干总线。

— Rx 身体活动：

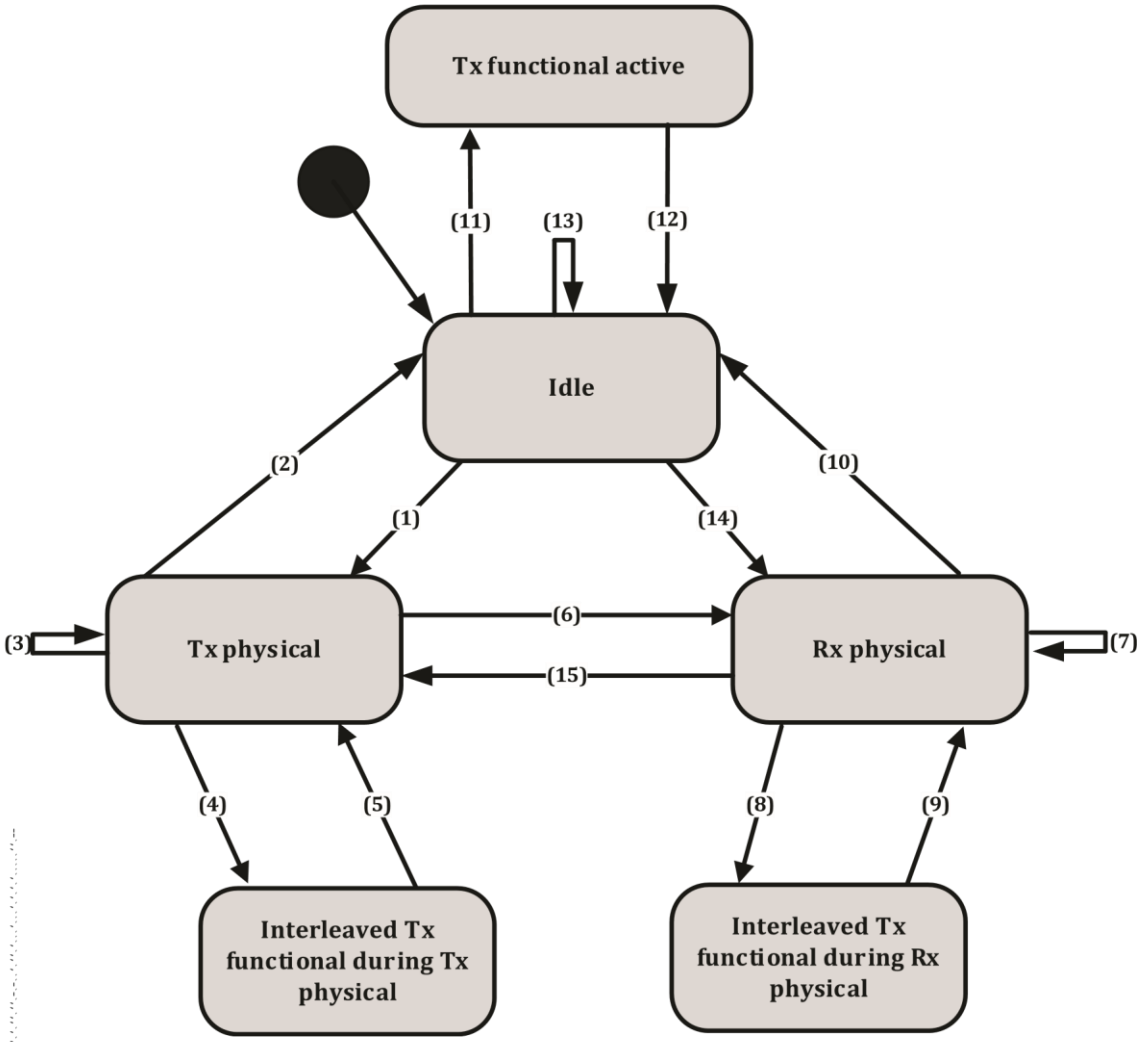
  在此状态下，主节点将传输从从节点路由到骨干总线。可以将功能寻址请求传输到 LIN 集群，但无法处理到从节点的进一步物理传输。

— Tx期间的交叉功能：

  在此状态下，主节点将功能寻址请求从骨干总线路由到 LIN 集群，同时向从节点的传输当前处于活动状态。功能寻址单帧 (SF) 可以传输，但在接收物理寻址传输时，活动从节点应忽略该帧。

— Rx 期间的交叉功能：

  在此状态下，主节点将功能寻址请求从骨干总线路由到集群，同时从从节点的接收当前处于活动状态。可以传输功能寻址单帧 (SF)，但在传输物理寻址响应时，活动从节点应忽略该帧。



Key

1 空闲状态 → Tx物理活动状态
— 触发器：从主干总线到从属节点的物理传输的开始。
— 作用：开始执行诊断主请求帧调度表并处理传输协议。

2 Tx物理活动状态→空闲状态
— 条件：从主干总线到集群的物理传输路由已完成，或者主干总线上发生传输协议传输错误（例如超时）。
— 操作：停止执行主请求帧调度表。

3 Tx 物理活跃状态 → Tx 物理活跃状态
— 条件：在操作前一个物理请求时触发了一个新的物理请求。
— 操作：接受新的传输请求，并停止先前待处理的物理请求。

4 Tx 物理活动状态 → Tx 物理状态期间交叉功能请求 — 条件：已收到来自主干总线的功能寻址请求。
— 操作：中断到从属节点的物理传输，并执行单个主请求帧调度，将功能寻址请求传输到集群上。

5 Tx 物理状态 → Tx 物理活动状态期间交叉功能 — 条件：功能寻址请求已路由到集群。
— 动作：继续向从节点进行中断的物理传输。

6 Tx 物理活动状态 → Rx 物理活动状态
— 条件：到从节点的物理传输已成功完成。
— 操作：停止执行主请求帧调度表，开始执行从属响应帧调度表并处理来自先前寻址的从属节点的传入响应。

7 Rx 物理活动状态 → Rx 物理活动状态
— 条件：来自从属节点的响应尚未启动或尚未完成，或者已收到响应待处理帧。
— 动作：传输从属响应帧调度表并处理 LIN 传输协议（即将传输从从属节点路由到主干总线）。

8 Rx 物理活动状态 → Rx 物理状态期间交叉功能请求 — 条件：已收到来自主干总线的功能寻址请求。
— 操作：中断从属响应帧调度的调度，并执行单个主请求帧调度表，以将功能寻址请求传输到集群上。

9 Rx 物理状态 → Rx 物理活动状态期间交叉功能 — 条件：功能寻址请求已路由到集群。
— 操作：重新开始执行从属响应帧计划并继续从从属节点中断的接收。

10 Rx物理活动状态→空闲状态
— 条件：从从属节点的接收已完成，或者在主干总线上发生传输协议错误，或者超时 P2 max 各自的 P2* max 已过去（根据ISO 14229-2 中定义的NRC 78 $_{16}$处理）。
— 操作：在“诊断交错模式”下，停止执行从属响应帧调度并恢复应用程序调度表。在“仅诊断模式”下，从属响应帧调度将继续，直到从 IDLE 模式开始新的传输。

11 空闲状态 → Tx功能激活状态
— 条件：已收到来自主干总线的功能寻址请求。
— 操作：执行单个主请求框架调度表，将功能寻址请求传输到集群上。

12 Tx功能活动状态→空闲状态
— 条件：功能寻址请求已被路由到集群。
— 行动：停止执行主请求框架计划。

13 空闲状态 → 空闲状态
— 条件：既没有从主干总线路由到从属节点的物理传输，也没有从从属节点路由到主干总线的任何响应。
— 操作：诊断交错模式。不执行任何主请求帧调度表或从响应帧调度表。仅诊断模式：执行从响应帧调度表。

14 空闲状态 → Rx 物理活动状态（仅适用于“仅诊断模式”）
— 条件：从属节点正在通过其中一个从属响应帧调度表向主节点的所有 LIN 集群广播发起的传输。
— 操作：处理来自响应从属节点的传入响应，并开始路由到主干总线。

15 无论当前活动连接如何，始终可以向主节点的所有 LIN 集群进行广播或在等待响应时传输新请求。当请求新的物理传输时，任何待处理的物理接收都会被拒绝。

图 27 — 主节点传输处理程序

仅当 SF 或 FF 中收到的 NAD 与先前物理请求的传输 NAD 匹配时，主机才应接受物理接收。

如果接收过程中出现错误，则应忽略此 NAD 的另一个响应，直到重新发出物理请求。

应接受具有相同 NAD 的多个有效物理响应，以支持 UDSonLIN 中定义的响应待处理帧，如 ISO 14229-1 和 ISO 14229-2 中所述。

9.7.3 从节点传输处理程序

图 28中定义的传输处理程序。这是为了允许在集群上进行无帧冲突的诊断通信。在诊断期间，通常不使用广播 NAD 。如果发生这种情况，从属节点将使用广播 NAD（7F $_{16}$）处理请求，就像它是从属节点配置的 NAD 一样。

定义了以下状态。

- 闲置的：

  在此状态下，从属节点不会在集群中接收或传输任何消息。它正在接受来自主节点的传入请求。它不会响应从属响应标头。

— 接收物理请求：

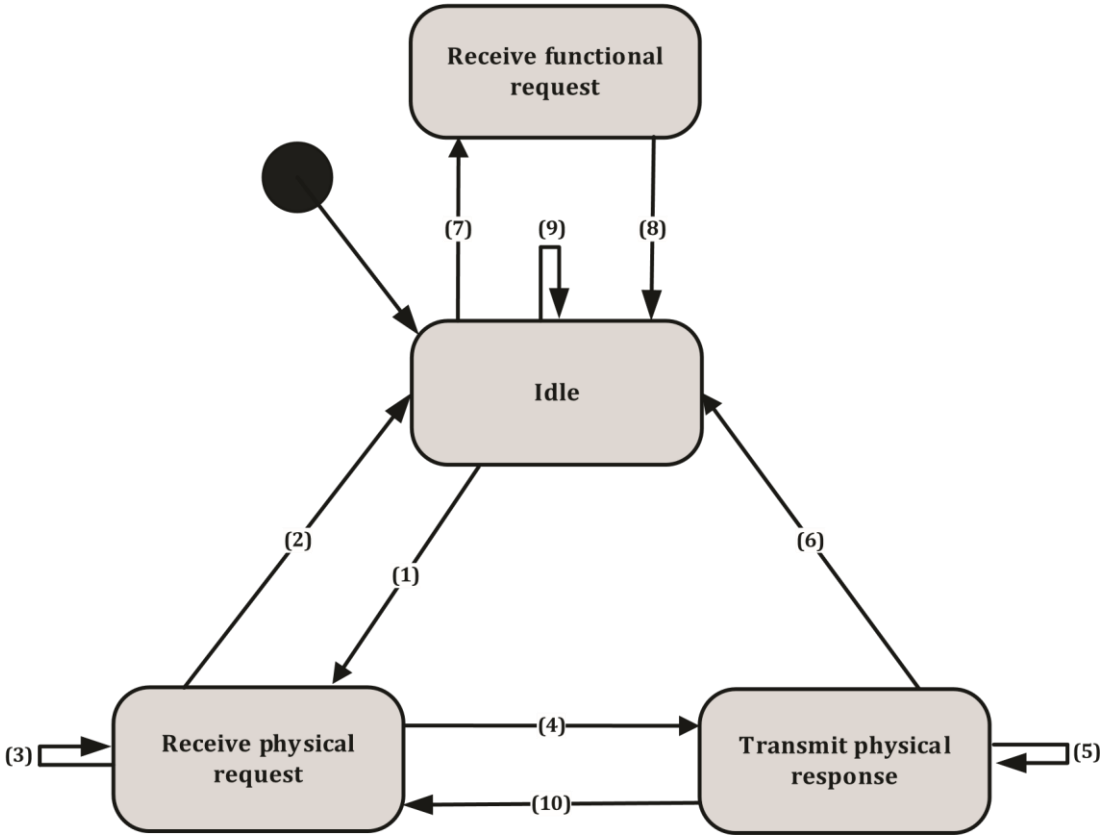
在此状态下，从节点正在接收和处理来自主节点的传输层帧。从节点应忽略来自主节点的任何交叉功能寻址请求。

— 发送物理响应：

在此状态下，从属节点当前仍在处理先前收到的请求，准备传输物理响应或实际上正在传输对先前收到的请求的响应。从属节点应忽略来自主节点的交叉功能寻址（NAD $7E_{16}$）请求。应接收新的物理请求并使从属节点丢弃当前请求或响应。如果新请求是发往从属节点的，则应处理该请求。

— 接收功能请求：

在此状态下，从节点正在接收来自主节点的功能传输。从节点不应响应任何从节点响应头。



Key

1 空闲状态→接收物理请求状态

— 条件：已收到主请求帧，其 NAD 与从属节点配置的 NAD 匹配。

— 操作：根据传输层要求开始处理物理请求。

2 接收物理请求状态→空闲状态

— 条件：发生了传输层错误，或收到了具有与从属节点配置的 NAD 不同的 NAD 的主请求帧，或收到了完全不需要响应的物理请求。

— 操作：停止处理物理请求。不响应从属响应标头。

3 接收物理请求状态 → 接收物理请求状态

— 条件：物理请求尚未完全接收，并且接收到新的主请求帧 (CF)，其 NAD 设置为从节点配置的 NAD。物理请求尚未完全接收，并且接收到新的物理请求，其 NAD 设置为从节点配置的 NAD。应忽略功能寻址请求。

— 操作：继续/重新启动物理请求。

4 接收物理请求状态→发送物理响应状态——条件：物理请求已完全接收。

— 操作：处理诊断请求。如果在处理前一个请求时收到新的物理请求，且该请求的 NAD 设置为从属节点配置的 NAD，则从属节点应丢弃当前请求或响应数据，并开始接收新请求。

5 传输物理响应状态 → 传输物理响应状态

— 条件：物理响应尚未完全传输。功能寻址请求应被忽略。

— 动作：按照传输层的要求继续响应从站响应帧。

6 传输物理响应状态→空闲状态

— 条件：物理响应已完全传输，发生 LIN 传输层错误或已收到 NAD 设置为与从节点配置的 NAD 不同的请求。— 操作：丢弃请求和响应数据。停止响应从属响应帧。

7 空闲状态→接收功能请求状态

— 条件：已收到 NAD 参数设置为功能 NAD 的主请求帧。 — 操作：根据传输层接收并处理主请求帧。不响应从属响应帧头。

8 接收功能请求状态→空闲状态

— 条件：功能请求已被处理。

— 操作：丢弃所有响应数据。停止响应从站响应帧。

9 空闲状态 → 空闲状态

— 条件：未收到请求且无待处理的响应。

— 动作：不响应任何从属响应帧。

10 发送物理响应状态 → 接收物理请求状态

— 条件：之前的响应传输尚未完全处理，并且收到了带有从属节点配置的 NAD 的新物理诊断主请求帧。

— 操作：丢弃响应数据。根据 LIN 传输协议要求开始接收和处理物理请求。

图 28 — 从节点传输处理程序

9.8 诊断服务优先级

在 LIN 网络中，MasterReq 和 SlaveResp 帧用于诊断通信，其中使用配置的 NAD 寻址从节点。除了会话层 UDS 诊断服务之外，本文档还指定了在数据链路层提供的节点配置和识别服务，以及会话/应用层上 NAD 范围 80_{16} - FF_{16}中的专有服务。由于这三个“服务实例”都访问相同的帧资源 MasterReq 和 SlaveResp，此外 LIN 仅限于半双工通信，因此可能会发生冲突，即一个实例的活动服务被另一个实例请求中断。

在网络设计层面，应通过为不同服务实例的使用定义不同的时间段来解决这些冲突，例如，如果从属节点重新配置，则不应使用 ISO 14229-7 UDSonLIN 服务。在 LIN 集群的正常生命周期内，不应使用专有服务。

为了解决主从节点数据链路层的这些冲突，提供了一种优先级方案，即优先级较高的服务会中断优先级较低的服务。

a) 节点配置和识别服务具有最高优先级。任何其他诊断通信都会中断。请注意，使用 AssignNAD 服务会将新配置的 NAD 分配给从属节点。这

b) ISO 14229-7 UDSonLIN 诊断服务具有中等优先级。这些服务在没有节点配置和识别服务处于活动状态时运行。

c) 专有服务 (NAD ≥ 80_{16}) 优先级较低。仅当没有其他诊断通信处于活动状态时，才会运行这些服务。

10 LIN节点能力语言(NCL)

10.1 一般规定

节点能力语言的目的是能够以标准化、机器可读的语法描述从属节点的可能性。

预计未来几年内，预制的现成从属节点的可用性将会增加。如果它们都附带一个节点功能文件 (NCF)，则可以为主节点生成 LIN 描述文件 (LDF)（参见第 12 条）和初始化代码（配置集群，例如重新配置冲突的帧标识符）。

如果任何集群的设置和配置都是全自动的，那么 LIN 即插即用开发就向前迈出了一大步。换句话说，集群中分布式节点的使用就和单 CPU 节点一样简单，物理设备直接连接到节点。

10.2 即插即用工作流程概念

10.2.1 一般规定

LIN 工作流概念允许实现无缝的设计和开发工具链，并提高开发速度和 LIN 集群的可靠性。

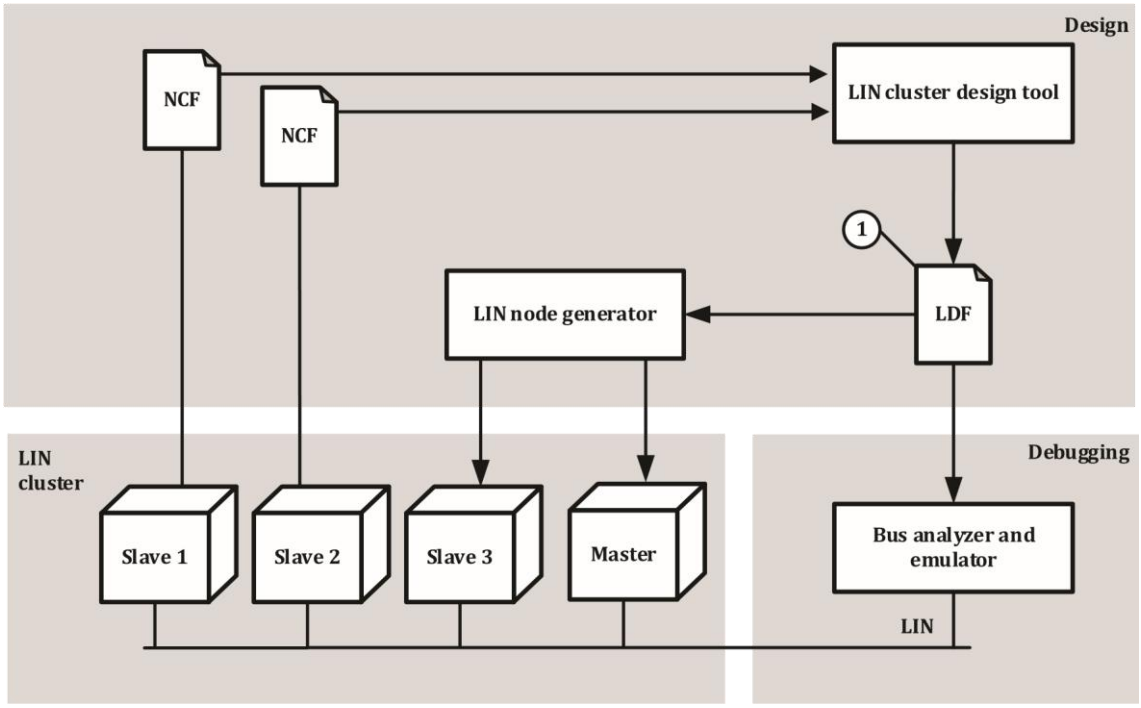
LDF 规范允许安全地分包节点，而不会因信息不兼容或网络过载等原因而危及 LIN 系统功能。它还是一种用于调试 LIN 集群（包括模拟未完成节点）的强大工具。

节点能力语言规范为现成的从属节点规范提供了标准化语法。这简化了标准从属节点的采购，并为自动生成节点的工具提供了可能性。因此，集群中从属节点的真正即插即用成为现实。

从节点连接到主节点，形成 LIN 集群。相应的节点功能文件由 LIN 集群设计工具解析，以在 LIN 集群设计过程中生成 LIN 描述文件 (LDF)。LDF 由 LIN 节点生成器解析，以在所需节点（图 29所示示例中的主节点和从节点 3 ）中自动生成 LIN 相关功能。

LIN 总线分析器/仿真器工具也使用 LDF 来进行集群调试。

图 29显示了集群开发分为三个方面：设计、调试和 LIN 物理集群。本规范侧重于设计阶段。



钥匙

1 LIN描述文件

图 29 — LIN 集群的开发

10.2.2 LIN 节点生成

LIN 集群的核心描述文件是 LIN 描述文件 (LDF)。基于此文件，可以生成集群中所有节点的通信驱动程序，此过程称为 LIN 节点生成。所有信号、帧以及主节点的调度表均在此文件中声明。

10.2.3 LIN 集群设计

创建 LDF 文件的过程称为 LIN 集群设计。当您设计一个全新的集群时，编写 LDF 文件（手动或借助计算机）是定义集群通信的有效方法。

但是，当您有现有的从属节点并想从头开始创建它们的集群时，就不那么方便了。如果定义的集群包含从属节点地址冲突或帧标识符冲突，则尤其如此。

通过接收带有每个现有从属节点的节点功能文件 NCF，LIN 集群设计步骤是自动的：只需将 NCF 文件添加到 LIN 集群设计工具中的项目中，它就会生成 LDF 文件。

如果您还想创建新的从属节点（图 29 ，从属节点 3），则过程会变得稍微复杂一些。要执行的步骤取决于所使用的 LIN 集群设计工具，该工具不属于 LIN 规范。一个有用的工具允许在生成 LDF 文件之前输入其他信息。（始终可以为不存在的从属节点编写一个虚构的 NCF 文件，因此将其包括在内。）

值得注意的是，生成的 LDF 文件反映了配置的网络；在激活集群流量之前，或者在应用程序通信之前通过节点配置服务（初始化计划表包含用于解决现有冲突的计划表命令）解决从属节点或框架之间原来的任何冲突。

10.2.4 调试

调试和节点仿真基于 LIN 集群设计中产生的 LDF 文件。

主机仿真增加了集群配置无冲突的要求。因此，仿真工具应能够读取 LIN 集群设计工具生成的重新配置数据。

11 节点能力文件(NCF)

NCF 为至少一个从属节点提供能力定义。

11.1 NCF语法概述

表 20所示。

表 20—NCF 使用的 BNF 语法

象征	意义
::=	::= 左侧的名称使用其右侧的语法来表达
<>	用于标记稍后指定的对象
|	竖线表示选择。竖线左侧或右侧应出现
bold	粗体文本是保留的，因为它是保留字，或者是强制标点符号
[ ]	方括号内的文字应出现一次或多次
( )	括号内的文字为可选，即应出现一次或零次
char_string	任何用引号括起来的字符串“像这样”
identifier	标识符。通常用于命名对象。标识符应遵循变量声明的正常 C 规则

integer	整数。整数可以是十进制或十六进制（ 16 ）格式。
real_or_integer	实数或整数。实数始终为十进制，且带有嵌入的小数点。

在使用此语法的文件中，注释可出现在任何地方。注释语法与 C++ 的语法相同，从 // 到行尾的任何内容以及包含在 /* 和 */ 分隔符中的任何内容都将被忽略。

保留文本和标识符区分大小写。

11.2 全局结构定义

节点能力文件的语法定义如下。

<node_capability_file_def>
<language_version_def>
(NCF_file_revision_def)
(LIN_sig_byte_order_big_endian_def)
[<node_definition>]

11.2.1 Nodecapabilityfilemarker

<node_capability_file_def> ::= node_capability_file; node_capability_file标签用于将该文件声明为节点能力文件（NCF）。

11.2.2 语言版本号定义

<language_version_def> ::=
LIN_language_version = char_string;

LIN_language_version应为“ISO17987:2015”。

11.2.3 NCF 修订

NCF_file_revision_def ::=

NCF_file_revision = <ncf_revision>;
<ncf_revision> ::= "int_major.int_minor.int_sub";

此修订号 ncf_revision 用于检查从属节点与 LIN 集群的一致性。通信定义中的更新（新信号、更改的信号映射等）由int_major、int_minor或int_sub的无符号 8 位整数值的增量表示。数据链路层通过 ReadByIdentifier B2 16服务将此修订版本提供给网络和应用程序。有关更多详细信息，请参阅 ISO 17987-3:2016，6.3.6.6。

11.2.4 大端信号编码变体

<LIN_sig_byte_order_big_endian_def> ::=
LIN_sig_byte_order_big_endian;

如果 NCF 中定义了此可选标签，则信号将按大端顺序映射到帧。有关更多信息，请参阅 ISO 17987-3:2016，5.2.2.6。

11.3 节点定义

<node_definition> ::= node
<node_name>{ <general_definition>

```
    <diagnostic_definition>
<frame_definition>     <encodin
g_definition>     <status_manag
ement>     (<free_text_definiti
on>)    } <node_name> ::= identi
fier
```

如果节点能力文件包含多个从属节点，则节点名称在文件中应是唯一的。声明的从属节点应被视为物理从属节点实例的类（模板）。

node_definition的属性在以下子条款中定义。

11.3.1 通用节点定义

```
<general_definition> ::=    general {
    LIN_protocol_version = <protocol_version>;     sup
plier = <supplier_id>;     function = <function_id>;
variant = <variant_id>;     bitrate = <bitrate_definit
ion>;     sends_wake_up_signal = "yes" | "no";    }
```

general_definition声明了指定与集群的一般兼容性以及一般节点属性的属性。

11.3.1.1 LIN协议版本号定义

```
<protocol_version> ::= char_string
```

指定从属节点使用的协议，范围应为“ISO17987:2015”、“2.2”、“2.1”、“2.0”、“<SAE J2602 协议版本标签>”

注意： 由于 NCF 定义以 LIN 2.0 开头，因此 LIN 1.3 不在列表中。

11.3.1.2 LIN产品识别

```
<supplier_id> ::= integer
<function_id> ::= integer
<variant_id>  ::= integer
```

supplier_id以 16 位数分配给每个供应商。function_id是供应商分配给产品的 16 位数，以使其唯一。最后， variant_id是一个 8 位值，用于指定变体，请参阅 ISO 17987-3。

11.3.1.3 比特率

```
<bitrate_definition> ::=
  automatic (min <bitrate>) (max <bitrate>) |   select
{<bitrate> [, <bitrate>]} |   <bitrate>
```

三种可能的bitrate_definition ：

- 自动的，

  从属节点可以采用总线上使用的任何合法比特率。如果添加了 min 和/或 max 字样，则可以使用从提供的比特率开始/直到提供比特率的任何比特率。

- 可选择，

  如果使用列出的比特率之一，则从属节点可以检测到比特率，否则失败。

- 固定的，

  仅可使用一种比特率。

鼓励标准化、现成的从属节点制造商构建自动从属节点，因为这为集群构建者提供了最大的灵活性。

<bitrate> ::= real_or_integer kbps
比特率指定在 1 kbit/s 至 20 kbit/s 范围内。

11.3.1.4 发送唤醒信号

如果从设备能够发送唤醒信号，则此参数设置为“yes”。否则，设置为“no”。

11.3.2 诊断定义

<diagnostic_definition> ::= diagnostic {
NAD = integer ([, integer]); | (integer to integer); diagn
ostic_class = integer; (P2min = real_or_integer ms;)
(STmin = real_or_integer ms;)
(N_As_timeout = real_or_integer ms;)
(N_Cr_timeout = real_or_integer ms;)
(support_sid { integer ([, integer]) };)
(max_message_length = integer;) }
Diagnostic_definition指定传输层和配置的属性。

NAD 属性定义初始节点地址；该值应根据
ISO 17987-3:2016，6.3.4.2。可以给出值列表或范围。范围是包含的，即两个值都包含在范围内。
如果给出了多个值，从属设备应根据物理属性动态选择给定 NAD 集内的其中一个值。

诊断类别定义支持的 I、II 或 III 类。

min的默认值在 ISO 14229-2 中指定为参数 P2 Server 。

min的默认值在9.4中指定。

N_As_timeout 、 N_Cs_timeout和N_Cr_timeout的默认值定义在
表 12 .

上述时序参数仅与诊断 II 类和 III 类从属节点相关。

max_message_length属性仅适用于诊断传输层；它定义诊断消息的最大长度。默认值定义为 4095。

support_sid列出了从属节点支持的所有 SID 值（节点配置、标识和诊断服务）。默认值为ISO 17987-3 规定的强制节点配置和标识服务 B2 16和 B7 16 。

11.3.3 框架定义

<frame_definition> ::= frames {
[<single_frame>] }
列出的帧应为所有由从属节点处理的无条件帧和事件触发帧。事件触发帧是指事件触发帧头，因此不包含任何信号。诊断帧应始终受支持，因此未列出。

<single_frame> ::=
<frame_kind> <frame_name> { <frame_propertie
s> (<signal_definition>)

```
  }
<frame_kind> ::= publish | subscribe
<frame_name> ::= identifier
```

发布或订阅的帧都按上述定义进行声明。frame_name是帧的符号名称。frame_kind从从属节点的角度确定（例如，传输的帧应为已发布帧）。

11.3.3.1 框架属性

```
<frame_properties> ::=   frame_length = i
nteger;   (min_period = integer ms;)
  (max_period = integer ms;)
  (event_triggered_frame = identifier;)
```

frame_length是 LIN 帧的长度（1 到 8）。

min_period和max_period的可选值用于指导工具生成计划表。

event_triggered_frame指的是事件触发帧，如果所描述的帧与其相关联。

当帧也由事件触发时，会应用一些限制，请参阅 ISO 17987-3:2016，5.2.4.3。

11.3.3.2 信号定义

```
<signal_definition> ::=   signals {
    [<signal_name> { <signal_properties> }]
  } <signal_name> ::= identifier
```

所有帧（诊断帧除外）都携带信号，这些信号根据signal定义进行声明。

```
<signal_properties> ::=   <init_value>
  size = integer;   offset =
integer;   (<encoding_nam
e>;)
<init_value> ::= <init_value_scalar> | <init_value_array>
<init_value_scalar> ::= init_value = integer
<init_value_array> ::= init_value = {integer ([, integer ])}
```

init_value指定从通电到首次由发布应用程序设置期间信号使用的值。init_value_scalar用于标量信号， init_value_array用于字节数组信号。init_value_array按大端顺序给出。

大小是为信号保留的位数，偏移量指定信号在帧中的位置（LIN 默认信号编码：帧中响应信号 LSB 的位数，参见 ISO 17987-3：2016，图 12，LIN 大端信号编码变体：帧中响应信号 MSB 的位数，参见 ISO 17987-3：2016，图 13）。

对于字节数组，大小和偏移量都应为八的倍数。

描述大小为 8 或 16 的信号是具有一个或两个元素的字节数组还是标量信号的唯一方法是通过分析 init_value ，即花括号对于区分数组和标量值非常重要。

encoding_name是对编码子句中定义的编码的引用，参见11.3.4 。

11.3.4 信号编码类型定义

编码旨在提供信号的表示和缩放属性。

```
<encoding_definition> ::=   encoding {
    [<encoding_name> {
      [<logical_value> |
      <physical_range> |
      <bcd_value>      |
      <ascii_value>]
    }]
  }
<encoding_name>  ::= identifier
<logical_value>  ::= logical_value, <signal_value> (, <text_info>); <physical_rang
e> ::= 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
```

signal_值、 min_值和max_值应在0至65 535的范围内。

max_value应大于或等于min_value 。如果原始值在 min 和 max 定义的范围内，则物理值应按照公式 (1)的定义计算：

物理值= （比例 r* aw_value ） +偏移量 （1）

11.3.5 状态管理

```
<status_management> ::=   status_managemen
t {     response_error = identifier;
    (fault_state_signals = identifier ([, identifier]);)   } <pu
blished_signal> ::= identifier
```

status_management指定主节点应监视哪些已发布的信号以确定从属节点是否按预期运行。

标识符各自引用信号定义中一个唯一的已发布信号，参见12.3.2 。参见 ISO 17987-3:2016, 5.5.4中response_error信号的定义和 ISO 14229-7:2015, 6.4.1 中故障状态信号的定义。

，旧式 LIN 2.0 从属节点也会设置response_error信号。这与后来的 LIN 标准修订 [LIN 2.1、LIN 2.2(A) 和 ISO 17987] 相结合。这可以在网络设计中考虑。

11.3.6 自由文本定义

```
<free_text_definition> ::=
free_text {     char_strin
g   }
```

free_text_definition用于在 LIN 集群设计工具中调出帮助文本、限制等。

自由文本定义中提供的典型信息是

— 从属节点的目的和物理世界交互，例如电机速度、功耗等，以及 — 与 LIN 标准的偏差。

11.4 NCF 示例

节点能力文件：

```
LIN_language_version = "ISO17987:2015";
NCF_file_revision = "1.07.04"; node step_motor
{   general {
    LIN_protocol_version = "ISO17987:2015";     supplier = 0x000
5; function = 0x0020; variant = 1;     bitrate = automatic min 1
0 kbps max 20 kbps;
    sends_wake_up_signal = "yes";
  }   diagnostic
{
    NAD = 1 to 3;     diagnostic_class =
2;
    P2_min = 100 ms;      ST_min
= 40 ms;
    support_sid { 0xB0, 0xB2, 0xB7 };
  }   frame
s {
    publish node_status {       length = 4; min_period = 10 ms; max_p
eriod = 100 ms;       signals {
        state       {init_value = 0; size = 8; offset = 0;}         fault_state {ini
t_value = 0; size = 2; offset = 9; fault_enc;}
        error_bit   {init_value = 0; size = 1; offset = 8;}
        angle       {init_value = {0x22, 0x11}; size = 16; offset = 16;}
      }
}
    subscribe control {       length = 1; max_period = 100 ms;       signals
{         command {init_value = 0; size = 8; offset = 0; position;}       }
    }
  }
  encoding {
```

```
    position {physical_value, 0, 199, 1.8, 0, "deg";}      fault_enc
{logical_value, 0, "no result";                 logical_value, 1, "f
ailed";                logical_value, 2, "passed";}
  }
  status_management { response_error = error_bit;                        fault_stat
e_signals = fault_state; }
  free_text { "step_motor signal values outside 0 - 199 are ignored" } }
```

12 LIN描述文件(LDF)

12.1 一般规定

本文档中描述的语言用于创建 LIN 描述文件。LIN 描述文件描述了一个完整的集群，还包含监控集群所需的所有信息。如果一个或多个节点不可用，这些信息足以对其进行有限的模拟。

LIN 描述文件可以作为组件之一，用于编写电子控制单元的软件，该电子控制单元应是集群的一部分。参考文献 [2] 中描述了应用程序接口 (API)，以便以统一的方式从不同的应用程序访问集群。然而，LIN 描述文件并未解决应用程序的功能行为。

LIN 描述文件的语法非常简单，可以手动输入，但鼓励开发和使用基于计算机的工具。如 11 中所述，节点功能文件提供了一种 (几乎) 自动生成 LIN 描述文件的方法。同一规范还给出了集群开发中可能工作流程的示例。

12.2 LDF 语法概述

表 21所示。

表 21—LDF 中使用的 BNF 语法

象征	定义
::=	::= 左侧的名称使用其右侧的语法来表达
<>	用于标记稍后指定的对象
|	竖线表示选择。竖线左侧或右侧应出现
bold	粗体文本是保留的 - 因为它是保留字，或者是强制标点符号
[ ]	方括号内的文字应出现一次或多次
( )	括号内的文字为可选，即应出现一次或零次
char_string	任何用引号括起来的字符串“像这样”

identifier	标识符。通常用于命名对象。标识符应遵循变量声明的正常 C 规则
integer	整数。整数可以是十进制或十六进制 ($_{16}$) 格式。
real_or_integer	实数或整数。实数始终为十进制，且带有嵌入的小数点。

在使用此语法的文件中，注释可出现在任何地方。注释语法与 C++ 的语法相同，从 // 到行尾的任何内容以及包含在 /* 和 */ 分隔符中的任何内容都将被忽略。

保留文本和标识符区分大小写。

12.3 LDF定义

LDF 的语法在12.2中指定。

12.3.1 全局结构定义

```
<LIN_description_file_def>
<LIN_protocol_version_def>
<LIN_language_version_def>
<LDF_file_revision_def>
<LIN_speed_def>
(<Channel_name_def>)
(<LIN_sig_byte_order_big_endian_def>)
<Node_def>
<Signal_def>
(<Diagnostic_signal_def>) <Frame_def>
(<Sporadic_frame_def>)
(<Event_triggered_frame_def>)
(<Diagnostic_frame_def>) <Node_attributes_def>
<Schedule_table_def>
(<Signal_encoding_type_def>) (<Signal_representatio
n_def>)
```

LIN描述文件的总体语法应如上。

12.3.1.1 LIN描述文件标记

```
<LIN_description_file_def> ::= LIN_description_file;
```

LIN_description_file标签用于将文件声明为 LIN 描述文件 (LDF)。

12.3.1.2 LIN协议版本号定义

```
<LIN_protocol_version_def> ::=
  LIN_protocol_version = char_string;
```

LIN_protocol_version定义 LIN 主协议版本，范围为“ISO17987:2015”、“2.2”、“2.1”、“2.0”、“1.3”、“<J2602 协议版本标签>”。主协议版本应等于或高于集群中定义的任何从属节点。

12.3.1.3 LIN语言版本号定义

```
<LIN_language_version_def> ::=
  LIN_language_version = char_string;
```

LIN_language_version应为“ISO17987:2015”。

12.3.1.4 LIN 文件修订号

```
<LDF_file_revision_def> ::=
  LDF_file_revision = "int_major.int_minor.int_sub";
```

此修订号用于检查主节点和从节点与 LIN 集群的一致性。通信定义中的更新（新信号、更改的信号映射等）由 int_major、int_minor 或 int_sub 的无符号 8 位整数值的增量表示。数据链路层通过 ReadByIdentifier B2 $_{16}$服务（从节点）向网络提供此修订版本，并向应用程序（主节点和从节点）提供此修订版本。有关更多详细信息，请参阅 ISO 17987-3:2016，6.3.6.6。

12.3.1.5 LIN速度定义

```
<LIN_speed_def> ::=
  LIN_speed = real_or_integer kbps;
```

此项设置集群的标称比特率。其范围应为 1 kbit/s 至 20 kbit/s。

12.3.1.6 频道后缀名称定义

```
<Channel_name_def> ::=  Channel_name = i
dentifier;
```

LDF 中所有命名对象的后缀。对于连接到多个集群的主节点，后缀是必需的。如果节点连接到多个集群（即使用多个 LDF），则使用它来避免命名冲突。如果给定，所有命名对象都应将此后缀添加到其名称中。

后缀名称需在命名对象前添加下划线“_”。

LDF 中的信号名称为“signal1”且Channel_name = “net1”，则生成的信号名称为“ signal1_net1 ”。

12.3.1.7 大端信号编码变体

```
<LIN_sig_byte_order_big_endian_def> ::=
  LIN_sig_byte_order_big_endian;
```

如果在 LDF 中定义了此可选标签，则信号将按大端顺序映射到帧。有关更多信息，请参阅 ISO 17987-3:2016，5.2.2.6。

12.3.2 信号定义

12.3.2.1 一般规定

信号定义子条款确定了集群中所有信号的名称及其属性。本子条款中的定义创建了一个信号标识符集。该集合中的所有标识符均应唯一。

```
<Signal_def> ::=
  Signals {
    [<signal_name>: <signal_size>, <init_value>, <published_by>[
                  ,<subscribed_by>];]
  }
<signal_name> ::= identifier
```

All signal_name identifiers shall be unique within the signal identifier set.

```
<signal_size> ::= integer signal_size
```

指定信号的大小。对于标量信号，其范围应为 1 位到 16 位；对于字节数组信号，其范围应为 8、16、24、32、40、48、56 或 64。

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

init_value指定所有用户节点应使用的信号值，直到收到包含该信号的帧。init_value_scalar用于标量信号， init_value_array用于字节数组信号。字节数组的initial_value应按大端顺序排列（即最高有效字节在前）。

描述大小为 8 或 16 的信号是具有一个或两个元素的字节数组还是标量信号的唯一方法是通过分析 init_value ，即花括号对于区分数组和标量值非常重要。

<published_by> ::= 标识符 <subscribed_by> ::= 标识符

posted_by标识符和subscribed_by标识符都应存在于节点标识符集中。

12.3.2.3 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;
    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/订阅者信息是预定义的，因此诊断信号在 LIN 描述文件中有一个单独的子句。

12.3.3 框架定义

12.3.3.1 一般规定

框架定义标识集群中所有框架的名称及其属性。定义创建框架标识符集（其符号名称）和关联的框架 ID 集（框架标识符）。这些集合中的所有成员都应是唯一的。

12.3.3.2 无条件帧

```
<框架定义> ::=
框架 {
[<frame_name>: <frame_id>, <published_by>, <frame_size> {
[<信号名称>, <信号偏移>; ]
}]
}
```

<frame_name> ::= 标识符

所有frame_name标识符在帧标识符集内必须是唯一的。

<frame_id> ::= 整数

frame_id指定帧标识符号，范围为0至 59。帧标识符对于帧标识符集内的所有帧均应是唯一的。

<published_by> ::= 标识符

posted_by标识符应存在于节点标识符集中。

<帧大小> ::= 整数

frame_size指定帧的大小，范围为 1 字节至 8 字节。

<信号名称> ::= 标识符

signal_name标识符应存在于信号标识符集中。

published_by标识符中指定的同一节点发布。

<信号偏移> ::= 整数

该值的范围是 0 至 （ 8 * frame_size - 1 ）。 signal_offset取决于所使用的信号编码类型，参见 12.3.1.7 。

LIN 默认信号编码（小端字节序）符合 ISO 17987-3:2016，图 12： signal_offset值指定信号在帧中的最低有效位的位置。信号的最低有效位首先传输。

符合 ISO 17987-3:2016 的 LIN 大端信号编码变体，图 13： signal_offset值指定了信号在帧中最高有效位的所在位置。

示例（LIN 默认信号编码）表 22显示了在具有四字节数据字段的帧中映射的十位信号。信号 S 的 LSB 位于偏移量 16（ signal_offset ），MSB 位于偏移量 25。

表 22—信号包装

字节0								字节1								字节2								字节3							
																年代	年代	年代	年代	年代	年代	年代	年代	年代	年代						
0						7815	**16**							23	24							31							

最先发送 最后发送

12.3.3.3 零散帧

<零星帧定义> ::=

零星帧 {

[<不规则帧名称>: <帧名称> ([, <帧名称>]);] }

<sporadic_frame_name> ::= 标识符

所有sporadic_frame_name标识符在帧标识符集内必须是唯一的。

<frame_name> ::= 标识符

所有frame_name标识符都应存在于帧标识符集中并引用无条件帧。如果需要传输多个声明的帧，则应选择首先列出的帧（优先排序）。所有frame_name标识符都应是主节点发布的无条件帧。此外，它们不应直接在与sporadic_frame_name相同的调度表中作为无条件帧进行调度。

12.3.3.4 事件触发帧

<事件触发帧定义> ::=

事件触发帧 {

[<事件触发名称>:

<碰撞解决计划表>,
<帧 ID>
[, <框架名称>];]
}
<event_trig_frm_name> ::= 标识符
所有event_trig_frm_name标识符在帧标识符集内必须是唯一的。

<collision_resolving_schedule_table> ::= 标识符
这是指调度表集中的调度表。此调度应在碰撞后自动激活。它应至少包含相关的无条件帧。

<frame_id> ::= 整数
frame_id指定帧 ID 号，范围为 0 至 59。该 ID 对于帧 ID 集内的所有帧均应是唯一的。

<frame_name> ::= 标识符
所有frame_name标识符都应存在于帧标识符集中，并引用无条件帧。

评论

帧的第一个字节携带相关帧的受保护标识符 (PID)，因此不能用于其他目的。

12.3.3.5 诊断帧

< 诊断框架定义 > ::=
诊断框架 {
主请求： 60 {
MasterReqB0, 0;
MasterReqB1,8;
MasterReqB2，16;
MasterReqB3,24;
MasterReqB4,32;
MasterReqB5,40;
MasterReqB6,48;
MasterReqB7，56; }
从属响应: 61 {
SlaveRespB0， 0;
从属RespB1，8;
SlaveRespB2，16;
SlaveRespB3，24;
SlaveRespB4，32;
SlaveRespB5，40;
SlaveRespB6，48;
SlaveRespB7，56;
} }
MasterReq 和 SlaveResp 保留帧名称用于标识诊断帧（参见 ISO 17987-3:2016, 5.2.4.5）并且在帧标识符集中须是唯一的。

如果使用符合 ISO 17987-3:2016, 5.2.2.6.2 的 LIN 大端信号编码变体，则每个信号的 MSB 用于帧信号映射：

< 诊断框架定义 > ::=
诊断框架 {
主请求： 60 {
MasterReqB0,7；
MasterReqB1，15；…
MasterReqB7，63；}
从属响应: 61 {
SlaveRespB0, 7；
SlaveRespB1，15；…
SlaveRespB7，63；
}
}

12.3.4 节点定义

12.3.4.1 一般规定

节点定义子条款确定所有参与节点的名称，并指定主节点的时间基准和抖动。本子条款中的定义创建了一个节点标识符集。该集合中的所有标识符均应唯一。

12.3.4.2 参与节点

<节点定义> ::=
节点 {
主站：<node_name>，<time_base> 毫秒，<jitter> 毫秒；
从属：<节点名称>（[，<节点名称>]）；
} <节点名称> ::= 标识符

nodes 子句列出了参与集群的物理节点。所有node_name标识符在节点标识符集内都应唯一。

node_name标识符指定主节点。

<时间基准> ::= 实数或整数

time_base值指定主节点中用于生成最大允许帧传输时间的时间基准。时间基准应以毫秒为单位指定。

<抖动> ::= 实数或整数

抖动应以毫秒为单位指定。有关time_base和抖动的更多信息，请参阅12.3.4 。

12.3.4.3 ISO17987 :2015、LIN2.2、LIN2.1 从节点的节点属性

节点属性提供有关单个从属节点行为的所有必要信息。

<节点属性定义> ::=
节点属性 {
[<节点名称> {
LIN_protocol = <协议版本>;configured_NAD = <诊断地址>; (ini
tial_NAD = <诊断地址>;)
<属性定义>；
}]
} <节点名称> ::= 标识符

所有node_name标识符都应存在于节点标识符集内，并引用从属节点。

<协议版本> ::= char_string

应在“ISO17987:2015”、“2.2”、“2.1”范围内。

<诊断地址> ::= 整数

diag_address指定已识别从节点的诊断地址，范围在3.1.2中定义。它应指定在解决任何集群冲突后用于从节点的唯一 NAD，即它在集群内应是唯一的。

如果没有给出initial_NAD，则initial_NAD与configured_NAD相同。

<属性定义> ::=

product_id = <supplier_id>, <function_id> (, <variant>); response_error = <signal_na

me>;

（故障状态信号 = <信号名称>（[, <信号名称>]);）

（P2min = 实数或整数毫秒;）

（STmin = 实数或整数毫秒;）

（N_As_timeout = real_or_integer 毫秒;）

（N_Cr_timeout = 实数或整数毫秒;）

<可配置框架定义>

<供应商 ID> ::= 整数

<函数 ID> ::= 整数

<变量> ::= 整数

supplier_id 、 function_id和variant_id范围在 11.3.1.2中定义。

变体 ID 是可选的，因为它是从属节点而不是集群的属性。

<信号名称> ::= 标识符

response_error信号的signal_name 标识符应存在于信号标识符集内，并引用一位标准信号，参见12.3.2.2 。 response_error信号应由指定的从属节点发布。有关更多信息，请参阅11.3.5中的状态管理。

Fault_state_signals是 LIN 从站节点的属性，用于诊断等级 I 和 II，参见 ISO 14229-7:2015，6.4.1 。

min的默认值为50 ms，ST min的默认值为 0 ms，请参阅 ISO 14229-7 作为参数 P2 Server 。

N_As_timeout和N_Cr_timeout的默认值在表12中定义。

可配置帧应列出从属节点处理的所有帧（无条件帧、事件触发帧和零星帧）。

configurable_frames_def 的定义：

<可配置框架定义> ::= 可配置框架 {

[<框架名称>;]

}

帧的顺序很重要，因为节点配置请求 AssignFrameIdentifierRange 依赖于顺序，参见 ISO 17987-3:2016 6.3.6.5。

12.3.4.4 LIN2.0从节点的节点属性

节点属性提供有关单个节点行为的所有必要信息。

<节点属性定义> ::=
节点属性 {
[<节点名称> {
LIN_protocol = <协议版本>;configured_NAD = <诊断地址>;
（产品 ID = <供应商 ID>, <功能 ID>, <变量>;）
（response_error = <signal_name>;）(P2_min = real_or_
integer ms;）
（ST_min = real_or_integer ms;）(configurable_frames {
[<框架名称> = <消息ID>;]
}）
}] }

<节点名称> ::= 标识符
<协议版本> ::= "2.0"
<供应商 ID> ::= 整数
<函数 ID> ::= 整数
<变量> ::= 整数
<消息 ID> ::= 整数

所有node_name标识符都应存在于节点标识符集内，并引用从属节点。supplier_id应在 0_{16} .. $7FFE_{16范围内}$， function_id应在 0_{16} .. $FFFE_{16范围内}$， variant应在 0_{10} .. $255_{10范围内}$， message_id应在 0_{16} ..FFFE $_{16}$范围内。

<信号名称> ::= 标识符

response_error信号的signal_name标识符应存在于信号标识符集内，并引用一位标准信号，参见12.3.2.2 。 response_error信号应由指定的从属节点发布。有关更多信息，请参阅11.3.5中的状态管理。

<诊断地址> ::= 整数

diag_address指定已识别节点的诊断地址，范围为 1_{10}到 127_{10}，如 LIN 诊断和配置规范中进一步定义。它应指定在解决任何集群冲突后用于节点的唯一 NAD，即它在集群内应是唯一的。

configurable_frames应列出节点处理的所有帧及其关联的唯一标识符message_id 。

12.3.4.5 LIN1.3从节点的节点属性

节点属性提供有关单个节点行为的所有必要信息。

<节点属性定义> ::=
节点属性 {
[<节点名称> {
LIN_protocol = <协议版本>;configured_NAD = <
诊断地址>;}]
}
<节点名称> ::= 标识符
<协议版本> ::= "1.3"

12.3.4.6 J2602 从属节点的节点属性

节点属性提供有关单个节点行为的所有必要信息。

<节点属性定义> ::=
节点属性 {
[<节点名称> {
LIN_protocol = <协议版本>;configured_NAD = <诊断地址>;
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 {
[<框架名称> = <消息ID>;]
})
（响应容忍度 = 实数或整数 %;）
（wakeup_time = 整数毫秒;）
（开机时间 = 整数毫秒;）
}]
}
<节点名称> ::= 标识符
<协议版本> ::= “J2602_1_1.0”
<供应商 ID> ::= 整数
<函数 ID> ::= 整数
<变量> ::= 整数
<消息 ID> ::= 整数

所有node_name标识符都应存在于节点标识符集内，并引用从属节点。supplier_id应在 0_{16}.. $7FFE_{16}$范围内， function_id应在 0_{16}.. $FFFE_{16}$范围内， variant应在 0_{10}.. 255_{10}范围内， message_id应在 0_{16}..$FFFE_{16}$范围内。

<信号名称> ::= 标识符
signal_name标识符应存在于信号标识符集内，并引用三位信号。
该信号应由指定节点在每个非诊断传输帧中发布。有关详细信息，请参阅 SAE J2602 标准。

<诊断地址> ::= 整数

diag_address指定已识别节点的诊断地址，范围为 1_{10}到 127_{10}，如 LIN 诊断和配置规范中进一步定义。它应指定在解决任何集群冲突后用于节点的唯一 NAD，即它在集群内应是唯一的。

configurable_frames应列出节点处理的所有帧及其关联的唯一标识符message_id 。

response_tolerance指定节点特定的帧接收容差。默认值为 40 % 。

12.3.5 时间表定义

调度表描述了总线上传输的帧和帧的时序。调度表中的有效帧是 ISO 17987-3:2016, 5.2.4 中定义的帧类型（保留帧除外）和下面列出的节点配置命令。

<计划表定义> ::=
计划表 {
[<计划表名称> {
[<命令> 延迟 <frame_time> 毫秒;]

}]
} <schedule_table_name> ::= 标识符
所有schedule_table_name标识符在计划表标识符集内都应是唯一的。

<命令> ::=
<框架名称> |

主请求 |
SlaveResp |
分配NAD {<节点名称>} |
数据转储 {<节点名称>, <D1>, <D2>, <D3>, <D4>, <D5>} |
保存配置 {<node_name>} |
AssignFrameId {<节点名称>, <框架名称>} |
AssignFrameIdRange {<节点名称>, <帧索引> (, <帧PID>, <帧PID>,
<帧_PID>, <帧_PID>) } |
自由格式 {<D1>, <D2>, <D3>, <D4>, <D5>, <D6>, <D7>, <D8>}
该命令指定了在帧槽中应执行的操作。提供帧名称即可传输指定的帧。

<frame_name> ::= 标识符
frame_name标识符应存在于帧标识符集中。如果frame_name引用事件触发帧或零星帧，则关联的无条件帧不得在同一调度表中使用。

<节点名称> ::= 标识符
node_name指的是一个从属节点，请参阅11.3 。

MasterReq和SlaveResp要么在12.3.3.5中定义为帧，要么如果省略此子条款，则自动定义。这些帧的内容通过服务提供，并在 ISO 17987-3:2016 第 6 条中指定。

AssignNAD生成 AssignNAD 请求，参见 ISO 17987-3:2016，6.3.4.2。

DataDump生成 DataDump 请求，参见 ISO 17987-3:2016，6.3.6.3。

SaveConfiguration生成一个 SaveConfiguration 请求，参见 ISO 17987-3:2016，6.3.6.4。
configured_NAD取自节点属性部分。AssignFrameId根据以下参数生成AssignFrameIdentifier请求，其内容为：NAD、 supplier_id和message_id取自node_name的节点属性， protected_id取自frame_name的框架定义，参见12.3.3 。

AssignFrameIdRange根据以下参数生成 AssignFrameIdentifierRange 请求，其内容：NAD 和帧的顺序（ frame_index ）取自节点属性，参见12.3.4.3 。

<frame_index> ::= 整数
frame_index将索引设置为第一个要分配 PID 的帧，参见12.3.4.3 。

<frame_PID> ::= 整数
如果给出了可选的四个frame_PID，则请求将包含这些值。如果没有给出frame_PID ，则四个帧的 PID 将从frame_name的帧定义中获取，请参阅12.3.3 。

此帧中的所有数据都是在 LDF 文件处理过程中固定和确定的。仅当主节点也支持此配置服务时，才支持此服务。

FreeFormat发送带有八个数据字节的固定主请求帧。例如，这可用于发布用户特定的固定帧。

<帧时间> ::= 实数或整数

frame_time指定帧槽的持续时间，参见 ISO 17987-3:2016，5.3.3。frame_time值应以毫秒为单位指定。

调度表的处理和切换由主应用程序控制，参见 ISO 17987-3:2016，5.3 中的描述和参考文献 [2] 中的调度表处理 API。

示例图 30显示了与调度表 VL1_ST1 相对应的时间线。假设 time_base （参见12.3.4.2 ）设置为 5 毫秒。sc

hedule_tables {

VL1_ST1 {

VL1_CEM_Frm1延迟15毫秒;

VL1_LSM_Frm1延迟15毫秒;

VL1_CPM_Frm1延迟15毫秒;

VL1_CPM_Frm2 delay 20 ms;

}

}

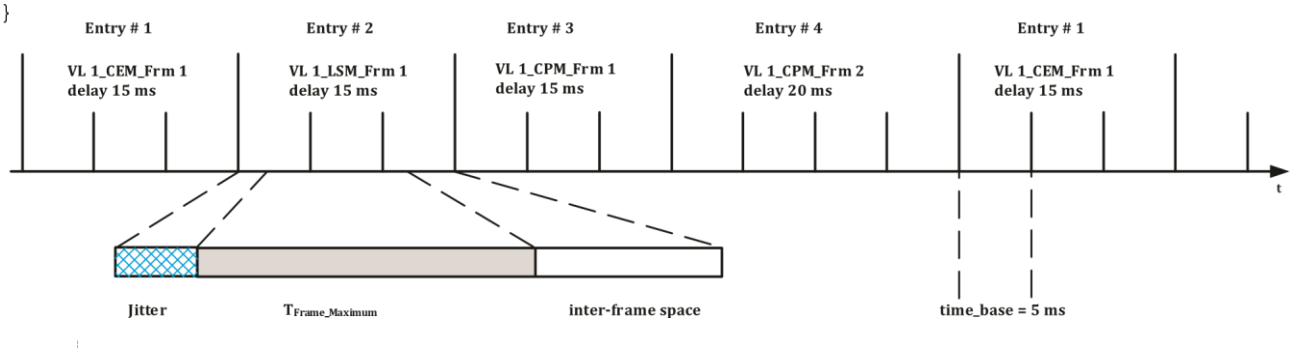


图 30—VL1_ST1 时间表

每个调度条目指定的延迟应长于抖动和最坏情况的帧传输时间。

12.3.6 信号编码类型定义

12.3.6.1 一般规定

信号编码类型旨在提供信号的表示和缩放属性。虽然此信息可用于在节点应用程序中生成自动缩放 API 例程，但这些 API 例程需要非常强大的节点。信号编码类型声明的主要目的是在总线流量分析工具中，它可以以易于访问的方式呈现记录的流量。

12.3.6.2 ASCII（ASC）

ASCII 数据使用一个字节代码来表示文本字符。ASCII 数据最常用于数据使用者是显示设备的情况，该显示设备可识别 ASCII 字符，因此无需进一步转换即可显示数据。最低有效 7 位表示从 0 到 127 的标准 ASCII 代码。最高有效位目前保留，但将来可能会被分配特殊功能。所有 ASCII 信号的位长应为 8 的倍数。

12.3.6.3 二进制编码的十进制（BCD）

当需要以半字节形式报告十进制数据时，使用二进制编码的十进制 (BCD) 编码，并且通常在数据消费者是显示设备的情况下使用。BCD 编码信号应为 4 位长。有效的 BCD 数据是十六进制字符 0-9，其编码在表 23中指定。

表 23—BCD 到十进制值的转换

BCD 值	十进制值
0 16	0
1 16	1
2 16	2
3 16	3
4 16	4
5 16	5
6 16	6
7 16	7
8 16	8
9 16	9
A 16至 F 16	无效的

示例 25 16将被解释为十进制的 37。作为两个 BCD 字符，该值被解释为十进制的 25。

12.3.6.4 信号编码

<信号编码类型定义> ::=
信号编码类型 {
[<信号编码类型名称> {
[<逻辑值> |
<物理范围> |
<bcd_值> |
<ASCII 值>]
}]
} <信号编码类型名称> ::= 标识符

所有signal_encoding_type_name标识符在信号编码类型标识符集内应是唯一的。

<逻辑值> ::= 逻辑值，<信号值>（，<文本信息>）；<物理范围> ::= 物理值，<最小值>，<最大值>，<比例>，
<偏移量> （，<文本信息>）；
<bcd_值> ::= bcd_值；
<ascii_值> ::= ascii_值；
<信号值> ::= 整数
<最小值> ::= 整数
<最大值> ::= 整数
<比例> ::= 实数或整数
<偏移量> ::= 实数或整数
<文本信息> ::= 字符串

signal_value 、 min_value和max_value 的取值范围为 0 至 65 535。max_value应大于或等于min_value 。如果原始值在 min 和 max 定义的范围内，则物理值应按照公式 (1) 的定义进行计算。

示例：V_battery 信号采用八位表示形式，如图31所示，即，分辨率高达 12 V 左右，并且具有三个超出范围的特殊值。

信号编码类型 { 电源状态 {

逻辑值，0，"关闭"；逻辑值，1，"开
启"；}
V_battery { logical_value, 0, "欠压"; physica
l_value, 1, 63, 0.0625, 7.0, "伏"; physical_v
alue, 64, 191, 0.0104, 11.0, "伏"; physical_v
alue, 192, 253, 0.0625, 1.3, "伏";
逻辑值，254，"过压"；逻辑值，255，"无效"；
}
}

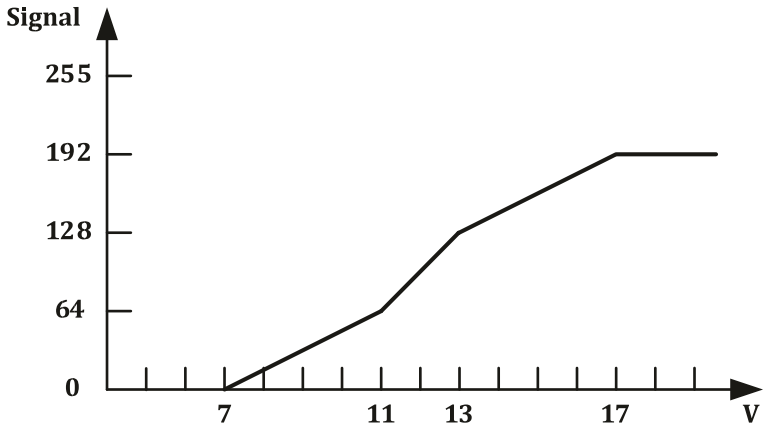


Figure 31 — Representation of V_battery

12.3.7 信号表示定义

信号表示声明用于将信号与相应的信号编码类型关联起来。

<信号表示定义> ::=
信号表示 {
[<信号编码类型名称>: <信号名称> ([, <信号名称>]);] }
<signal_encoding_type_name> ::= 标识符

signal_encoding_type_name标识符应存在于信号编码类型标识符集中。

<信号名称> ::= 标识符
signal_name标识符应存在于信号标识符集中（标量和字节数组信号均适用）。每个信号只能与一个signal_encoding_type_name相关联，并且不能嵌套在signal_group_name中。

12.4 LDF 示例

LIN_description_file;
LIN_protocol_version = "ISO17987:2015";
LIN_language_version = "ISO17987:2015";
LDF_file_revision = "14.23.01";
LIN_speed = 19.2 kbps;
Channel_name = "DB";
Nodes {

```
  Master: CEM, 5 ms, 0.1 ms;
  Slaves: LSM, RSM;
}
Signals {
  InternalLightsRequest: 2, 0, CEM, LSM, RSM;
  RightIntLightsSwitch: 8, 0, RSM, CEM;
  LeftIntLightsSwitch: 8, 0, LSM, CEM;
  LSMerror: 1, 0, LSM, CEM;
  RSMerror: 1, 0, RSM, CEM;
  IntTest: 2, 0, LSM, CEM;
}

Frames {
  CEM_Frm1: 0x01, CEM, 1 {
    InternalLightsRequest, 0;
  }
  LSM_Frm1: 0x02, LSM, 2 {
    LeftIntLightsSwitch, 8;   }
  LSM_Frm2: 0x03, LSM, 1 {
    LSMerror, 0;
    IntTest, 1;
  }
  RSM_Frm1: 0x04, RSM, 2 {
    RightIntLightsSwitch, 8;   }
  RSM_Frm2: 0x05, RSM, 1 {
    RSMerror, 0;
  }
}
Event_triggered_frames {
  Node_Status_Event : Collision_resolver, 0x06, RSM_Frm1, LSM_Frm1;
}
Node_attributes {   RSM {
    LIN_protocol = "2.1";     configured_NAD = 0x20;     product_id =
0x4E4E, 0x4553;     response_error = RSMerror;
    P2_min = 150 ms;     ST_min = 50 ms;     configurable_f
rames {       Node_Status_Event;
      CEM_Frm1;
      RSM_Frm1;
      RSM_Frm2;
    }   }
  LSM {
    LIN_protocol = "ISO17987:2015";     configured_NAD = 0x21;     i
nitial_NAD = 0x01;     product_id = 0x4A4F, 0x4841;     response_err
or = LSMerror;     fault_state_signals = IntTest;
```

```
    P2_min = 150 ms;     ST_min = 50 ms;     configurable_f
rames {       Node_Status_Event;
      CEM_Frm1;
      LSM_Frm1;
      LSM_Frm2;
    }   }
}
Schedule_tables {   Configuration_Schedule {
    AssignNAD {LSM} delay 15 ms;
    AssignFrameIdRange {LSM, 0} delay 15 ms;
    AssignFrameIdRange {RSM, 0} delay 15 ms;     SaveConfiguration {LSM} delay 10 m
s;
    SaveConfiguration {RSM} delay 10 ms;
  }
  Normal_Schedule {
    CEM_Frm1 delay 15 ms;
    LSM_Frm2 delay 15 ms;
    RSM_Frm2 delay 15 ms;
    Node_Status_Event delay 10 ms;
  }
  MRF_schedule {
    MasterReq delay 10 ms;
  }
  SRF_schedule {
    SlaveResp delay 10 ms;
  }
  Collision_resolver { // Keep timing of other frames if collision
    CEM_Frm1 delay 15 ms;
    LSM_Frm2 delay 15 ms;
    RSM_Frm2 delay 15 ms;
    RSM_Frm1 delay 10 ms;  // Poll the RSM node
    CEM_Frm1 delay 15 ms;
    LSM_Frm2 delay 15 ms;
    RSM_Frm2 delay 15 ms;
    LSM_Frm1 delay 10 ms;  // Poll the LSM node
  }
}
Signal_encoding_types {   Dig2Bit {
    logical_value, 0, "off";     logical_value, 1, "on";
logical_value, 2, "error";     logical_value, 3, "void";   }
  ErrorEncoding {
    logical_value, 0, "OK";     logical_value, 1, "error
";   }
  FaultStateEncoding {
```

```
    logical_value, 0, "No test result";     logical_value, 1, "failed
";     logical_value, 2, "passed";     logical_value, 3, "not used
";   }
  LightEncoding {
    logical_value, 0, "Off";     physical_value, 1, 254, 1, 100, "lux";
logical_value, 255, "error";
  }
}
Signal_representation {
  Dig2Bit: InternalLightsRequest;
  ErrorEncoding: RSMerror, LSMerror;
  FaultStateEncoding: IntTest;
  LightEncoding: RightIntLightsSwitch, LeftIntLightsSwitch; }
```

参考书目

[1] ISO 17987-4，*道路车辆 — 本地互连网络（LIN）— 第 4 部分：电气物理层（EPL）规范（12V/24V）*

[2] ISO/TR 17987-5，*道路车辆 — 本地互连网络（LIN）— 第 5 部分：应用程序编程接口（API）*

[3] ISO 17987-6，*道路车辆 — 本地互连网络（LIN）— 第 6 部分：协议一致性测试规范*

[4] ISO 17987-7，*道路车辆 — 本地互连网络（LIN）— 第 7 部分：电气物理层（EPL）一致性测试规范*

[5] ISO/IEC 7498-1，*信息处理系统 — 开放系统互连 — 基本参考模型：基本模型 — 第 1 部分*

[6] ISO/IEC 10731，*信息技术 - 开放系统互连 - 基本参考模型 - OSI 服务定义约定*