

中华人民共和国国家标准

GB/T 28029.2—2020
部分代替 GB/T 28029.1—2011

轨道交通电子设备 列车通信网络(TCN) 第 2-1 部分:绞线式列车总线(WTB)

Electronic railway equipment—Train communication network (TCN)—
Part 2-1: Wire Train Bus (WTB)

(IEC 61375-2-1:2012, MOD)

2020-03-06 发布

2020-10-01 实施

国家市场监督管理总局 发布
国家标准化管理委员会

目 次

前言 Ⅲ

引言 Ⅵ

1 范围 1

2 规范性引用文件 1

3 术语、定义、缩略语和约定 1

 3.1 术语和定义 1

 3.2 缩略语 15

 3.3 约定 16

 3.4 总则 21

 3.5 一致性测试 25

4 物理层 26

 4.1 概述 26

 4.2 拓扑结构 26

 4.3 介质规范 27

 4.4 介质连接 32

 4.5 节点规范 35

 4.6 线路单元规范 38

 4.7 收发器规范 41

 4.8 由介质决定的信号表示 46

5 链路层控制 52

 5.1 编址 52

 5.2 帧和报文 52

 5.3 报文格式和协议 59

 5.4 介质分配 75

 5.5 初运行 77

 5.6 链路层接口 115

6 实时协议 129

 6.1 概述 129

 6.2 变量传送服务和协议 131

 6.3 消息传送服务和协议 150

 6.4 传送和存储数据的表示和编码 237

7 应用层 256

 7.1 过程数据编排 256

 7.2 WTB 线路故障位置检测 261

8 列车网络管理 267

 8.1 总则 267

8.2 管理者、代理者及其接口	268
8.3 管理对象	270
8.4 服务和管理消息	278
8.5 接口过程	331
附录 A (资料性附录) 本部分与 IEC 61375-2-1:2012 相比的结构变化情况	336
参考文献.....	339

前 言

GB/T 28029《轨道交通电子设备 列车通信网络(TCN)》分为以下 12 个部分:

- 第 1 部分:基本结构;
- 第 2-1 部分:绞线式列车总线(WTB);
- 第 2-2 部分:绞线式列车总线(WTB)一致性测试;
- 第 2-3 部分:TCN 通信规约;
- 第 2-4 部分:TCN 应用规约;
- 第 2-5 部分:以太网列车骨干网(ETB);
- 第 2-6 部分:车地通信;
- 第 2-7 部分:基于电台的无线列车骨干网(WLTB);
- 第 3-1 部分:多功能车辆总线(MVB);
- 第 3-2 部分:多功能车辆总线(MVB)一致性测试;
- 第 3-3 部分:CANopen 编组网(CCN);
- 第 3-4 部分:以太网编组网(ECN)。

本部分为 GB/T 28029 的第 2-1 部分。

本部分按照 GB/T 1.1—2009 给出的规则起草。

本部分代替了 GB/T 28029.1—2011《牵引电气设备 列车总线 第 1 部分:列车通信网络》中的第 2 章“实时协议”、第 4 章“绞线式列车总线(WTB)”及第 5 章“列车网络管理”(除 5.3.3 和 5.4.3)等相关内容,与 GB/T 28029.1—2011 相比,主要技术变化如下:

- 修改了“范围”的内容(见第 1 章,2011 年版的 1.1)。
- 修改了“规范性引用文件”(见第 2 章,2011 年版的 1.2)。
- 修改了“术语、定义、缩略语和约定”(见第 3 章,2011 年版的 1.3、1.4、1.5)。
- 修改了“TCN WTB 设备配置可选项”一图(见图 8,2011 年版的图 7)。
- 修改了“发送器测试电路图中的重载测试电路的阻性负载值由“ $0.42R_i$ ”为“ $0.75R_i$ ”(见图 24, 2011 年版的图 128)。
- 修改了“节点请求的特征周期,表示为基本周期 T_{bp} 的 2^n 倍。 $n=0\sim 128$ ”为“节点请求的特征周期,表示为基本周期 T_{bp} 的 2^n 倍。 $n=0\sim 7$ ”(见 5.5.2.2,2011 年版的 4.7.2.1)。
- 修改了“LS_REGULAR_SLAVE 的含义:节点在 REGULAR_SLAVE 状态或 TEACHING_MASTER 状态”为“LS_REGULAR_SLAVE 的含义:节点处于常规从设备或节点处于学习从状态”(见表 20,2011 年版的 4.8.4.2.2)。
- 增加了“网关站”相关内容(见 6.3.2.4)。
- 修改了“/* bit0 */”为“/* bit7 */”,“/* bit1 */”为“/* bit6 */”(见 6.3.10.2.4,2011 年版的 2.3.10.2.4)。
- 增加了“应用层”相关内容(见第 7 章)。
- 修改了“ $g_0\sim g_7$ 、 $g_8\sim g_{15}$ 等”由升序改为降序排列,修改为“ $g_7\sim g_0$ 、 $g_{15}\sim g_8$ 等”(见图 263, 图 264, 2011 年版的 5.4.6.7.3、5.4.6.8.2)。
- 修改了“呼叫写组索引”的内容编排(见 8.4.5.8.2,2011 年版的 5.4.6.8.2)。
- 修改了“应答写组索引”的内容编排(见 8.4.5.8.3,2011 年版的 5.4.6.8.3)。
- 修改了全文中所有图解式位序排列:将位序 $0\sim 7$ 升序排列修改为 $7\sim 0$ 降序排列;将位序 $0\sim$

15 排列修改为 15~0 排列;将文本式(0)~(7)升序排列修改为降序排列(7)~(0)(见图 43、表 2、5.5.2.3,2011 年版的图 147、表 2、4.7.2.2)。

本部分使用重新起草法修改采用 IEC 61375-2-1:2012《轨道交通电子设备 列车通信网络(TCN) 第 2-1 部分:绞线式列车总线(WTB)》。

本部分与 IEC 61375-2-1:2012 相比在结构上有较多调整,附录 A 中列出了本部分与 IEC 61375-2-1:2012 的章条编号对照一览表。

本部分与 IEC 61375-2-1:2012 相比存在技术性差异,这些差异涉及的条款已通过在其外侧页边空白位置的垂直单线(|)进行了标示,具体技术性差异及其原因如下:

——增加了范围中“规定”的内容,以符合 GB/T 1.1 的要求(见第 1 章)。

——关于规范性引用文件,本部分做了具有技术性差异的调整,以适应我国的技术条件,调整的情况集中反映在第 2 章“规范性引用文件”中,具体调整如下:

- 用等同采用国际标准的 GB/T 3454 代替了 ITU-T V.24;
- 用等同采用国际标准的 GB/T 7421 代替了 ISO/IEC 13239;
- 用等同采用国际标准的 GB/T 13000 代替了 ISO/IEC 10646;
- 用等同采用国际标准的 GB/T 15273.1 代替了 ISO 8859-1;
- 用等同采用国际标准的 GB/T 15629.2 代替了 ISO/IEC 8802-2;
- 用 GB/T 16262(所有部分)代替了 ISO/IEC 8824(所有部分),两项标准各部分之间的一致性程度如下:
 - GB/T 16262.1—2006 信息技术 抽象语法记法一(ASN.1) 第 1 部分:基本记法规范(ISO/IEC 8824-1:2002,IDT);
 - GB/T 16262.2—2006 信息技术 抽象语法记法一(ASN.1) 第 2 部分:信息客体规范(ISO/IEC 8824-2:2002,IDT);
 - GB/T 16262.3—2006 信息技术 抽象语法记法一(ASN.1) 第 3 部分:约束规范(ISO/IEC 8824-3:2002,IDT);
 - GB/T 16262.4—2006 信息技术 抽象语法记法一(ASN.1) 第 4 部分:ASN.1 规范的参数化(ISO/IEC 8824-4:2002,IDT)。
- 用修改采用国际标准的 GB/T 25119 代替了 IEC 60571;
- 用修改采用国际标准的 GB/T 28029.3—2020 代替了 IEC 61375-2-2:2012;
- 用修改采用国际标准的 GB/T 28029.9 代替了 IEC 61375-3-1;
- 删除了 IEC 61375-1、ISO/IEC 8825 和 ISO/IEC 9646。

——删除了正文中没有用到的术语和定义(见 IEC 61375-2-1:2012 的第 3 章);

——删除了缩略语“AVA、BER、DIN、EIA、EP、ERRI、LLC、ORE、PICS、PTA、RIC”(见 IEC 61375-2-1:2012 的 3.2);

——修改了“发送器测试电路图中的重载测试电路的外接电阻负载值由“ $0.42R_i$ ”为“ $0.75R_i$ ”,因发送器测试电路图中的重载测试电路的阻性负载值为端接器阻值和外接阻值的并联值,当外接阻值为 $0.75R_i$ 时,重载测试电路的阻性负载值约为 $0.42R_i$,满足标准要求(见图 24);

——修改了“节点请求的特征周期,表示为基本周期 T_{bp} 的 2^n 倍。 $n = 0 \sim 128$ ”为“节点请求的特征周期,表示为基本周期 T_{bp} 的 2^n 倍。 $n = 0 \sim 7$ ”,因如程序注释所列,节点请求的特征周期为基本周期的 1~128 倍,即 $2^n = 1 \sim 128$,相应 $n = 0 \sim 7$ (见 5.5.2.2);

——修改了“/* bit0 */”为“/* bit7 */”,“/* bit1 */”为“/* bit6 */”,因为统一表 148 程序注释与图解说明(见 6.3.10.2.4);

——修改了全文中所有图解式位序排列,将位序 0~15 排列修改为 15~0 排列,因为按照 3.3.5 传送数据规范统一全文图解式位序排列方式(见 5.2.3、8.3.2.1、8.4.1.6)。

本部分还做了下列编辑性修改：

- 增加了 IEC 61375-2-1:2012 的 4.1.4 遗漏条款 b)“节点的方向 1 连向末端 1,方向 2 连向末端 2”(见 4.2.4)。
 - 修改了 4.4.5 的表 6。
 - 删除了 4.7.2.1 的条款 e)及条款 f)前面的编号 e)及 f)。
 - 修改了 IEC 61375-2-1:2012 的 4.7.1.5.3 的图 33 及 4.7.1.5.4 的图 34 的画法(见 4.8.1.5.4、4.8.1.5.5)。
 - 修改了 5.4.2.1 的“ $T_{ip} = (2n \times T_{bp})$ ”为“ $T_{ip} = (2'' \times T_{bp})$ ”。
 - 增加了 5.5.4.8.9 的遗漏条款 c):“消息轮询:主设备在下一个周期相开始之前剩余时间内轮询节点以获取消息数据,见 5.5.4.8.12。”。
 - 修改了“Unsigned1 为“UNSIGNED”。
 - 修改了 IEC 61375-2-1:2012 中的部分错误:
 - 5.6.4.2.2 中的“LS_REGULAR_SLAVE”的含义“node in state REGULAR_SLAVE or TEACHING_MASTER”中的“TEACHING_MASTER”有误,在本部分中将其修改为“节点处于常规从设备或节点处于学习从状态”;
 - 6.2.3.3.5.1 中的“‘BA’H(= 442)”有误,在本部分中将其修改为“‘1BA’H(= 442)”;
 - 6.3.6.5.4 中的“nk_eot BOOLEAN1--always FALSE (0)”中的“nk_eot”有误,在本部分中将其修改为“nk_e”;
 - 8.4.4.9.3 中的“sif_code = 39”有误,在本部分中将其修改为“sif_code = 29”。
 - 将 IEC 61375-2-1:2012 第 2 章的 UIC 557 移入参考文献中,因为 UIC 557 在本部分是资料性引用。
 - 调整了部分列项编号。
- 请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。
- 本部分由国家铁路局提出。
- 本部分由全国牵引电气设备与系统标准化技术委员会(SAC/TC 278)归口。
- 本部分起草单位:中车株洲电力机车研究所有限公司、中车青岛四方车辆研究所有限公司、中车大连电力牵引研发中心有限公司、中车长春轨道客车股份有限公司、中车青岛四方机车车辆股份有限公司、中车株洲电力机车有限公司、中国铁道科学研究院集团有限公司机车车辆研究所。
- 本部分主要起草人:韩露、宁侨、刘伟、王锋、周学勋、唐柳、李海龙、车聪聪、颜罡、朱广超。
- 本部分所代替标准的历次版本发布情况为:
- GB/T 28029.1—2011。

引 言

GB/T 28029 的本部分规定了列车通信网络中的一个组成部分——绞线式列车总线(WTB),它是主要的串行数据通信总线之一,用于经常相互联挂和解联的重联车辆,见图 1。

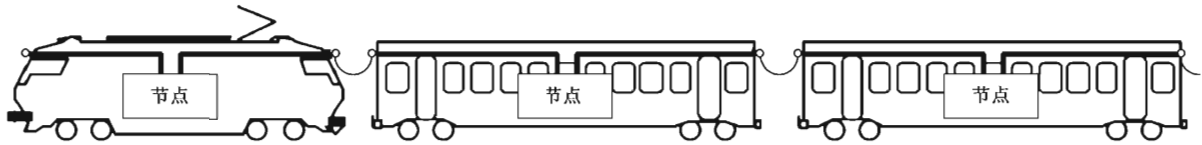


图 1 绞线式列车总线

本部分定义了列车通信网络(TCN)的通信接口。

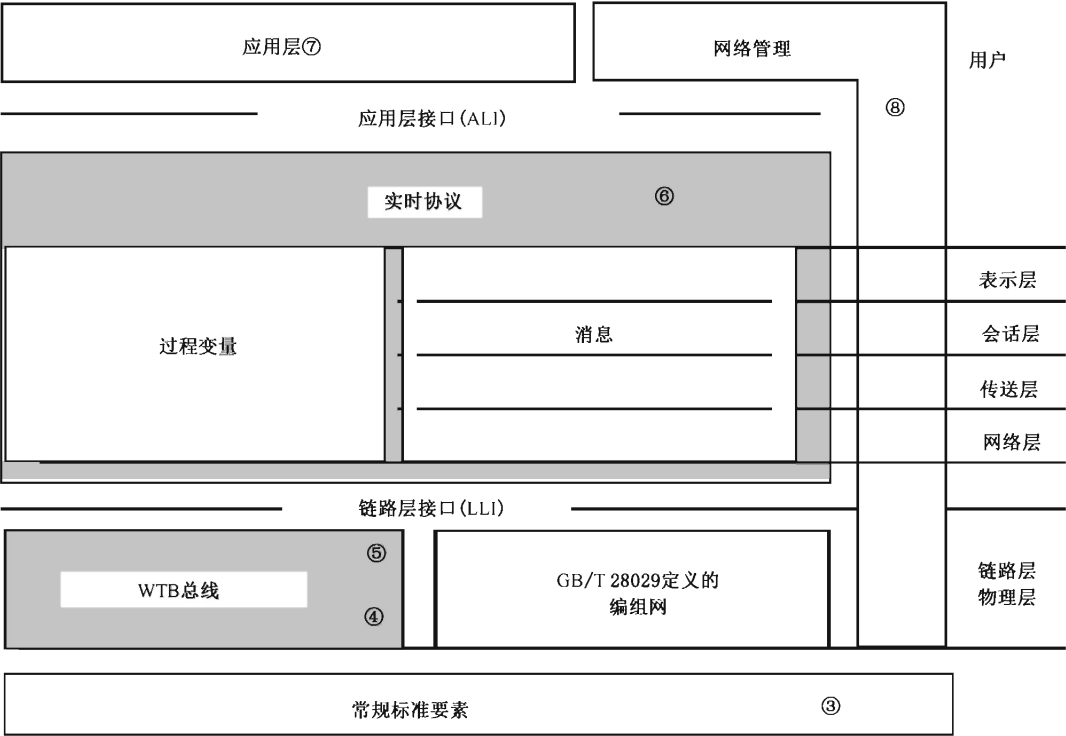
TCN 分为两层网络架构,即列车骨干网和列车编组网:

- a) 对于开式列车(见定义)中的互连车辆,如 UIC 列车,本部分规定了被称为绞线式列车总线(WTB)的列车总线;
 - b) 为了连接标准车载设备,可使用编组网,如多功能车辆总线(MVB)。
- 在 TCN 体系中,WTB 以提供两种通信服务的实时协议为其特征,它提供两种通信服务:
- a) 过程变量,一种分布式的实时数据集,通过广播周期性的刷新;
 - b) 消息,只有在需要时用下述两种方式传送:
 - 单播(点对点)或/和;
 - 多播。

WTB 提供通用的网络管理对整个网络进行调试、试运行及维护。

MVB 和 WTB 共享实时协议和网络管理,其他的编组网络需要与 WTB 的实时协议和网络管理适配。

TCN 的结构与 GB/T 9387.1 定义的开放式互连模型(OSI 模型)类似,见图 2。



注：图中的数字标号与本部分章条号相对应。

图 2 TCN 的分层

本部分包括 8 章和 1 个附录：

- 第 1 章：范围；
- 第 2 章：规范性引用文件；
- 第 3 章：术语、定义、缩略语和约定；
- 第 4 章：物理层；
- 第 5 章：链路层控制；
- 第 6 章：实时协议：
 - 变量：链路层接口及应用层接口；
 - 消息：链路层接口、协议，应用层接口；
 - 数据表示。
- 第 7 章：应用层：
 - 过程数据编排；
 - WTB 线路故障位置检测。
- 第 8 章：列车网络管理：
 - 网络配置、监视及控制。

附录 A(资料性附录)：本部分与 IEC 61375-2-1:2012 相比的结构变化情况。

轨道交通电子设备 列车通信网络(TCN)

第 2-1 部分:绞线式列车总线(WTB)

1 范围

GB/T 28029 的本部分规定了绞线式列车总线(WTB)。

本部分适用于开式列车中编组间及编组内的数据通信。

应用本部分的列车通信总线(WTB)能实现开式列车中各个编组的互操作性。编组内部的数据通信网络(例如 MVB)与 TCN 兼容,供应商提供 WTB 和编组网络兼容性证明。

若供应商与用户协商同意,本部分也可用于闭式列车及多单元列车。

注:本部分未考虑公共汽车、无轨电车等车辆。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件,仅注日期的版本适用于本文件。凡是不注日期的引用文件,其最新版本(包括所有的修改单)适用于本文件。

GB/T 3454 数据终端设备(DTE)和数据电路终接设备(DCE)之间的接口电路定义表(GB/T 3454—2011,ITU-T V.24:2000,IDT)

GB/T 7421 信息技术 系统间远程通信和信息交换 高级数据链路控制(HDLC)规程(GB/T 7421—2008,ISO/IEC 13239:2002,IDT)

GB/T 13000 信息技术 通用多八位编码字符集(UCS)(GB/T 13000—2010,ISO/IEC 10646:2003,IDT)

GB/T 15273.1 信息处理 八位单字节编码图形字符集 第一部分:拉丁字母一(GB/T 15273.1—1994,idt ISO 8859-1:1987)

GB/T 15629.2 信息技术 系统间远程通信和信息交换 局域网和城域网 特定要求 第 2 部分:逻辑链路控制(GB/T 15629.2—2008,ISO/IEC 8802-2:1998,IDT)

GB/T 16262(所有部分) 信息技术 抽象语法记法一(ASN.1)[ISO/IEC 8824(所有部分)]

GB/T 25119 轨道交通 机车车辆电子装置(GB/T 25119—2010,IEC 60571:2006,MOD)

GB/T 28029.3—2020 轨道交通电子设备 列车通信网络(TCN) 第 2-2 部分:绞线式列车总线(WTB)一致性测试(IEC 61375-2-2:2012,MOD)

GB/T 28029.9 轨道交通电子设备 列车通信网络(TCN) 第 3-1 部分:多功能车辆总线(MVB)(GB/T 28029.9—2020,IEC 61375-3-1:2012,MOD)

IEC 60807(所有部分) 频率低于 3 MHz 的矩形连接器(Rectangular connectors for frequencies below 3 MHz)

IEEE 754 二进制浮点运算(Standard for Binary Floating-Point Arithmetic)

ITU-T Recommendation Z.100 SDL 语言[Specification and Description Language (SDL)]

UIC 556 列车总线上的信息传送[Information transmission in the train (train-bus)]

3 术语、定义、缩略语和约定

3.1 术语和定义

下列术语和定义适用于本文件。

3.1.1

地址 address

通信参与方标识符,根据层不同可有几种类型。

3.1.2

代理者 agent

站中的应用进程,代表管理者访问本地管理对象。

3.1.3

应用层 application layer

OSI 模型中的最高层,直接接口应用。

3.1.4

应用层接口 application layer interface; ALI

由应用层提供的服务的定义。

3.1.5

应用消息适配器 application messages adapter; AMA

应用实现消息服务时直接调用的代码。

3.1.6

应用消息接口 application messages interface; AMI

消息服务的定义。

3.1.7

应用进程 application process

实际开放系统中的一个元素,其为特定应用执行信息处理。

3.1.8

应用监视接口 application supervision interface; ASI

代理者可用的监视服务的定义。

3.1.9

应用变量适配器 application variables adapter; AVA

应用实现变量服务时直接调用的代码。

3.1.10

应用变量接口 application variables interface; AVI

变量服务的定义。

3.1.11

辅助通道 auxiliary channel

用于检测附加节点的通道。

3.1.12

基本周期 basic period

总线活动被分为若干周期,其最短的即为基本周期。

注: 它由周期相(用于周期数据)和偶发相(用于消息数据和监视数据)组成。

3.1.13

大开端 big-endian

存储和发送数据的一种顺序机制。

注: 在该机制下多位位组数据的最高有效部分存于存储器的最低位组地址,且首先发送。

3.1.14

位填充 bit-stuffing

GB/T 7421 规定的用于防止把帧数据误译为标志序列的方法。

注：它在每 5 个连续的“1”后插入一个附加的“0”，而在接收时再把该“0”去掉。

3.1.15

广播 broadcast

几乎同时把同一信息传送给多个目的的过程。

注：在 TCN 中广播被认为是不可靠的，即有的目的可能收到了该信息，而有的目的可能没收到该信息。

3.1.16

总线 bus

为仲裁等目的，向附着其上的所有参与者几乎同时广播同一信息，使得所有设备获得其状态的相同指示的通信介质。

3.1.17

总线控制器 bus controller

负责通信链路层的处理器或集成电路。

3.1.18

总线开关 bus switch

WTB 节点内将两个方向上的电缆节在电气上连接起来的开关或继电器。

3.1.19

呼叫者 caller

启动消息交换的应用进程。

3.1.20

校验序列 check sequence

基于在所发送有用数据上附加根据有用数据计算得出的校验和或循环冗余校验(CRC)码的检错方法。

3.1.21

校验变量 check variable

一种反价布尔类型的过程变量，用于保护另一过程变量。

3.1.22

校验偏置 check offset

校验变量在数据集中的位偏置。

3.1.23

闭式列车 closed train

由一组编组构成的列车，其组成在正常运行中不会改变。

示例：如地铁、城郊列车或高速列车组。

3.1.24

组成 composition

构成列车的编组的数量和特征。

3.1.25

组态 configuration

对网络拓扑结构、连到网络上的设备、这些设备的能力和产生的通信的定义。

注：还可包括在进入常规运行前将组态信息载入设备的操作。

3.1.26

连接确认 connect confirm

消费者对生产者连接请求的响应。

3.1.27

连接请求 connect request

生产者向消费者发送消息的第 1 个数据包。

3.1.28

编组 consist

在正常运行中不会分离的单个车辆或一组车辆。

注：一个编组可包含 0 个、1 个或多个编组网。

3.1.29

编组网 consist network

互连编组内设备的总线。

注：如遵守或采用本文档中描述的 TCN 实时协议的 MVB。

3.1.30

坚固性 consistency

组成一个数据集的所有要素的读写都是在一种不可分割的操作中进行的，则该数据集是坚固的。

3.1.31

消费者 consumer

传输层中消息的接收者。

注：参见“生产者”。

3.1.32

会话 conversation

应用层的数据交换，由呼叫消息和应答消息（后者在多播协议中没有）组成。

注：一次会话起始于第一个连接请求帧，终止于收到应答消息最后一个确认或是不再期望能收到应答消息。

3.1.33

数据报文 datagram

含有为转发到最终目的所需全部信息，但不含以前帧内容信息的帧。

注：数据报文不使用以前建立起来的连接，且在链路层不确认。

3.1.34

数据集 dataset

在一个过程数据帧中传送的所有过程变量。

3.1.35

分界符 delimiter

含有非法编码（即非 1 也非 0）的信号序列，用于确定帧的开始（起始分界符）和终止（终止分界符）。

注：改写 GB/T 16657.2—2008，定义 3.9。

3.1.36

目的设备 destination device

链路层中帧的接收者。

注：参见“源设备”。

3.1.37

设备 device

连到一条或多条总线上的部件。

3.1.38

设备地址 device address

标识总线上的设备。

注：MVB 总线上设备地址为 12 位；WTB 总线上设备地址为 8 位，低 6 位为节点地址。连接几种总线的设备对每一总线可有不同的设备地址，某些特殊设备如中继器，仅参与物理层，因而没有设备地址。

3.1.39

方向 1 **direction 1**

WTB 节点的一个方向。

3.1.40

方向 2 **direction 2**

WTB 节点的另一个方向。

3.1.41

终止分界符 **end delimiter**

介质返回闲置状态前的帧终止序列。

3.1.42

末端节点 **end node**

端接它所连接的两个总线段,但不把它们恒定连接起来的节点。

3.1.43

扩展电缆 **extension cable**

在主干电缆中插入节点所用的电缆,每路线由两根独立的绞线对组成,截面可能小于主干电缆。

3.1.44

终点实体 **final**

网络层上数据包或确认包的接收者。

注:当两个设备在同一总线内通信时,终点实体即位于目的设备上(参见“起始实体”)。

3.1.45

标志 **flag**

由“1”和“0”组成的序列,用于帧的起始和终止分界。

注:在发送的数据中若出现与标志相同的数据,则用位填充的方法加以修改,参见 GB/T 7421。

3.1.46

帧 **frame**

发送器在一个时间槽内所发送的连续符号序列,在两个时间槽间总线处于闲置状态。

3.1.47

帧校验序列 **frame check sequence; FCS**

GB/T 7421 规定的 16 位 FCS。

3.1.48

帧数据 **frame data**

在起始分界符和终止分界符(MVB)间所发送的数据,或在前导码及终止分界符(WTB)间所发送的数据。

3.1.49

加电清除 **fritting**

采用接点上加破坏电压的方法,对氧化的接点进行电清除。

3.1.50

功能 **function**

与另一个应用进程交换消息的应用进程。

3.1.51

功能索引 **function directory**

向站标识符映射功能标识或做相反映射的索引。

3.1.52

功能标识符 **function identifier**

功能的 8 位标识符。

3.1.53

功能码 **F code**

在主帧中用于指示请求和所期望的响应从帧的长度。

3.1.54

网关 **gateway**

在应用层不同总线间的连接,需要对与应用有关的数据进行分析和协议转换。

3.1.55

组地址 **group address**

节点所属的多播组的地址。

3.1.56

组索引 **group directory**

指示节点所属的多播组的索引。

3.1.57

高级数据链路控制 **high-level data link control; HDLC**

数据传送用的一套标准化协议,包括数据传送用的 GB/T 7421。

3.1.58

HDLC 数据 **HDLC data**

以 HDLC 帧传送的数据。

3.1.59

初运行 **inauguration**

列车组成改变时执行的操作。

注:它给出 WTB 上所有节点相对于主设备的地址、它们的取向及同一总线中所有已命名节点的描述符。

3.1.60

特征周期 **individual period**

同一源的相同过程数据两次连续发送的间隔。

注:特征周期是基本周期的 2ⁿ 倍。

3.1.61

事例 **instance**

同一对象、进程的多个实现之一。

3.1.62

完整性 **integrity**

在系统部件出现故障时识别并拒绝错误数据的系统特性。

3.1.63

中间节点 **Intermediate node**

使其所连接的两个总线节之间建立连续性而不终结的节点。

3.1.64

跨接电缆 **jumper cable**

连接相邻两个车辆主干电缆的电缆。

注:其截面可能大于主干电缆,在使用 UIC 电缆时,连接器用手动连接。车辆间通常有两根跨接电缆。

3.1.65

线路 line

无冗余总线。

注：一个单线总线由一根线路组成，一个双线总线由两根线路组成。

3.1.66

线路单元 line unit

提供与线路电气连接的所有电路。

3.1.67

链路地址 link address

链路层地址，用于标识哪个总线和哪个设备地址发送或接收数据包。

3.1.68

链路控制 link control

指示帧类型的 HDLC 帧中的字段。

3.1.69

链路数据 link data

链路层传送的但与它不相关的数据。

3.1.70

链路报头 link header

与链路层有关的消息数据帧的一部分。

3.1.71

链路层 link layer

OSI 模型中在同一总线上各设备之间建立点对点 and 广播连接的层。

3.1.72

链路层接口 link layer interface

链路层与上一通信层间的接口。

3.1.73

链路层管理 link layer management

为管理链路层的链路层控制接口。

3.1.74

逻辑地址 logical address

不绑定于专用设备的地址。

示例：过程数据地址。

3.1.75

宏循环 macro cycle

一个宏周期中包含的基本周期数。

3.1.76

宏周期 macro period

最长的特征周期，此后周期性通信回复到同样的模式，以 ms 计。

3.1.77

主通道 main channel

接收主总线通信的通道。

3.1.78

管理消息 management message

网络管理中管理者与代理者间交换的消息。

3.1.79

管理者 manager

站中专用于网络管理的功能。

注：它通过系统地址发送管理呼叫消息。

3.1.80

编排 marshalling

对数据集中的过程变量分配应用地址或命名，在 WTB 上它与节点类型及版本有关。

3.1.81

主设备 master

在总线上主动地向各从设备发送信息的设备。

注：它可给某个从设备发送权，以使该从设备在有限时间内发送一个从设备帧。

3.1.82

主帧 master frame

主设备发送的帧。

3.1.83

介质访问控制 medium access control

链路层的子层，控制对介质的访问（仲裁、主权转移、轮询）。

3.1.84

介质 medium

信号的物理载体。

示例：电缆、光纤等。

3.1.85

介质连接单元 medium attachment unit

与传送介质耦合的设备。

3.1.86

消息 message

以一个或几个包传送的数据项。

3.1.87

消息传送 messages

TCN 中的一种传送服务。

3.1.88

消息数据 message data

链路层在消息传送中偶然发送的数据。

3.1.89

信使 messenger

端到端消息通信及与应用接口的通信栈。

3.1.90

多播 multicast

同一消息向由组地址标识的一组接收者传送的过程。

注：即使该组包含所有接收者也使用术语“多播”。

3.1.91

多功能车辆总线 multifunction vehicle bus; MVB

连接可编程的站及简单传感器/执行器的编组网。

3.1.92

多单元列车 multiple unit train

由一组闭式列车组成的列车,在正常运行中该组组成可能改变。

3.1.93

网络 network

以共同约定的方式交换信息的可能不同通信系统的集合。

3.1.94

网络地址 network address

标识网络中的功能或站的地址,可以是用户地址或系统地址。

3.1.95

网络报头 network header

与网络层有关的消息数据帧的一部分。

3.1.96

网络层 network layer

OSI 模型中负责不同总线间路由选择的层。

3.1.97

网络管理 network management

远程配置、监视、诊断和维护网络所必须的操作。

3.1.98

节点 node

WTB 上的设备,可作为列车总线与车辆总线之间的网关。

3.1.99

节点地址 node address

WTB 上的节点地址(6 位),等于 WTB 上 8 位设备地址中的低 6 位。

3.1.100

节点描述符 node descriptor

为 24 位的数据结构,为节点指明节点周期和节点密钥。

3.1.101

节点索引 node directory

将节点地址映射为设备地址(在 WTB 上是一一映射)的索引。

3.1.102

节点密钥 node key

由应用选择的 16 位标识符,用于标识节点的类型和版本。

注:主设备在每次组成改变后交换数据前向其他所有节点分发节点密钥。

3.1.103

节点周期 node period

WTB 上一个节点所希望的特征周期。

注:若没有出现过负荷,则与特征周期相同。

3.1.104

八位位组 octet

存于存储器内或作为一个单元进行传送的 8 位字节。

3.1.105

开式列车 open train

由一个或一组编组构成的列车,其组成在正常运行中可能改变。

注：如国际 UIC 列车。

3.1.106

起始实体 origin

网络层中包(数据或确认)的发送者,当两个设备在同一总线上通信时,起始实体位于源设备上。

注：参见“终点实体”。

3.1.107

包(数据包) packet

以一个消息数据帧发送的消息(信息、确认或控制)单位。

3.1.108

周期 period

时间单位,一周期后周期性事件重复发生。

3.1.109

周期性数据 periodic data

按一个特征周期间隔,周期性发送的过程数据。

3.1.110

周期扫描表 periodic list

一个宏循环的每个周期里要轮询的节点、地址或设备列表。

3.1.111

周期相 periodic phase

主设备根据周期扫描表,轮询周期性数据的相。

3.1.112

物理地址 physical address

WTB 节点地址,用于标识同一总线上的通信设备。

3.1.113

轮询 polling

为接收从帧而进行的主帧发送。

3.1.114

端口 port

一种含有发送和接收数据的存储器结构,写新值将复盖原值(缓冲区,不是排队)。

注：端口为总线及应用提供了同时访问的方法。

3.1.115

前导码 preamble

WTB 上以接收器同步为目的作为一帧开头的信号序列。

3.1.116

表示层 presentation layer

OSI 模型中负责数据表示及转换的层。

3.1.117

过程数据 process data

链路层在过程变量传送中周期性地广播的源寻址数据。

3.1.118

过程变量 process variable

表达过程状态的变量。

注：如速度、制动命令。

3.1.119

生产者 producer

传输层中消息的发送者。

注：参见“消费者”。

3.1.120

发布者 publisher

广播数据集的源。

注：参见“用户”。

3.1.121

过程变量名 PV name

过程变量的标识符。

3.1.122

过程变量集 PV set

属于同一数据集中的一组过程变量。

3.1.123

队列 queue

以先进先出方式对若干有序帧进行存储的一种存储结构。

3.1.124

机箱 rack

含有一个或多个属于同一段内设备的装置。

3.1.125

接收器 receiver

可从物理介质上接收信号的电子设备。

3.1.126

接收队列 receive queue

设备中接收消息数据的队列。

3.1.127

常规运行 regular operation

与列车初运行(WTB)相对的正常总线活动。

3.1.128

中继器 repeater

物理层总线段间的连接。

注：它提供了使总线扩展到无源方法限制之外的能力。各互连段工作在同一速率和同一协议之下。由中继器引入的时延约为1位的时长。

3.1.129

应答者 replier

响应呼叫请求,接收呼叫消息并发送应答消息的应用进程。

3.1.130

路由器 router

网络层两个总线间的连接。

注：它根据网络地址从一条总线向另一条总线转发数据报文。

3.1.131

扫描 scan

为监视目的以确定的顺序对各设备进行的轮询。

3.1.132

节 section

段的一部分,节与节间的连接为无源的且无需端接器。

3.1.133

段 segment

可挂设备的一段电缆。

注:段的两端接等于其特征阻抗的端接器。段可由几个节(非端接)用连接器连接而成。

3.1.134

分段 segmentation

将长消息分割成几个较短的帧,以便发送。

3.1.135

发送队列 send queue

设备中发送消息数据的队列。

3.1.136

服务 service

子系统提供给用户的能力和特性。

3.1.137

会话报头 session header

会话层中消息数据帧的一部分。

3.1.138

会话层 session layer

OSI 模型中负责建立和关闭通信的层。

3.1.139

A 侧 side A

基准于 WTB 节点方向的编组一侧。

3.1.140

B 侧 side B

基准于 WTB 节点方向的编组另一侧。

3.1.141

从设备 slave

从总线上接收信息或向总线发出信息以响应主设备请求的设备。

3.1.142

从帧 slave frame

从设备发送的帧。

3.1.143

源设备 source device

链路层中帧的发送者。

注:参见“目的设备”。

3.1.144

偶发性传送 sporadic transmission

当一个网络外部事件需要时进行的按需发送。

注:也称非周期的、事件驱动的、需求驱动的发送。

3.1.145

偶发性数据 sporadic data

按需传送的携带消息数据或监视数据的数据帧。

3.1.146

偶发相 sporadic phase

基本周期的后半部分,专用于消息数据和总线管理数据的按需传送。

3.1.147

站 station

一个有消息通信能力的设备。

注:与简单设备不同,它支持一个代理者的功能。

3.1.148

站索引 station directory

将站标识符映射到链路地址或做相反映射的索引。

3.1.149

站标识符 station identifier

站的 8 位标识符。

3.1.150

站状态字 station status word

描述站的状态及能力的 16 位描述符。

3.1.151

强主 strong master

强节点是当前的主设备,在被降到弱节点状态前不会放弃总线主权。

3.1.152

强节点 strong node

由应用选出成为强主的节点。

注:一个总线段中只可有一个强主。

3.1.153

残段 stub

在电气总线抽头处分支的 T 形连接,用于将一个设备连到线路上。

3.1.154

用户 subscriber

广播数据集的宿之一。

注:参见“发布者”。

3.1.155

监视数据 supervisory data

链路层监视所需的在一个总线内传送的数据。

3.1.156

系统地址 system address

管理者与代理者间交换管理消息的网络地址,由节点地址及站标识符组成。

3.1.157

抽头 tap

从段上进行抽头的地方,抽头是一个三路电分岔。

3.1.158

报文 telegram

由主帧和相应的从帧构成的整体。

3.1.159

端接器 terminator

终结一段电气传输线的电路。

注：理想值为其特征阻抗。

3.1.160

端接开关 terminator switch

在 WTB 总线段的末端插入端接器的开关。

3.1.161

拓扑 topography

描述连接到列车骨干网上的节点的数据结构。

注：包括它们的地址、朝向、位置及节点描述符。

3.1.162

拓扑结构 topology

一个给定网络中可能的电缆互连形式及设备数。

3.1.163

拓扑计数器 topography counter

节点中的计数器，它在每次新的初运行时累加。

3.1.164

通信存储器 traffic store

由网络 and 用户二者共同访问的共享存储器，它包含过程数据端口。

3.1.165

列车通信网络 train communication network

连接轨道交通机车车辆车载可编程电子设备的数据通信网络。

3.1.166

列车总线 train bus

列车骨干网 train backbone

连接列车中各编组且遵守 TCN 协议的总线。

3.1.167

列车网络管理 train network management

TCN 网络管理的各种服务。

3.1.168

收发器 transceiver

发送器与接收器的组合。

3.1.169

发送器 transmitter

可在物理介质上发送信号的电子器件。

3.1.170

传送数据 transport data

传输层中传送的数据。

3.1.171

传输层 transport layer

OSI 模型中负责端对端流量控制和差错恢复的层。

3.1.172

主干电缆 trunk cable

沿编组布置的电缆,不同于扩展电缆或跨接电缆。

3.1.173

用户地址 user address

功能之间交换用户消息的网络地址。

注:由节点地址(或组地址)及功能标识符组成。

3.1.174

用户消息 user message

用户功能之间交换的消息。

3.1.175

变量传送 variables

一种 TCN 传送服务。

3.1.176

变量偏置 var_offset

过程变量在数据集内的位偏移量。

3.1.177

弱主 weak master

弱节点是当前主设备,当发现有更强的主设备存在时将放弃主权。

3.1.178

弱节点 weak node

可自发地接管总线主权,但在检测到更强的节点时将释放总线主权的节点。

3.1.179

绞线式列车总线 wire train bus;WTB

适用于编组经常联挂和解联的列车总线。

3.2 缩略语

下列缩略语适用于本文件。

ALI:应用层接口(Application Layer Interface)

AMA:应用消息适配器(Application Messages Adapter)

AMI:应用消息接口(Application Messages Interface)

ANSI:美国国家标准研究所(American National Standard Institute)

ASI:应用监视接口(Application Supervision Interface)

ASN.1:抽象语法标记一(Abstract Syntax Notation One)

AVI:应用变量接口(Application Variables Interface)

BR:位速率(Bit Rate)

BT:位时间(Bit Time)

CRC:循环冗余校验(Cyclic Redundancy Check)
FCS:帧校验序列(Frame Check Sequence)
HDLC:高级数据链路控制(High-level Data Link Control)
IEC:国际电工委员会(International Electrotechnical Commission)
IEEE:电气与电子工程师协会(Institute of Electrical and Electronics Engineers)
ISO:国际标准化组织(International Standard Organisation)
ITU:国际电信联盟(International Telecommunication Union)
LFLD:链路故障位置检测(Line Fault Location Detection)
LME:层管理实体(Layer Management Entity)
LMI:层管理接口(Layer Management Interface)
MAC:介质访问控制(Medium Access Control)
MAU:介质连接单元(Medium Attachment Unit)
MIB:管理信息库(Management Information Base)
MVB:多功能车辆总线(Multifunction Vehicle Bus)
NRZ:不归零(Non-Return to Zero)
OSI:开放式系统互连(Open System Interconnection)
PDM:过程数据编排(Process Data Marshalling)
RTP:实时协议(Real-Time Protocols)
SDL:规范及描述语言(Specification and Description Language)
TCN:列车通信网络(Train Communication Network)
TNM:列车网络管理(Train Network Management)
UIC:国际铁路联盟(International Union of Railways)
WTB:绞线式列车总线(Wire Train Bus)

3.3 约定

3.3.1 数值基础

如无其他说明,本部分所有数值采用十进制表示。

模拟量和小数值用点号分开。

示例 1:电压为 20.0 V。

二进制和十六进制值按 ASN.1(见 GB/T 16262)的约定表示。

示例 2:十进制数 20 的 8 位二进制编码='0001 0100'B,十六进制编码='14'H。

3.3.2 命名约定

TCN 标准中的关键字首字母大写。

若是复合字,则字的不同部分间以空格连接。

若数据结构由关键字组成,则其类型由以下划线分隔的相同关键字组成。

当关键字对应的值在消息中发送时,相应的字段有与类型相同的名字,但需小写。

当值作为参数传递时,参数与消息中的字段有相同的名字。

在 SDL 图中,相应的变量与类型有相同的名字,但没有下划线。

示例:

拓扑计数器(Topo Counter)是链路层的一个计数器。

其类型是 Topo_Counter,为六位无符号数(UNSIGNED6)。
当其值在消息中传送时,相应的字段称为 topo_counter。
当其值传送通过过程接口时,参量被称为 topo_counter,其 C-type 是 Type_Topo_Counter。
在 SDL 图中,表示该计数器的变量称为 TopoCounter。

3.3.3 时间命名约定

以小写字母开头的时间值(如 t_mm)是可测量的时间间隔。
以大写字母开头的时间值(如 T_reply)是参数或超时值。

3.3.4 过程接口约定

过程接口用一组服务原语定义,这组原语表示了服务用户和服务提供者之间抽象的、实现无关的交互。
在本部分中,这些原语用带类型参量的 ANSI C 语法中的过程来表示。
应被看作仅提供了语义描述,并没有隐含特定的实现或语言。允许提供相同语义的任何接口。
不能对该接口的语法声明一致性。只要提供了规定的服务,实现可自由改变过程名或参数名、增加参数或拆分过程。

在 ANSI C 语法中接口过程使用 Courier 字体定义。
过程名、变量及参数小写。

示例 1:lm_send_request。

常数及类型定义大写。

示例 2:UNSIGNED32。

过程或类型名加前缀:

——对变量传送服务:

- lp_ 或 LP_ 链路层;
- ap_ 或 AP_ 应用层。

——对消息传送服务:

- MD_ 一般消息;
- lm_ 或 LM_ 链路层;
- nm_ 或 NM_ 网络层;
- tm_ 或 TM_ 传输层;
- sm_ 或 SM_ 会话层;
- am_ 或 AM_ 应用层。

表 1 是用于过程和类型的示例。

表 1 接口过程规范模板示例

定义	服务或数据类型在此表示。 在指示过程的情况,触发调用的事件在此指明,用“when”开始。 服务过程名及参数在此定义。 在指示过程的情况,规定过程类型。 区分输入参数、输出参数和返回参数
----	--

表 1 (续)

语法	<pre>MD_RESULT lm_send_request (/* example */ unsigned destination, UNSIGNED8 link_control, MD_PACKET * p_packet, ENUM8 * status);</pre>	
输入	提供给过程的输入参数不准许修改	
	destination	数据类型“unsigned”与编译器相关
	link_control	参数通过引用传递,过程不能修改。数据类型是 8 位字
	p_packet	“*”表示指向 p_packet 数据结构的指针,p_packet 数据结构类型 MD_PACKET 在本部分别处定义
输出	输出参数由调用修改	
	status	类型 ENUM8 是 8 位枚举类型
结果	结果参数是可选的输出参数,用于表示调用的成功或失败,但不是服务必须的	
	MD_RESULT	结果参数为以下类型: 对 AMI 为 AM_RESULT; 对 LMI 为 MD_RESULT; 对 LPI 为 LP_RESULT; 对 AVI 为 AP_RESULT。 模板对每个过程规定了各自的出错代码。 若结果参数仅取以下两值,则可不必解释: xx_OK = 0 成功结束; xx_FAILURE <> 0 出错。 结果也可作为参数列表中的输出参数返回,这取决于实现过程
用法	过程模板之后列出的规则指示宜如何使用过程。虽然使用规则不是强制性的,但不遵守这些规则可能导致不可预知的结果	
注: 本表中表示的数据结构为接口规范,不宜与总线上传送的相同数据结构的格式混淆,见 3.3.5。		

3.3.5 传送数据规范

被传送数据的格式,不论是单帧还是整个消息,都以两种形式规定:

- a) 图解式,不规范但可直观显示消息结构;
- b) 基于 ASN.1 的文本式,编码规则在 6.3 中规定。

示例 1: 表 2 给出了一个消息的图解式,相应的文本式在表 3 中给出。

表 2 消息结构示例

		首先传送的八位位组								其次传送的八位位组							
位	—>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		snu	gni	node_id						station_id							
2		next_station_id								rsv1	Tvd	topo_counter					
4		tnm_key								sif_code (=3)							
6		参数 1															
8		参数 2								参数 3						参数 4	
参数 5:ARRAY [n] OF (下面的字段重复 n 次)																	
参数 5.1																	
参数 5.2																	
参数 5.3:STRING32																	
(CHARACTER8)										最后一个字符或 0							

^八位位组

位编号以一个字节或一个字中 2 的幂的形式表示。位编号不表示总线上传送顺序,有两种传送顺序:最高有效位(MSB)优先和最低有效位(LSB)优先。

在图解式中,每行用于表示一个 16 位字,但在第 5 章中,每行是 8 位的。

参数数组由顶部和左部的重复框处理。

重复可嵌套(见表 2 中的参数 5.3)。

若参数长度大于 3 个字,应为其分配 3 行,而中间一行应有阴影框。

表 3 文本式消息示例(对应于表 2)

Call_Mgt_Message::=RECORD		
{		
snu	BOOLEAN1 (=1),	——“1”表示消息使用系统编址
gni	BOOLEAN1 (=0),	——“0”表示终点实体是一独立设备
node_id	UNSIGNED6,	——6 位终点实体节点地址
station_id	UNSIGNED8,	——8 位站标识符
next_station_id	UNSIGNED8,	——8 位下一站标识符
rsv1	BOOLEAN1 (=0),	——该位恒为 0
tvd	BOOLEAN1,	——该位指示拓扑计数器是否有效
topo_counter	UNSIGNED6,	——6 位拓扑计数器
tnm_key	UNSIGNED8,	——通知网络管理呼叫消息
sif_code	UNSIGNED8,	——对每一个管理消息有不同的 SIF_code16 位的值。
parameter1	INTEGER16,	如果参数不足 16 位,则右对齐且符号扩展(也就是
parameter2	INTEGER8,	1 个八位位组是作为第 2 个八位位组来传送)
parameter3	UNSIGNED6,	该值在字的高八位位组发送
par4	ANTIVALENT2,	该值在低八位位组中发送,但其低 2 位作为参数 4
parameter5	ARRAY [n] OF	参数 4 有 2 位
		参数 5 表示应重复 n 次的结构数据,包含:

表 3 (续)

{		
parameter5.1	INTEGER16,	——重复字段的第一个参数
parameter5.2	UNIPOLAR4.16,	——重复字段的第二个参数
parameter5.3	STRING32	——第三个参数为一字符串,(最多 32 个 8 位字符的数组);
		——以 1 个“0”字符结尾,或用 2 个“0”字符结尾以校准到 16 位字边界上的字符串;
		——32 个“0”字符组成的空字符串;
		——字符串的实际长度为“0”前有效字符数
}		
{		
}		

字段名首字母小写,其类型首字母大写。有时,同一类型用作传送格式时,仅首字母大写;而用作 C_type 时,整个类型大写。

示例 2:Am_Result(传送格式)和 AM_RESULT(接口过程的 C_type)。

3.3.6 状态图约定

传送协议状态机按照 GB/T 15629.2(逻辑链路层)以表格形式描述。GB/T 15629.2 规定了状态机在可能状态之间的转移关系。

状态间的转移由事件支配。事件可来自网络层(入境包)、会话层(命令)或超时。

在离开该状态前执行与事件相关的动作。该动作定义了下一个状态。

图 3 给出了状态转移图的一个实例。

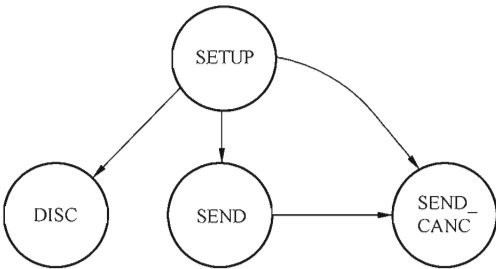


图 3 状态转移实例

从 SETUP 状态开始,状态机可转移到三种状态:解联(DISC)、发送(SEND)或取消发送(SEND_CANC)。

这些状态间的转移如表 4 所示。

表 4 状态转移表

当前状态	事件	动作	下一状态
SETUP (起动)	rcv_DR	close_send (DR_reason);	DISC
	rcv_CC AND (conn_ref = CC_conn_ref)	IF (eot) THEN close_send (AM_OK); ELSE credit := CC_credit; send_not_yet := credit; send_data_or_cancel; END;	DISC SEND or SEND_CANC
	TMO AND (rcp_cnt = MAX_REP_CNT)	close_send(AM_CONN_TMO_ERR)	DISC

根据表 4,有三个事件导致 SETUP 状态转移到 DISC 状态:

- a) rcv_DR(收到解联请求,网络事件)。相应的动作是关闭连接后转移到 DISC 状态。
- b) 另一网络事件(收到有正确引用的连接确认),导致转移到 DISC、SEND 或 SEND_CANC 状态之一,如何转移取决于发送数据的输出或取消过程。
- c) 预定的超时条件(rcp_cnt = MAX_REP_CNT),也导致连接关闭。

3.4 总则

3.4.1 设备间接口

本部分定义了编组内设备的数据通信接口,该接口作为设备到编组网的连接,如图 4 所示。

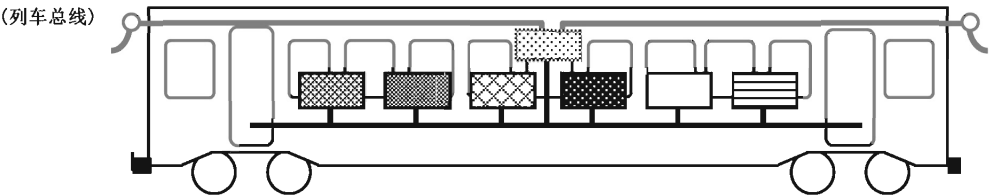


图 4 设备间接口

编组网主要是为有互操作性和互换性需求的设备互连而设计的,但不排除用于其他场合。

3.4.2 编组间接口

本部分定义了编组间的数据通信接口,该接口作为编组上节点到列车总线的连接,如图 5 所示。

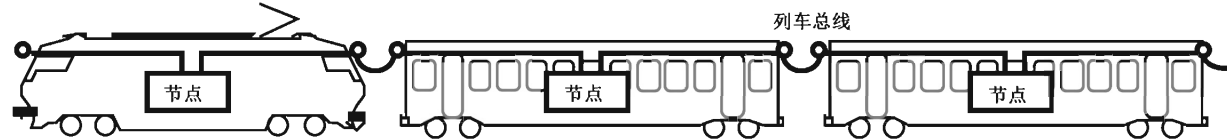


图 5 编组间接口

对于列车总线,本部分规定了 WTB。WTB 主要是为开式列车中编组互连而设计的串行数据通信总线,但不排除用于其他场合。

注:开式列车的定义见第 3 章。

3.4.3 实时协议

本部分定义的 TCN 体系是由列车总线和编组网构成的两层体系,如图 6 所示。

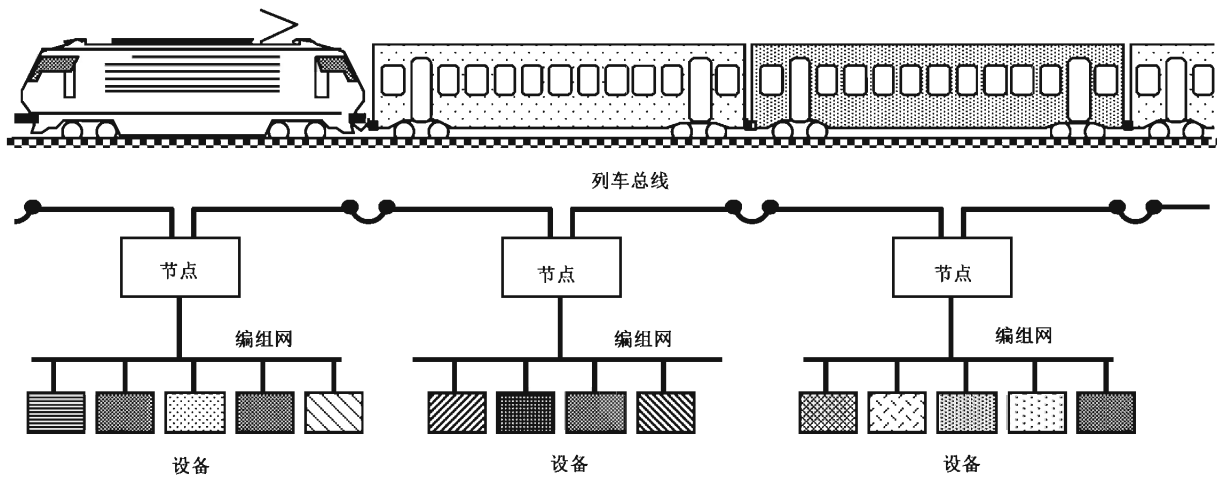


图 6 列车总线和编组网

对于通信协议,第 6 章规定了 WTB 上所有节点使用的实时协议(RTP)。编组网上的设备可使用相同的 RTP(例如 MVB),或使编组网协议适应 WTB 节点 RTP。

RTP 规定了 TCN 提供的接口,包含两种基本服务:变量传送和消息传送。

RTP 规定了处理路由选择、流量控制及差错恢复的传送协议。

RTP 规定了总线需要提供给传送协议的接口,尤其是以下两种基本服务:

- a) 过程数据:周期的、源寻址广播;
- b) 消息数据:偶发的、无连接传送。

3.4.4 网络管理

对于网络管理,第 8 章规定了 TCN 网络管理(TNM)。TNM 作为管理者与代理者间交换的消息集来提供基本服务。

3.4.5 组态

本部分可部分使用,也可整体使用。例如,可能使用:

- a) 无编组网或与非 MVB 的编组网联用的 WTB。
- b) 与非 WTB 且非 MVB 的其他总线联用的 RTP。

图 7 展示了对应于三种应用领域的三种组态:

- a) 开式列车组态展示了需要自动组态的开式列车(如 UIC 列车)。WTB 用作标准的列车总线,最多支持 32 个节点。每个编组可有 0 个、1 个或多个节点。每个节点最多可连接 15 个编组网(MVB 或其他)。

- b) 多单元列车组态展示了两个相连的闭式列车。当这些闭式列车经常联挂和解联时,WTB 可用作标准的列车总线。但若可用其他方法组态时,也可用其他总线(如 MVB)来作为列车总线。编组网可穿越多个车辆。
- c) 闭式列车组态展示了在闭式列车上编组网(如 MVB)即用作列车总线又用作车辆总线。

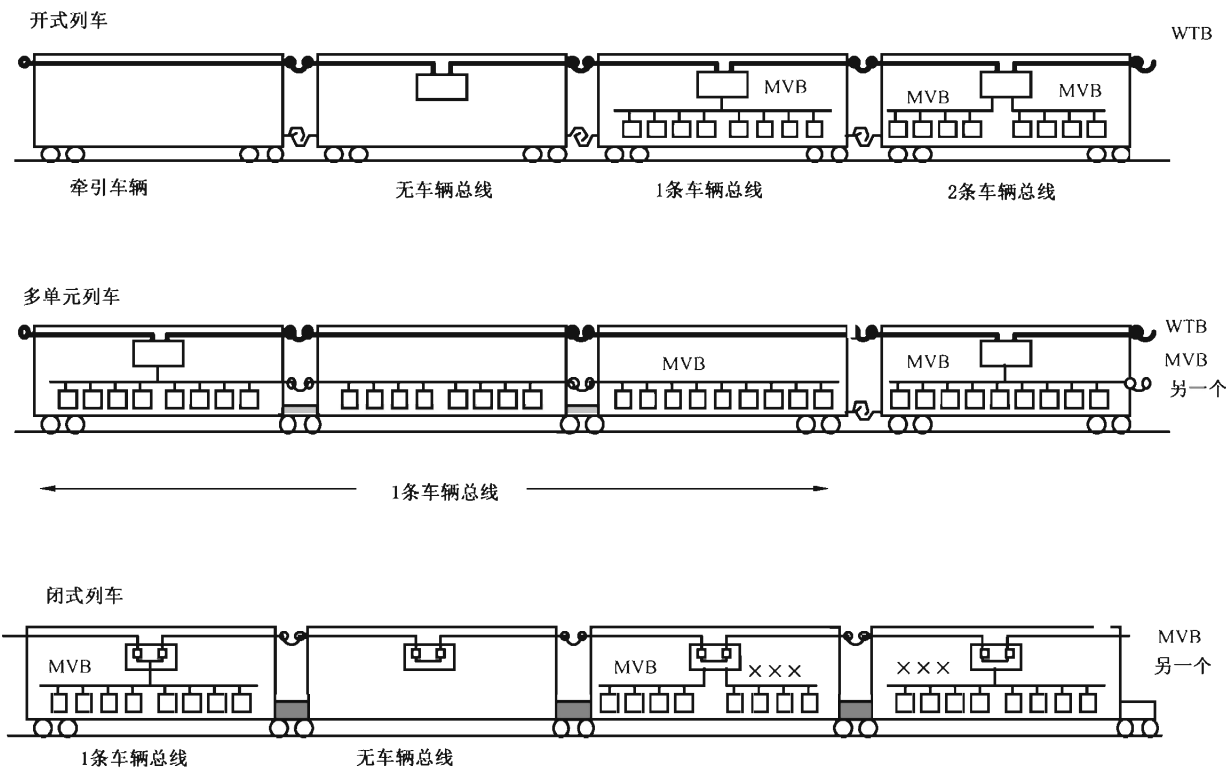


图 7 TCN 的组态

在图 7 的三种组态中,使用 MVB 编组网连接车载设备,但也可使用其他总线作为编组网。

3.4.6 标准设备结构

3.4.6.1 概述

本条规定了 TCN 设备允许的可选项。这些可选项在相应的章条中描述,并综合在图 8 中。

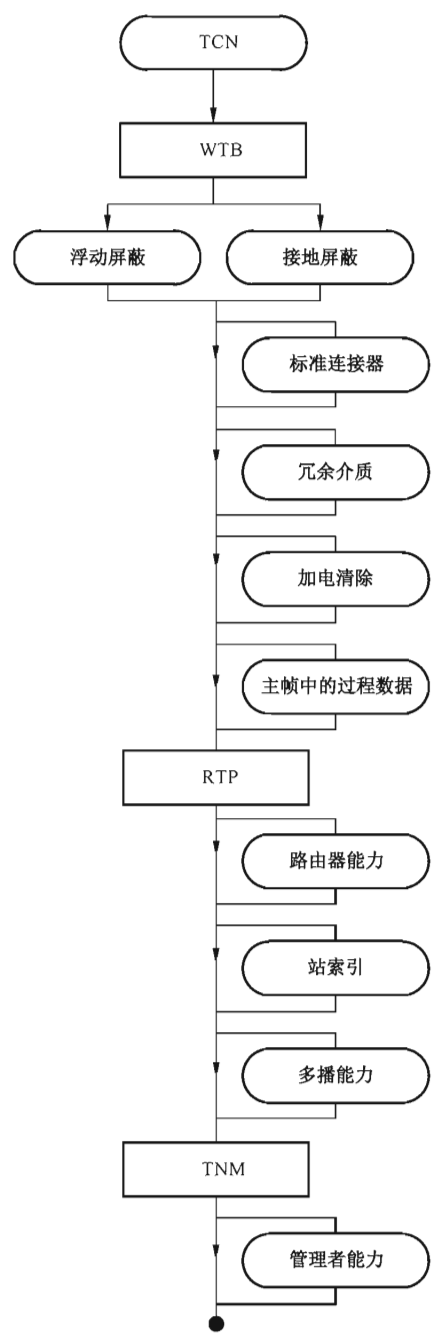


图 8 TCN WTB 设备配置的可选项

3.4.6.2 TCN 设备

TCN 兼容的 WTB 设备应实现至少一个 WTB 总线 MAU。

3.4.6.3 WTB 可选项

- WTB MAU 应可配置成浮动屏蔽或接地屏蔽。
- WTB MAU 可使用本部分规定的连接器。
- WTB MAU 可使用本部分规定的冗余介质。
- WTB MAU 可实现本部分规定的加电清除。

WTB MAU 可按本部分规定在主帧中传送过程数据。

3.4.6.4 RTP 可选项

- TCN 设备应实现变量传送服务。
- 除了 MVB 1 类设备外,其他 TCN 设备应实现消息传送服务。
- TCN 设备若有一个以上 MAU,则可实现路由器功能。
- TCN 节点可实现节点索引。
- TCN 设备可实现站索引。
- TCN 设备可实现多播协议。

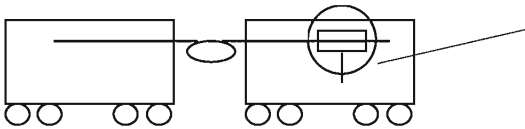
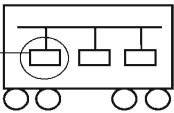
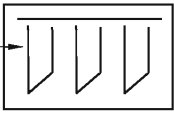
3.4.6.5 TNM 可选项

- TCN 设备应实现代理者功能。
- TCN 设备可实现管理者功能。

3.5 一致性测试

为申明对 TCN 标准的一致性,设备应通过一组测试。
为确信根据表 5 可保证互操作性,GB/T 28029.3—2020 中规定了一致性测试。

表 5 互操作性测试

接口等级	GB/T 28029	终端用户
<div>1 级</div> <div>机车车辆间接口</div> <div>(列车总线)</div> <div></div>	<div>协议</div> <div>物理的/电气的</div> <div>节点连接器</div> <div>电缆规范</div> <div>(Z:衰减)</div>	<div>用户消息</div> <div>机械的(连接器/电缆)</div>
<div>2 级</div> <div>设备间接口</div> <div>(车辆总线)</div> <div></div>	<div>协议</div> <div>物理的/电气的</div> <div>机械的(连接器/电缆)</div>	<div>用户消息</div>
<div>3 级</div> <div>印制板或内部器件间接口</div> <div></div>	<div>未包含</div>	<div>未包含</div>

4 物理层

4.1 概述

本章规定了 WTB 的物理介质,即一种工作速率为 1.0 Mbit/s 的屏蔽绞线式总线对。
本章旨在为不同的节、节点和连接器提供一种尽可能统一的信号传播电气介质。

4.2 拓扑结构

4.2.1 总线节

WTB 总线应由下列类型总线节互连的节点组成:

- a) 沿编组分布的干线电缆(连贯车辆只有干线电缆);
- b) 连接不同编组干线电缆的跨接电缆;
- c) 延伸干线电缆以到达节点的扩展电缆。

4.2.2 耦合器

连接器和连接盒可用于装配节点和电缆节。
每个编组含有总线 and 一定数目的节点。

4.2.3 节点

在常规运行中,每个节点应插入到干线电缆中并连接两个总线节:

- a) 位于总线末端的节点,即末端节点,应电气终结连接到其上的两个总线节;
- b) 位于总线中间的节点,即中间节点,应电气连接到其上的两个总线节。

连接到一个节点的两个电缆节应被命名为方向 1(Direction_1)和方向 2(Direction_2)。

在一列车上,可每个编组一个节点,或者每节车辆多个节点,如图 9 所示。

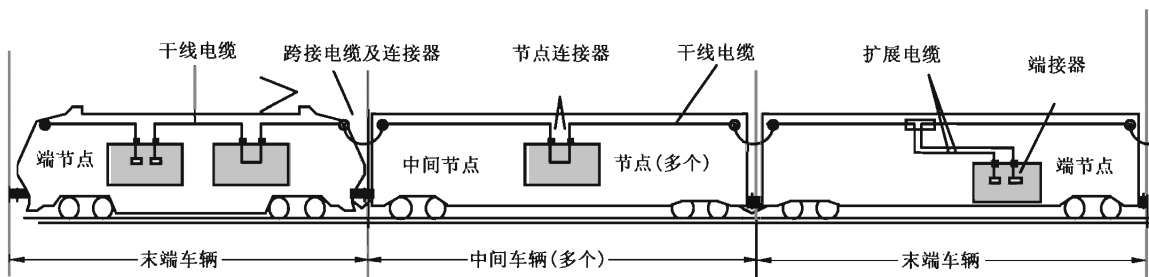


图 9 列车组成(图中两个中间节点)

4.2.4 编组朝向

节点的朝向对于左右识别是非常关键的,应遵循以下约定:

- a) 编组的一个末端标识为末端 1(Extremity 1),另一个末端标识为末端 2(Extremity 2);
- b) 节点的方向 1 连向末端 1,方向 2 连向末端 2;
- c) 若方向 1 指向北,则编组的西侧命名为 A 侧,东侧命名为 B 侧;
- d) 节点 A 侧和 B 侧的命名遵循所在编组同样的约定。

注 1: 一个节点的方向 1 可指向另一个节点的方向 1 或方向 2,除非两个节点在同一编组内。因为一个编组相对于另一个编组的朝向是不可预测的。

注 2：定义编组的末端 1 为没有停放制动的编组末端。

4.2.5 编组规范(资料性)

由于供应商可提供单个编组或节点而不是整列车,因此总线、编组和节点的规范分别制定。

本条规定整条总线和每个节点的特性。对于特殊应用,从中推导出编组特性。若车辆、编组或节点数未超出范围,则列车的一致性得以满足。

下列计算适用于在 UIC 556 中规定的列车。其他应用,如重载运输,也可采用类似方法计算。

UIC 556 中参考列车由 22 节车辆组成,电缆长 860.0 m,没有使用中继器。通常每个编组只有一个节点,对于 32 个节点的列车,最多 10 个由单节车辆(如可驾驶拖车)组成的编组可支持两个节点。

根据 4.6.3,额定频率下节点衰减信号小于 0.3 dB,因此整列车由节点引起的信号衰减不超过 $32 \times 0.3 \text{ dB} = 9.6 \text{ dB}$ 。

根据 4.7.3,接收器能处理的动态范围是 20.0 dB,因此整列车由电缆、连接器以及其他因素引起的信号衰减不能超过 $20.0 \text{ dB} - 9.6 \text{ dB} = 10.4 \text{ dB}$ 。

分配给每个车辆的最大衰减是 $10.4 \text{ dB} / 22 = 0.5 \text{ dB}$ 。

该值是在移除节点且短接两端连接器时测得的。测量中考虑了跨接电缆的衰减,如图 10 所示。

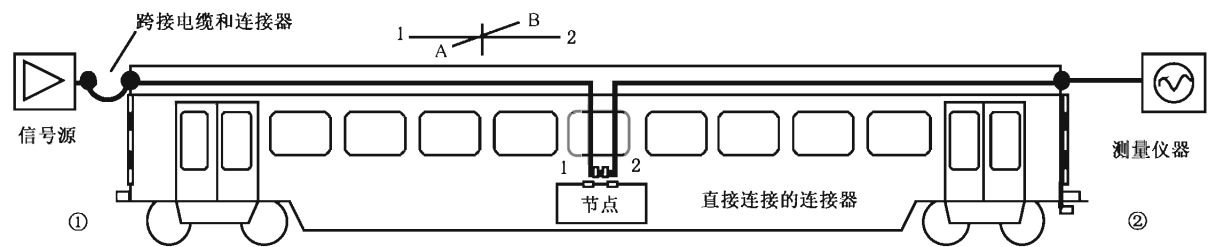


图 10 车辆衰减测量

由于电缆的弯曲和延伸,每节车辆的电缆长度大约为车辆长度的 150%。假设一节车辆的长度是 26.0 m,则列车上介质长度最大达到 860 m($22 \times 26 \text{ m} \times 1.5 = 858.0 \text{ m}$)。

为满足这些要求,列车介质衰减宜小于 10.4 dB/860.0 m 或 12.0 dB/km。由于跨接电缆、连接器和接头处都可能引入较高的衰减,因此宜使用衰减小于 10.0 dB/km 的干线电缆(见 4.3.4)。

同样的原理也适用于其他畸变参数。测量方案见 4.6.3.1。

当车辆装有冗余的 A 线和 B 线时,分别测试每条线路。

4.3 介质规范

4.3.1 拓扑结构

节点应插入 WTB 电缆,每个节点与两个总线节相连,如图 11 所示。

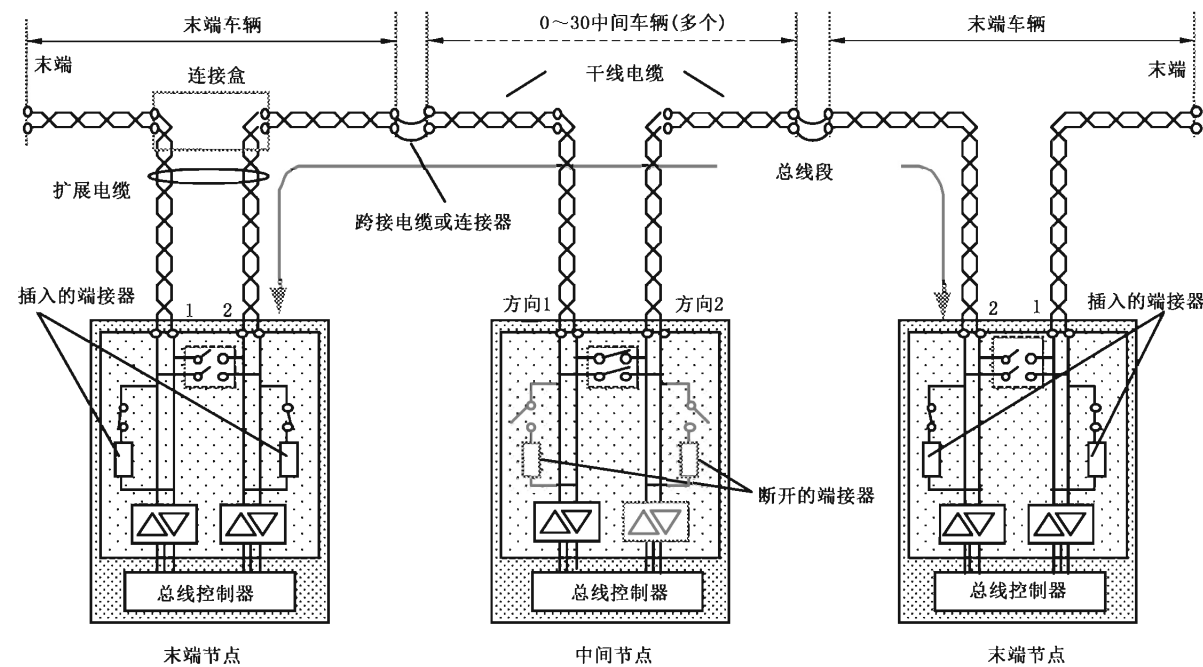


图 11 常规运行下连接的节点

节点应能：

- a) 在连接到其上的两个总线节间建立电气连续,以按中间节点动作；
 - b) 通过端接器(阻抗调节网络)电气终止连接到其上的总线节,以按末端节点动作。
- 末端节点应可在其总线节上独立地进行发送或接收,而中间节点应只有一个收发器使能。

4.3.2 冗余介质(可选)

本条定义了一种冗余方案,在此方案中每个节点通过独立的线路单元连接到两条线路上,如图 12 所示。

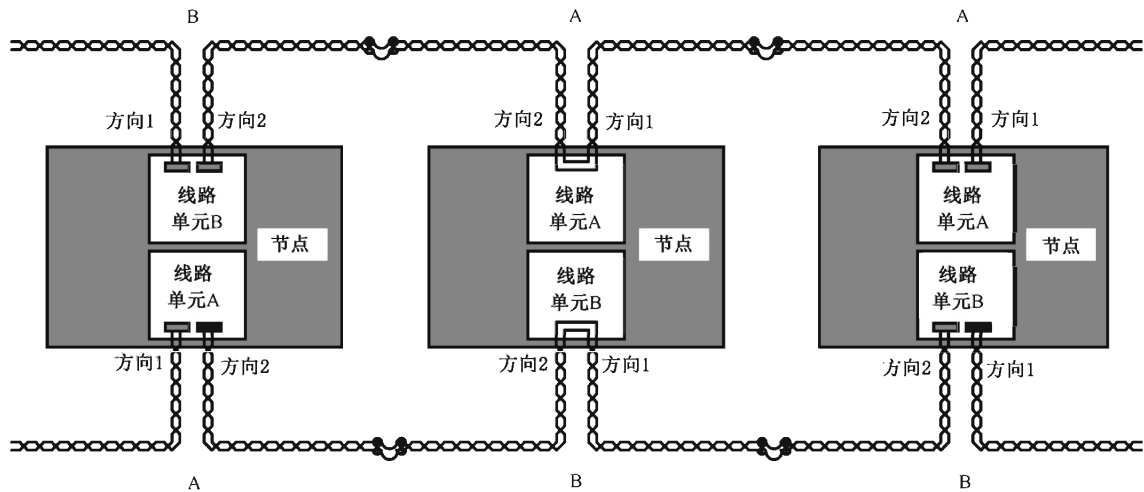


图 12 双线连接

若使用该选项,应遵循下列规范:

- a) 两条线路应标识为 A 线和 B 线;
- b) 同一编组内所有节点,该标识应一致;
- c) 属于不同线路的电缆应区分标记;
- d) A 线和 B 线的方向 1 和方向 2 应配置相同。

注 1: 在车辆之间不可能仅用一根线路连接时,双线介质方案是强制性的。

注 2: A 线布置在编组的 A 侧,B 线布置在编组的 B 侧。

注 3: 因为编组的朝向是不可预知的,一个编组的 A 线可能与另一编组的 B 线相连,如图 12 所示。

4.3.3 总线配置规则

4.3.3.1 概述

本条适用于以最大可能长度工作的总线。

除非有其他规定,所有电量值应是在输入信号频率 $1.0 \times (1 \pm 0.01\%) \text{ MHz}$ 、差分幅值 $\pm 4.0 \text{ V}$ ($8.0 V_{pp}$) 的正弦输入信号下测得的。

4.3.3.2 信号速率

由于采用曼彻斯特编码,其对应的信号频率为 1.0 MHz ($BT = 1.0 \mu\text{s}$, $BR = 1.0 \text{ MHz}$),因此所有总线段应以相同的 $1.0 \times (1 \pm 0.01\%) \text{ Mbit/s}$ 速率工作。

4.3.3.3 节点和电缆引起的延时

总线上任意两个节点之间的端到端传输延时 T_{pd} 不应超过 $60.0 \mu\text{s}$ 。

注 1: 对于给定应用,传输延时可根据式(1)计算。

$$T_{pd} = (L \times 6.0 + R \times T_{rd}) \dots\dots\dots (1)$$

式中:

L —— 电缆长度(包括干线电缆、扩展电缆和跨接电缆),单位为米(m);

6.0 —— 近似于带载传输线的传输延时,单位为纳秒每米(ns/m);

R —— 中继器的数目;

T_{rd} —— 中继器的传输延时,单位为微秒(μs)。

注 2: WTB 在 860.0 m 长度内不需使用中继器,但中继器在某些应用中是有用的。

注 3: 本条与 5.2.2.2 和 4.8.2.3 中规定的容许延时相符。

4.3.3.4 节点和电缆引起的衰减

同一总线段上任意两个节点之间的总电压衰减不应超过 20.0 dB 。此值是用频率在 0.5 BR 和 2.0 BR 之间的正弦信号测得的。

注 1: 衰减是与节点数和总电缆长度成比例的。

注 2: 本条与 4.7.3 中规定的接收器动态范围相符。

4.3.3.5 节点和电缆引起的抖动

在最大长度和支持最大节点数时,一个端结的总线段增加的边沿抖动对比理想过零点不应超过 $\pm 0.1 \text{ BT}$ 。

测试条件:

——线路由差分幅值为 $4.0 \times (1 \pm 10\%) V_{pp}$ 、中心点在 0.0 V 、源阻抗为 $22.0 \times (1 \pm 10\%) \Omega$ 的信号源驱动;

——驱动信号是曼彻斯特编码的、重复周期至少为 511 位的“0”“1”伪随机序列。

注 1: 由于总线节、残段、连接器或负载集束之间的阻抗不匹配而引起的干扰和反射可引起过零点时刻的抖动。

注 2: 本条摘自 GB/T 15629.3 或负载集束, 与 4.7.3 中规定的接收器的容许抖动相符。

4.3.3.6 冗余线路间的时滞(可选)

任意两个节点之间的 A 线和 B 线传输延时最大差值不应超过 $T_{\text{skew}}=30.0\ \mu\text{s}$ 。

注: 本条与 4.8.2.5.1 中规定的接收器容许时滞相符。

4.3.4 电缆规范

4.3.4.1 机械规范

所有电缆节应由一对带护套的屏蔽绞式的导线组成(屏蔽双绞线)。

导线对每米应至少 12 绞。

干线电缆的横截面积宜为 $0.75\ \text{mm}^2$ (AWG 18)。

跨接电缆的横截面积宜为 $1.34\ \text{mm}^2$ (AWG 16)。

若使用 4.4.5 中 Sub-D 连接器作间接连接, 则扩展电缆的每根导线横截面积不应超过 $0.56\ \text{mm}^2$ (AWG 20)。

4.3.4.2 标记

双绞线的每一根导线应分别标识为 X 和 Y, 屏蔽层标识为 S。

电缆的每一根线应清晰标记。

在所有连线和接合点处应标记。

4.3.4.3 特性阻抗

所有总线节应存在 $Z_w=120.0\times(1\pm10\%)\ \Omega$ 的差分特性阻抗, 该值是用频率为 0.5BR 和 2.0 BR 之间的正弦信号测得的。

4.3.4.4 电缆衰减

在 1.0 BR 频率时电缆对正弦信号的衰减应小于 10.0 dB/km, 在 2.0 BR 频率时, 电缆对正弦信号的衰减应小于 14.0 dB/km。

4.3.4.5 分布电容

在 1.0 BR 频率时, 电缆的差分(线对线)分布电容不应超过 65 pF/m。

4.3.4.6 电缆对屏蔽层的电容不平衡

在 1.0 BR 频率时, 电缆对屏蔽层的电容不平衡不应超过 1.5 pF/m。

4.3.4.7 串扰抑制

在同一扩展电缆中有两对导线时, 从一对导线到另一对导线的信号抑制在 0.5 BR~2.0 BR 频率范围内应大于 55.0 dB。

4.3.4.8 屏蔽品质

按照 GB/T 28029.3—2020 中 5.6.1.2 的规定测量, 电缆转移阻抗在 20.0 MHz 频率时应小于 $20.0\ \text{m}\Omega/\text{m}$ 。

按照 GB/T 28029.3—2020 中 5.6.1.2 的规定测量, 电缆差分转移阻抗应小于 $2.0\ \text{m}\Omega/\text{m}$ 。

4.3.4.9 连接器品质

注：本条不适用于车辆之间的连接器。

所有电缆连接应提供导线和屏蔽层的连续性，且连接电阻小于 10.0 mΩ。

按照 GB/T 28029.3—2020 的 5.6.1.2 规定测量，连接器管脚与屏蔽层之间的转移阻抗在 20.0 MHz 频率时应小于 20.0 mΩ，任意两个管脚之间的转移阻抗应小于 2.0 mΩ。

4.3.5 屏蔽概念

4.3.5.1 概述

为满足不同应用，规定了下列两种屏蔽概念：

- 接地屏蔽概念(首选)；
- 浮动屏蔽概念。

4.3.5.2 接地屏蔽概念

当应用接地屏蔽概念时：

- 屏蔽层应直接与每个节点的节点地相连；
- 在车辆之间，跨接电缆不应建立屏蔽的连续性，如图 13 所示。

注 1：屏蔽层宜尽可能地与地相连，例如：在车辆的末端、机箱的外壳等处，以便在较短路径内回环感应电流，防止感应电流通过屏蔽层产生电磁干扰。这要求车体有良好的传导性，以防止牵引设备大的漏电流。

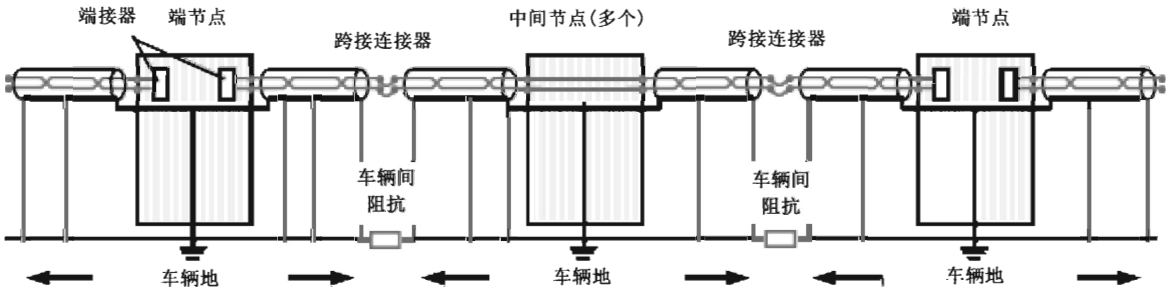


图 13 接地屏蔽概念

注 2：接地屏蔽概念由 UIC 558 规定。

4.3.5.3 浮动屏蔽概念

当应用浮动屏蔽概念时：

- 当屏蔽层没有与节点连接时，屏蔽层应与地隔离；
- 屏蔽层应通过一个阻容(RC)电路与每一节点的节点地相连，RC 电路的参数分别为： $R_s=47.0\times(1\pm5\%)k\Omega$ ，并联电容 $C_s=100.0\times(1\pm10\%)nF$ ，750.0 V，如图 14 所示。

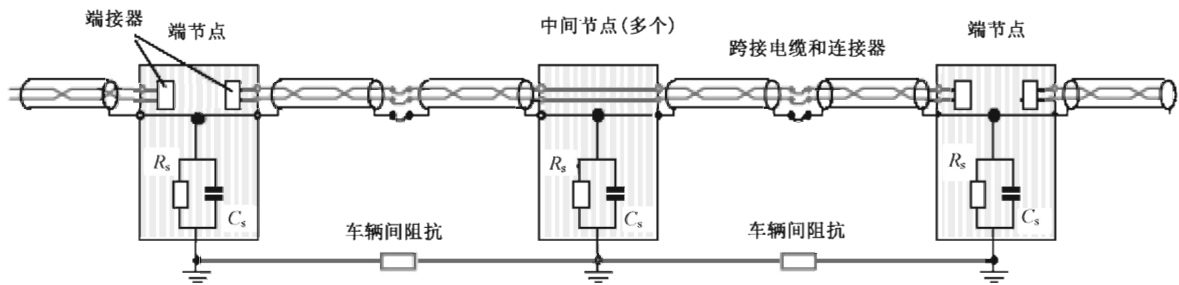


图 14 浮动屏蔽概念

4.3.6 端接器

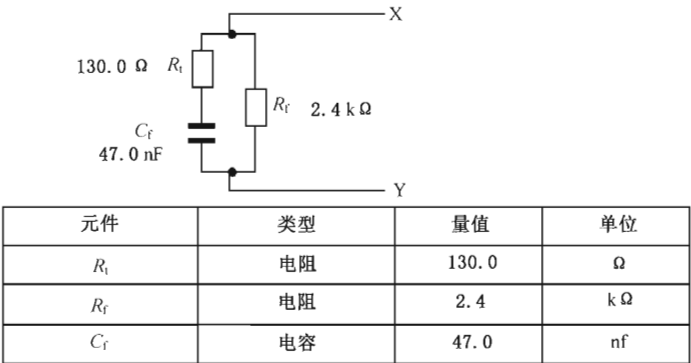
末端节点应电气上终止通过端接器连接到其上的两个总线段。

端接器应为在 $0.5 \text{ BR} \sim 2.0 \text{ BR}$ 频率范围之内相位角小于 0.087 rad 的无极性阻抗，其值为 $Z_w \times (1 \pm 5\%)$ 。

端接器应与电缆屏蔽层隔离。

端接器在 X 和 Y 之间的直流阻抗应为 $2.4 \text{ k}\Omega$ ，且能消耗至少 1.0 W 的持续功率（即使在不要求加电清除的应用中）。

示例：图 15 为一个推荐电路。



说明：

R_i ——电阻；

R_f ——电阻；

C_f ——电容。

图 15 端接器

4.4 介质连接

4.4.1 概述

- 本条规定了两种连接节点的方法：
- a) 直接节点连接，不通过连接器直接将节点插入干线电缆；
 - b) 间接节点连接，通过连接器将节点与干线电缆相连。

4.4.2 节点连接点标识

节点应将与之相连的两个总线节标识为方向 1 和方向 2,仅对该节点。
节点应将与之相连的两条线路标识为 A 线和 B 线。若只使用一条线路,应标识为 A 线。
线路单元的电缆连接点应如下命名:

- a) A 线方向 1 为:A1X、A1Y 和 A1S;
- b) A 线方向 2 为:A2X、A2Y 和 A2S;
- c) B 线方向 1 为:B1X、B1Y 和 B1S;
- d) B 线方向 2 为:B2X、B2Y 和 B2S。

4.4.3 直接节点连接

直接节点连接应将节点插入电缆,并用螺栓或其他满足电气和机械要求的紧固件连接,如图 16 所示。

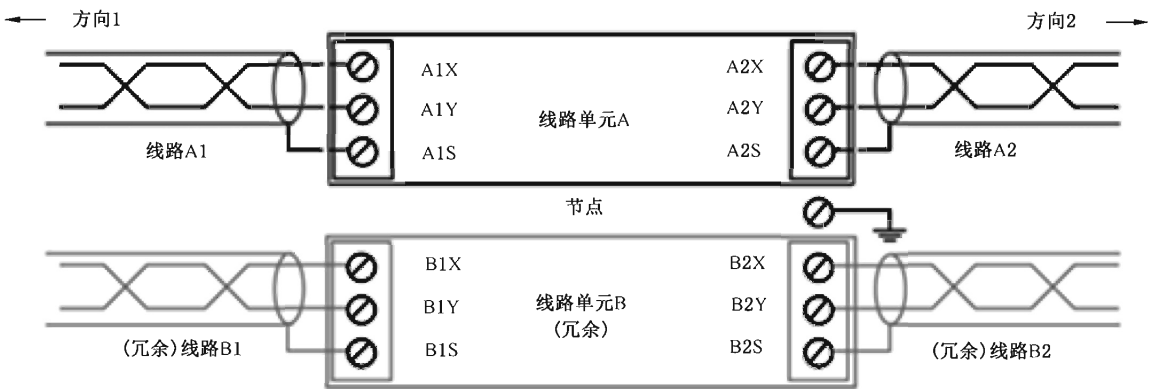


图 16 直接节点连接(可选双线)

应能移除节点并将两个方向的电缆相连,以提供电缆和屏蔽的连续性。

4.4.4 间接节点连接

间接节点连接应在单线连接中使用两个连接器,在双线连接中使用 4 个连接器,如图 17 所示。

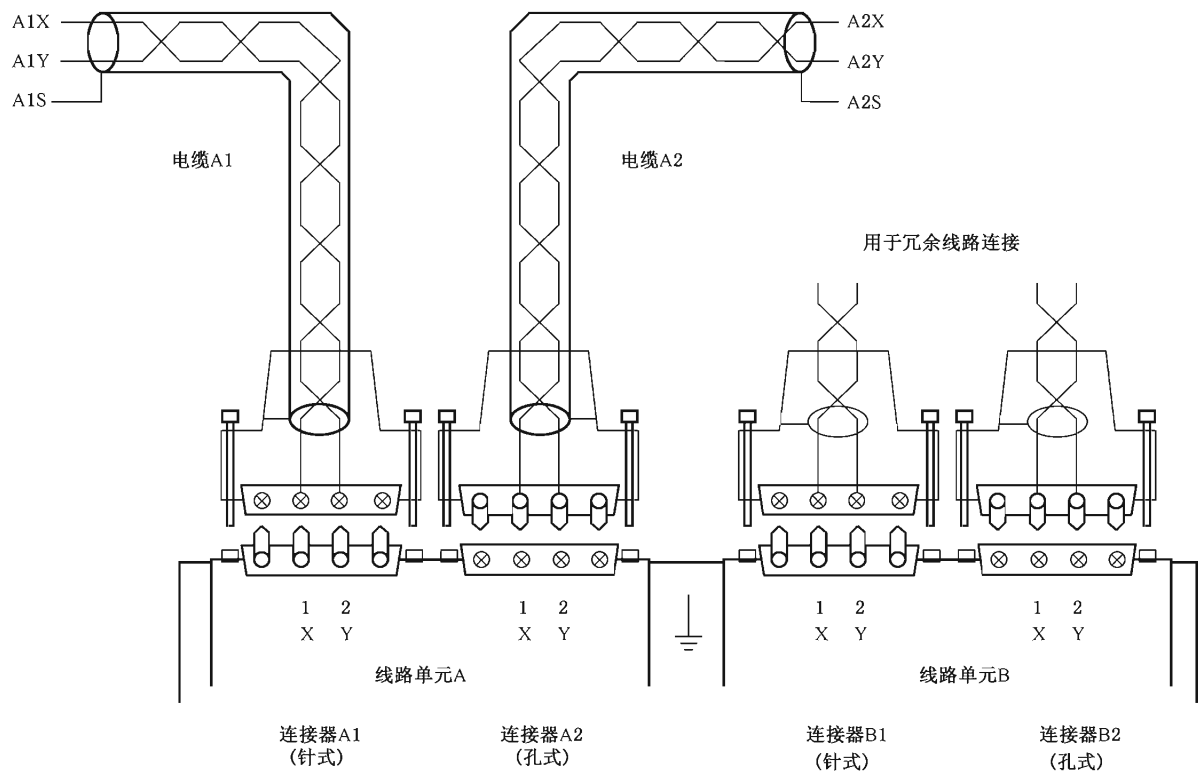


图 17 间接节点连接

4.4.5 连接器(可选)

有互换性要求时,间接节点连接应按下列规则连接到电缆:

- 连接器应为 SUB-D 型连接器(见 IEC 60807)。
- 连接器应具有导电的屏蔽外壳,使得:
 - 当采用接地屏蔽概念时,外壳应与电缆屏蔽层连接并在紧固后与插座保持电气接触;
 - 当采用浮动屏蔽概念时,外壳可与电缆屏蔽层隔离。
- 连接器应采用公制螺纹。
- 连接器应具有以下极性和布置:
 - 在方向 1,线路单元上应使用针式连接器,电缆上应使用孔式连接器;
 - 在方向 2,线路单元上应使用孔式连接器,电缆上应使用针式连接器;
 - 同一线路的连接器垂直布置时,上部连接器应为方向 1,下部连接器应为方向 2,上部线对应为 A 线;
 - 同一线路的连接器水平布置时,左侧连接器为方向 1,右侧连接器为方向 2,前视节点时左部线对为 A 线。
- 应能将两个方向的电缆连接器连接并紧固在一起,以提供电缆和屏蔽的连续性。
- 连接器(针式或孔式)的引脚布置应如表 6 和图 18 所示。

表 6 WTB 连接器引脚布置

引脚号	信号定义
1	X 正线
2	Y 负线
3	保留(用于屏蔽层)
4	保留
5	保留
6	保留
7	保留
8	保留
9	保留

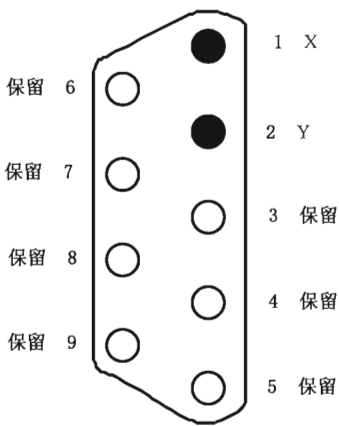


图 18 WTB 连接器前视图

注：宜在采用浮动屏蔽概念时优选插座与节点外壳电气隔离。

4.5 节点规范

4.5.1 节点组成

4.5.1.1 概述

节点应通过其介质连接单元(MAU)连接到介质上。

单线连接的 MAU 应包括：

- a) 一个线路单元；
- b) 一个方向转换器；
- c) 一个主通道和一个辅助通道。

示例：中间设置带总线开关的 MAU，如图 19 所示。

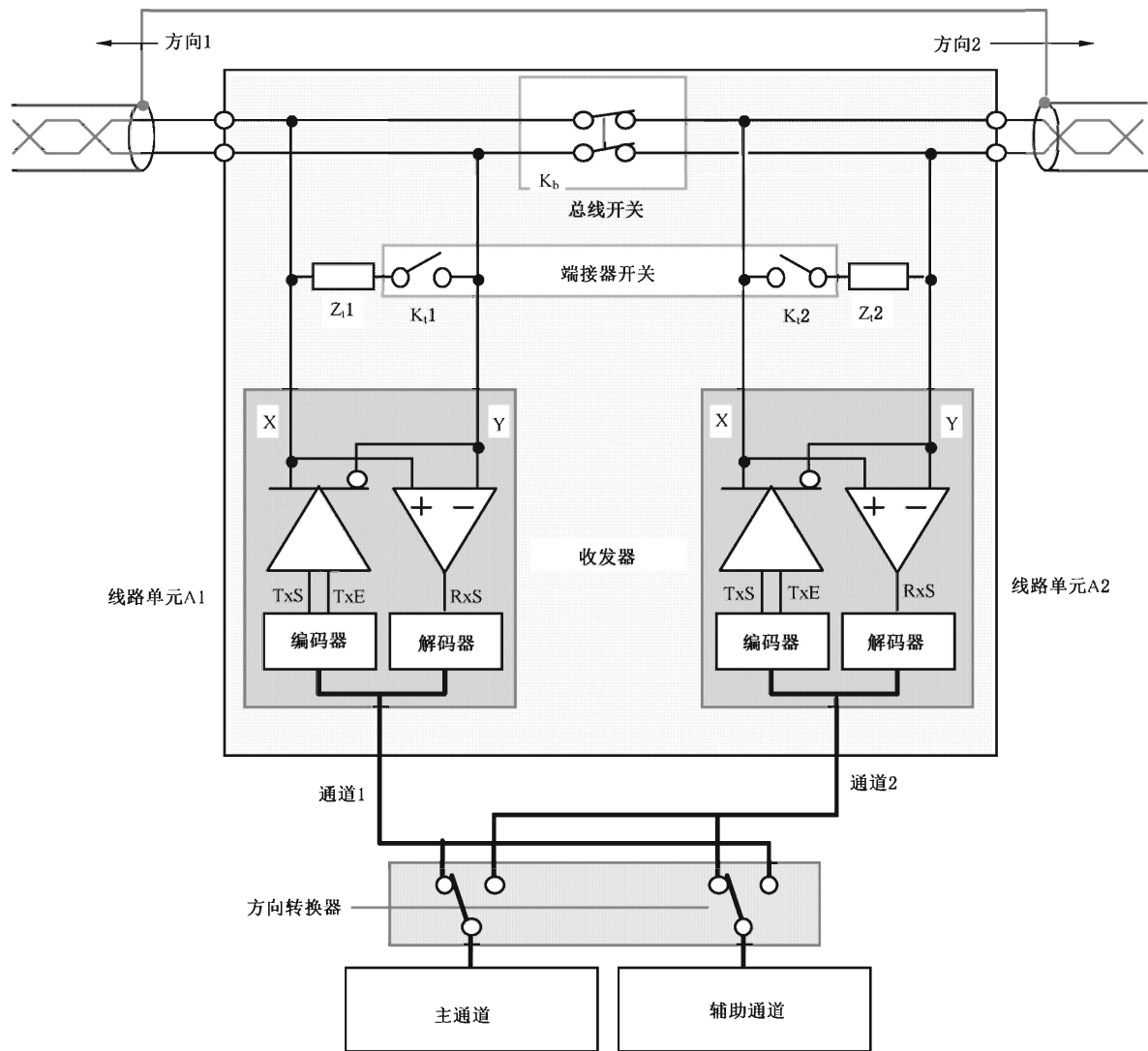


图 19 MAU 结构示例

4.5.1.2 线路单元组成

一个线路单元应包含：

- a) 一个总线开关(K_b)，用于两个方向的连接或断开。
- b) 两个端接器开关 $K_{1,1}$ 和 $K_{1,2}$ ，每个方向一个。末端设定时在节点插入端接器($Z_{1,1}$ 、 $Z_{1,2}$)，中间设定时移除端接器。
- c) 两个收发器(发送器/接收器)电路，每个方向一个。每个发送器由二进制信号 T_xS (信号)和 T_xE (使能)控制。接收器的输出信号是 R_xS (数字信号或模拟信号)。收发器通过适当的方法，例如采用变压器，实现与线路的电气隔离。
- d) 两个曼彻斯特编码/解码器，每个收发器一个。编码/解码器可集成在各自的发送器或接收器内。曼彻斯特编码/解码器的输出和输入规定为一个调制解调器接口。
- e) 与隔离开关相连的过压保护/短路保护电路(图 19 中未画出)。

注 1：连接到冗余介质的节点有两个线路单元。

注 2：若使用了这样的继电器，开关 K_b 、 $K_{1,1}$ 或 $K_{1,2}$ 可是同一机械继电器的触点。

4.5.1.3 主通道和辅助通道

主通道和辅助通道应都具有向线路单元发送和从线路单元接收 HDLC 帧和控制信号的能力。

注：辅助通道仅用于检测附加节点和接收其地址时发送和接收帧。相较于主通道，辅助通道的操作可简化。

4.5.1.4 方向转换器

方向转换器应使主通道与方向 1 相连且辅助通道与方向 2 相连，或相反连接。

注：方向转换器并不是必须的物理组成部分，其可由逻辑或软件方式实现。

4.5.2 节点和开关设置

4.5.2.1 概述

本条描述中间设定或末端设定的节点特性。

4.5.2.2 中间设定

中间设定时，节点应：

- a) 在两个总线段间建立连续性(K_b 闭合)；
- b) 切除端接器 $Z_{i,1}$ 和 $Z_{i,2}$ ($K_{i,1}$ 和 $K_{i,2}$ 断开)；
- c) 通过收发器 1 或收发器 2 将主通道与线路连接；
- d) 关闭辅助通道和未用的收发器。

注：非工作节点(禁止或掉电)和不位于总线末端的节点使用中间设定。

4.5.2.3 末端设定

末端设定时，节点应：

- a) 隔离两个总线段(K_b 断开)；
- b) 插入两个端接器($K_{i,1}$ 和 $K_{i,2}$ 闭合)；
- c) 使辅助通道与一个方向相连、主通道与另一个方向相连。

注：未命名的节点、在总线末端的节点(特例：没有从设备的主设备)或处于休眠模式的节点使用末端设定。

4.5.3 双线单元(可选)

在使用双线方案时，应遵循下列规则：

- 设计用于双线连接的 MAU 应通过独立的线路单元与 A 线和 B 线连接；
- 节点设置(末端设定或中间设定)应等同适用于两个线路单元；
- 应可在保持一条线路正常操作时移除另一条线路。

示例：冗余线路工作的 MAU 如图 20 所示。转换逻辑允许从 A 线或 B 线接收信号。

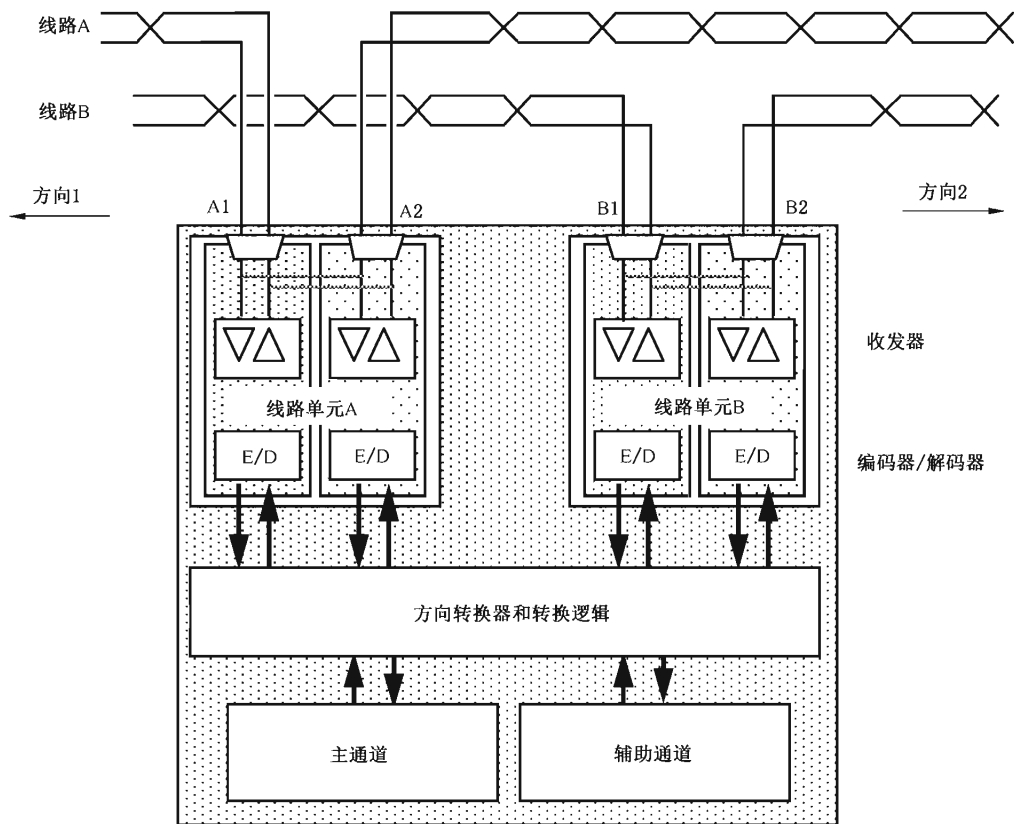


图 20 有冗余线路单元的节点

4.6 线路单元规范

4.6.1 概述

尽管规范中只提及线路单元 A,但若使用了冗余介质,则这些规范也适用于线路单元 B。

4.6.2 电气隔离

节点外壳与 A1X、A1Y、A2X、A2Y 中任何一点间的绝缘电压和绝缘电阻应超过 GB/T 25119 中规定的值。

注：这些值在目前的版本中为 0.50 kV(r.m.s)和 1.0 MΩ。

4.6.3 线路单元的插入损失

4.6.3.1 衰减测量

测量插入损失时,信号发生器(内阻为 Z_i)产生的正弦信号通过 20.0 m 长的电缆施加于点 A1X 和 A1Y,在另一根连接到 A2X 和 A2Y 的 20.0 m 长的电缆的末端用电压表(与值为 Z_i 的阻抗并联)测量。反之亦然。如图 21 所示。

衰减定义为以下两个差分电压的比值,用分贝(dB)表示:

- a) 第一个电压在节点被移除、电缆连接器耦合时设置为 $4.0 V_{pp}$;
- b) 第二个电压是节点(末端设定或中间设定,根据测试情况而定)插入时的测量值。

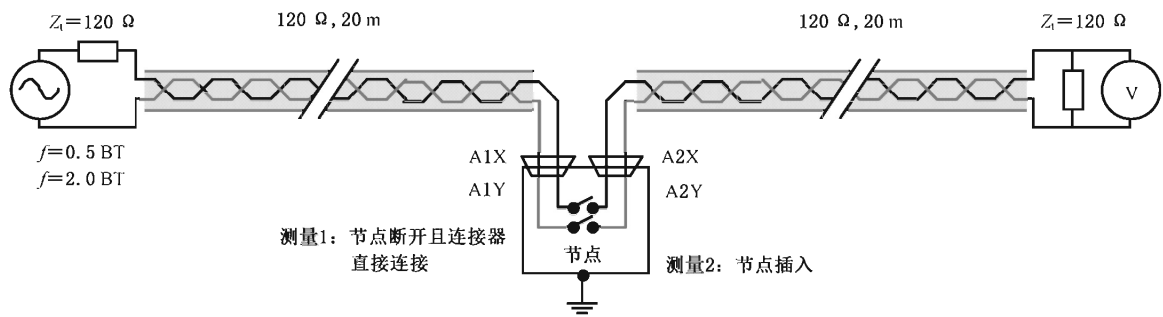


图 21 衰减测量

4.6.3.2 末端设定的节点

在末端设定(K_b 断开、 K_i1 和 K_i2 闭合)方式下,线路单元应提供一个 1.0 BR 的信号,施加于 A1X 和 A1Y 或 A2X 和 A2Y 之间,阻抗与 4.3.6 中规定的端接器相符。

在末端设定方式下,在 A1X 和 A1Y 之间施加信号且在 A2X 和 A2Y 之间测量,线路单元衰减信号应大于 55.0 dB,反之亦然。

4.6.3.3 中间设定的节点

在中间设定(K_b 闭合、 K_i1 和 K_i2 断开)方式下,线路单元在下列两种情况之一时:

- 接收器正常工作,发送器处于高阻状态;
- 接收器和发送器都没有供电。

应衰减正弦信号:

- 在 0.5 BR 和 1.0 BR 之间小于 0.3 dB;
- 在最大到 2.0 BR 时小于 0.4 dB。

节点应呈现一个至少 1 MΩ 的电阻,以承受加于 A1X 和 A1Y 或 A2X 和 A2Y 之间的正或负的正 48.0 V 直流电压。

4.6.4 开关规范

所有连接到总线的开关(K_b 、 K_i 等):

- a) 断开时,绝缘耐压应至少 500.0 V(r.m.s);
- b) 闭合时,初始接触电阻应小于 0.050 Ω;
- c) 闭合时,10⁷ 个周期后接触电阻应小于 0.100 Ω;
- d) 包括回跳时间在内,从一种设置切换到另一种设置的时间应小于 10.0 ms。

注:继电器可是固态的或机械式的。

4.6.5 线路单元屏蔽层的连接

在每一个节点,方向 1 和方向 2 上的电缆屏蔽层应通过插座连接在一起,其接触电阻小于 0.010 Ω。线路单元应提供下列两种方法连接屏蔽层和节点外壳:

- a) 如 4.3.5.2 规定,直接通过低阻抗连接(接地屏蔽);
- b) 如 4.3.5.3 规定,通过 RC 网络连接(浮动屏蔽)。

示例:节点屏蔽层连接原理,如图 22 所示。

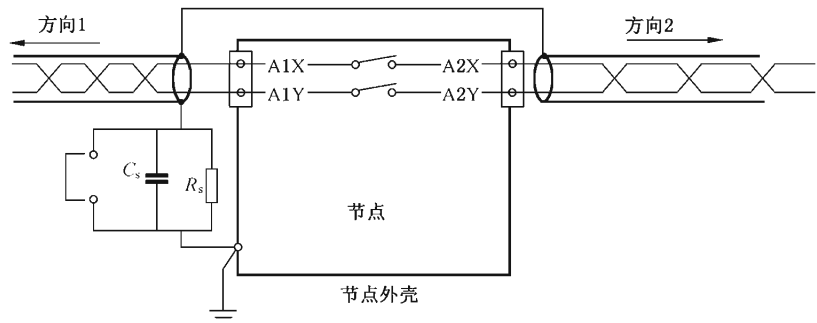


图 22 线路单元中的屏蔽层接地

4.6.6 加电清除(可选)

4.6.6.1 概述

为克服继电器和连接器中接触氧化,节点中可使用加电清除,即在两个方向的线 X 和 Y 之间施加一连续电压。

若使用了加电清除,则适用本条。

注 1: 未规定非冗余介质加电清除的用法。

注 2: 使用加电清除的节点和未使用加电清除的节点可在同一总线中混合使用。

4.6.6.2 加电清除电源和负载

支持加电清除的节点应为每一个方向提供一个加电清除电压源和电压负载。

在冗余物理介质时,节点应提供两个独立的加电清除电压源,一个方向一个;同时应为另一个节点的加电清除提供电压负载,如图 23 所示。

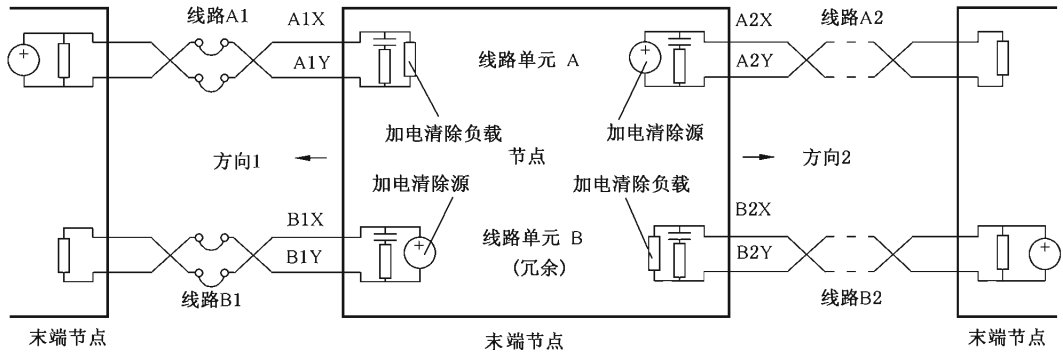


图 23 加电清除电源和负载

加电清除电源的正极应与 A2X 相连,冗余线上与 B1X 相连。

加电清除电源的负极应与 A2Y 相连,冗余线上与 B1Y 相连。

加电清除电源应提供一个 48.0 V(+20%, -10%) 的直流电压。该值是当相应线路及其插入的端接器连接到 4.3.6 中规定的端接器或开路时在连接点(A2X/A2Y 或 B1X/B1Y)测得的。

加电清除电源纹波在 0.5 BR~2.0 BR 范围内应低于 0.100 V_{pp}。

加电清除电源输出电流不应超过直流 80.0 mA。

加电清除电源应有遵循 GB/T 25119 的输入输出隔离。

加电清除电源应与线路去耦,例如通过 0.10 H 的电感或其他满足节点插入损耗的方式。

加电清除电源的开通时间常数应在 0.5 ms~5.0 ms 的范围内。

加电清除电源的关断时间常数应在 0.5 ms~5.0 ms 的范围内。

同一节点内两个加电清除电源之间的衰减在 0.5 BR~2.0 BR 范围内应大于 50.0 dB。

4.6.6.3 应用加电清除

末端节点应在其激活的辅助通道上接通加电清除电源(未命名节点两个方向的辅助通道都是激活的,休眠模式节点没有激活的辅助通道)。

注 1: 节点可脉冲式地施加加电清除电源,例如为减少消耗。

注 2: 只要满足了电磁干扰级别的要求,就可在主通道上接通加电清除电源。

4.7 收发器规范

4.7.1 约定

连接到冗余总线上的 MAU 有四个收发器,分别命名为 A1、A2、B1 和 B2。在非冗余布置中,只使用收发器 A1 和 A2。下列规范适用于任意一个收发器。

除非另行规定,否则应保持下列缺省测量条件:

- 收发器的特性在电缆节与节点连接处的 X 和 Y 之间测得。
- 所有被测量的电压均为 X 和 Y 之间的差分电压($U_x - U_y$)。
- 测量发送器时,接收器电路处于正常接收状态。测量接收器时,发送器电路处于高阻状态。
- 所有电阻值的精度为 $\pm 1\%$,所有电容值的精度为 $\pm 10\%$ 。

4.7.2 发送器

4.7.2.1 发送器负载

为定义发生器特性,规定了四种测试电路以模拟电缆和节点:

- 轻载测试电路模拟开路线路(如处于末端设定状态的节点)。其阻性负载值等于端接器阻值。
- 重载测试电路模拟满负载总线。阻性负载值等于端接器阻值的 0.42 倍。
- 闲置测试电路模拟没有阻性负载的 860.0 m 长电缆。每个电容值为 $1.3 \times (1 \pm 10\%) \text{ nF}$,每个电阻值为 $27.0 \times (1 \pm 1\%) \Omega$ 。
- 短路测试电路模拟一种线路故障。其只包含电流测量电路。

这些电路如图 24 所示。

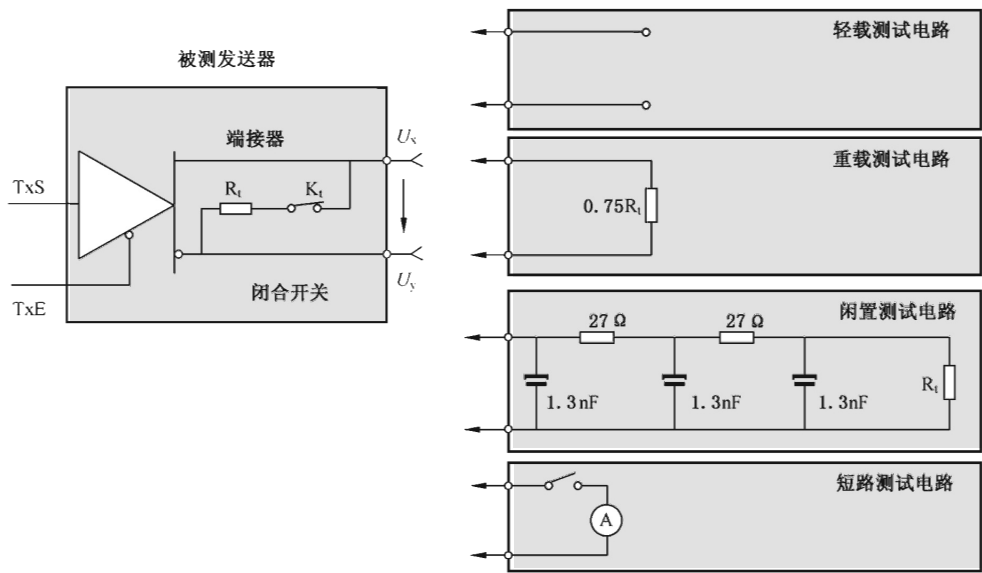


图 24 发送器测试电路

测试是节点在末端设定方式(K_b 断开, K_t 闭合)下进行的。
线路单元的端接器在测试电路规范中予以考虑。

4.7.2.2 发送器输出信号

注：由于使用了数据编码，发送器在前导码和终止分界符之间产生长度为 1 位(1.0 BT)或 0.5 位(0.5 BT)的脉冲。
以下规定对 0.5 BT 和 1.0 BT 的正负脉冲都适用。

发送器应是一个差分驱动器。

输出信号是节点连接点上的差分电压($U_x - U_y$)。

当与 4.7.2.1 中定义的重载测试电路或轻载测试电路连接时，发送器应遵循下列规定，如图 25 所示：

- a) 输出信号应正负交替。
- b) 在重载测试电路中，输出信号的幅值应至少为 $\pm 3.0\text{ V}$ ，在轻载测试电路中，输出信号的幅值应最大为 $\pm 7.0\text{ V}$ 。
- c) 峰值定义为输出信号的最大幅值。信号从峰值时刻到在下一个过零点前 $0.100\text{ }\mu\text{s}$ 的时间内幅值下降不应超过 20%。相对于平均电压下降，在这段时间内的幅值振铃不应超过峰值的 5%。
- d) 输出信号的压摆率在任何时间点应小于 0.20 V/ns ，在过零点前后的 100.0 ns 内应大于 0.03 V/ns 。
- e) 输出信号过冲，定义为最大幅值与稳态幅值的差除以稳态幅值，不应超过 10%。
- f) 输出信号的边沿畸变，定义为理想过零点与实际过零点的时间差，不应超过 1 BT 时间的 $\pm 2\%$ 。

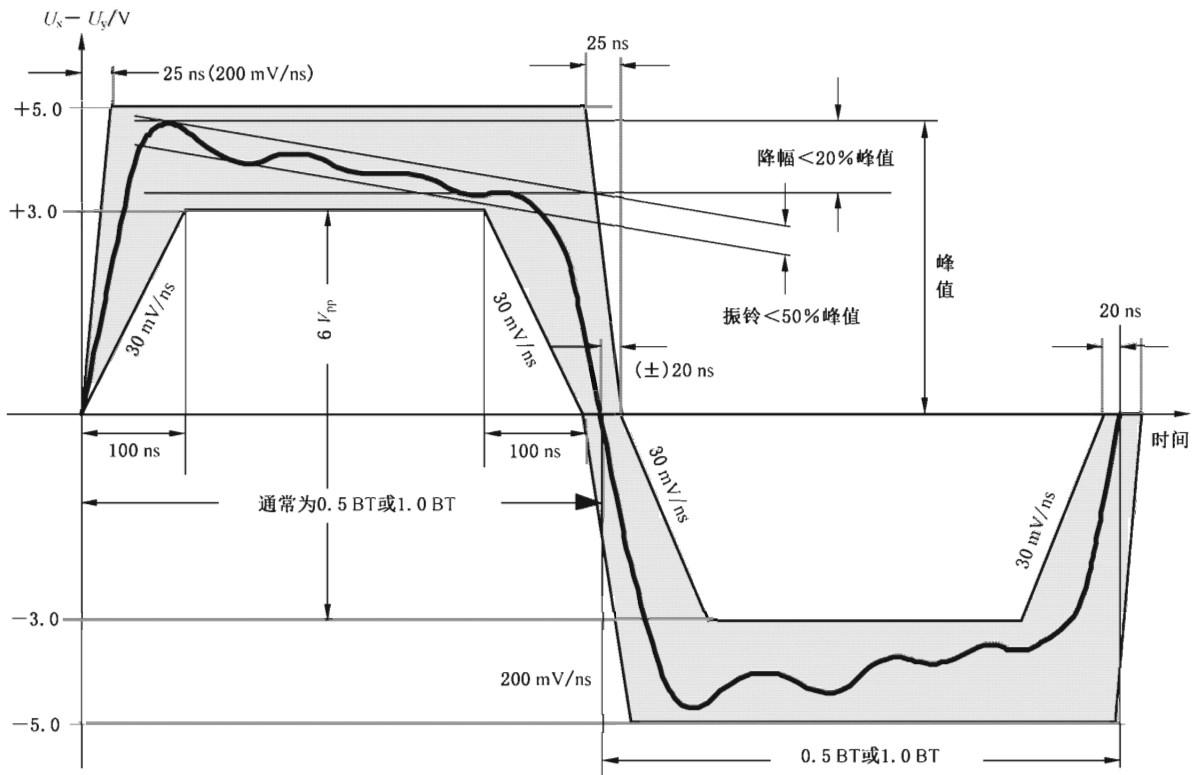


图 25 发送器脉冲波形

注：由于加电清除电容与变压器串联，因此会产生电压下降。

4.7.2.3 发送器噪声

在 1.0 kHz~4.0 BR 频率范围内,发送器在没有发送数据时所产生的任何噪声不应超过 5.0 mV(r.m.s)。

4.7.2.4 发送器帧尾

发送器产生的帧尾应在下列条件下测试：

- a) 发送器发送最长可能的帧；
- b) 帧数据位是一个由“1”和“0”符号组成的伪随机序列；
- c) 帧以 4.8.1.4 中规定的终止分界符结束；
- d) 发送器驱动 4.7.2.1 的闲置测试电路；
- e) 在发送器禁止前平均差分幅值大于 4.5 V。

在这些条件下,输出信号应有以下限制,如图 26 所示：

- a) 在上一次由负到正跳变 100.0 ns 后,输出信号应在 2.0 BT±0.100 μs 的时间内维持在 3.000 V 以上；
- b) 在上一次由负到正跳变后的 3.0 BT 内,输出信号应下降到 1.100 V 以下；
- c) 从输出信号第一次到达 1.100 V 开始的 20.00 μs,输出信号幅值不应超过 0.100 V；
- d) 从输出信号第一次到达 1.100 V 开始的 64.0 μs,输出信号幅值不应超过 0.025 V。

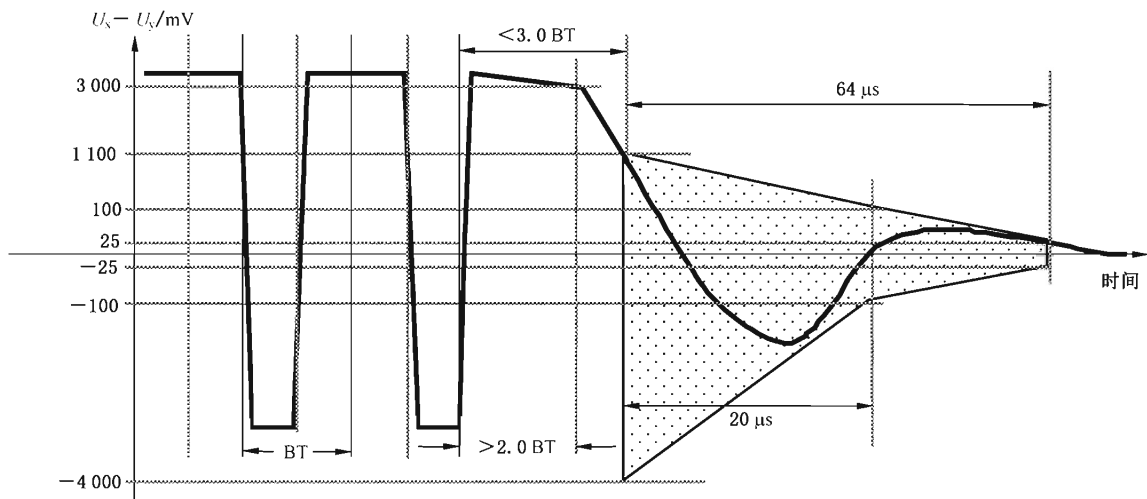


图 26 发送器信号和闲置

注：在发送器闲置后，线路振铃可通过平衡每一位元的信号来减到最小。通过平衡 4.8.1.4 中规定的终止分界符，可进一步减小振铃。

4.7.2.5 发送器容错

发送器无论是否使能，都应容许在连接点处应用短路测试电路（见 4.7.2.1）直到达到热稳定，且应在移除短路测试电路后恢复正常运行。

短路电路电流不应超过 1.0 A。

注：对于一致性测试，在 1 h 后达到热稳定。

4.7.2.6 发送器防闲聊

若发送时间超过防闲聊时间（ T_{jabber} ），每个发送器应有一独立的电路使其与总线隔开。 T_{jabber} 值等于最长可能帧（包括前导码、终止分界符、位填充）的时间 + 20%。

4.7.3 接收器规范

4.7.3.1 接收器信号特性

接收器宜解码如图 27 所示形状的信号，该信号施加于连接点处：

- 当接收信号幅值小于 0.100 V 时，接收信号的斜率大于 2.0 mV/ns；
- 当接收信号在从前一个过零点 100.0 ns 开始、持续至少 $(0.5 BT - 350.0 \text{ ns})$ 或 $(1.0 BT - 350.0 \text{ ns})$ 的时间段内保持在 0.300 V 以上时，其峰值在 0.330 V ~ 5.00 V 之间变化。

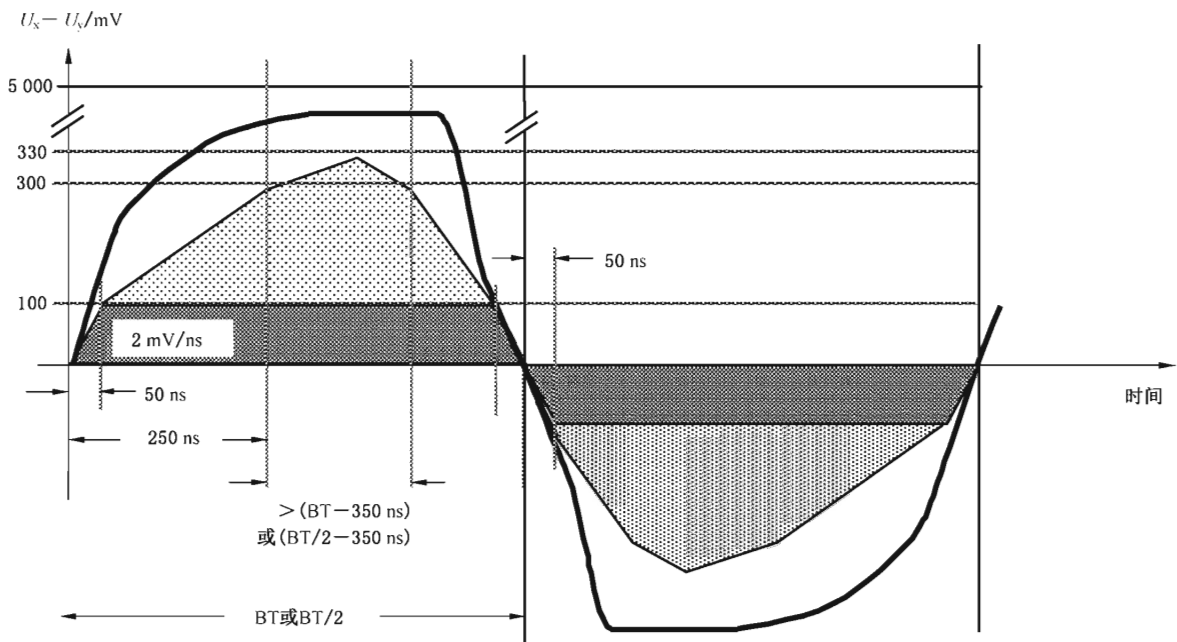


图 27 接收器信号包络线

若帧丢失或无效帧(见 4.8.1.5.5)、错误的帧长度或错误的帧数据位(FCS 错误),则检测到帧错误。

4.7.3.2 接收器极性

TxS 信号的高电平应对应于正的差分电压($U_x - U_y$),此电压也应与接收器 RxS 信号的高电平相对应。

TxS 信号的低电平应对应于负的差分电压($U_x - U_y$),此电压也应与接收器 RxS 信号的低电平相对应。

线路处于空闲状态时,RxS 的状态未定义。

4.7.3.3 接收器灵敏度

当 4.7.3.1 定义的接收信号的幅值在其最小值到最大值范围内变化时,接收器在接收到的 $3 \times 3 \times 10^6$ 帧(帧长度为 64 个随机数据位、速率为 1 000 fps)中检测到的帧错误不应大于 3。

注:根据图 27,接收器的工作电压范围在 0.330 V~5.000 V 之间,约 23.6 dB。考虑到 4.3.3.4 规定的介质衰减小于 20.0 dB,因此噪声裕量约为 4.0 dB。

4.7.3.4 接收器不灵敏度

当接收信号(见 4.7.3.1)小于 0.100 V 时,接收器不应解码有效帧(见 4.8.1.5.4)。

4.7.3.5 接收器边沿畸变

当测试信号边沿的过零点在预期过零点 ± 0.1 BT 时间内时(如图 28 所示),接收器在接收到的 $3 \times 3 \times 10^6$ 个帧(帧长度为 64 个随机数据位、速率为 1 000 fps)内检测到的帧错误不应大于 3。

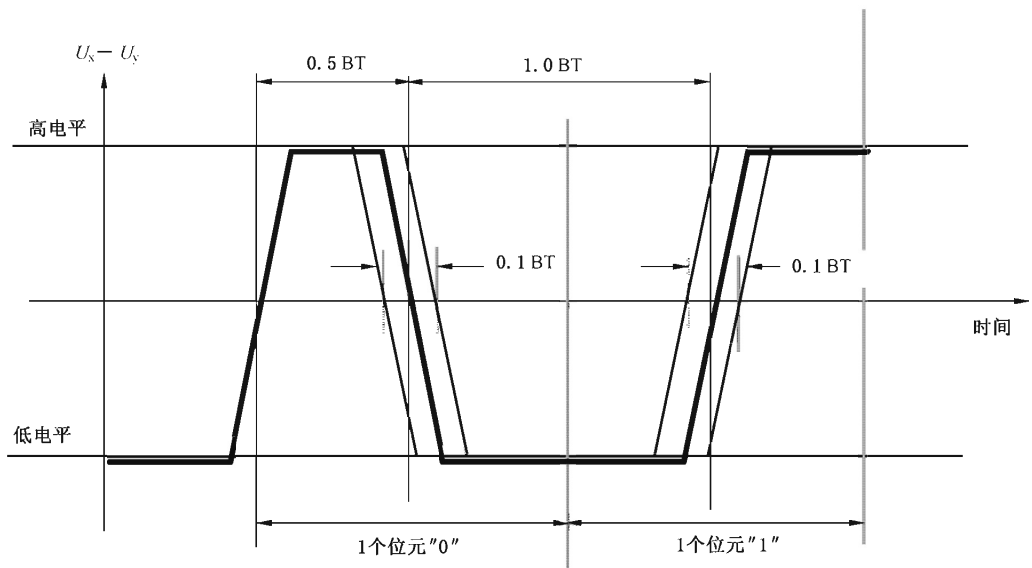


图 28 接收器边沿畸变

4.7.3.6 接收器噪声抑制

当工作在下列两种情况时,接收器在接收到的 3×10^6 帧[帧长度为 64 个随机数据位、速率为 1 000 fps、信号幅值为 0.700 V(1.400 V_{pp})]内检测到的帧错误不应大于 3:

- 在外壳和两条数据线间存在频率范围为 65.0 Hz~1.5 MHz、幅值为 4.000 V(r.m.s)的共模正弦信号;
- 在 X 和 Y 之间存在频率范围为 1.0 kHz~4.0 MHz、幅值为 0.140 V(r.m.s)的加性准高斯白噪声。

4.8 由介质决定的信号表示

4.8.1 帧的编码与解码

4.8.1.1 约定

编码和解码假定发送或接收的信号为二进制,无偏置电平。在线路空闲时,接收的电平信号未定义。

RxS 代表从线路接收到的理想(模拟)信号。

如图 29 所示,在进入空闲状态前,帧应作为以前导码开始、终止分界符结束的正、负电平序列传送。

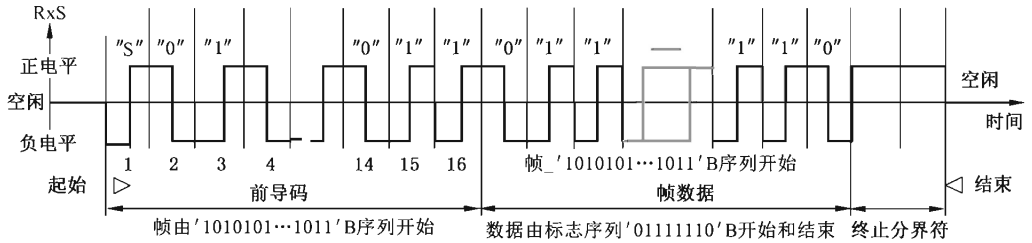
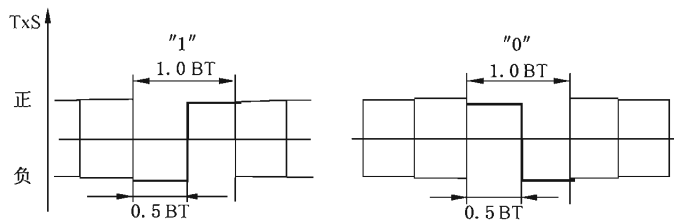


图 29 线路上理想的帧(16 位前导码)

4.8.1.2 位编码

如图 30 所示,前导码和帧数据位应采用曼彻斯特码编码:
——位元前半部分为负电平、位元中间跳变为正电平的位应编码为“1”;
——位元前半部分为正电平、位元中间跳变为负电平的位应编码为“0”。



说明:
1.0 BT ——1 个位时间;
0.5 BT ——1/2 个位时间。

图 30 位编码

在帧的终止分界符到达之前,帧的边沿距离应是一个位时间(“0”接“1”或“1”接“0”)或半个位时间(连“0”序列或连“1”序列)。

4.8.1.3 前导码编码

帧应以前导码开始。前导码以起始位 S(发送作“1”位)开始,随后是若干个(“0”, “1”)位对,最后以“1”位结尾,如图 31 所示。

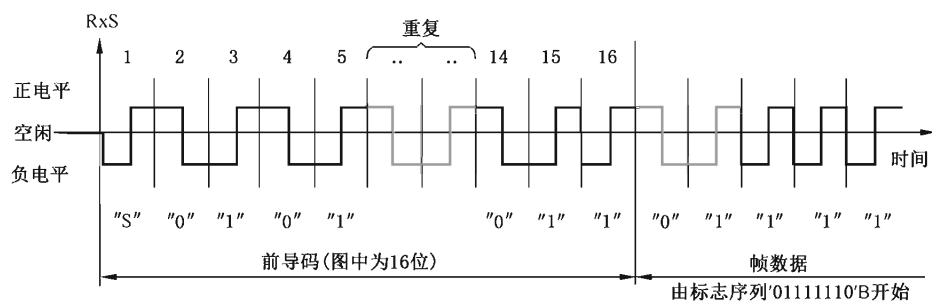


图 31 前导码

在前导码起始位和最后位“1”之间,应有最少 7 个最多 15 个(“0”“1”)位对。
解码器可通过解码前导码检查 RxS 的极性,但若 X 和 Y 接反,不应自动反向信号。
注: 以下章条中只考虑 16 位前导码。

4.8.1.4 终止分界符

帧应由终止分界符结束。终止分界符使线路维持 2.0 BT 宽的正电平。
为补偿失衡,在正电平后可附加一个 2.0 BT 宽的负电平,如图 32 所示。

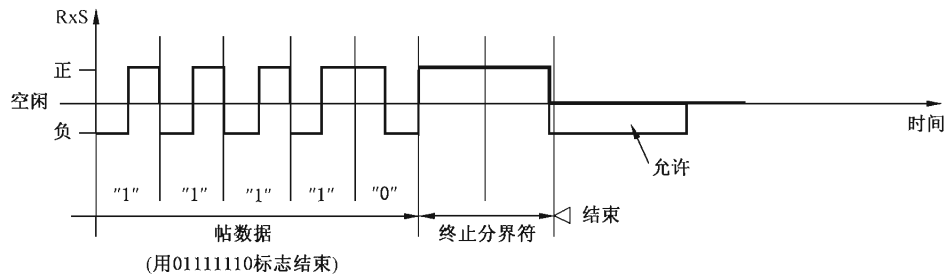


图 32 终止分界符

注 1: 没有负电平补偿的终止分界符将使线路失衡,引起线路振铃(见 4.7.2.4)。宜使用补偿脉冲,但不强制,以允许使用不产生补偿脉冲的商业电路。

注 2: 由于使用 HDLC 标志(见 5.2.1),帧最后一位为“0”。

4.8.1.5 信号品质监视

4.8.1.5.1 概述

下列规范假定解码器产生两个信号:载波检测(CS)信号和信号品质错误(SQE)信号。这两个信号用于信号品质监视和冗余切换。

4.8.1.5.2 载波检测信号

在检测到按 4.8.1.3 规定的前导码的最后一位后的 0.5 BT 内,解码器应使 CS 信号有效。

在检测到终止分界符或检测到既非“0”非“1”也非终止分界符的位后的 0.5 BT 内,解码器应使 CS 信号无效。

4.8.1.5.3 信号品质错误信号

在检测到按 4.8.1.3 规定的前导码的最后一位后的 0.5 BT 内,解码器应使 SQE 信号无效。

当 CS 信号有效时,在检测到既非“0”非“1”也非终止分界符的位后的 0.5 BT 内,解码器应使 SQE 信号有效。

4.8.1.5.4 有效帧

有效帧应定义为一个包含了前导码、一定数目的“0”和“1”位以及终止分界符的帧。

示例:一个有效帧及相应的 CS 和 SQE 信号,如图 33 所示。

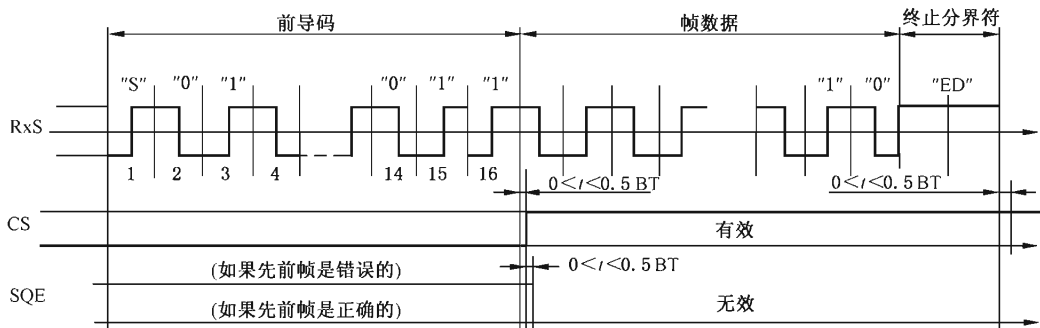


图 33 有效帧、RxS、CS 和 SQE 信号

为了冗余控制的目的,一个有效帧在前导码后应至少有 8 个数据位。

4.8.1.5.5 无效帧

当 CS 信号有效时,若 SQE 信号有效的时间超过 0.5 BT,则此帧应定义为无效帧。

示例:接收到错误帧的信号时序,如图 34 所示。

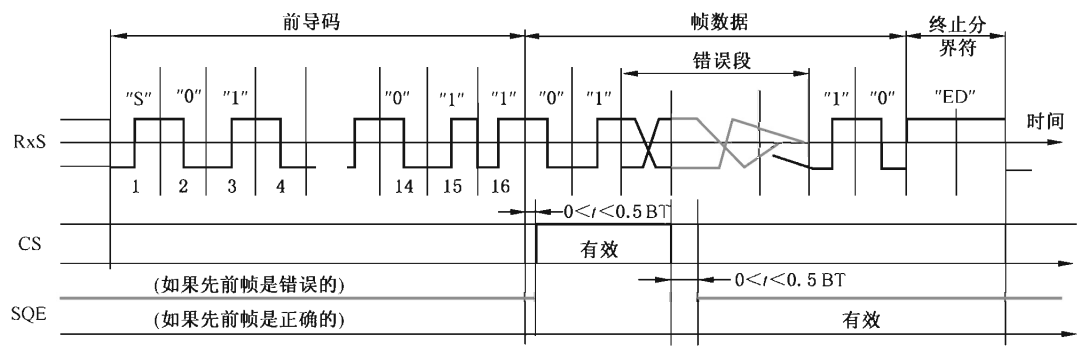


图 34 错误帧、RxS、CS、SQE 信号

若 SQE 信号有效,解码器应忽略所有数据直到接收到下一个前导码。

4.8.2 双线处理(可选)

4.8.2.1 概述

本条定义了一种可选的冗余机制。若使用该选项,则下列规范对方向 1 和方向 2 以及 A 线和 B 线都适用。

4.8.2.2 原理

节点在 A 线和 B 线上同时发送相同的数据,节点只从其中一条线路接收数据,同时监视另一条线路。接收数据的线路称作信任线,被监视的线路称作监视线。

每个节点基于其物理层产生的信号或链路层请求,选择其信任线和监视线,与其他节点无关。

为保持介质无关性,信任线的选择依赖于线路单元产生的信号,因为这些信号在线路单元接口中定义。

4.8.2.3 时滞

由于信号在 A 线和 B 线上的延时不同,因此信号时滞(时间差)在发送器、接收器或线路上其他地方也不同,如图 35 所示。

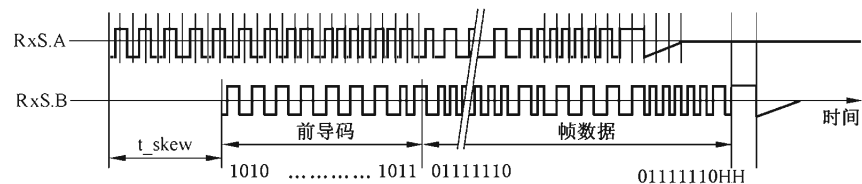


图 35 冗余线路(接收器)

4.8.2.4 冗余发送

带有冗余线路单元的 MAU 应在 A 线和 B 线(A1 线和 B1 线,或 A2 线和 B2 线)上发送相同的信号。

在线路单元的输出端测得的同一方向 A 线和 B 线间信号时滞不应超过 $T_{\text{skew}_t}=1.0\ \mu\text{s}$ 。

4.8.2.5 冗余接收

4.8.2.5.1 接收时滞

接收器允许的最大接收时滞应为 $T_{\text{skew}_r}=32.0\ \mu\text{s}$ 。

4.8.2.5.2 线路受扰

每个连接到节点的总线节应有一个“线路受扰(Line_Disturbance)”信号。A 线的线路受扰信号叫 DA1 和 DA2,B 线的线路受扰信号叫 DB1 和 DB2。

线路受扰信号应有效,若:

- 解码器使能了该线路的 SQE 信号;
- 在冗余线路的线路单元使其 CS 信号之后的 T_{skew_r} 时间内,解码器没有产生 CS 信号(帧丢失)。

在以下情况线路受扰信号应无效:

- 当解码器接收到按 4.8.1.5.4 定义的有效帧。

在非冗余模式中,未使用的线路应被认为受到永久干扰。

示例:如图 36 所示,A1 线上的所有帧都是有效帧,B1 线是受扰的。

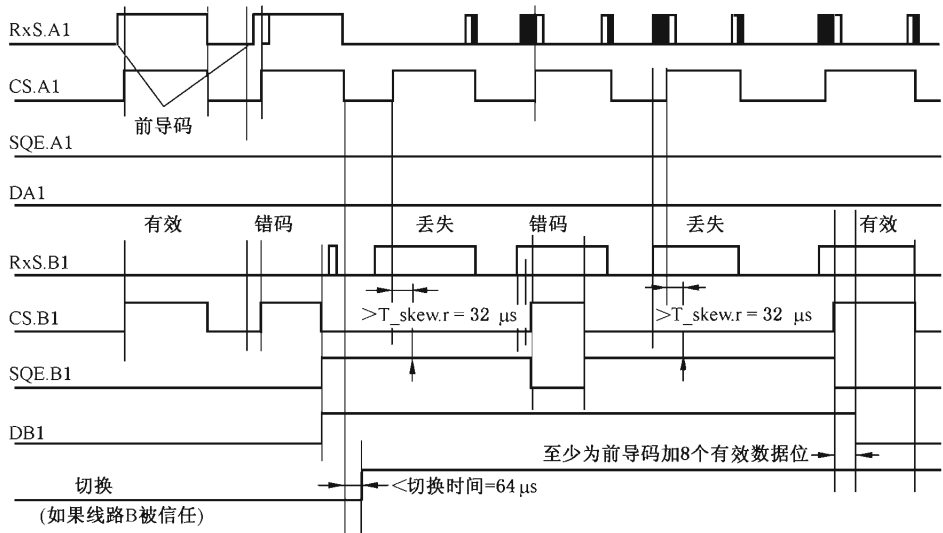


图 36 线路受扰信号

4.8.2.6 线路切换

(末端)节点的方向 1 和方向 2 可信任不同的线路(例如 A1 线和 B2 线)。

切换单元应能在 $T_{\text{switchover}}=64.0\ \mu\text{s}$ 的时间内交换信任线和监视线,若:

- a) 信任线的线路受扰信号有效而监视线的线路受扰信号无效；
- b) 链路层请求切换(如帧长度、FCS 或协议发生差错)。

4.8.2.7 MAU 报告

MAU 应向链路管理层报告：

- a) 通过哪条线路接收到了帧数据；
- b) 每条线路每个方向上线路受扰信号的每次生效；
- c) 用于节点报告(见 5.5.2.3)的四个线路受扰信号(DA1、DA2、DB1、DB2)的状态。

注：若一条线路完全失效(或未连接)，在失效发生后线路受扰信号将触发一次。但是，若线路被中断，根据中断所在位置，线路受扰信号在每一帧的后面都能触发。

4.8.3 线路单元接口

线路单元接口定义了进入和离开线路单元的信号。

这个接口可保留在节点内部，但建议该接口可用于测试。该接口在一致性测试时未包括。

下列规范使接口更容易，且定义了在本部分其他章条中用到的信号。

若开放，线路单元接口应包含 GB/T 3454 中分别为每一个收发器定义的调制解调信号以及附加的控制信号，如表 7 所示。

表 7 线路单元接口信号

名称	表示	GB/T 3454 的章条号	方向	含义
GND	信号地	102	—	公共返回
TxD	发送数据	103	至线路单元	帧数据，无前导码或终止分界符，作为 NRZ 信号提供，由 TxC 提供时钟
RxD	接收数据	104	来自线路单元	NRZ 序列，不包括帧前导码和终止分界符，由 RxC 提供时钟
RTS	请求发送	105	至线路单元	命令发送前导码；清除此信号命令产生终止分界符
CTS	清除发送	106	来自线路单元	通告已发送前导码，随后是请求的数据
TxC	发送器时钟	114	来自线路单元	收发器产生，给要发送的 TxD 数据提供时钟
RxC	接收器时钟	115	来自线路单元	解码器产生，给已接收的 RxD 数据提供时钟
SQE	信号品质错误	不属于 GB/T 3454	来自线路单元	如 4.8.1.5.3 描述
CS	载波检测	不属于 GB/T 3454	来自线路单元	如 4.8.1.5.2 描述
Kx	开关控制信号	不属于 GB/T 3454	至线路单元	为两条线路定义至少两个基本开关设置：末端设定和中间设定。 另外，开关控制信号还可将收发器与线路隔离或连接

5 链路层控制

5.1 编址

链路层应使用 8 位标识符作为源设备地址和目的设备地址。

节点的主通道应以范围在 1(“00000001”B)~63(“00111111”B)之间的设备地址寻址,该地址由初运行过程分配。

主设备的主通道应接收主设备地址 1(“00000001”B)。

设备地址 0(“00000000”B)应是自身固有地址,不应发送。

设备地址 64~126 和 128~254 保留给将来使用。

地址 255(“11111111”B)应是所有节点侦听的广播地址。

未命名节点应在其两个通道上响应地址 127(“01111111”B)。

节点的辅助通道应响应未命名地址。

节点地址是一个已命名从设备地址或主设备地址。

5.2 帧和报文

5.2.1 帧数据格式

帧数据格式应遵循 GB/T 7421 中定义的 HDLC 格式,如图 37 所示。

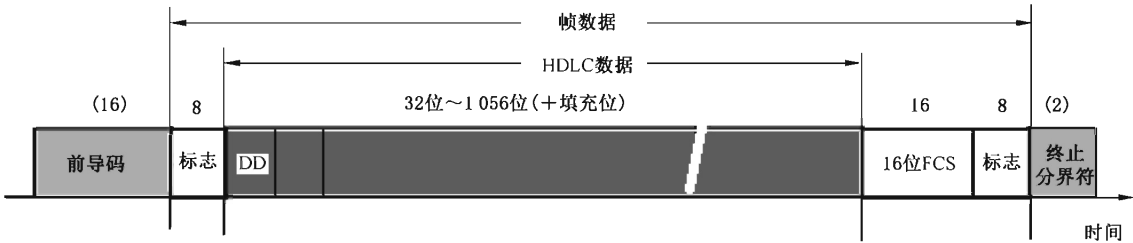


图 37 HDLC 帧结构

如 GB/T 7421 规定,帧数据应以标志“01111110”开始。

开始标志后应接以 HDLC 数据。HDLC 数据最少 32 位,最多 1 056 位(不计算 GB/T 7421 中的填充位)。

HDLC 数据应为八位位组的整数倍(GB/T 7421 的一个限制)。

HDLC 数据的第一个八位位组应为 5.1 规定的 8 位目的设备地址。

HDLC 数据的第二个八位位组不应解释为 GB/T 7421 的控制字段。

HDLC 数据后应接以错误检测代码,此代码应为 GB/T 7421 规定的 16 位帧校验序列(FCS)。

帧应由与开始标志相同的结束标志结束。

结束标志不能用作下一帧的开始标志。

发送器不应发送 GB/T 7421 的“空闲”或“取消”序列。

帧是八位位组组成的序列。按 GB/T 7421 规定,每个八位位组的最低有效位应首先发送。

注 1: 位顺序约定保证了在八位位组内位发送顺序的唯一性。作为 GB/T 7421 的一个例外,FCS 的两个八位位组先发送最高有效位。

注 2: HDLC 的位填充防止了标志序列出现在两标志之间的数据中;发送器在每 5 个连续“1”的数据后插入一个“0”,接收器去除每 5 个“1”后的“0”。填充位对链路层是不可见的,但可使帧长度最多增加 20%。

5.2.2 报文定时

5.2.2.1 约定

总线段应由一个称为主设备的节点控制,该节点按自身步调在总线上发送数据。其他节点是从节点,只有在被主设备请求时才可发送数据。

末端节点被认为是辅助通道的主设备。

节点内主设备功能和从设备功能是不同的。

总线通信应由称为报文的帧对组成。帧对由主设备发送的主帧和在规定时间内从设备响应的从帧组成。

帧间隔时间应为测量到的从终止分界符的最后一次跳变起到下一帧前导码起始位的中间跳变止的时间段。

示例:报文定时,如图 38 所示。

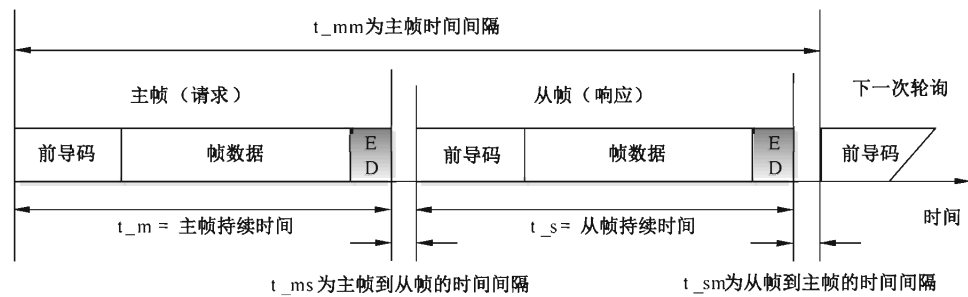


图 38 报文定时

5.2.2.2 应答延时的计算

对于给定总线,应答延时(T_{reply})是在主设备处测得的主帧结束与响应该主帧的从帧开始之间的最大延时。

应答延时由传输延时、解码和访问延时组成。

T_{reply} 是一个配置参数,其告知主设备,若没有收到从帧,在发送下一个主帧之前应等待多长时间。

示例:有 17 个节点(16 个总线节)的组成,主设备在总线的一个末端,如图 39 所示。

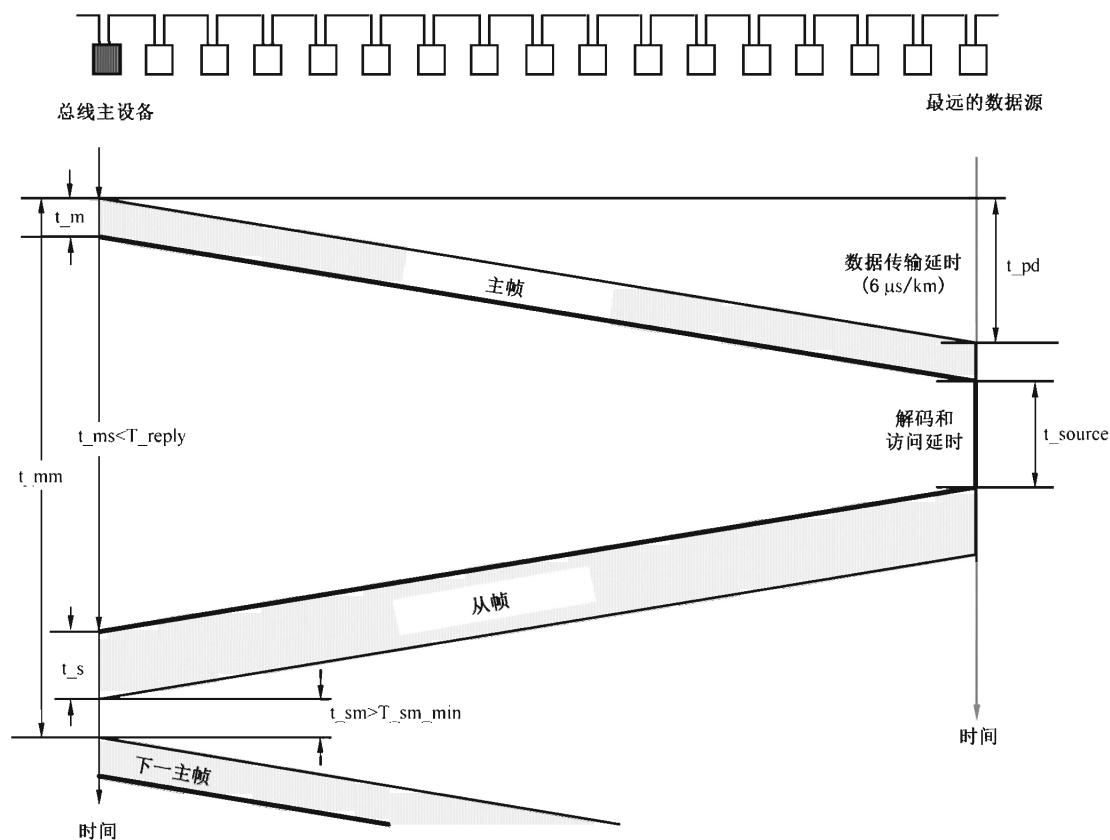


图 39 帧间间隔示例

对于给定应用,最差情况下应答延时 T_{reply} 计算见式(2):

$$T_{reply} = 2 \times T_{pd} + T_{source_max} \dots\dots\dots (2)$$

式中:

T_{reply} ——最差情况下应答延时,单位为微秒(μs);

T_{source_max} ——在源设备上主帧解码和应答的时间,单位为微秒(μs)(见 5.2.2.4.2);

T_{pd} ——在给定应用中末端节点之间的帧在最差情况下的传输延时,单位为微秒(μs)(见 4.3.3)。

帧超时计算见表 8。

表 8 帧超时计算

位时间	主通道	辅助通道
应用数据位	1 024 位	16 位
帧头	32 位	32 位
CRC	16 位	16 位
合计位数	1 072 位	64 位
位填充(最差情况 $\times 1.2$)	1 286 位	77 位
2 \times 标志位	16 位	16 位

表 8 (续)

位时间	主通道	辅助通道
终止分界符	2 位	2 位
前导码最大长度	32 位	32 位
合计位数	1 336 位	127 位
1.0 Mbit/s 时传输时间	1 336.0 μs	127.0 μs
$2 \times$ 传输延时	120.0 μs	120.0 μs
从节点响应时间	300.0 μs	800.0 μs
合计延时	1 756.0 μs	1 047.0 μs

5.2.2.3 冲突

当多个发送器同时激活时冲突发生。这种情况通常仅发生在同时发送的不同总线段末端节点之间。无法区分冲突与寂静或无效帧。

5.2.2.4 发送帧间隔

5.2.2.4.1 主设备侧

在主通道,主设备应期望在主帧发送结束后的 $T_{\text{main_max}}=1.756\text{ ms}$ 时间内,完整地接收响应该主帧的从帧,且在接收完从帧的 $T_{\text{sm_min}}=0.064\text{ ms}$ 后或超时($1.756\text{ ms}+0.064\text{ ms}=1.820\text{ ms}$)结束后,可开始发送下一主帧,如图 40 所示。

在辅助通道,末端节点应期望在 $T_{\text{aux_max}}=1.047\text{ ms}$ 时间内,完整地接收到响应其主帧的从帧,且在接收完检测响应帧的 $T_{\text{sm_min}}=0.064\text{ ms}$ 后或超时($1.047\text{ ms}+0.064\text{ ms}=1.111\text{ ms}$)结束后,可开始发送下一主帧。

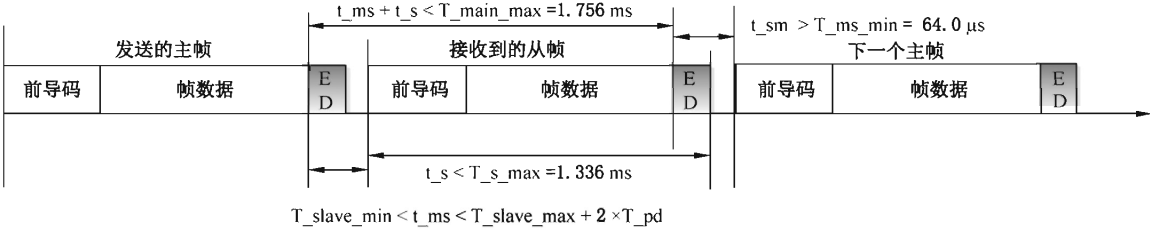


图 40 在主设备侧测量的帧间隔

5.2.2.4.2 从设备侧

- 被寻址的从设备应按图 41 所示开始发送帧:
- 不早于接收到主帧结束后的 $T_{\text{source_min}}=64.0\text{ }\mu\text{s}$;
 - 不迟于接收到主帧结束后的 $T_{\text{source_max}}=0.300\text{ ms}$ 。但对于检测响应帧,该时间在初运行期间增加到 0.600 ms ,在常规运行期间增加到 0.800 ms 。

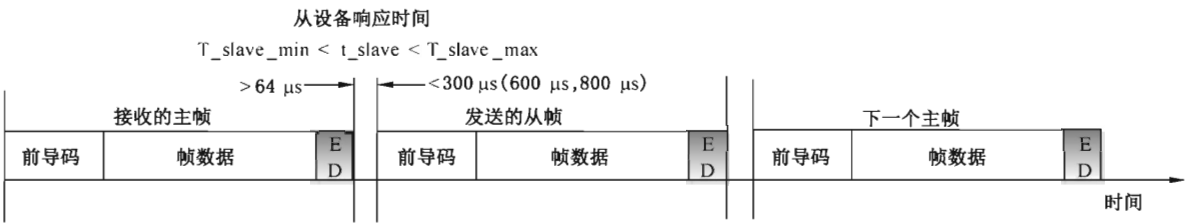


图 41 在从设备侧测量的帧间隔

5.2.3 HDLC 帧组成

HDLC 数据由开始标志和校验序列之间的数据组成,不包括填充位,如图 42 所示。

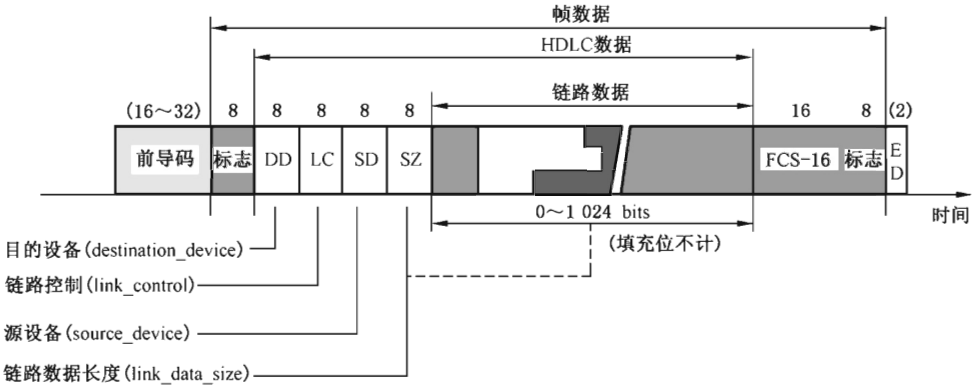
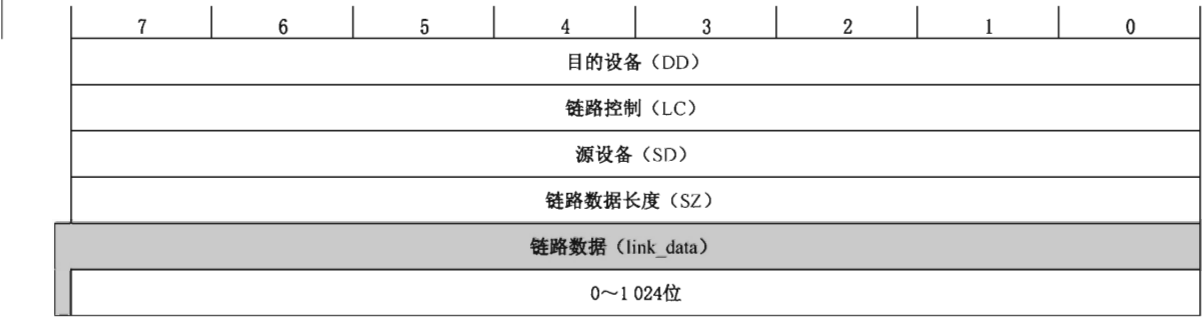


图 42 HDLC 数据格式

HDLC 数据应由下列字段组成,如图 43 所示。

HDLC_Data ::= RECORD

```
{
    destination_device UNSIGNED8  ——8 位的节点地址或广播地址,在响应帧中,缺省值为主设备地址
    link_control          Link_Control  ——8 位的链路控制
    source_device         UNSIGNED8  ——8 位的发送帧的源节点地址,在请求帧中,缺省值为主设备地址
    link_data_size        UNSIGNED8  ——8 字节节的链路数据长度,以八位位组的整数倍表示,若链路数据
                                   字段为空,则它将为 0
    link_data             Link_Data   ——WORD8 类型的数组 ARRAY[link_data_size]。在 0~1 024 数据
                                   位之间,见 5.3
}
```



注：图中使用 DD、LC、SD、SZ 以节省空间。

图 43 HDLC 数据格式

5.2.4 链路控制字段

链路控制字段区分：

- a) 请求帧(主帧)；
- b) 响应帧(从帧)。

链路控制字段区分三种类型的报文：

- a) 过程数据报文：用于更新分布式过程数据集；
- b) 消息数据报文：用于消息传送；
- c) 监视数据报文：用于总线监视和初运行。

链路控制字段允许被轮询的节点通过以下 4 位指示偶发性传送请求、状态改变或初运行条件：

- a) “A 位”(Attention)：请求发送消息数据；
- b) “C 位”(Change)：节点状态改变；
- c) “I 位”(Inhibit)：禁止初运行；
- d) “RI 位”(Remote Inhibit)：远端禁止初运行。

链路控制字段应编码成如表 9 规定的 8 位数据。

表 9 链路控制字段编码

		编码							
帧类型		7	6	5	4	3	2	1	0
过程数据和消息数据	过程数据请求/ 过程数据响应	M	0	A	C	I	0	0	0
	消息数据请求/ 消息数据响应	M	0	A	C	0	1	1	1
监视数据	检测请求/ 检测响应	M	1	0	0	0	0	0	0
	状态请求/ 状态响应	M	1	0	RI	0	0	0	1
	中间设定请求/ 中间设定响应	M	1	0	0	0	0	1	0
	末端设定请求/ 末端设定响应	M	1	0	0	0	0	1	1
	消名请求	M	1	0	0	0	1	0	0
	命名请求/ 命名响应	M	1	0	0	0	1	0	1
	拓扑请求/ 拓扑响应	M	1	0	0	0	1	1	0
	存在请求/ 存在响应	M	1	0	RI	I	1	1	1

在表 9 中未规定的位组合保留，且应忽略。

链路控制类型文本格式规定如下：

```

Link_Control ::= RECORD
{
    mq          ENUM1          ——最高位
    {
        SR      (0),          ——“0”：从设备响应
        MQ      (1)          ——“1”：主设备请求
    },
    sup          ENUM1
    {
        PM      (0),          ——过程数据或消息数据
        SP      (1)          ——监视数据
    },
    ONE_OF [sup]
    {
        [PM]      RECORD
        {
            a_bit    BOOLEAN1,  ——在过程数据响应帧或消息数据响应帧中若“A位”置位则为“1”
            c_bit    BOOLEAN1,  ——在过程数据响应帧或消息数据响应帧中若“C位”置位则为“1”
            i_bit    BOOLEAN1,  ——在过程数据请求帧或过程数据响应帧中若“I位”置位则为“1”
            pom      ENUM3
            {
                PROCESS_DATA    (0),  ——过程数据请求帧或过程数据响应帧
                MESSAGE_DATA    (7)  ——消息数据请求帧或消息数据响应帧
            }
        },
        [SP]      RECORD          ——监视数据请求帧或监视数据响应帧
        {
            res0      WORD1 (0),  ——保留位, = 0
            rem_inh    BOOLEAN1,  ——在状态响应帧或存在响应帧中若“RI位”置位则为“1”
            i_bit      BOOLEAN1,  ——在存在请求帧或存在响应帧中若“I位”置位则为“1”
            supervisory_type ENUM3  ——区分监视数据
            {
                DETECT    (0),  ——检测请求帧/检测响应帧
                STATUS    (1),  ——状态请求帧/状态响应帧
                SETINT     (2),  ——中间设置请求帧/中间设置响应帧
                SETEND     (3),  ——末端设置请求帧/末端设置响应帧
                UNNAME     (4),  ——消名请求帧
                NAMING     (5),  ——命名请求帧/命名响应帧
                TOPOGRAPHY(6),  ——拓扑请求帧/拓扑响应帧
                PRESENCE   (7)  ——存在请求帧/存在响应帧
            }
        }
    }
}

```

5.2.5 “A位”“C位”和“I位”的处理

过程数据响应帧和消息数据响应帧应置下列位为“1”，以指示异步事件：

- a) “A 位”应置位,只要消息数据的发送队列有待发送的帧。
- b) “C 位”应置位以指示节点状态改变,且当节点接收到状态请求帧时应复位。
- c) “I 位”应用下列方法置位,以禁止初运行:
 - 只要节点上的应用禁止初运行,节点应在所有过程数据和监视数据中置位“I 位”(消息数据响应不应置位“I 位”)。
 - 主设备应将从其命名的每一个节点接收到的过程数据响应帧中“I 位”进行“或”运算,并拷贝到其发送的所有存在请求帧中的“I 位”。
 - 末端节点应将在存在请求帧中接收到的“I 位”拷贝到其检测请求帧、检测响应帧和存在响应帧中的“I 位”。
- d) “RI 位”应用下列方法置位,以禁止初运行,末端节点应将从远端组成的检测响应帧中读到的“I 位”插入到其存在响应帧和状态响应帧中的“RI 位”。

5.2.6 帧长度、帧校验序列和协议错误

下列条款仅适用于通过自身地址或广播地址接收到的帧。

接收器应忽略帧,并认为接收此帧的线路受扰,若:

- a) 帧校验序列错误;
- b) 帧长度与在链路数据长度字段中指示的长度不匹配;
- c) 从帧类型与之前主帧类型(过程数据帧、消息数据帧、监视数据帧)不同。

若该帧不是检测请求帧、检测响应帧或命名请求帧,辅助通道应忽略帧并报告协议错误。

若该帧是检测请求帧或检测响应帧,主通道应忽略帧。

主设备应忽略响应主帧的第二个从帧(没有干扰的冲突)。

5.3 报文格式和协议

5.3.1 链路数据字段

链路层控制区分过程数据、消息数据和监视数据。

考虑到寻址限制,HDLC 数据的定义包括链路报头:

```

HDLC_Data ::= RECORD
{
    ONE_OF[link_control.sup]           —— 依赖于链路控制
    {
        [PM]ONE_OF[link_control.pom]   —— 过程数据或消息数据
        {
            [PROCESS_DATA]
            ONE_OF[link_control.mq]     —— 请求或响应
            {
                [MQ]Process_Data_Request,
                [SR]Process_Data_Response
            }
            [MESSAGE_DATA]
            ONE_OF[link_control.mq]     —— 请求或响应
            {
                [MQ]Message_Data_Request,
                [SR]Message_Data_Response
            }
        }
    }
}
  
```

},
[SP]Supervisory Data — 监视数据
}
}

链路数据中的前 4 个八位位组形成链路报头,且在所有帧中具有相同的格式和含义。

5.3.2 过程数据

5.3.2.1 行为

主设备应通过发送过程数据请求帧请求另外一个节点(或自己)发送过程数据或发送过程数据至某个节点(可选),被寻址的节点应使用过程数据响应帧响应并广播给其他所有节点。

过程数据报文由过程数据请求帧和其后的过程数据响应帧组成,如图 44 所示。

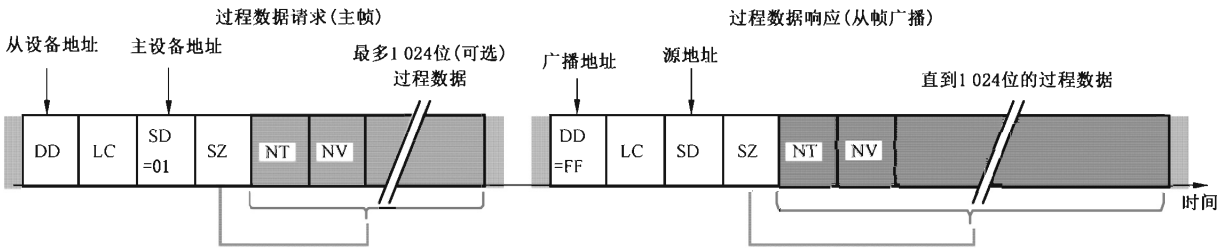


图 44 过程数据报文

5.3.2.2 过程数据请求帧

过程数据请求帧应具有下列格式,如图 45 所示。

Process_Data_Request ::= RECORD
{
 destination_device UNSIGNED8 — 节点地址或自轮询时主设备地址
 link_control Link_Control — 过程数据请求
 source_device UNSIGNED8 — 主设备地址
 link_data_size UNSIGNED8 — = 0 或(可选 1~128)
 ARRAY[link_data_size] OF WORD8 — 过程数据(可选)
 — 由应用定义的内容
}

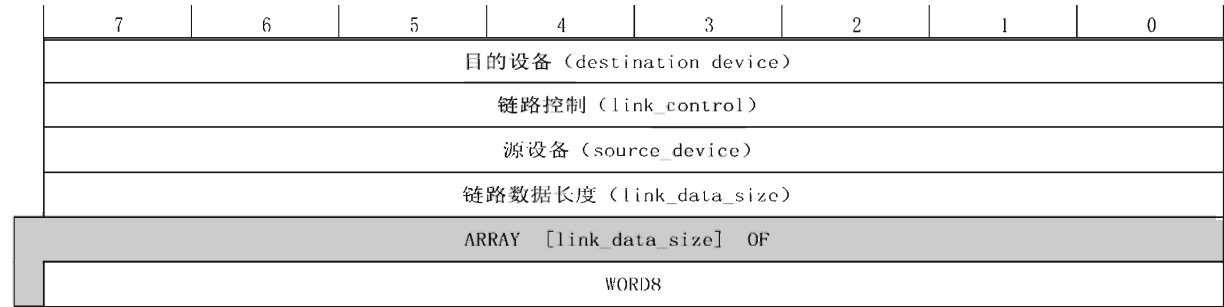


图 45 过程数据请求帧格式

5.3.2.3 过程数据响应帧

过程数据响应帧应具有下列格式,如图 46 所示。

```
Process_Data_Response ::= RECORD
{
  destination_device    UNSIGNED8    ——目的设备地址=广播地址
  link_control          Link_Control ——过程数据响应
  source_device         UNSIGNED8    ——节点或主设备地址
  link_data_size        UNSIGNED8    ——(0~128)
  ARRAY [link_data_size] OF WORD8    ——由应用定义的内容;
                                     建议前两个八位位组为节点密钥且应用检查是否与从拓扑中接收
                                     到的该节点的节点密钥相符
}
```

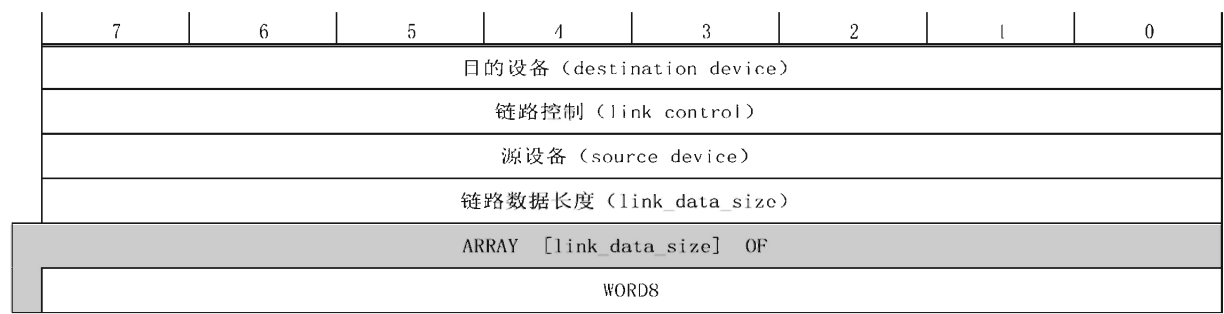


图 46 过程数据响应帧格式

5.3.3 消息数据

5.3.3.1 行为

主设备应使用消息数据请求帧请求其他节点(或自己)发送消息数据,被寻址的节点应使用消息数据响应帧响应,消息数据响应帧目的地址为单个节点地址或广播地址,如图 47 所示。

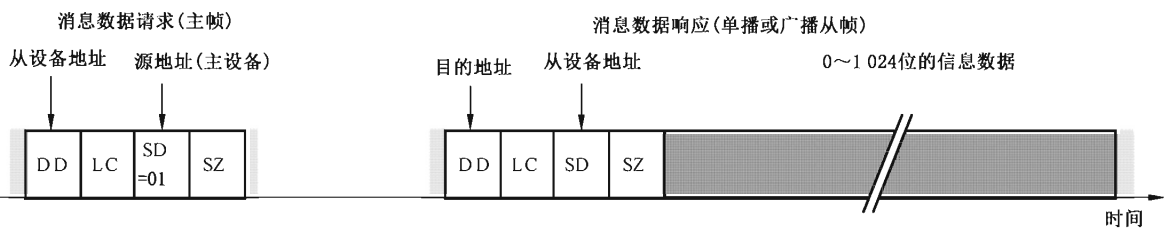


图 47 消息数据报文

注：消息数据结构在第 6 章中规定。

5.3.3.2 消息数据请求帧

消息数据请求帧应具有下列格式,如图 48 所示。

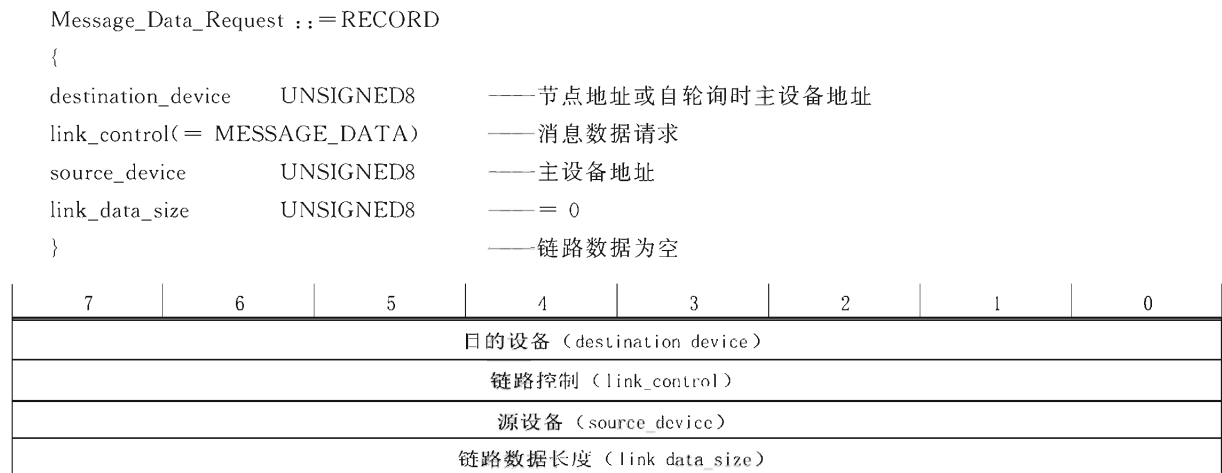


图 48 消息数据请求帧格式

5.3.3.3 消息数据响应帧

消息数据响应帧应具有下列格式,如图 49 所示。

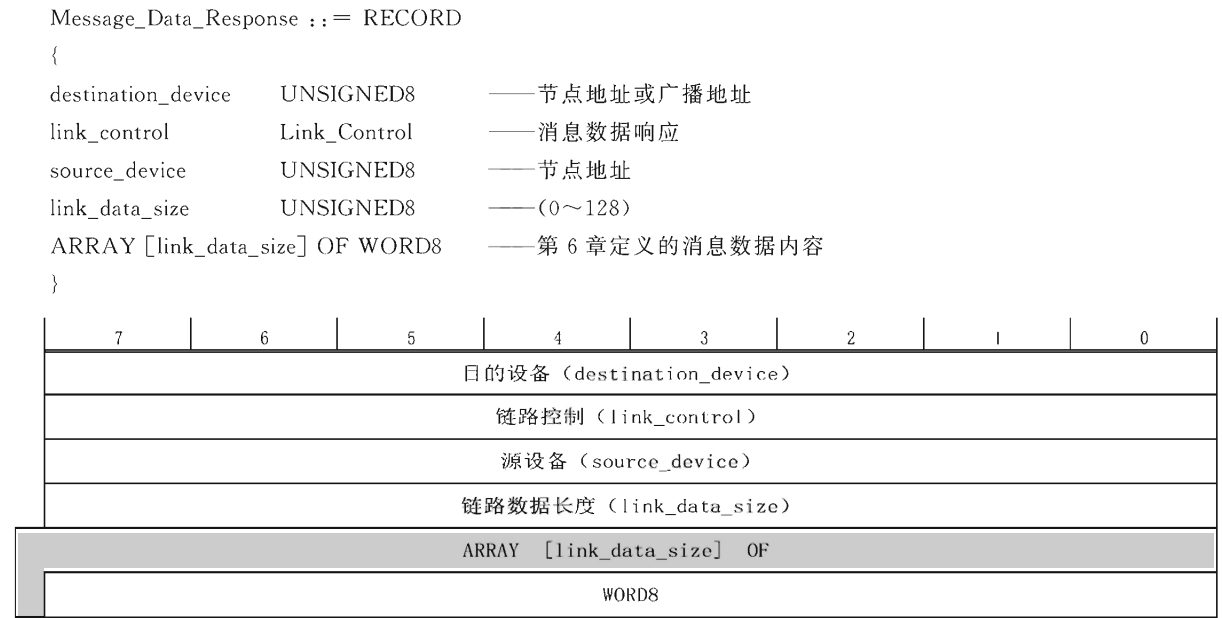


图 49 消息数据响应帧格式

5.3.4 监视数据

5.3.4.1 行为

主设备应通过监视数据请求帧请求节点发送监视数据或发送监视数据给节点,被寻址的源节点应使用监视数据响应帧响应,如图 50 所示。

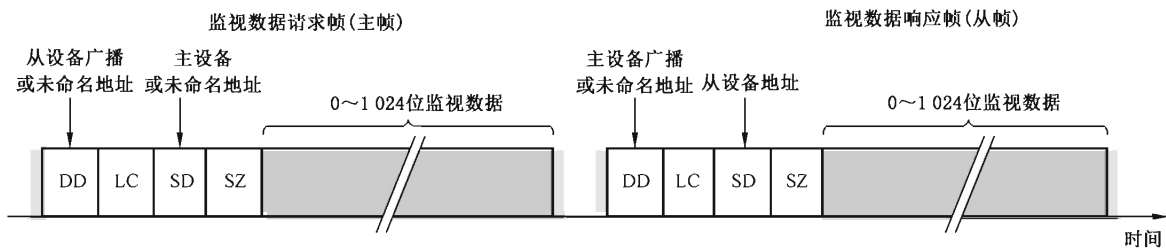


图 50 监视报文

在辅助通道上,末端节点可扮演主设备角色。

5.3.4.2 监视报文格式

监视帧应具有如下格式。

```
Supervisory_Data ::= ONE_OF[link_control.mq]
{
  [MQ]                Supervisory_Data_Request,
  [SR]                Supervisory_Data_Response
}

Supervisory_Data_Request ::= ONE_OF[link_control.supervisory_type]
{
  [DETECT]            Detect_Request,
  [PRESENCE]          Presence_Request,
  [STATUS]            Status_Request,
  [NAMING]            Naming_Request,
  [SETINT]            SetInt_Request,
  [SETEND]            SetEnd_Request,
  [TOPOGRAPHY]        Topography_Request,
  [UNNAME]            Unname_Request
}

Supervisory_Data_Response ::= ONE_OF[link_control.supervisory_type]
{
  [DETECT]            Detect_Response,
  [PRESENCE]          Presence_Response,
  [STATUS]            Status_Response,
  [NAMING]            Naming_Response,
  [SETINT]            SetInt_Response,
  [SETEND]            SetEnd_Response,
  [TOPOGRAPHY]        Topography_Response
}
```

5.3.5 检测报文

5.3.5.1 行为

末端节点应通过检测请求帧向另一节点指示其存在,另一节点(若存在且能应答)应使用检测响应帧响应。

检测报文如图 51 所示。

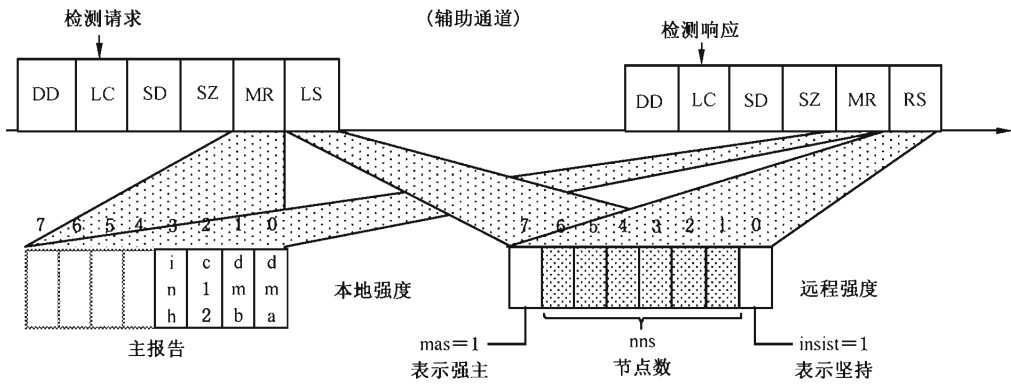


图 51 检测报文

5.3.5.2 检测请求帧

检测请求帧应具有下列格式,如图 52 所示。

Detect_Request ::= RECORD
{
destination_device UNSIGNED8 —— 未命名节点地址
link_control Link_Control —— 检测请求
source_device UNSIGNED8 —— 未命名节点地址
link_data_size UNSIGNED8 —— = 2
master_report Master_Report —— 见 5.5.2.6
local_strength Composition_Strength —— 请求节点的本地组成强度的拷贝(见 5.5.2.5)
 —— ins 位(坚持位)置位
}

7	6	5	4	3	2	1	0
目的设备 (destination_device)							
链路控制 (link_control)							
源设备 (source_device)							
链路数据长度 (link_data_size)							
主设备报告 (master_report)							
强主 (mas)	组成中已命名设备数 (nns)					坚持 (ins)	

图 52 检测请求帧格式

5.3.5.3 检测响应帧

检测响应帧应具有如下格式,如图 53 所示。

Detect_Response ::= RECORD

{		
destination_device	UNSIGNED8	——广播地址
link_control	Link_Control	——检测响应
source_device	UNSIGNED8	——未命名节点地址
link_data_size	UNSIGNED8	——= 2
master_report	Master_Report	——与检测请求帧相同(用于其他组成)
remote_strength	Composition_Strength	——响应节点的远端组成强度,若远端组成坚持则远端节点置位“ins”位(见 5.5.2.5)
}		

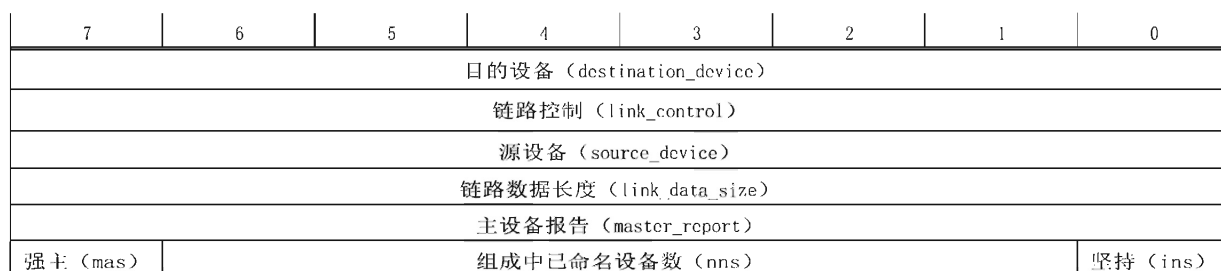


图 53 检测响应帧格式

5.3.6 存在报文

5.3.6.1 行为

主设备应通过存在请求帧请求末端节点指示其存在及另一个组成存在的可能性,末端节点应使用存在响应帧响应,如图 54 所示。

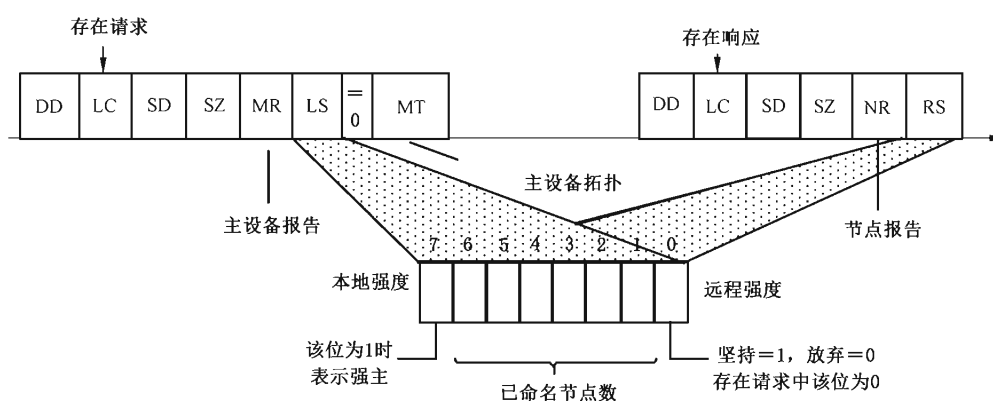


图 54 存在报文

5.3.6.2 存在请求帧

存在请求帧应具有如下格式,如图 55 所示。

Presence_Request ::= RECORD
{
 destination_device UNSIGNED8 —— 末端节点地址
 link_control Link_Control —— 存在请求
 source_device UNSIGNED8 —— 主设备地址
 link_data_size UNSIGNED8 —— = 4
 master_report Master_Report —— 见 5.5.2.6
 local_strength Composition_Strength —— 主设备本地组成强度的拷贝, ins 位 = 0
 reserved1 WORD4 (= 0) —— 保留, = 0
 master_topo Master_Topo —— 见 5.5.2.8
}

7	6	5	4	3	2	1	0
目的设备（destination_device）							
链路控制（link_control）							
源设备（source_device）							
链路数据长度（link_data_size）							
主设备报告（master_report）							
强主（mas）	组成中已命名设备数（nns）						坚持（ins）
保留1（reserved1）							
主设备拓扑（master_topo）							

图 55 存在请求帧格式

5.3.6.3 存在响应帧

存在响应帧应具有如下格式,如图 56 所示。

Presence_Response ::= RECORD
{
 destination_device UNSIGNED8 —— 广播地址
 link_control Link_Control —— 存在响应
 source_device UNSIGNED8 —— 末端节点地址
 link_data_size UNSIGNED8 —— = 2
 node_report Node_Report —— 见 5.5.2.3
 remote_strength Composition_Strength —— 末端节点远端组成强度的拷贝,若 ins 位 = 1 则表明另一组成坚持
}

7	6	5	4	3	2	1	0
目的设备 (destination_device)							
链路控制 (link_control)							
源设备 (source_device)							
链路数据长度 (link_data_size)							
节点报告 (node_report)							
强主 (mas)	组成中已命名设备数 (nns)						坚持 (ins)

图 56 存在响应帧格式

5.3.7 状态报文

5.3.7.1 行为

主设备应通过状态请求帧请求节点状态,从设备应使用状态响应帧响应,如图 57 所示。

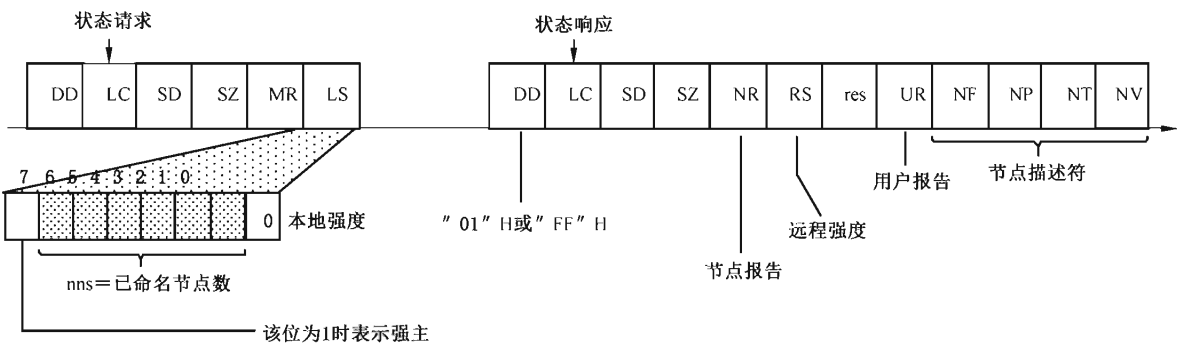


图 57 状态报文

5.3.7.2 状态请求帧

状态请求帧应具有如下格式,如图 58 所示。

```
Status_Request ::= RECORD
{
    destination_device    UNSIGNED8    —— 节点地址
    link_control          Link_Control  —— 状态请求
    source_device         UNSIGNED8    —— 主设备地址
    link_data_size        UNSIGNED8    —— = 2
    master_report         Master_Report —— 见 5.5.2.6
    local_strength        Composition_Strength —— 主设备本地组成强度,ins 位=0
}
```

7	6	5	4	3	2	1	0
目的设备 (destination_device)							
链路控制 (link_control)							
源设备 (source_device)							
链路数据长度 (link_data_size)							
主设备报告 (master_report)							
强主 (mas)	组成中已命名设备数 (nns)						坚持 (ins)

图 58 状态请求帧格式

5.3.7.3 状态响应帧

状态响应帧应具有如下格式,如图 59 所示。

Status_Response ::= RECORD
{
 destination_device UNSIGNED8 ——主设备地址或广播地址
 link_control Link_Control ——状态响应
 source_device UNSIGNED8 ——节点地址
 link_data_size UNSIGNED8 —— = 8
 node_report Node_Report ——见 5.5.2.3
 remote_strength Composition_Strength ——末端节点远端组成强度,中间节点为 0
 reserved1 WORD8(= 0) ——保留, = 0
 user_report user_Report ——见 5.5.2.4
 node_descriptor Node_Descriptor ——见 5.5.2.2
}

7	6	5	4	3	2	1	0
目的设备（destination_device）							
链路控制（link_control）							
源设备（source_device）							
链路数据长度（link_data_size）							
节点报告（node_report）							
强主（mas）	组成中已命名设备数（nns）						坚持（ins）
保留1（reserved1）							
用户报告（user_report）							
节点帧长度（node_frame_size）							
保留1（reserved1）					节点周期（node_period）		
节点类型（node_type）							
节点版本（node_version）							

图 59 状态响应帧格式

5.3.8 中间设定报文

5.3.8.1 行为

主设备应通过发送中间设定请求帧请求从设备将其开关设置为中间设定状态,从设备应以中间设定响应帧确认,如图 60 所示。



图 60 中间设定报文

5.3.8.2 中间设定请求帧

中间设定请求帧应具有如下格式,如图 61 所示。

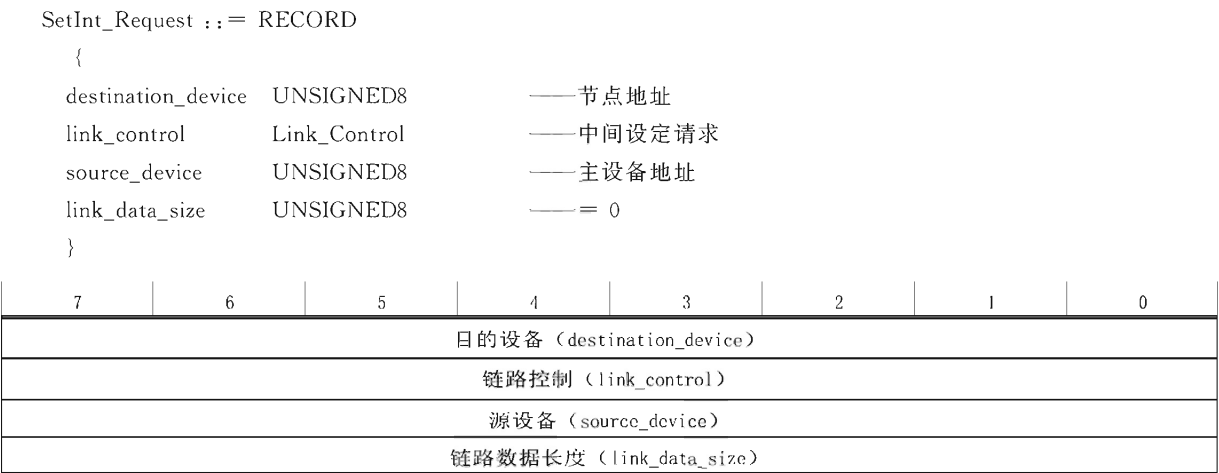


图 61 中间设定请求帧格式

5.3.8.3 中间设定响应帧

中间设定响应帧应具有如下格式,如图 62 所示。

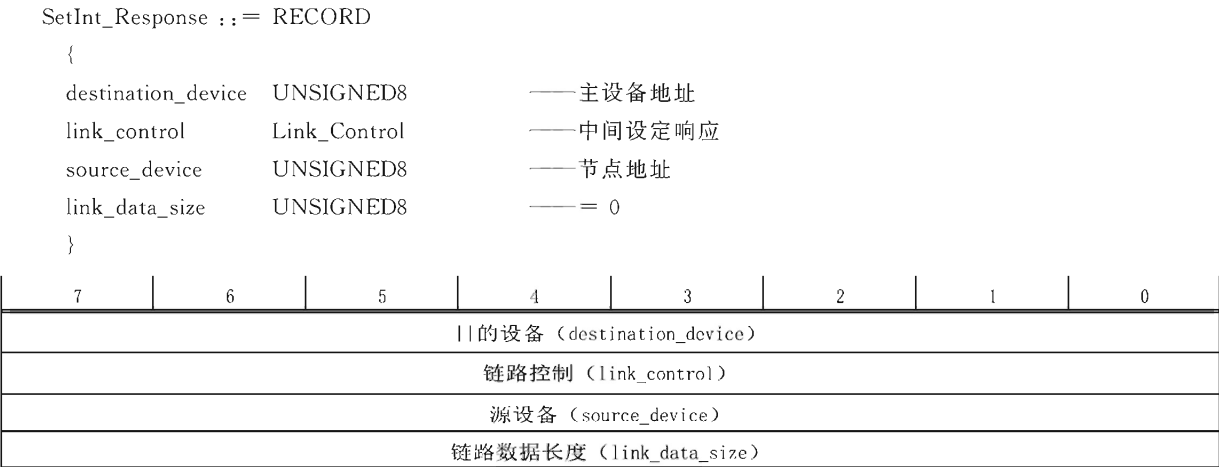


图 62 中间设定响应帧格式

5.3.9 命名报文

5.3.9.1 行为

主设备应通过命名请求帧将其分配的地址和强度传送给从设备,从设备应以命名响应帧确认,如图 63 所示。

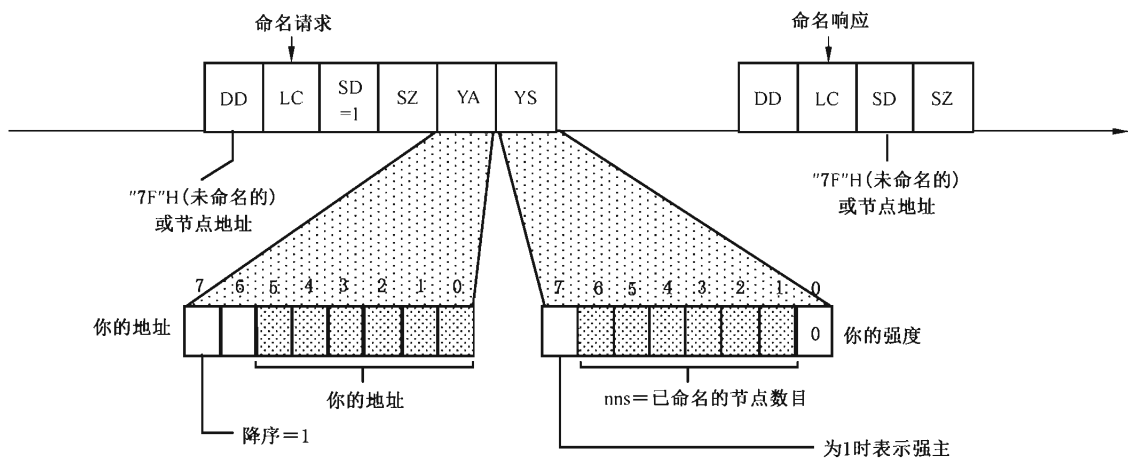


图 63 命名报文

5.3.9.2 命名请求帧

命名请求帧应具有如下格式,如图 64 所示。

Naming_Request ::= RECORD
{
 destination_device UNSIGNED8 ——节点地址或未命名节点地址
 link_control Link_Control ——命名请求
 source_device UNSIGNED8 ——主设备地址
 link_data_size UNSIGNED8 ——= 2
 dir1 BOOLEAN1 ——若以升序命名,则= 1
 rsv1 WORD1 ——保留,= 0
 your_address UNSIGNED6 ——由主设备分配
 your_strength Composition_Strength ——由主设备分配,ins 位= 0
}

7	6	5	4	3	2	1	0
目的设备 (destination_device)							
链路控制 (link_control)							
源设备 (source_device)							
链路数据长度 (link_data_size)							
方向 (dir1)	保留1 (rsv1)	你的地址 (your_address)					
强主 (mas)	组成中已命名设备数 (nns)						坚持 (ins)

图 64 命名请求帧格式

5.3.9.3 命名响应帧

命名响应帧应具有如下格式,如图 65 所示。

Naming_Response ::= RECORD
{
 destination_device UNSIGNED8 ——主设备地址或未命名节点地址
 link_control Link_Control ——命名响应
 source_device UNSIGNED8 ——节点地址或未命名节点地址(见 5.3.9)
 link_data_size UNSIGNED8 —— = 0
}

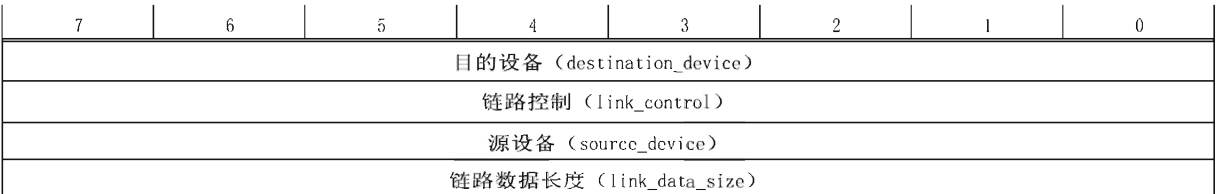


图 65 命名响应帧格式

5.3.10 消名报文

5.3.10.1 行为

主设备应通过广播消名请求帧请求所有从设备消名,从设备不应响应,如图 66 所示(无消名响应帧)。



图 66 消名报文

5.3.10.2 消名请求帧

消名请求帧应具有如下格式,如图 67 所示。

Unname_Request ::= RECORD
{
 destination_device UNSIGNED8 ——广播地址
 link_control Link_Control ——消名请求
 source_device UNSIGNED8 ——主设备地址
 link_data_size UNSIGNED8 —— = 0
}

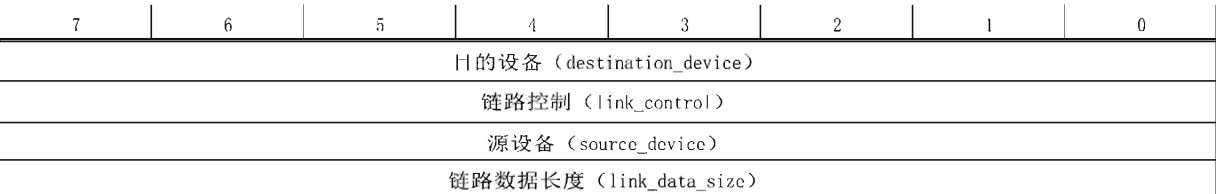


图 67 消名请求帧格式

5.3.11 末端设定报文

5.3.11.1 行为

主设备应通过末端设定请求帧请求从设备切换到末端设定状态并采用新的组成强度,从设备应以末端设置响应帧响应,如图 68 所示。



图 68 末端设定报文

5.3.11.2 末端设定请求帧

末端设定帧应具有如下格式,如图 69 所示。

SetEnd_Request ::= RECORD
{
destination_device UNSIGNED8 ——节点地址
link_control Link_Control ——末端设定请求
source_device UNSIGNED8 ——主设备地址
link_data_size UNSIGNED8 ——= 2
reserved1 WORD8(0) ——= 0
local_strength Composition_Strength ——主设备所见的本地组成强度(见 5.5.2.5)
}

7	6	5	4	3	2	1	0
目的设备 (destination device)							
链路控制 (link_control)							
源设备 (source device)							
链路数据长度 (link_data size)							
保留1 (reserved1)							
强度 (mas)		组成中已命名设备数 (nns)					坚持 (ins)

图 69 末端设定请求帧格式

5.3.11.3 末端设定响应帧

末端设定响应帧应具有如下格式,如图 70 所示。

```
SetEnd_Response ::= RECORD
{
    destination_device    UNSIGNED8    ——主设备地址
    link_control          Link_Control ——末端设定响应
    source_device         UNSIGNED8    ——节点地址
    link_data_size        UNSIGNED8    —— = 0
}
```

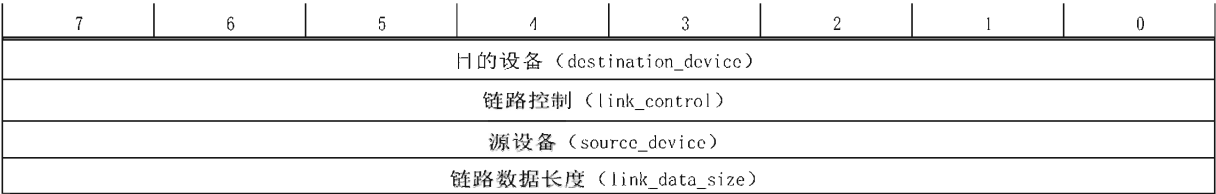


图 70 末端设定响应帧格式

5.3.12 拓扑报文

5.3.12.1 行为

主设备应通过拓扑请求帧将其拓扑结构告知从设备,从设备应以拓扑响应帧确认,如图 71 所示。

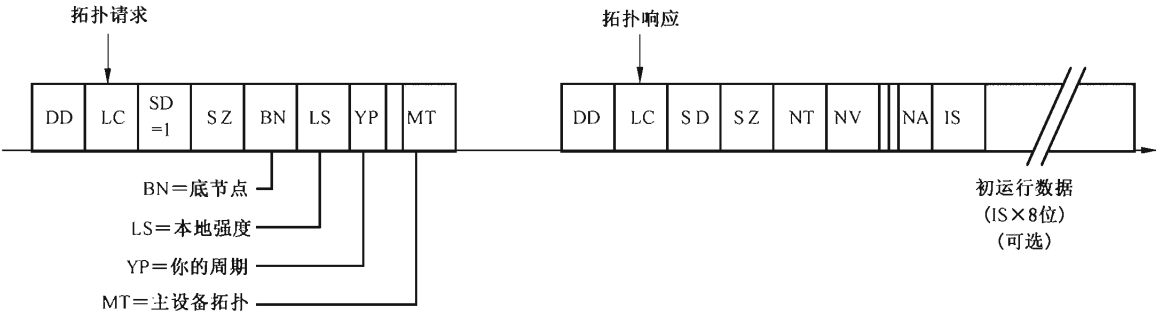


图 71 拓扑报文

5.3.12.2 拓扑请求帧

拓扑请求帧应具有如下格式,如图 72 所示。

```
Topography_Request ::= RECORD
{
    destination_device    UNSIGNED8    ——节点地址(可是主设备地址)
    link_control          Link_Control ——拓扑请求
    source_device         UNSIGNED8    ——主设备地址
    link_data_size        UNSIGNED8    —— = 4
    bottom_node           UNSIGNED8    ——主设备在方向 1 上末端节点的地址
    local_strength        Composition_Strength ——主设备所见的编组强度(local_strength.ins= 0)
    your_period           UNSIGNED4    ——指定的特征周期(见 5.5.2.2 和 5.4.2)
    master_topo           Master_Topo  ——见 5.5.2.8
}
```

图 72 拓扑请求帧格式

7	6	5	4	3	2	1	0
目的设备（destination_device）							
链路控制（link_control）							
源设备（source_device）							
链路数据长度（link_data_size）							
底节点（bottom_node）							
强主（mas）	组成中已命名设备数（nns）						坚持（ins）
你的周期（your_period）							
主设备拓扑（master_topo）							

图 72 (续)

5.3.12.3 拓扑响应帧

拓扑响应帧应具有如下格式,如图 73 所示。

Topography_Response ::= RECORD
{
 destination_device UNSIGNED8 ——广播地址
 link_control Link_Control ——拓扑响应

 source_device UNSIGNED8 ——源节点地址(可是主设备地址)
 link_data_size UNSIGNED8 ——0~128
 node_type Node_Type ——节点密钥第一部分
 node_version Node_Version ——节点密钥第二部分
 sam BOOLEAN1 ——若与主设备同方向则为“1”
 rsv1 WORD1 ——保留,=0
 node_address UNSIGNED6 ——由初运行给出的节点地址
 inaug_data_size UNSIGNED8 ——初运行数据长度,1~124 个八位位组
 inauguration_data ARRAY [inaug_data_size] OF WORD8 ——应用定义的初运行数据
}

7	6	5	4	3	2	1	0
目的设备（destination_device）							
链路控制（link_control）							
源设备（source_device）							
链路数据长度（link_data_size）							
节点类型（node_type）							
节点版本（node_version）							
方向（sam）	保留（rsv1）	节点地址（node_address）					
初运行数据长度（inaug_data_size）							
ARRAY [inaug_data_size] OF							
WORD8							

图 73 拓扑响应帧格式

注：初运行数据长度字段与链路数据长度字段冗余是有意为之的,用于合理性检查。

5.4 介质分配

5.4.1 组织

5.4.1.1 概述

下列规范适用于常规运行,即当只有一个主设备且总线能传送应用数据时。当初运行结束时进入常规运行,当组成改变时退出常规运行。

注:在多个节点中对主设备的选择在 5.5 中定义。主设备的选择不属于本条描述的介质分配的一部分。

5.4.1.2 基本周期

5.4.1.2.1 基本周期结构

主设备应将总线活动分为固定的周期,即称作“基本周期”。

一个基本周期应分为两个相,如图 74 所示:

- a) 用于传送周期数据的周期相;
- b) 用于传送监视数据和/或消息数据的偶发相。

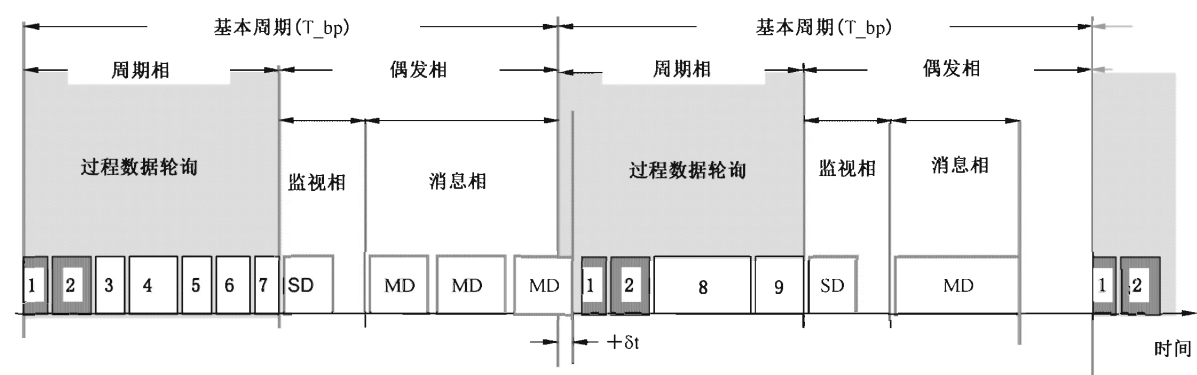


图 74 基本周期结构

5.4.1.2.2 基本周期时长

基本周期时长应为 $T_{bp} = 25.0\text{ ms} \pm 1.0\text{ ms}$ 。

在基本周期开始后,只要周期相没有结束,主设备不可发送消息数据请求帧或监视数据请求帧。

注:周期相可能延时 δt 开始。 δt 用于轮询及发送最长可能的消息数据帧或监视数据帧,大约为 2.0 ms。若没有其他偶发数据在发送,下一个基本周期期望按计划时间开始。

5.4.2 周期相

5.4.2.1 特征周期

同一节点连续两次轮询之间的时间间隔叫作特征周期,即 T_{ip} 。

特征周期应是基本周期 T_{bp} 的 2^n 倍,即 $T_{ip} = (2^n \times T_{bp})$ 。

注:特征周期由每个节点的应用定义。在初运行期间,每个节点在其节点描述符(见 5.5.2.2)中通告其期望的特征周期和帧长度。

总线上任意节点的最长特征周期叫作宏周期。

5.4.2.2 周期扫描表

周期扫描表是在宏周期的每个基本周期内轮询的所有节点列表。周期扫描表也定义了在每个基本

周期中留给偶发相的时间。

主设备应根据在初运行期间每个节点期望的特征周期和从每个节点接收到的过程数据长度来配置周期扫描表。

主设备应在基本周期内均衡地发送轮询,以便留出每个基本周期的 40% 给偶发相。

在宏周期内,若周期相平均占基本周期的 60% 以上,则具有最长周期的节点的特征周期应加倍,直到周期相平均占基本周期的 60% 以下。

若该操作不足以使周期相平均占基本周期的 60% 以下,则具有次长周期的节点的特征周期应加倍,且如需要则依此类推直到具有最短周期的节点的特征周期加倍为止。

每个节点选择的特征周期应在拓扑请求帧中以“你的周期”字段传送给每个节点。

5.4.2.3 末端节点轮询

主设备应通过发送存在请求帧在一个基本周期中轮询一个末端节点,且在下一个基本周期中轮询另一个末端节点,末端节点应通过广播存在响应帧予以响应。

注:存在报文允许所有节点监视总线的完整性,且允许主设备检测另一组成的存在。

5.4.2.4 出错条件及处理

当节点停止响应时,主设备不应从其周期扫描表中移除该节点,而应继续轮询该节点直到下一次初运行或节点重新插入总线。

注 1:只有通过一次新的初运行才能从组成中移除故障节点。

当某节点连续三次停止响应轮询时,预定了该节点过程数据的节点应能向其应用指示该节点丢失,且若该节点恢复则应指示该节点重新插入。

注 2:过程数据的宿时间监视提供丢失节点监视。

在常规运行期间节点停止观察预期通信(如末端节点丢失、主设备丢失、没有轮询)的行为应在 5.5.4.9.3 中定义。

5.4.3 偶发相

5.4.3.1 事件通报

节点应通过设置过程数据响应帧或消息数据响应帧中的“A 位”和“C 位”来请求偶发性发送。

若在周期相期间有多个节点声明有偶发性数据发送,主设备应以巡回方式处理这些请求,以使在再次服务同一个节点前已服务所有其他请求。

所有监视数据请求(“C 位”)应在消息数据请求(“A 位”)之前处理。

5.4.3.2 消息列表

主设备应将在先前过程数据帧或消息数据帧中“A 位”置位的节点的地址插入到其消息列表中。

主设备应通过消息数据请求帧轮询指示有变化的节点,且当主设备轮询节点获取消息数据时应将该节点从消息列表中移除,除非消息数据响应帧中也置位“A 位”。

5.4.3.3 监视列表

主设备应将在先前过程数据帧或消息数据帧中“C 位”置位的节点的地址插入到其监视列表中。

主设备应通过状态请求帧轮询指示有变化的节点,且当主设备接收到该节点状态响应帧时应将该节点地址从监视列表中移除。

注:此表通常为空白。该表包含有在总线延伸时的末端节点地址、描述符有变化的节点地址或宣称休眠请求变化(设置休眠或退出休眠)的节点地址。

5.4.3.4 后台扫描(可选)

主设备可向未包含在其消息列表或监视列表之中的节点轮询消息数据或状态数据。

注：不能发送过程数据但能发送消息数据的节点通过后台扫描轮询。

5.5 初运行

5.5.1 概述

5.5.1.1 地址分配

初运行应为每个节点分配一个地址，如图 75 所示。

——从主设备开始，方向 1 上的节点从 63 开始降序编号，最后一个命名的节点为底节点；

——从主设备开始，方向 2 上的节点从 02 开始升序编号，最后一个命名的节点为顶节点。

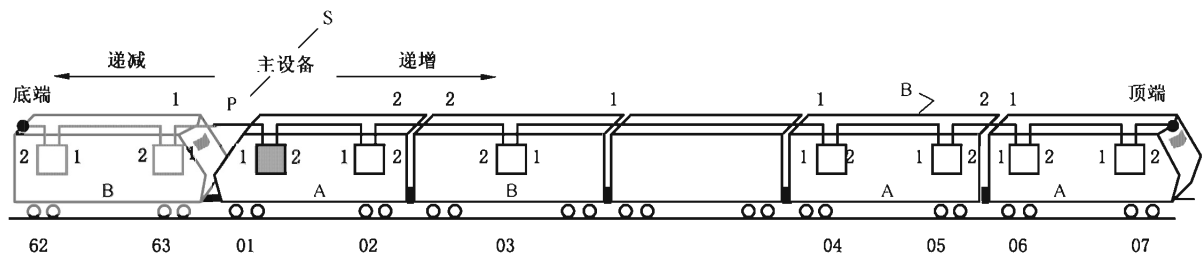


图 75 节点位置编号

5.5.1.2 节点分级

应用可通过将节点分为如下等级实现不同的主权策略：

- 强节点；
- 弱节点；
- 从节点。

5.5.1.3 强节点

强节点由应用提升为主设备。

执行主权的强节点叫作强主。

强节点可由应用降为弱节点。若节点曾是强主，则将向所有节点指示其被降级，并作为弱主保持对总线的控制，直到指定了强节点为止。

注 1：通常，在一个组成中，应用只提升一个节点作为强节点。若总线上存在一个以上强节点，初运行过程将导致总线分成与强节点个数一样多的独立总线段，因为每个总线段只能有一个主设备。

注 2：强节点允许将主权与某些应用功能(如与头车)绑定在一起。这种绑定对在过程数据请求帧中包含过程数据是需要的(见 5.3.2)。

5.5.1.4 弱节点

弱节点是应用允许成为主设备的节点。

执行主权的弱节点叫作弱主。

通常，在一个组成中，应用指定多个或所有节点为弱节点。

在没有强节点的组成中，平等对待所有节点的竞争结果保证了有且仅有一个弱节点成为弱主，其他所有节点成为从节点。

若弱主检测到强节点或另一个控制了更多从节点的弱主的存在,则该弱主将被降级且成为另一个主设备的从节点。

若被应用提升为强节点的弱节点还不是主设备,则其变成强主并初运行总线。

注:在没有明确的应用命令时,允许弱节点操作总线。弱节点能通过执行初运行和指定另一(弱)节点为主设备来克服主设备失效。

5.5.1.5 从节点

从节点是在任何时候应用都不准许成为主设备的节点。

注:从节点不可参与总线恢复。这种模式在测试或与强节点连接中 useful。

5.5.2 描述符

5.5.2.1 概述

下列数据结构在监视数据帧和总线管理消息中使用。数据结构以传送符定义。对于链路层接口,相应的 C 数据类型在 5.6.4 中定义。

5.5.2.2 节点描述符

每个节点应实现节点描述符以标识其特性。

节点描述符应以下列数据结构表示,如图 76 所示。

Node_Descriptor :: = RECORD
{
 node_frame_size UNSIGNED8, ——该节点过程数据响应帧的链路数据长度
 reserved1 WORD5(=0), ——保留, = 0
 node_period UNSIGNED3, ——节点请求的特征周期,表示为基本周期 T_bp 的 2ⁿ 倍。
 n = 0 ~ 7:
 0: 1 T_bp (25.0 ms)
 1: 2 T_bp (50.0 ms)
 2: 4 T_bp (100 ms)
 3: 8 T_bp (200 ms)
 4: 16 T_bp (400.0 ms)
 5: 32 T_bp (800.0 ms)
 6: 64 T_bp (1.6 s)
 7: 128 T_bp (3.2 s)
 node_type UNSIGNED8 ——节点密钥第一部分
 node_version UNSIGNED8 ——节点密钥第二部分
}

7	6	5	4	3	2	1	0			
节点帧长度 (node_frame_size)										
保留1 (reserved1)					节点周期 (node_period)					
节点类型 (node_type)										
节点版本 (node_version)										

图 76 节点描述符格式

注 1: 对于特定应用,节点描述符需得到供应商和用户之间一致同意。

- 注 2：节点描述符在命名响应帧、状态响应帧和拓扑响应帧中作为内部变量使用。
- 注 3：用于国际联运客车的 UIC 节点描述符参加 UIC 556。
- 注 4：应用可插入节点密钥作为每个过程数据帧的头两个字节，以保证有更好的防错保护（见 5.6.2.2）。

5.5.2.3 节点报告

每个节点应实现节点报告，以报告线路受扰和组成改变。

节点报告应以下列结构表示，如图 77 所示。

```
Node_Report ::= BITSET8
{
    da1          (7)      ——线路 A1 受扰,拷贝 4.8.2.5 的 DA1
    da2          (6)      ——线路 A2 受扰,拷贝 4.8.2.5 的 DA2
    db1          (5)      ——线路 B1 受扰,拷贝 4.8.2.5 的 DB1
    db2          (4)      ——线路 B2 受扰,拷贝 4.8.2.5 的 DB2
    int          (3)      ——节点处于中间设定状态时置位,处于末端设定状态时复位
    dsc          (2)      ——节点描述符改变时置位,节点接收到新的拓扑时复位
    slp          (1)      ——节点拟进入低功耗模式时置位,休眠请求取消时复位
    sam          (0)      ——节点与主设备朝向相同时置位,相反时复位
}
```

7	6	5	4	3	2	1	0
A1线受扰 (da1)	A2线受扰 (da2)	B1线受扰 (db1)	B2线受扰 (db2)	中间设定 (int)	描述符改变 (dsc)	休眠模式 (slp)	同向指示 (sam)

图 77 节点报告格式

5.5.2.4 用户报告

每个节点应实现用户报告，以传送一个应用专属的八位位组给其他应用，例如报告应用专属干扰。

用户报告应以一个八位位组表示，如图 78 所示。

```
user_Report ::= WORD8 ——应用定义的
```

7	6	5	4	3	2	1	0
用户报告位7 (ur7)	用户报告位6 (ur6)	用户报告位5 (ur5)	用户报告位4 (ur4)	用户报告位3 (ur3)	用户报告位2 (ur2)	用户报告位1 (ur1)	用户报告位0 (ur0)

图 78 用户报告格式

注：应用可通过链路层服务设置用户报告中的每一位。

5.5.2.5 组成强度

每个节点应实现一个本地组成强度变量(LocStr)，用以指示组成中有多少个节点以及其主设备是强主还是弱主。

每个节点应为每一个辅助通道实现一个远端组成强度变量，即 RemStr(1)和 RemStr(2)，用以指示节点在方向 1 或方向 2 上检测到的远端组成的强度。

注：这些变量仅在相应通道激活时使用。中间节点不需要这些变量。

组成强度应以下列结构表示，如图 79 所示。

Composition_Strength :: = RECORD
{
 mas BOOLEAN1, ——若本组成主设备是强主则置位,弱主则复位
 nns UNSIGNED6, ——组成中已命名节点的数目
 ins BOOLEAN1 ——若组成在冲突中坚持则置位,放弃则复位
}

7	6	5	4	3	2	1	0
强主 (mas)	组成中已命名设备数 (nns)						坚持 (ins)

图 79 组成强度格式

为简化强度比较,本地组成强度或远端组成强度应作为一个无符号八位位组值计算,见式(3):

$$\text{RemStr 或 LocStr} = (\text{mas} \times 128) + 2 \times \text{nns} + \text{ins} \quad \dots\dots\dots (3)$$

示例:

- RemStr = 0 没有检测到更多的节点;
- RemStr = 1 发现未命名节点;
- RemStr = 2 单个不坚持弱主;
- RemStr = 3 单个坚持弱主;
- RemStr = 4 有一个从节点的不坚持弱主;
- RemStr = 13 有 5 个从节点的坚持弱主;
- RemStr > 128 由强主命名的组成。

5.5.2.6 主报告

每个节点应实现主报告,以报告冗余线的干扰并允许识别干扰的方向。
主设备报告应以下列结构表示,如图 80 所示。

Master_Report :: = BITSET8
{
 rsv1, (=0) ——保留,设置为 0
 rsv2, (=0) ——保留,设置为 0
 rsv3, (=0) ——保留,设置为 0
 rsv4, (=0) ——保留,设置为 0
 inh, ——若本组成任意节点禁止初运行则置位
 c12, ——若此帧送到相对于本节点的方向 2 则置位
 dmb, ——若主通道的 B 线受扰则置位(拷贝 DB1 或 DB2)
 dma ——若主通道的 A 线受扰则置位(拷贝 DA1 或 DA2)
}

7	6	5	4	3	2	1	0
保留1 (rsv1)	保留2 (rsv2)	保留3 (rsv3)	保留4 (rsv4)	禁止指示 (inh)	方向2指示 (c12)	B线受扰 (dmb)	A线受扰 (dma)

图 80 主报告

5.5.2.7 拓扑计数器

每个节点应实现拓扑计数器。拓扑计数器是一个以 64 为模数的计数器,节点每接收到一个连续的拓扑则计数器加 1 或 2。拓扑计数器不取 0 值,直接从 63 增到 1。

拓扑计数器应以下列结构表示,如图 81 所示。

Topo_Counter::= UNSIGNED6 ——0 = 不使用

7	6	5	4	3	2	1	0
X	X	拓扑计数器 (topo_counter)					

图 81 拓扑计数器格式

- 注 1: 实时协议使用拓扑计数器,以保证初运行时列车总线上交换的信息的坚固性。
- 在暂时掉电后,节点应确保改变拓扑计数器,以与掉电前的值不同。
- 注 2: 拓扑计数器可存储在非易失存储器(非易失存储器用于快速重新插入)中。当供电恢复时,节点能获取先前的拓扑计数器值并加以改变。
- 在冗余切换之后,新激活的节点应确保改变拓扑计数器,以与先前激活节点的值不同。
- 注 3: 可以一个节点使用偶数值的拓扑计数器,而另一个节点使用奇数值的拓扑计数器。节点每次收到连续拓扑时,拓扑计数器以 2 递增。这种方法确保了拓扑计数器各自保证偶数/奇数值。在冗余切换时,拓扑计数器从偶数值变成奇数值,或相反。

5.5.2.8 主设备拓扑

每个节点应实现主设备拓扑。主设备拓扑是一个 12 位的计数器,主设备每发布一个连续拓扑时计数器加 1。

主设备拓扑应以下列结构表示,如图 82 所示。

11	10	9	8	7	6	5	4	3	2	1	0
主设备拓扑 (master_topo)											

图 82 主设备拓扑格式

- 当节点第一次成为主设备时,12 位的主设备拓扑应随机初始化。
- 主设备每发布一次拓扑时,应将主设备拓扑加 1。
- 注: 该计数器允许暂时断开节点,以检查节点是否重新加入到正确的组成。

5.5.2.9 初运行计数器

每个节点应实现一个 16 位的初运行计数器,以记录节点初运行的次数,传递未命名节点状态。

注: 该计数器用于诊断。

5.5.3 其他组成检测(资料性)

5.5.3.1 检测协议

- 末端节点:
- 检测与开放式末端连接的附加节点的存在。

——向该附加节点报告自身存在。

为此,末端节点向其开放式末端(或在单个主设备时向其两个末端)发送包含有其本地组成强度的检测请求帧。

该检测请求帧初始通过置位其“坚持位(ins 位)”指示在有竞争发生时本组成将坚持。

在所有情况下,末端节点(已命名节点或未命名节点)在帧间间隔的时间限制(见 5.2.2.4)内,以检测响应帧(或另一个检测请求帧)响应已接收到的检测请求帧。

接收的末端节点将远端组成强度与其自身本地组成强度相比较:

——若远端组成强度弱于本地组成强度,则该节点保持其“坚持位”置位。

——若远端组成强度强于或等于本地组成强度,则该节点复位其“坚持位”且放弃。

——若两个组成都由强主命名,则该节点保持其“坚持位”置位。

接收到存在请求帧的节点采用存在响应帧中指定的强度。

末端节点在随后每个存在响应帧中报告:

- a) 另一节点的存在;
- b) 远端组成的强度;
- c) 在强度相同时,本地组成决定放弃或坚持。

5.5.3.2 冲突避免规则

因为两个独立组成的末端节点异步地发送检测请求帧,所以当节点期望检测响应帧时可能接收到错误的帧或另一个检测请求帧。下列规则减少了竞争:

- a) 在通过主通道接收到命名请求帧或状态请求帧后最迟 $400.0\ \mu\text{s}$,仍未处于常规运行的末端节点应通过辅助通道发送检测请求帧;
- b) 在通过主通道接收到存在请求帧后最迟 $400.0\ \mu\text{s}$,处于常规运行的末端节点应通过辅助通道发送检测请求帧,以有 50% 的可能性避免重复冲突;
- c) 单独的主设备应通过其两个辅助通道以至少每 $29.0\ \text{ms}$ 、至多每 $21.0\ \text{ms}$ 发送一次检测请求帧,在 $(25.0 \pm 4.0)\ \text{ms}$ 的范围内随机发送以避免重复冲突;
- d) 在发送检测请求帧后,末端节点应忽略在 $T_{\text{detecting_response}} = 1.047\ \text{ms}$ 内接收到的既不是检测响应帧又不是检测请求帧的帧,且应忽略在此时间之后接收到的既不是检测请求帧又不是命名请求帧的帧;
- e) 主设备不应以高于末端节点能跟随的速率来发送检测请求帧。这由状态请求帧、存在请求帧和命名请求帧协议保证。

注 1: 同时也是末端节点的主设备向自身发送存在请求帧。

注 2: 同时也是末端节点的主设备向自身发送状态请求帧。

示例:图 83 给出了一个典型检测过程的时序图。

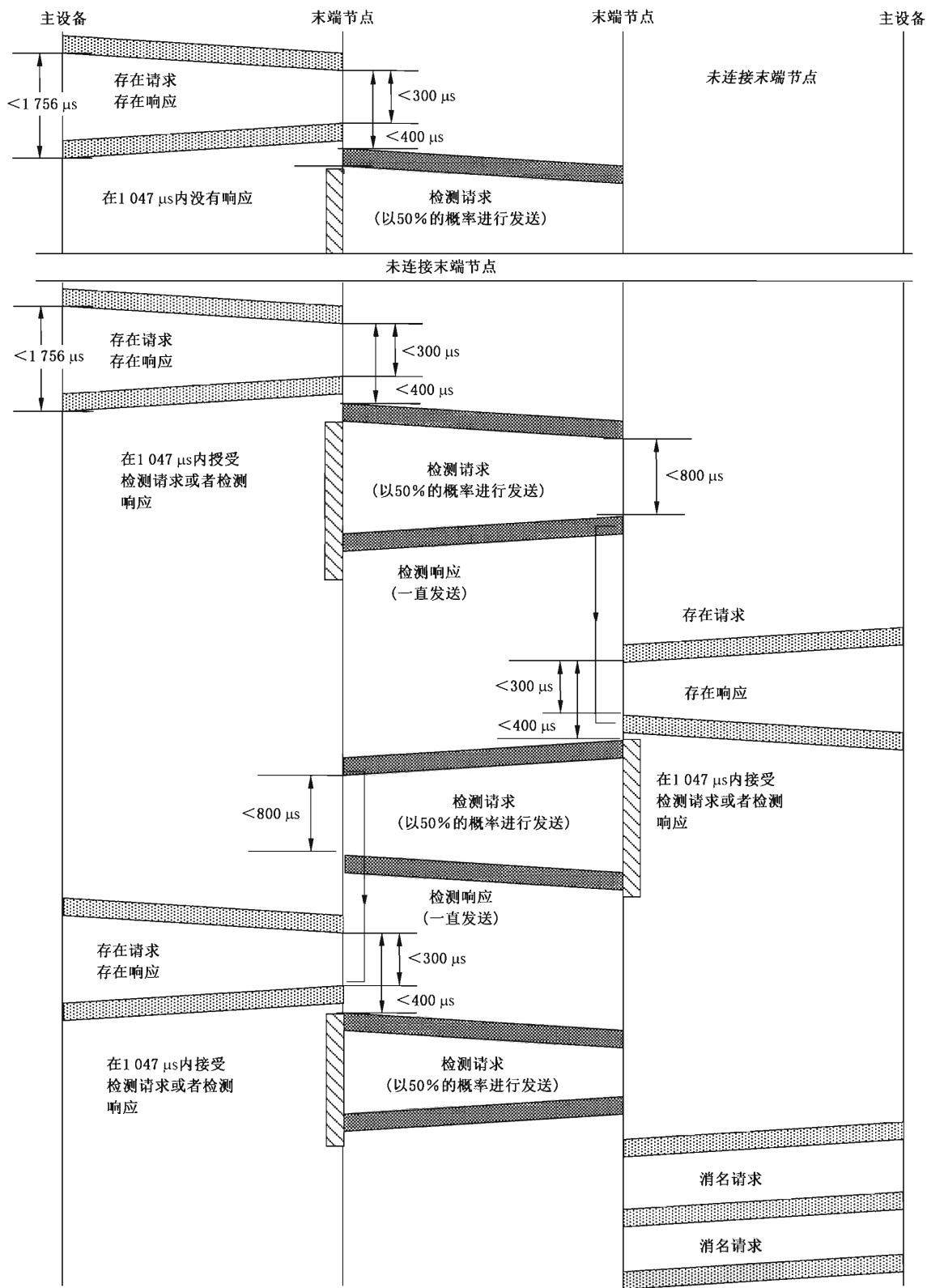


图 83 检测协议时序图

5.5.4 初运行状态图

5.5.4.1 节点结构

节点应处于图 84 所示的主要状态之一。这些主要状态又被分成一些小状态,在以下章条中定义。

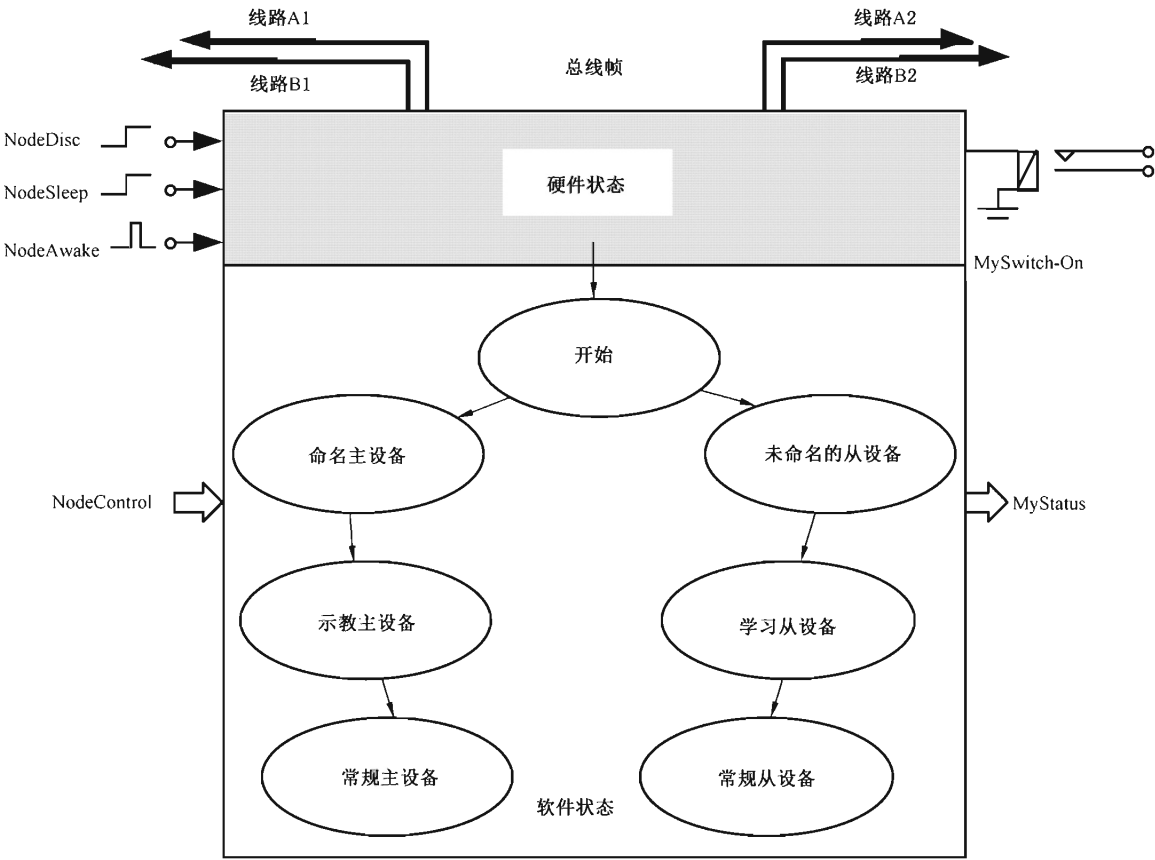


图 84 节点主要状态和应用设置

为节约能量,节点可处于低功耗休眠模式。由于在休眠模式下节点没有充分供电,一些状态以及状态的转移应在 MAU 硬件中实现,这些状态被认为是硬件状态,且相应的控制和状态变量由硬件符号化。其他状态被认为是软件状态。

5.5.4.2 初运行进程结构

在概念上,节点的活动分成 3 个进程,如图 85 所示:

- 一个主进程。主进程在所有已命名节点中都激活。在已命名从设备中,该进程附加在主设备的方向上。在主设备中,该进程指向由主设备命名的第一个从设备。
- 两个相同的辅助进程。每个辅助进程附加在一个方向上。辅助进程只在总线开放末端方向的末端节点上才激活。

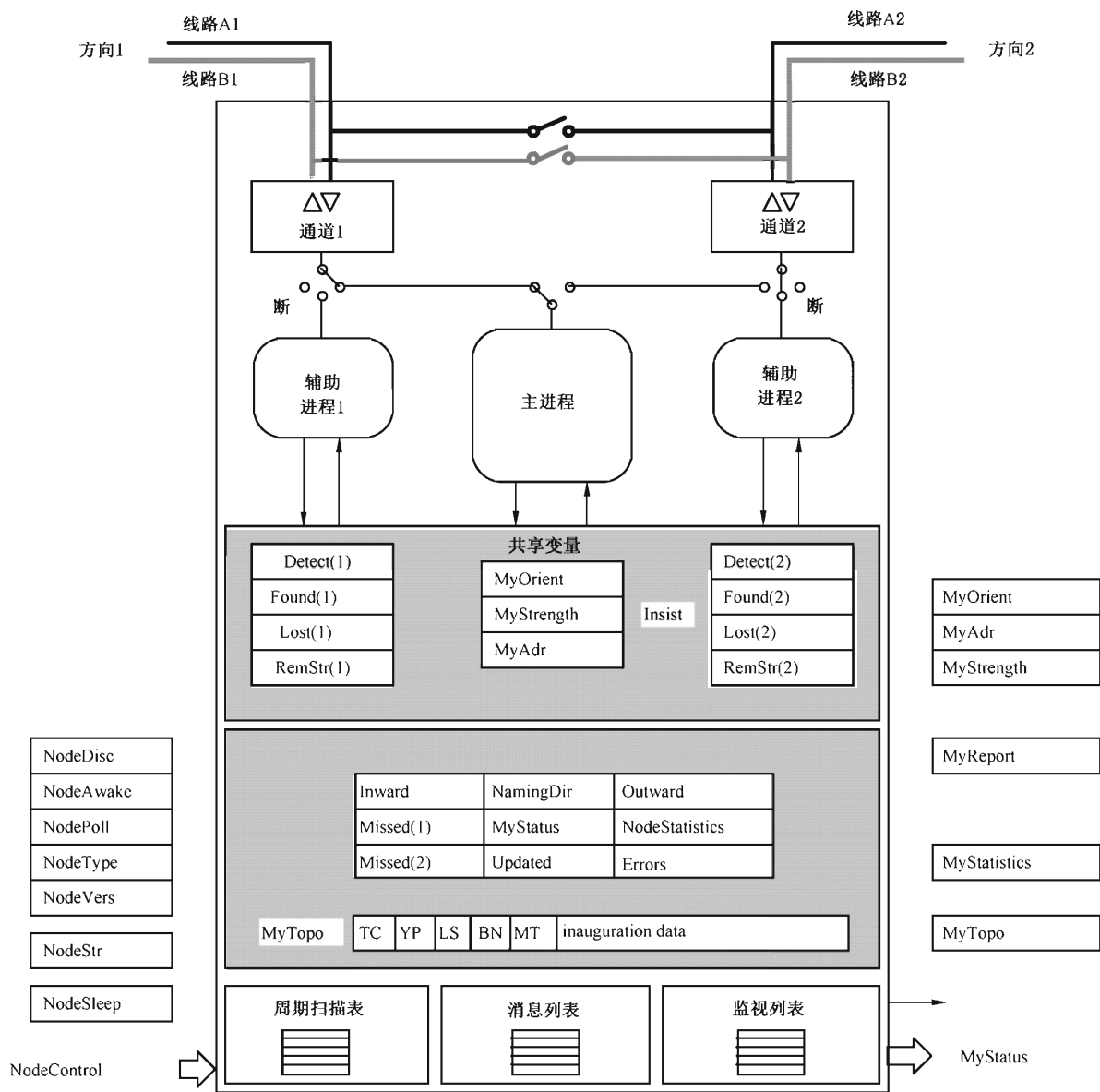


图 85 节点进程(末端设定)

在本部分中,主进程和辅助进程都假定为循环进程,即它们以预先确定的时间间隔(缺省值为每个基本周期)运行且在间隔结束前由自身停止。一个节点的主进程和辅助进程与其他节点的这些进程并行运行。

这些进程通过以下方式相互控制(且控制自己):

- a) 通过在总线上发送帧;
- b) 通过启动超时;
- c) 通过公共(被轮询的)变量。

进程内转移通过以下方式触发:

- a) 通过控制变量;
- b) 通过总线上接收到的帧;
- c) 通过超时。

转移由同一进程或其他进程的公共(被轮询的)变量影响。

节点的状态通过“MyStatus”结构可见。

5.5.4.3 规范语言

初运行进程使用 SDL/GR 语言(ITU-T Z.100)规定。

为简化图表,下列约定适用:

- a) 作为 SDL 的一个限制,没有事件排队,即给定状态下的节点只考虑进程在该状态下发生的事件。
- b) 为简化图表,包含了“IF/ELSE”语句。布尔变量一般采用 YES/NO、“1”/“0”或“TRUE”/“FALSE”。
- c) 状态名或宏名大写。当宏只有一个状态时,该状态与宏同名,但只有一个指向宏的入口。
- d) 对每个状态,都可能有一个与状态同名的定时器与之关联(例如:定时器 T_named_slave 用于 NAMED_SLAVE 状态),且在每次进入状态时重启定时器。
- e) 若没有规定下一个状态,则操作返回到其开始时状态。

5.5.4.4 节点变量

5.5.4.4.1 数据结构 NodeControl

节点应由表 10 规定的 NodeControl 数据结构控制。

表 10 数据结构 NodeControl

变量	类型	含义
NodeAwake	BOOLEAN1	若 NodeSleep 和 NodeInhibit 未置位,则将节点从休眠状态唤醒并使其参与初运行
NodeDisc	BOOLEAN1	将节点设置为中间设定非激活状态;由上电电路清除置位;当供电不足(例如小于标称值 70%)时,通常置位;当节点经历不可恢复故障时,也可置位
NodeSetUp	RECOED	给定节点配置数据的命令,尤其是 NodeDescriptor 和 NodeStrength(从节点/弱节点/强节点)等配置数据
NodeInhibit	BOOLEAN1	禁止节点引起初运行,除非是从失效中恢复
NodeSleep	BOOLEAN1	将节点设置为末端设定低功耗休眠状态,但不阻止节点苏醒若节点检测到活动。当总线宜断电时(例如:电池充电已停止 45 min 以上),通常置位

5.5.4.4.2 数据结构 MyStatus

节点应通过表 11 规定的 MyStatus 数据结构使其状态对应用可用。

表 11 数据结构 MyStatus

变量	类型	含义
MyTopo	RECORD	在最新拓扑发布中接收到的拓扑的当前版本号;该变量是拓扑的拷贝(在所有节点收到相同版本号前可能不一致)
MyOrient	ANTIVALENT2	指示节点指向与主设备同向还是反向。“00”B:出错,“01”B:同向,“10”B:反向,“11”B:未定义

表 11（续）

变量	类型	含义
MyDir	BOOLEAN1	若节点与主设备同向则为 TRUE,通过命名请求帧接收
MyAdr	UNSIGNED6	通过命名请求帧接收到的地址(your_address),或未命名节点的未命名地址,或主设备地址
MyReport	NodeReport	采用节点在状态响应帧(见 5.3.7)中发送给主设备的格式的节点报告
MyStrength	Composition_Strength	节点从辅助通道上命名请求帧,或主通道上末端设定请求帧,状态请求帧或拓扑请求帧接收到的节点本地组成强度
MySwitchOn	BOOLEAN1	若节点将上电则为“1”,节点向低功耗模式切换则为“0”
MyStatistics	RECORD	发送和接收的帧的统计信息及错误计数

5.5.4.4.3 共享变量

主进程与辅助进程通过共享变量交换信息。

假定这些变量被各自进程轮询,即在主进程与辅助进程之间没有异步信号。

因为这些变量可同时被多个进程访问,变量可由锁定/解锁(LOCK/UNLOCK)以便坚固地读取和修改(例如:避免两个方向同时命名一个未命名节点)。

标明(1,2)的变量专属于每个辅助进程。

MyStatus 数据结构中所有变量都是共享变量。

其他共享变量在表 12 中规定。

表 12 节点共享变量

变量	类型	含义
Detect(1) Detect(2)	BOOLEAN1 BOOLEAN1	方向 1 和/或方向 2 使能检测。“1”使能,“0”禁止
Found(1) Found(2)	BOOLEAN1 BOOLEAN1	由检测到另一个节点的辅助进程或由远端组成强度(RemSer)不为 0 的存在响应帧置位;当该方向节点已命名、设置为中间节点状态或变为未命名时复位
Lost(1) Lost(2)	INTEGER8 INTEGER8	检测已检测到但仍未命名的节点丢失的计数器
Insist	BOOLEAN1	若末端节点在两个方向都坚持则为“1”,若任一方向检测到更强组成则为“0”
LocStr	Composition_Strength	节点所属组成的强度
RemStr(1) RemStr(2)	Composition_Strength Composition_Strength	由自身辅助进程检测到的或从监视帧远端强度字段里接收到的另一组成强度

5.5.4.4.4 主进程变量

表 13 所列的变量用于主进程。其他本地变量出现在相应宏或过程的 SDL 图表中。

表 13 主进程变量

变量	类型	含义
Topo	RECORD	拓扑,每个已命名节点一项
TopoCounter	Topo_Counter	见 5.5.2.7
MasterTopo	Master_Topo	见 5.5.2.8
Inward	ANTIVALENT2	主设备方向(对于已命名从节点)
Outward	ANTIVALENT2	背离主设备方向(对于已命名从节点)
NamingDir	ANTIVALENT2	主设备命名方向(0:没有,1:底,2:顶)
Missed(1) Missed(2)	UNSIGNED8 UNSIGNED8	检测每个方向末端节点的丢失
Errors	UNSIGNED8	错误计数器,当发现不利情况时增加计数,当该发现输出有利时清零,因此对各种发现作了隐含说明
C_bit	BOOLEAN1	只要该位置位,所有过程数据响应帧或消息数据响应帧将使 C 位置位

主设备运行依赖表 14 中规定的列表。

表 14 主进程列表

变量	类型	含义
Periodic_List[n]	RECORD	在一个特定基本周期中为过程数据而轮询的地址列表,n 表示基本周期在宏周期中的位置
Message_List	RECORD	为消息数据而轮询的节点列表,按检测到的传送请求(由 A 位指示)的顺序排列
Supervisory_List	RECORD	为监视数据而轮询的节点列表,按检测到的传送请求(由 C 位指示)的顺序排列

5.5.4.5 辅助进程

5.5.4.5.1 进程

如图 86 所示,辅助进程检测不属于组成的节点的存在性。只有末端设定状态的节点才执行辅助进程。独立节点有两个激活的辅助进程。帧交换在辅助通道上完成。

辅助进程由两种状态组成:一种是“检测响应”(DETECTING_RESPONSE)状态,只有节点已命名且不坚持时才进入该状态;另一种是“检测请求”(DETECTING_REQUESTS)状态,只要辅助进程激活即进入该状态。

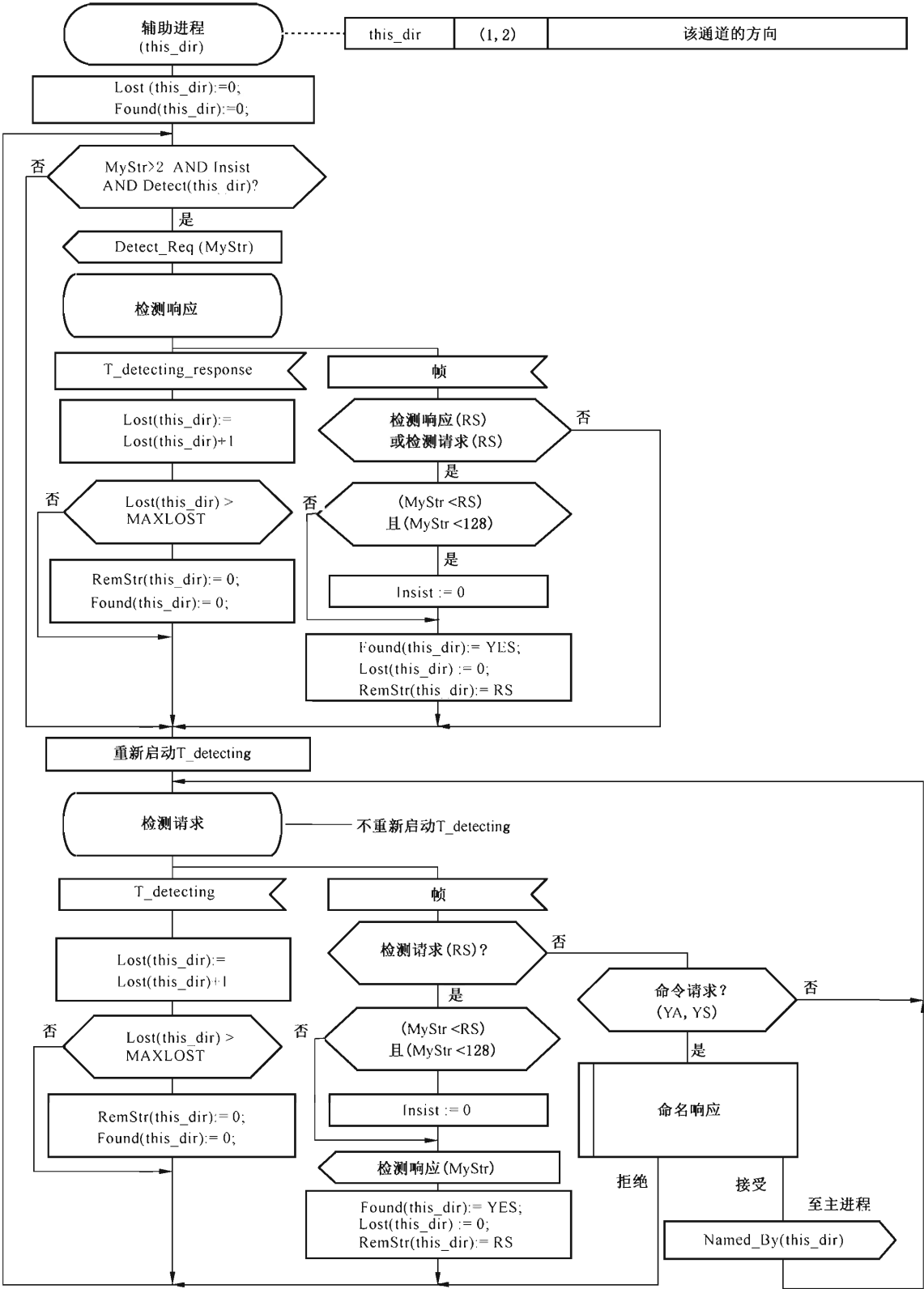


图 86 辅助进程状态

5.5.4.5.2 “检测响应”状态

处于“检测响应”状态不放弃的已命名节点应发送检测请求帧,期待检测响应帧或检测请求帧。

处于“检测响应”状态的节点应期待:

- a) 检测响应超时 $T_{\text{detecting_response}}$,进入“检测请求”状态。
- b) 检测响应帧或检测请求帧:
 - 记录远端组成的存在和强度($\text{Found}(\text{this_dir}) = \text{"1"}; \text{Lost} = 0$);
 - 比较彼此强度,若另一组成较强则放弃,“insist”= 0;
 - 进入“检测请求”状态。
- c) 其他,忽略且进入检测请求状态。

注 1: 检测请求帧是由从节点发送的唯一主帧。

注 2: 一些情况可导致节点无条件切除。

5.5.4.5.3 “检测请求”状态

辅助进程在检测请求状态时应期待:

- a) 检测超时 $T_{\text{detecting}}$:
 - 若节点在 $T_{\text{detecting}}$ 期间没有收到帧,则应递增该方向的“Lost”计数器,并进入“检测响应”状态;
 - 若节点在 MAXLOST 个连续检测周期内没有接收到任何帧,则应复位变量“Found”,置“ $\text{RemStr}(\text{this_dir})$ ”为 0,并进入“检测响应”状态。
- b) 检测请求帧:
 - 记录另一个节点的存在($\text{Found}(\text{this_dir}) = \text{"1"}; \text{Lost} = \text{"0"}$);
 - 比较彼此强度,若请求者具有较大或相同的强度且自身不是强主则放弃,置“Insist”为 0;
 - 发送检测响应帧。
- c) 命名请求帧:
 - 若辅助进程先前没有收到一个较强组成的检测请求帧, (“ $\text{Found}(1) = \text{"0"}$ ”且“ $\text{Found}(2) = \text{"0"}$ ”),则应声明接收到命名请求帧的线路受扰,并信任另一线路即使另一线路已受扰;
 - 否则,应执行“命名响应”(NAMING_RESPONSE)宏,若成功执行则终止辅助进程。
- d) 其他类型帧,声明接收到此帧的线路受扰,信任另一线路即使另一线路已受扰,不复位 $T_{\text{detecting}}$,并进入“检测请求”状态。

注 1: 节点期望通过辅助通道接收的帧仅有检测请求帧和命名请求帧。若节点没有事先收到检测请求帧就收到命名请求帧,则解释为协议错误。

注 2: 由于检测请求帧可能因冲突而导致误码,为避免重复冲突,检测超时 $T_{\text{detecting}}$ 是一个在均值前后的随机数。

5.5.4.5.4 “命名响应”宏

如图 87 所示,“命名响应”宏为节点分配地址:

- 若节点已命名,则应忽略命名请求帧。只有当节点已通过另一通道命名时才会发生这种情况。
- 若节点未命名,则应以命名响应帧响应。节点应停止该方向上的检测,并将通道分配给其主通道。

注: 检测节点是否未命名要求对 MyAdr 进行排他的读操作,因为另一辅助进程也可能访问。

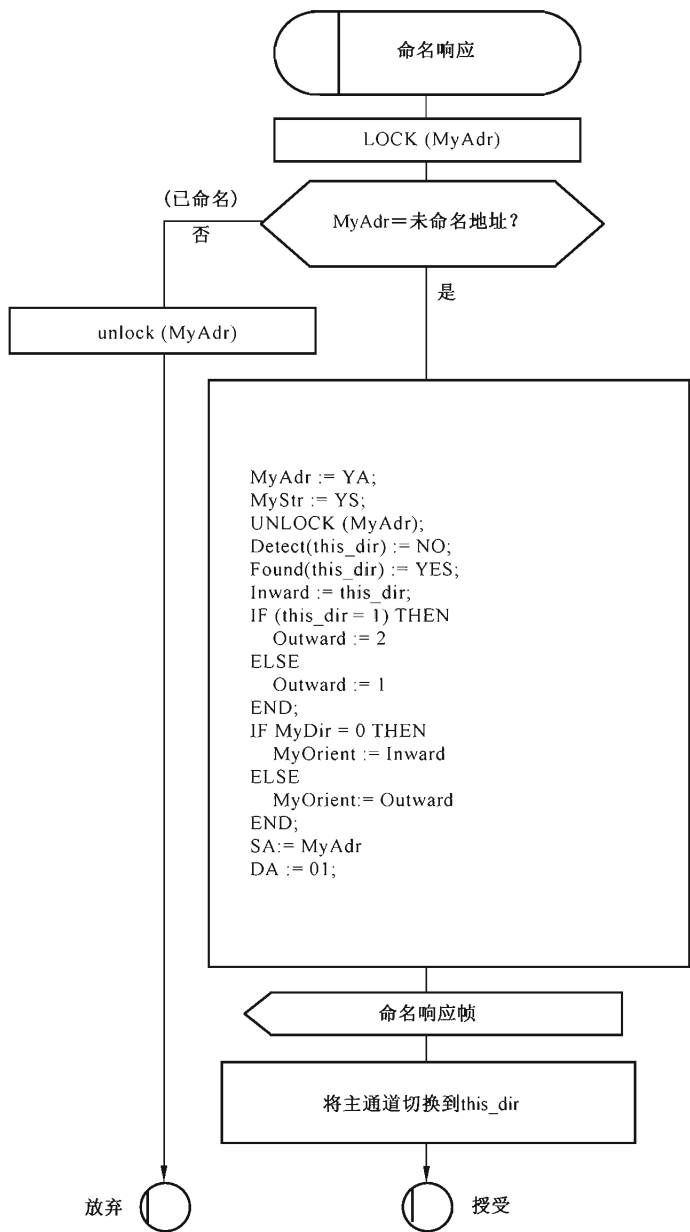


图 87 命名响应宏

5.5.4.6 主进程主要状态

如图 88 所示,主进程分为一些主要状态。
这些状态细分为多个小状态。小状态由宏(只出现一次)或过程(可出现多次)表示,并在以下章条中描述。

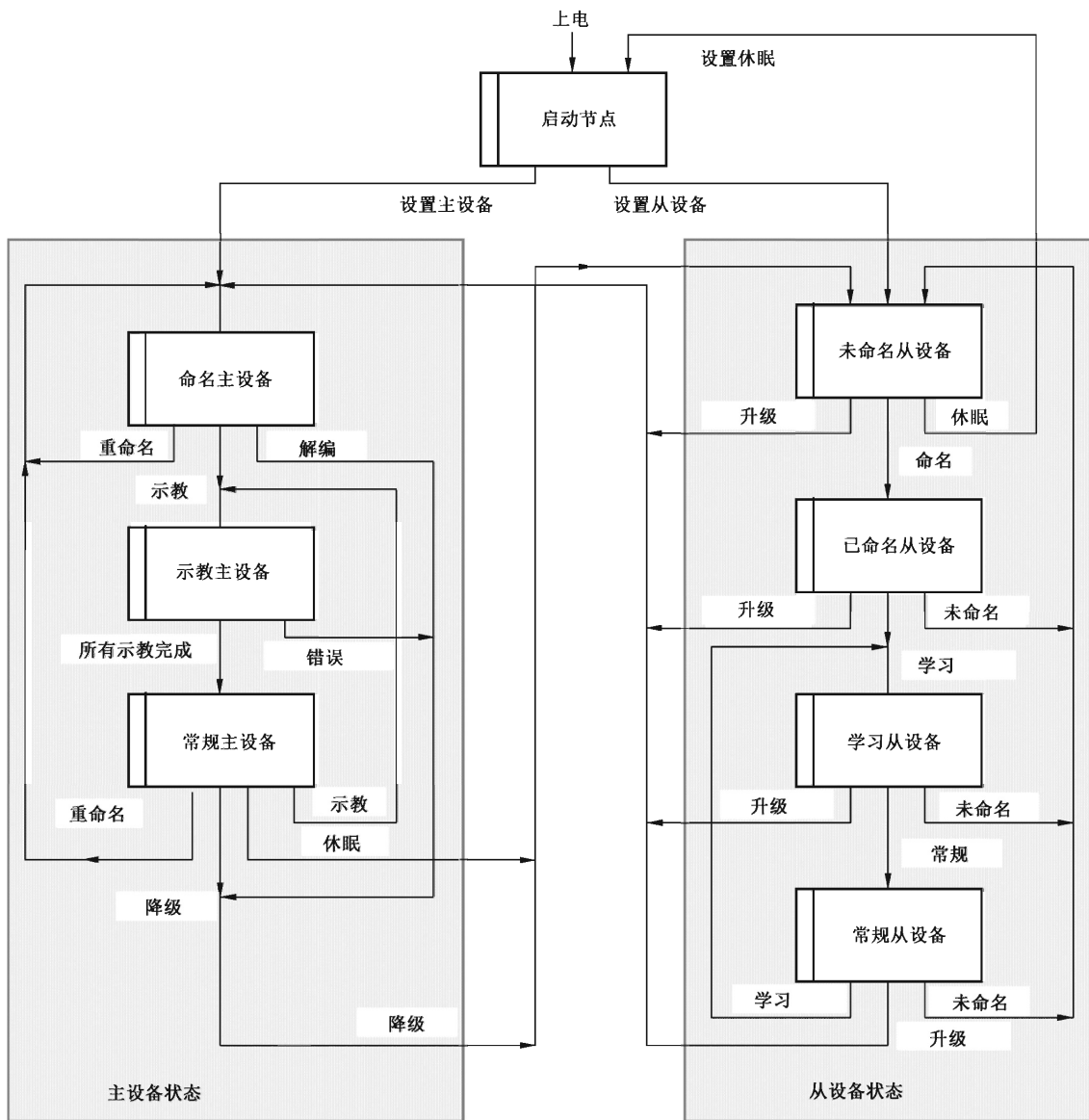


图 88 主进程状态

“启动节点”(START_NODE)状态在表 15 中描述。

表 15 启动节点状态

启动节点	<p>节点在失效、复位或上电后进入“启动节点”状态。</p> <p>节点从节点无源的断电状态启动,或从节点等待总线或应用指示恢复满功耗的低功耗状态启动。</p> <p>然后应用程序通过 NodeSetup 命令配置节点。节点若配置为强节点,则立即进入命名主设备状态;若配置为从节点或弱节点,则进入未命名从设备状态</p>
------	--

节点作为主设备工作的状态在表 16 中列出。

表 16 主设备状态

命名主设备	<p>初始化节点为孤立主设备。设置节点为末端设定状态,两个检测通道均激活。在两个方向上开始发送检测帧以搜索其他节点。</p> <p>当主设备检测到节点时,命名节点并更新其节点状态列表。当在两个方向上均不再检测到可命名节点时,主设备结束命名并进入示教主设备状态。</p> <p>若弱主发现更强组成,则该弱主降级并返回到未命名从设备状态。在此之前,应消名其命名过的节点(解散组成)。</p> <p>若主设备发现不可恢复的错误,则通过返回到命名主设备状态重新开始过程</p>
示教主设备	<p>主设备通过逐一请求每个节点分发拓扑,向所有其他节点广播其节点描述符和初运行数据。</p> <p>若分发成功,主设备进入常规主设备状态。</p> <p>若分发失败,主设备解散组成并返回命名主设备状态</p>
常规主设备	<p>在常规运行时,主设备轮询节点。</p> <p>主设备处理以下例外:</p> <ul style="list-style-type: none">——任意节点描述符或状态改变;——主设备强度改变。 <p>在以下场景,主设备退出常规主设备状态:</p> <ul style="list-style-type: none">——总线收缩(不再检测到末端节点);——总线延伸(命名附加节点)

节点作为从设备工作的状态在表 17 中列出。

表 17 从设备状态

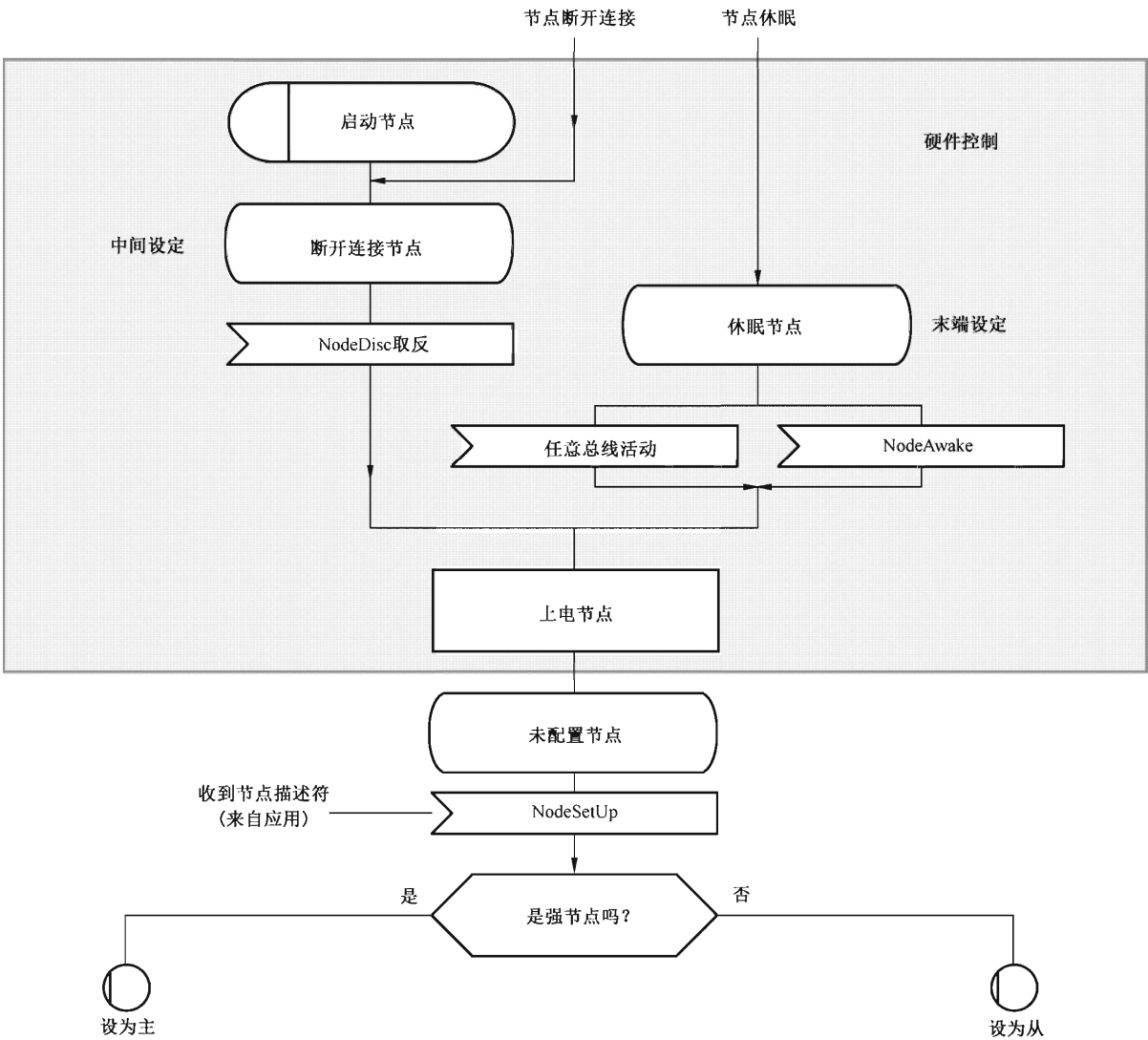
未命名从设备	<p>未命名从设备状态是节点在末端设定下,两个辅助进程均侦听(但不发送)。当接收到主设备命名帧或由应用提升为强节点或弱节点时,退出此状态</p>
已命名从设备	<p>命名请求帧导致节点将其主通道转向主设备。节点在另一个方向上的辅助进程仍激活。主设备定期发送状态请求帧以检测其开放末端是否有更多节点。</p> <p>一旦检测到未命名节点,主设备将本节点切换成中间设定状态,关闭其辅助进程。</p> <p>当节点接收到指示命名过程结束的拓扑请求帧或拓扑响应帧时,退出此状态</p>
学习从设备	<p>已命名从节点接收所有其他节点的初运行数据,并向所有其他节点广播其初运行数据。节点将其初运行计数器加 1。</p> <p>当接收到主设备指示进入常规运行的存在请求帧或指示常规运行的任意帧时,节点退出此状态</p>
常规从设备	<p>在此状态,假定节点已更新完整拓扑。</p> <p>若是末端节点,节点激活的辅助进程检测总线延伸,并通过存在响应帧向主设备报告。</p> <p>所有节点监视两个末端节点是否存在;若连续 3 个存在响应帧没有感知到末端节点,则节点返回到未命名从设备状态。</p> <p>若节点已经更新,则期望定期轮询其过程数据并接收其他节点的过程数据。</p> <p>若接收到给自己的主设备指示组成改变而地址未改变的拓扑响应帧或拓扑请求帧,则节点返回到学习从设备状态</p>

在所有从设备状态中,从设备可被提升为强主并直接进入命名主设备状态。

5.5.4.7 “启动节点”宏

5.5.4.7.1 概述

如图 89 所示,启动节点(START_NODE)宏定义节点在成为主设备或从设备之前经历的由硬件和软件控制的状态。



5.5.4.7.2 “断开连接节点”状态

当应用程序置位 NodeDisc 指示节点将掉电或经历了严重破坏时,节点应无条件进入“断开连接节点”(DISCONNECTED_NODE)状态。

此状态是未上电节点的初始状态。在此状态,节点应处于中间设定模式。

当应用上电节点且复位 NodeDisc 信号时,节点应退出此状态。然后节点应进入未配置节点状态。

5.5.4.7.3 “休眠节点”状态

当 NodeSleep 命令置位且 NodeDisc 未置位时,节点应进入“休眠节点”(SLEEPING_NODE)状态。

节点应处于末端设定模式低功耗状态。
以下情况节点应退出此状态：

- a) 当应用置位 NodeAwake 命令时；
- b) 当节点检测到总线活动时。

当退出休眠状态时，节点应置位 MySwitchOn 以全功耗供电节点，且进入未配置节点状态。除了“断开连接节点”状态和“休眠节点”状态外，其他所有状态下 MySwitchOn 信号应由硬件置位。

- 注 1：进入此状态将中断总线。若总线上有活动，节点将被唤醒。因此，应用维持 NodeSleep 信号足够长的时间，以避免另一节点再将此节点唤醒。
- 注 2：总线开关是带电的，因为若完全断电将使节点进入中间设定模式。
- 注 3：总线活动由不带 SQE 信号的载波检测（见 4.8.1.5）定义，或由任何其他指示在任一通道上接收到波形完好的帧的方法定义。

5.5.4.7.4 “未配置节点”状态

“未配置节点”(UNCONFIGURED_NODE)状态下，节点应处于末端设定模式，并等待其节点描述符和初运行数据。

在接收到带有有效节点描述符的 NodeSetUp 命令后，节点应退出此状态。

若是强节点，节点应进入命名主设备宏；若是弱节点或从节点，节点应进入未命名从设备宏。

5.5.4.8 主设备状态

5.5.4.8.1 “请求响应”过程

如图 90 所示，请求响应(REQUEST_RESPONSE)过程发送请求帧并等待相应的响应帧。

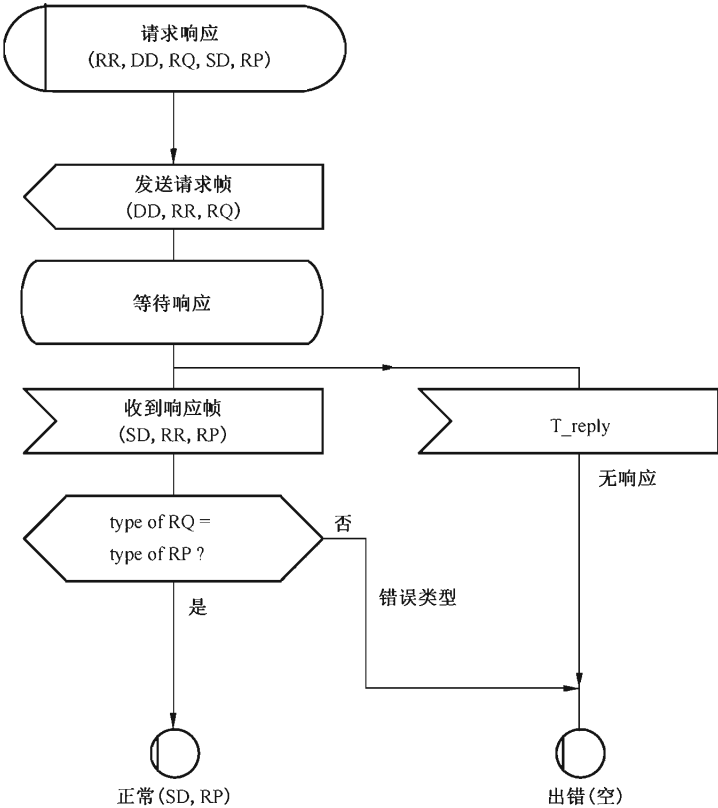


图 90 “请求响应”过程

参数包括：

- a) RR: 请求/响应类型, 为{存在、状态、命名、拓扑}等类型之一；
- b) DD: 目的设备；
- c) SD: 源设备；
- d) RP: 响应参数(依赖于 RR, 根据不同类型的帧定义)；
- e) RQ: 请求参数(依赖于 RR, 根据不同类型的帧定义)。

在下列条件, 此过程退出：

- a) 从设备进行了响应；
- b) 在超时 T_reply 之后。

5.5.4.8.2 “末端设定”和“中间设定”过程

如图 91 所示, 若节点不处于相应状态, 则这两个过程分别将节点设置为末端设定状态和中间设定状态。

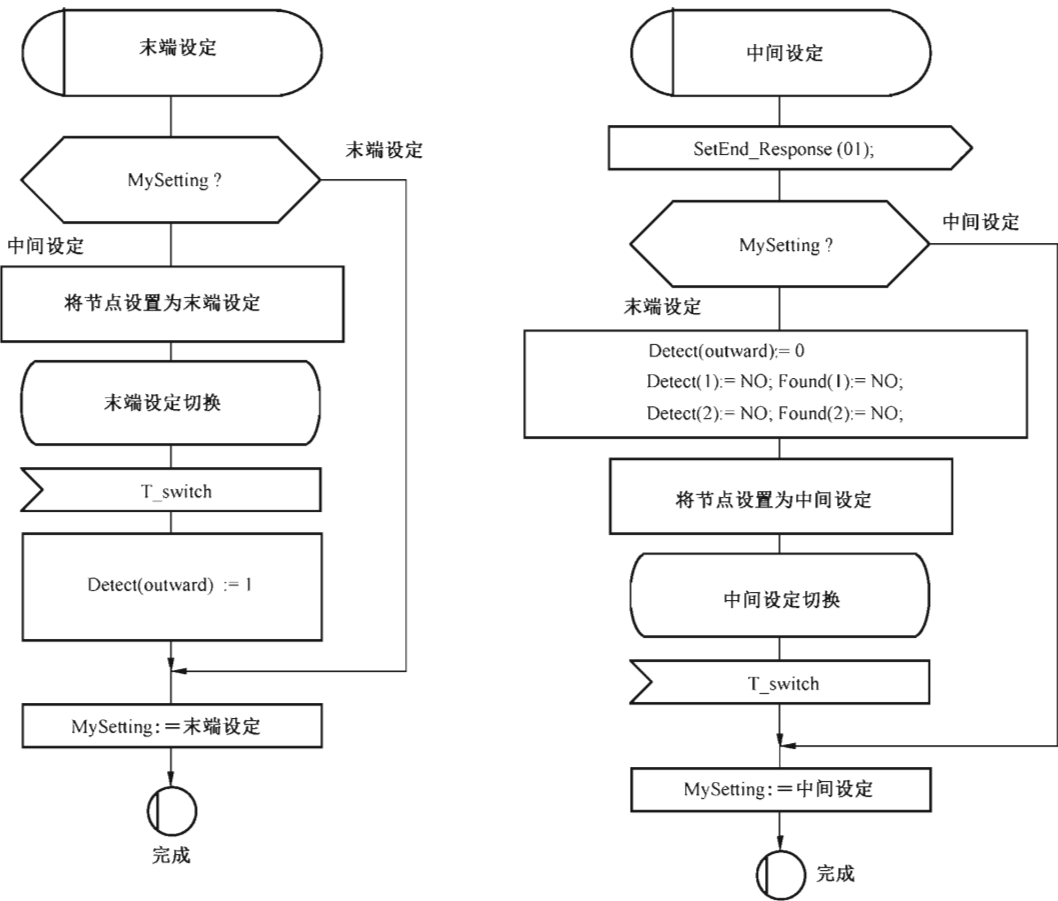


图 91 “末端设定 (SET_TO_END)”和“中间设定 (SET_TO_INT)”过程

变量 Detect(1)或 Detect(2)分别控制两个辅助进程的启停。

若将被设置为末端模式的节点本身是主设备, 或节点正在消名, 则 Detect(1)和 Detect(2)将都为“1”。

5.5.4.8.3 “初始化主设备”宏

如图 92 所示, 进入“初始化主设备”(INIT_MASTER)宏将节点配置为主设备(从节点升级为强主

或由于没有总线活动而使未命名从节点成为主设备的结果)。

节点首先将自己初始化为末端设定模式(若还不是),设置其地址和强度,设置空拓扑,使能两个检测通道并开始发送检测请求帧。在此配置中,主设备按轮询自身一样动作。

设置结束后,主设备已准备好命名其可能发现的其他节点。

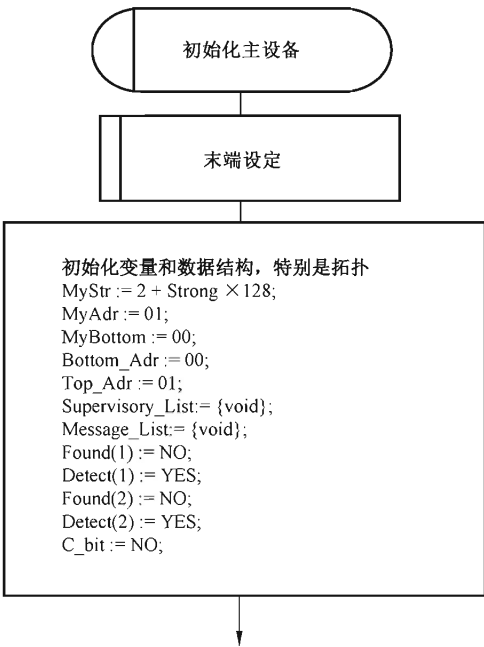


图 92 “初始化主设备”宏

5.5.4.8.4 “命名主设备”宏

如图 93 所示,命名主设备(NAMING_MASTER)宏包含主设备命名其他节点时的各种状态。主设备通过执行初始化主设备宏进入此状态。

主设备应在 AWAIT_PERIOD 状态等待 T_naming_master 周期的开始。若 T_naming_master 延时已经结束,则主设备立即进行下一步。

主设备应执行“末端询问”宏询问末端节点是否发现了另一组成的存在,并根据末端询问宏的输出相应动作:

- a) 若检测到的组成较强,主设备应进入“消名主设备”宏解散当前组成。
- b) 若检测到的组成较弱或与当前组成强度相同,主设备应等待检测到的另一组成来解散(此过程需要多长时间不能确定)。
- c) 若末端节点没有响应,主设备应记录错误并稍后重试。在同一方向上连续 3 个错误应导致主设备通过执行“消名主设备”宏解散当前组成。
- d) 若末端节点报告一个未命名节点,主设备应将该节点命名为新的末端节点。
- e) 若在同一方向上连续 3 次轮询没有检测到可命名节点,主设备应进入“示教主设备”状态。

注 1: 主设备每个 T_naming_master 周期命名一个节点。

注 2: 在命名主设备状态中,总是允许初运行。

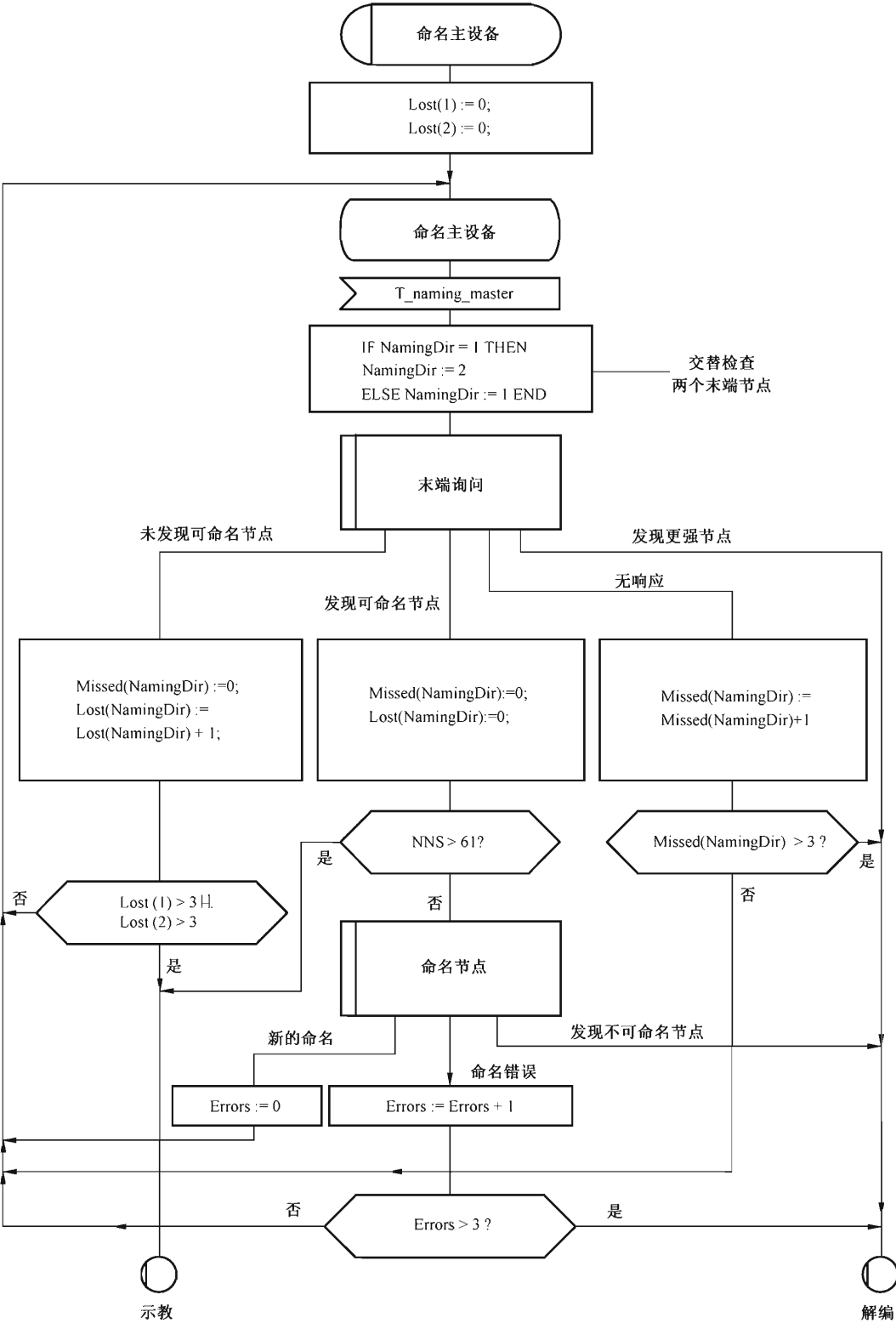


图 93 “命名主设备”宏

5.5.4.8.5 “末端询问”宏

如图 94 所示,“末端询问”(ASK_END)宏检查在“NamingDir”方向上末端节点的存在,并请求末端

节点报告可能的远端组成。这隐式包含了主设备本身是末端节点的情况。

根据检测到的远端组成强度,主设备区分 3 种情况:

- a) 没有发现可命名节点(没有节点或节点已命名);
- b) 发现未命名节点(远端节点将接受命名);
- c) 发现更强的组成(节点)(远端节点属于更强组成)。

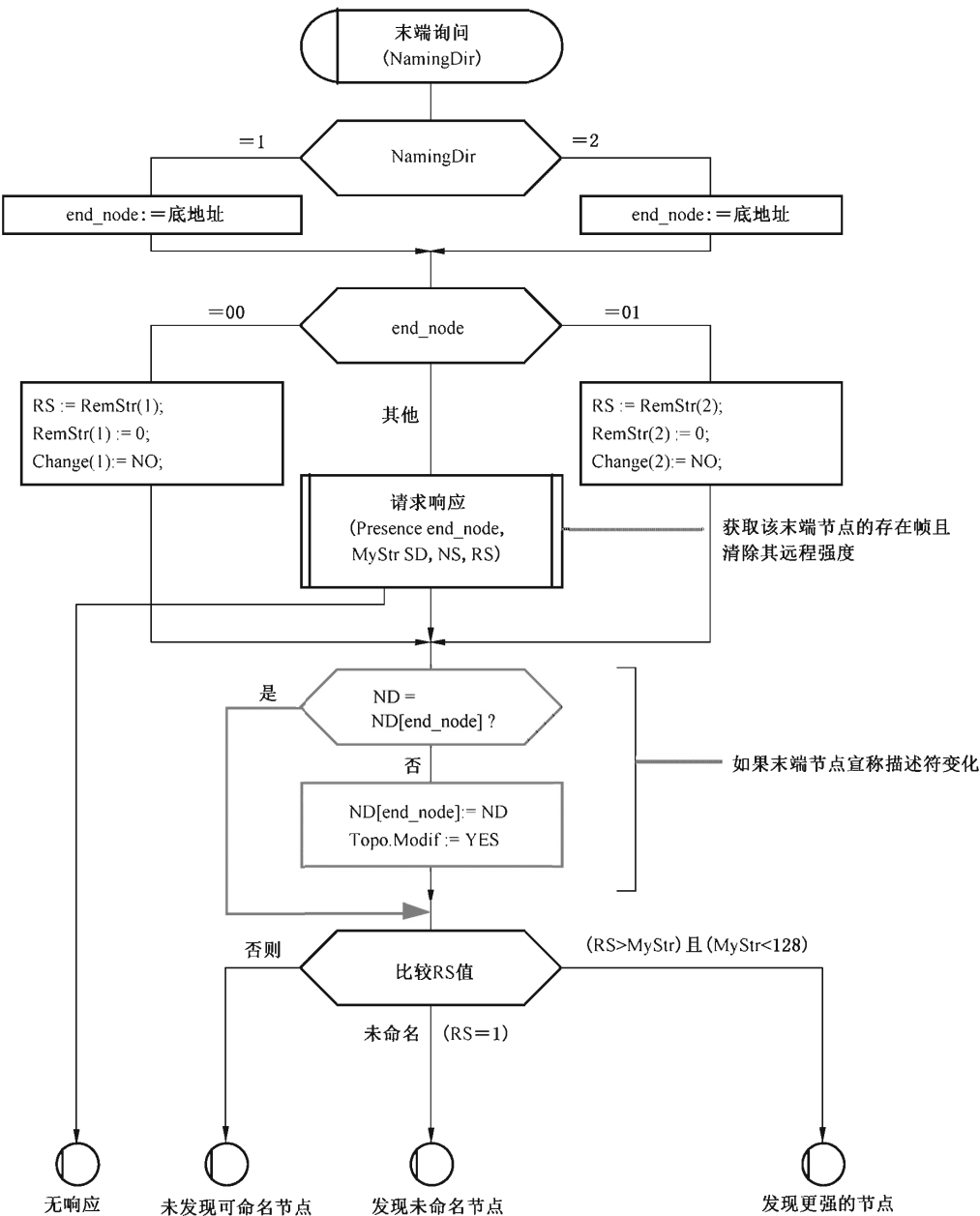


图 94 “末端询问”宏

5.5.4.8.6 “命名节点”过程

如图 95 所示,当末端节点指示存在未命名节点时,调用“命名主设备”宏的“命名节点”(NAME_ONE)过程。

该过程带有一个参数:命名方向。

若主设备本身是末端节点且已命名另一侧的节点,则应进入中间设定模式而不发送命令到总线上。

否则,主设备应保持在末端设定模式,并发送中间设定请求帧给在命名方向上的末端节点。

若没有收到中间设定响应帧,主设备应等待 T_{switch} 并重复发送中间设定请求帧。

若 3 次尝试后没有收到中间设定响应帧,主设备应退出“命名节点”过程,并报告命名错误。

否则,在接收到中间设定响应帧后,主设备应等待延时 T_{switch} ,并按如下协议给未命名节点发送命名请求帧,包括节点地址(your_address)和组成强度(your_strength):

- a) 主设备第一次尝试时应发送目的设备地址为未命名地址的命名请求帧;
- b) 若第一次尝试没有接收到命名响应帧,主设备应等待 $T_{\text{aux_main}}$,并再次发送命名请求帧,其目的设备地址为已分配的节点地址;
- c) 若第二次尝试没有接收到命名响应帧,主设备应再次发送命名请求帧,其目的设备地址为未命名地址;
- d) 若第三次尝试没有接收到命名响应帧,主设备应等待 $T_{\text{aux_main}}$,并再次发送命名请求帧,其目的设备地址为已分配的节点地址;
- e) 若第四次尝试在 T_{reply} 时间内没有接收到命名响应帧,主设备应再次发送命名请求帧,其目的设备地址为未命名地址;
- f) 若第五次尝试没有接收到命名响应帧,主设备应等待 $T_{\text{aux_main}}$,并再次发送命名请求帧,其目的设备地址为已分配的节点地址;
- g) 若第六次尝试在 T_{reply} 时间内没有接收到命名响应帧,主设备应通过发送末端设定请求帧将先前末端节点恢复为末端设定模式,并等待 T_{switch} 以断开开关,然后退出“命名节点”过程,并报告状态“ unnameable_found ”。

否则,若命名成功,主设备应更新拓扑、末端节点地址和组成强度,并应等待 $T_{\text{aux_main}}$,然后向最新命名的节点发送状态请求帧。

若仅在信任线接收到状态响应帧而监视线受扰,主设备应等待 1.2 ms 后重复发送状态请求帧(并等待状态响应帧),重复发送 3 次状态请求帧以评估线路质量,并在其下一个主设备报告中报告结果。

若 3 次发送状态请求帧都没有接收到响应,则主设备应退出“命名节点”过程,并报告“ unnameable_found ”状态。

否则,若接收到状态响应帧,主设备应确定拓扑、末端节点地址及组成强度,并退出“命名节点”过程,报告“ new_named ”。

注 1: 未命名节点期望通过其辅助通道接收命名请求帧和发送命名响应帧。

注 2: 延时 $T_{\text{aux_main}}$ 允许从辅助通道切换到主通道。在此时间内,总线通信被中断。

注 3: 新命名节点通过以命名响应帧响应接受命名,或通过不响应拒绝命名。

注 4: 若节点已命名但响应丢失,则使目的设备地址为未命名地址的尝试将失败。相反,若命名未发生,则使目的设备地址为已分配的节点地址的尝试将失败。

注 5: 已命名节点返回状态响应帧,包含其节点描述符和其已检测到的节点的远端强度。

注 6: 在存在干扰时,3 次重发状态请求帧允许主设备区分帧丢失和主设备与末端节点间冗余线路丢失。

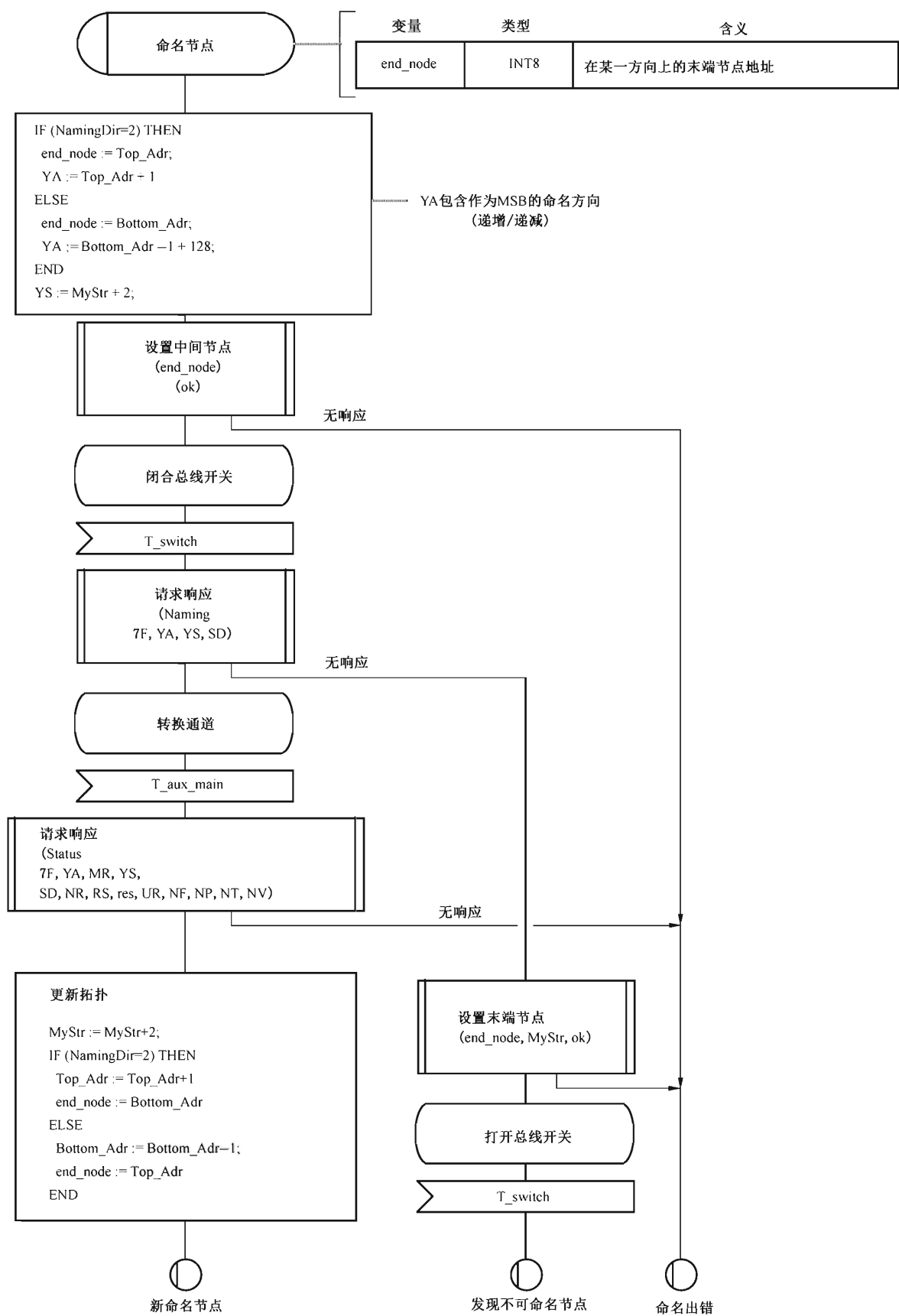


图 95 “命名节点”过程

5.5.4.8.7 “示教主设备”宏

如图 96 所示,当主设备不再检测到可命名节点或检测到在其节点描述符指示变化的节点时,主设备分发拓扑信息给所有已命名的节点。

为此,主设备应增加 Topo_Counter 和 Master_Topo。

节点应按地址升序,从底节点开始,到顶节点结束,包括自身(自轮询),向所有已命名节点连续发送 3 次拓扑请求帧。

在以下情况,节点应退出“示教主设备”(TEACHING_MASTER)状态:

- 当连续 3 次发送拓扑请求帧之后没有接收到拓扑响应帧时,应进入“消名主设备”宏;
- 当从所有节点接收到拓扑响应帧时,应进入“常规主设备”宏。

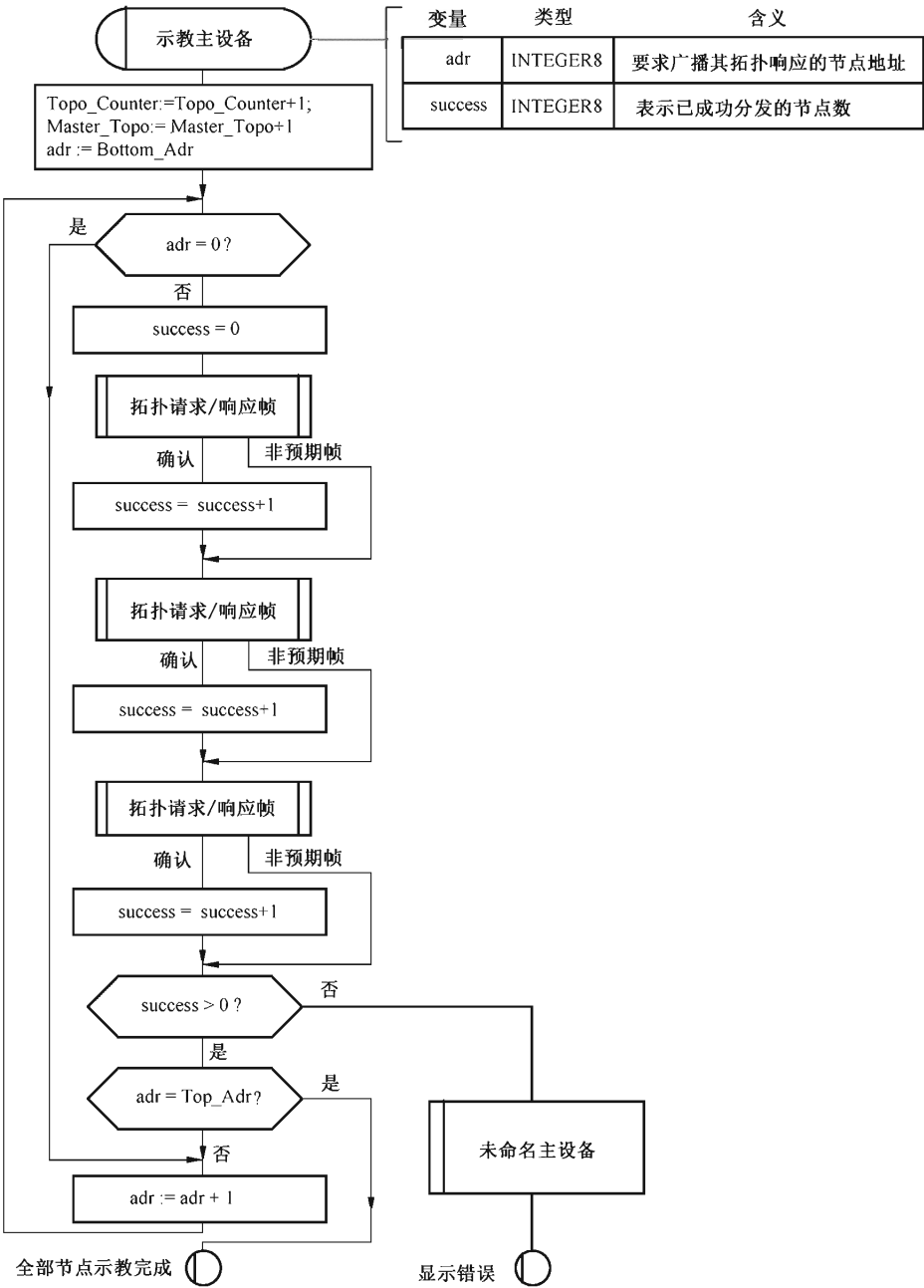


图 96 “示教主设备”宏

5.5.4.8.8 “消名主设备”宏

如图 97 所示,主设备通过消名所有节点解散其组成。为此,主设备应在 1.0 ms 内连续 3 次广播消名请求帧。然后,若是强节点则进入“命名主设备”宏,否则进入“未命名从设备”宏。

注:为保证所有节点接收到消名请求帧,应有延时。从设备在进入末端设定模式前也有相同延时。



图 97 “消名主设备”(UNNAMING_MASTER)宏

5.5.4.8.9 “常规主设备”宏

如图 98 所示,主设备处于常规运行状态。

“常规主设备”(REGULAR_MASTER)宏分为 3 个功能块:

- a) 周期轮询:主设备轮询其周期扫描表中的节点以获取过程数据。此阶段在 5.5.4.8.11 中详细描述。
 - b) 监督轮询:
 - 在每个基本周期,主设备应给一个末端节点发送存在请求帧;在同一方向上任意 4 个连续的存在请求帧之间的间隔时间应小于 $6.5 \times T_{bp}$ 。为达到这个要求,一个便捷的方法是在基本周期的固定比例处发送存在请求帧,例如在每个基本周期的开始处发送。
 - 主设备检测 C 位置位的节点状态。此检测应在周期相之后进行。
 - c) 消息轮询:主设备在下一个周期相开始之前剩余时间内轮询节点以获取消息数据,见 5.5.4.8.12。
- 在下列情况,主设备应退出“常规主设备”宏:
- a) 若连续 3 次轮询主设备未感知末端节点存在,则解散其组成,并进入“命名主设备”宏;
 - b) 若检测到另一个可命名组成的存在且允许初运行,则主设备执行“消名主设备”宏以解散其组成,然后进入“命名主设备”宏;

- c) 若检测到组成的变化,则进入“示教主设备”宏;
 - d) 若发现比自身强的组成且允许初运行,则主设备在进入“未命名从设备”宏之前解散其组成;
 - e) 若已经由应用命令设置为休眠模式或被解联,则进入“未命名从设备”宏。
- 注:当所有其他节点也处于未命名从设备状态时,“未命名从设备”宏使节点进入休眠模式。

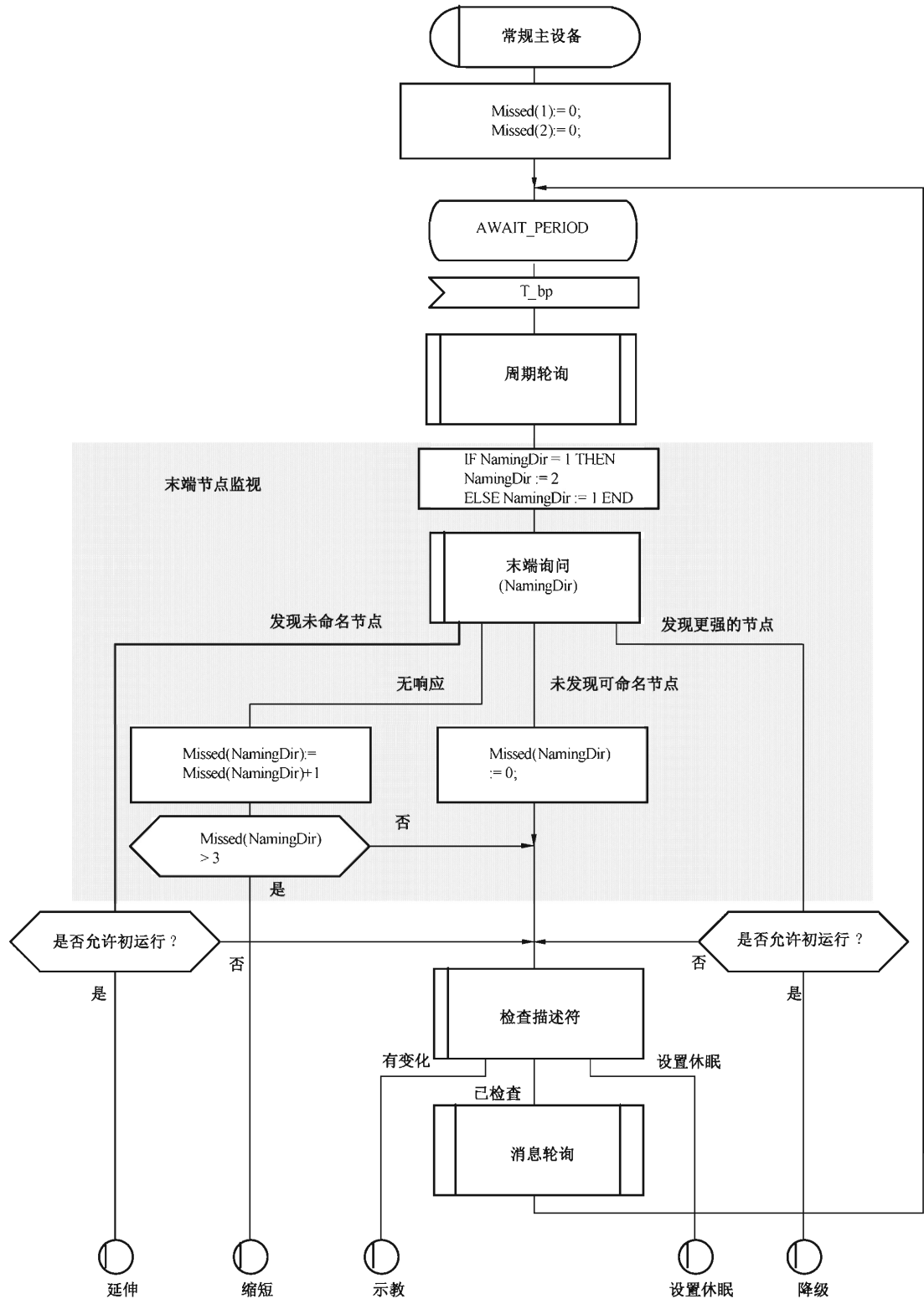


图 98 “常规主设备”宏

5.5.4.8.10 “检查描述符”宏

如图 99 所示,“检查描述符”(CHECK_DESC)宏检查请求改变描述符或请求设置为休眠模式的中间节点。

末端节点的情况单独处理,因为末端节点还可能附带指示总线延伸。

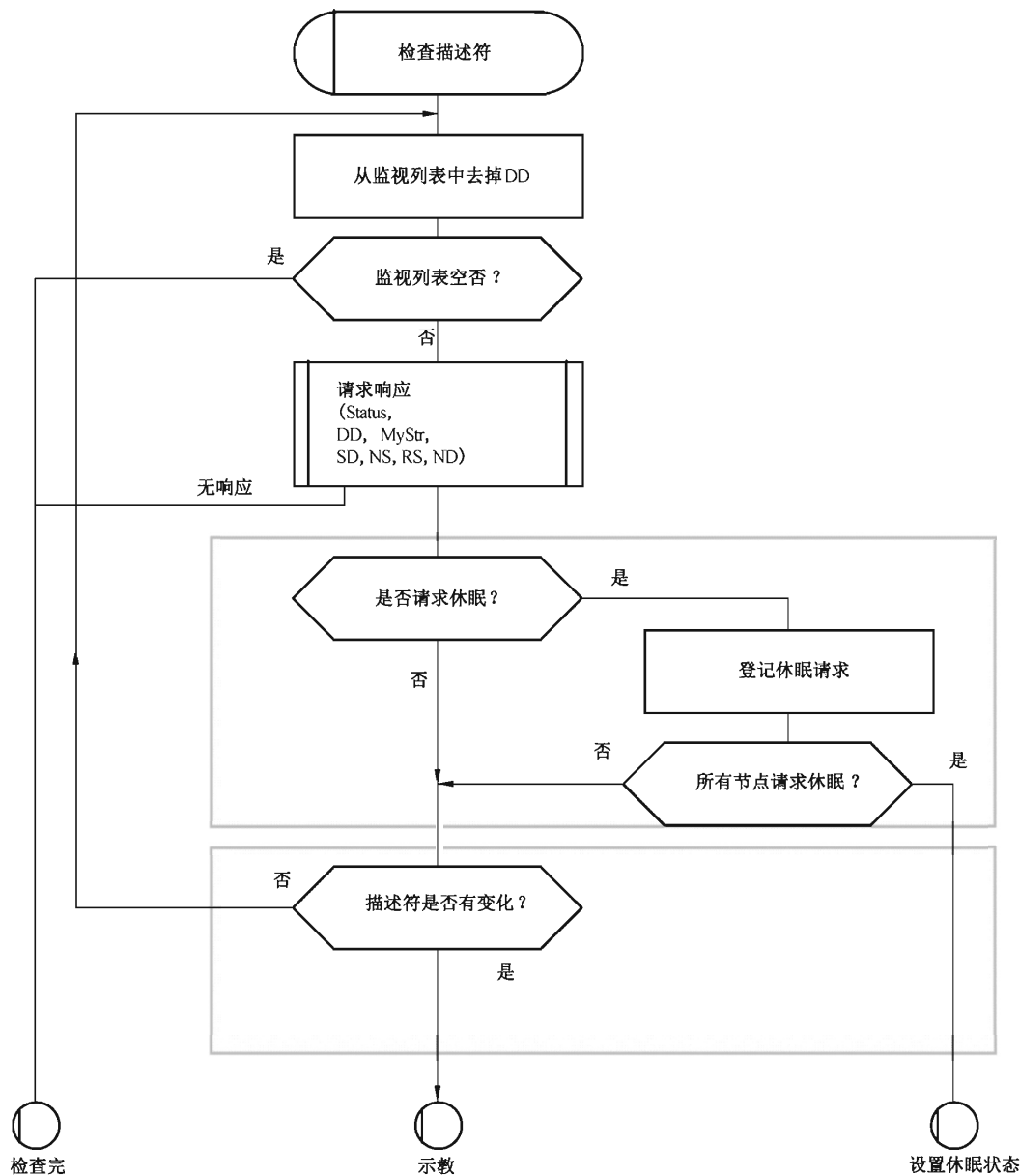


图 99 “检查描述符”宏

5.5.4.8.11 “周期轮询”宏

如图 100 所示,主设备根据周期扫描表扫描节点,周期扫描表列出了在此周期中要轮询的地址。当轮询节点的过程数据时,主设备应:

- a) 记录被轮询节点的存在性及其过程数据;
- b) 记录因节点描述符的改变而导致的组成改变(“C 位”);
- c) 记录通过置位“A 位”请求消息数据传送的节点;

- d) 记录通过置位“**I** 位”禁止初运行的节点。
- 若没有接收到过程数据响应帧,主设备应在同一基本周期内不重复过程数据请求帧。
- 当自轮询时,主设备应在发送过程数据响应帧之前给自己发送过程数据请求帧。

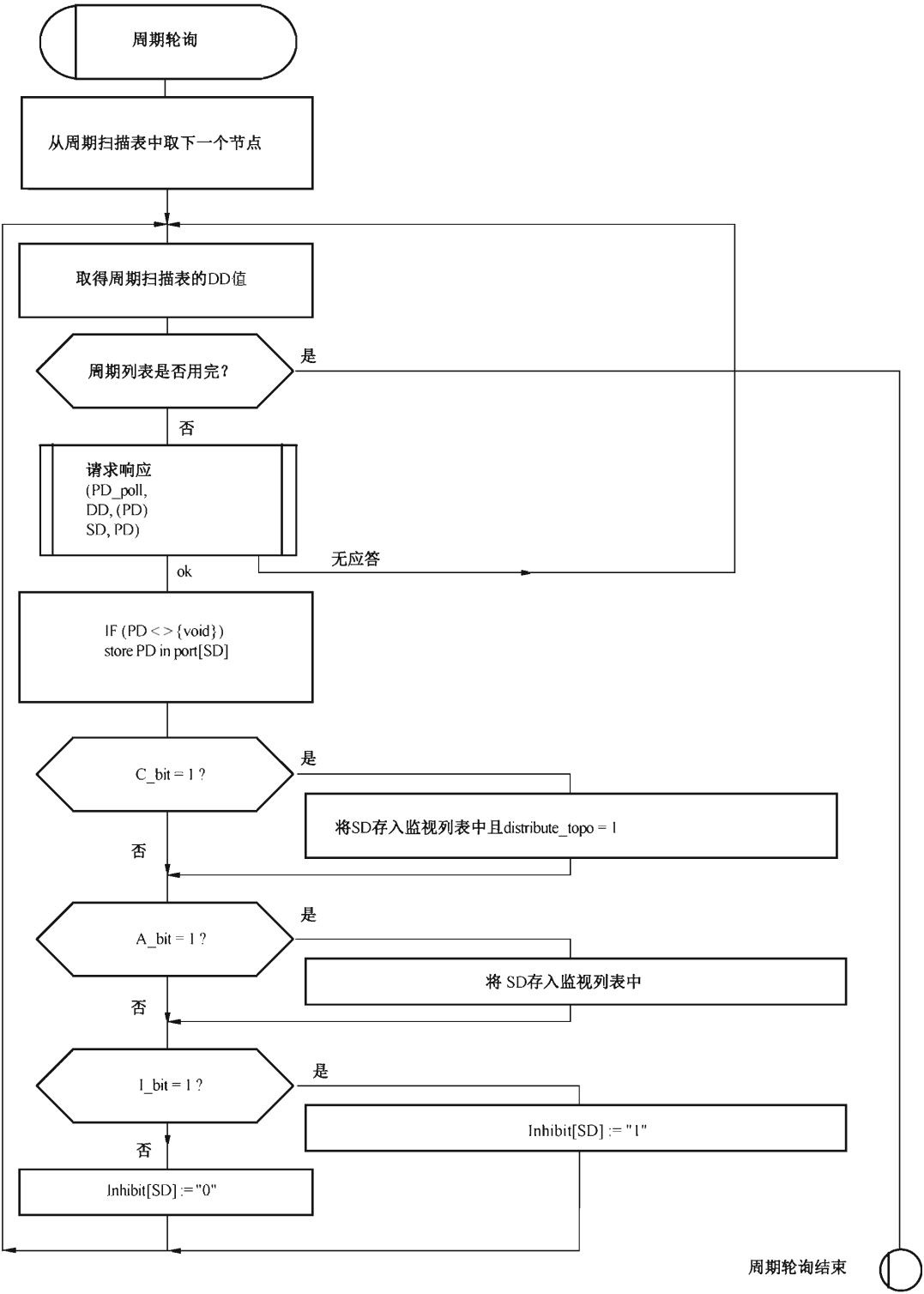


图 100 “周期轮询”宏

5.5.4.8.12 “消息轮询”宏

如图 101 所示,若在下一周期相开始前还有充足时间,主设备应发送消息数据。
若没有接收到消息数据响应帧,主设备在同一基本周期内不应重复消息数据请求帧。
当自轮询时,主设备应在发送消息数据响应帧之前给自己发送消息数据请求帧。
在下一周期相开始之前,若没有充足时间发送一个完整的帧,则节点应退出此状态。然后节点应返回 AWAIT_PERIOD 状态。

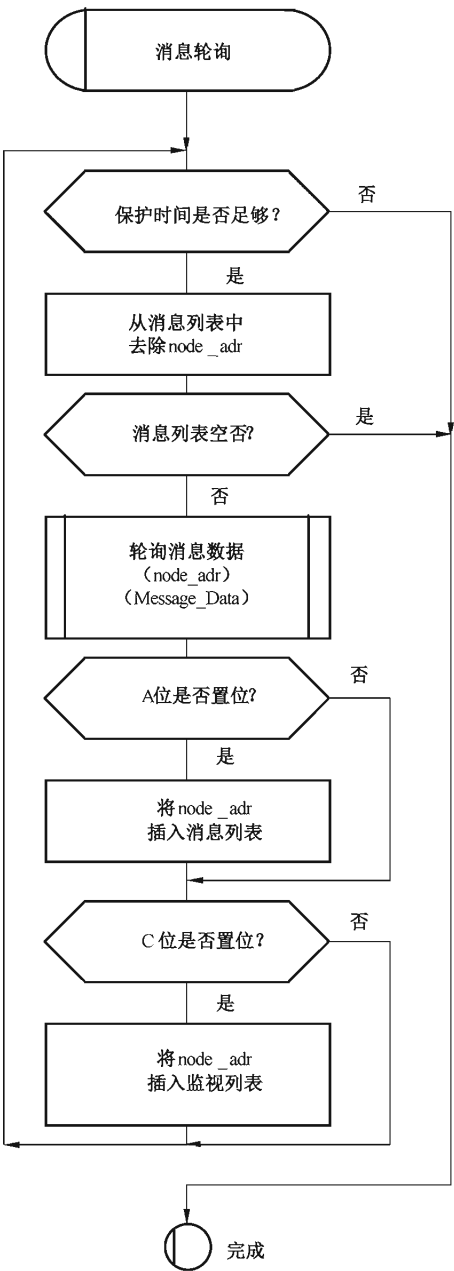


图 101 “消息轮询”(MESSAGE_POLL)宏

5.5.4.9 从设备状态

5.5.4.9.1 “未命名从设备”宏

如图 102 所示,节点应将自身设置为末端设定状态且其两个辅助通道都处于侦听状态,并在

AWAIT_NAMING 状态等待命名。

在进入 AWAIT_NAMING 状态时,节点应复位定时器 $T_{\text{await_naming}}$ 且期待:

- a) 定时器 $T_{\text{await_naming}}$:
- 若从应用接收到节点休眠(NodeSleep)命令,则切换到中间设定模式,并进入低功耗状态 NODE_SLEEP;
 - 若配置为弱节点,则进入“命名主设备”状态;
 - 否则,返回到“未命名从设备”(UNNAMED_SLAVE)状态。
- b) 辅助通道的 NameBy 信号:
- 将其主通道锁定到已命名方向;
 - 进入“已命名从设备”状态。

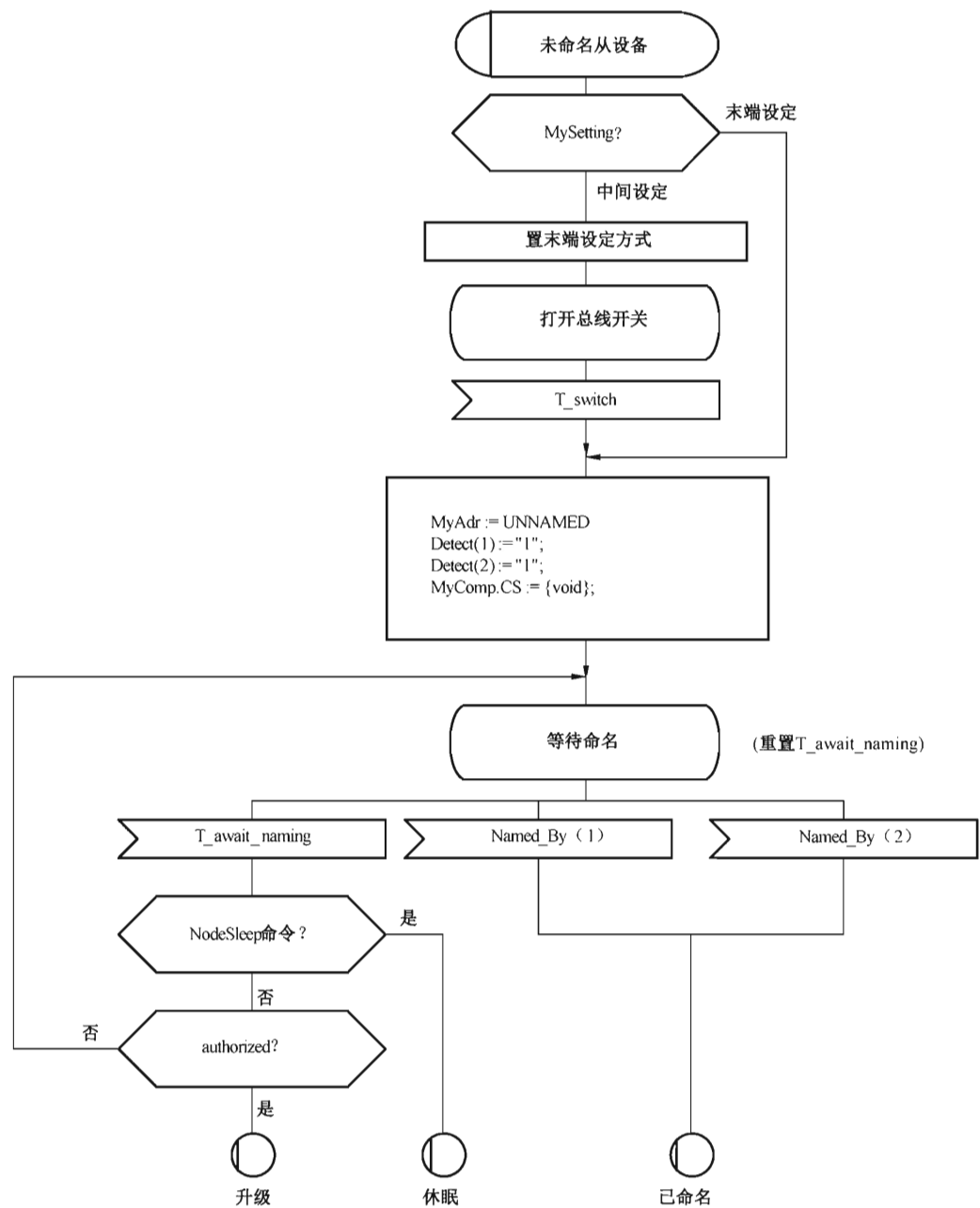


图 102 “未命名从设备”状态

5.5.4.9.2 “已命名从设备”宏

如图 103 所示,在“已命名从设备”(NAMED_SLAVE)状态下,节点已命名,且为响应主设备请求可在末端设定模式和中间设定模式之间切换。

在进入“已命名从设备”状态下时,节点应复位定时器 $T_{\text{named_slave}}$ 并期待:

- a) 节点描述符改变:
 - 若已被提升为强节点,则节点进入“命名主设备”宏;
 - 否则,置位其 C 位,并返回“已命名从设备”状态。
- b) 超时 $T_{\text{named_slave}}$,进入“未命名从设备”状态。
- c) 命名请求帧,向主设备发送命名响应帧。
- d) 中间设定请求帧,向主设备发送中间设定响应帧,若仍未处于中间设定模式则进入该模式。
- e) 末端设定请求帧,向主设备发送末端设定响应帧,若仍未处于末端设定模式则进入该模式。
- f) 状态请求帧,广播状态响应。
- g) 状态响应帧,记录状态以检查线路冗余。
- h) 消名请求帧:
 - 等待 $T_{\text{aux_main}}$ (以允许所有节点接收 3 个消名请求帧之一);
 - 将定时器 $T_{\text{await_naming}}$ 的阈值设置为最大值;
 - 返回“未命名从设备”状态。
- i) 拓扑请求帧或拓扑响应帧,与在“学习从设备”状态下相同响应,并进入“学习从设备”状态。
- j) 其他,增加错误计数器。

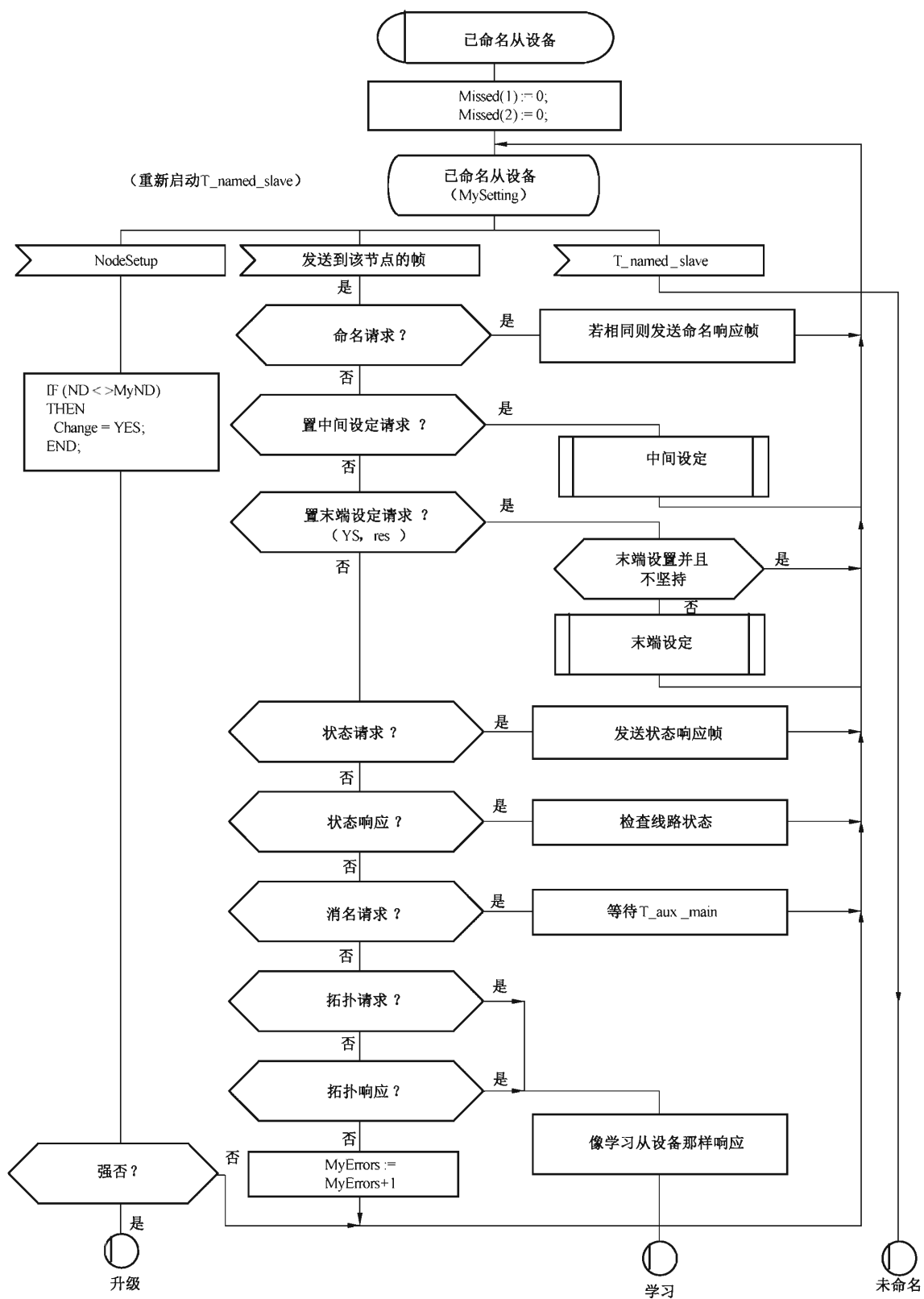


图 103 “已命名从设备”状态

注 1：在命名过程中，总线监视通过状态响应帧进行，而不管是否为末端节点。

注 2：在此状态下，节点忽略消息数据请求帧、消息数据响应帧、过程数据请求帧、过程数据响应帧、存在请求帧和存在响应帧。

5.5.4.9.3 “学习从设备”宏

如图 104 所示,节点在监视总线的同时,接收来自所有其他节点的拓扑信息。节点既不改变其设置,也不改变其名字。

在进入“学习从设备”(LEARNING_SLAVE)状态时,节点应复位定时器 $T_{\text{learning_slave}}$ 并期待:

- a) 节点描述符改变:
 - 若已被提升为强节点,则进入“命名主设备”状态;
 - 否则,应置位其 C 位。
- b) 超时 $T_{\text{learning_slave}}$,不改变其定时器 $T_{\text{learning_slave}}$ 的值,进入“未命名从设备”状态。
- c) 消名请求帧:
 - 等待 $T_{\text{aux_main}}$ (以允许所有节点接收 3 个消名请求帧之一);
 - 将定时器 $T_{\text{await_naming}}$ 的阈值设置为最大值 $T_{\text{await_max}}$;
 - 进入“未命名从设备”状态。
- d) 状态请求帧,向主设备发送状态响应帧。
- e) 状态响应帧,记录状态。
- f) 拓扑请求帧,广播其拓扑响应帧。
- g) 拓扑响应帧,更新其拓扑,检查是否拥有完整拓扑,是否已接收到具有相同 Master_Topo 的所有拓扑请求帧。若是,则置 $\text{Updated} := \text{TRUE}$ 。
- h) 存在请求帧、存在响应帧、过程数据请求帧、过程数据响应帧、消息数据请求帧或消息数据响应帧,与在“常规从设备”状态下相同响应,并进入“常规从设备”状态。
- i) 其他,增加错误计数器。

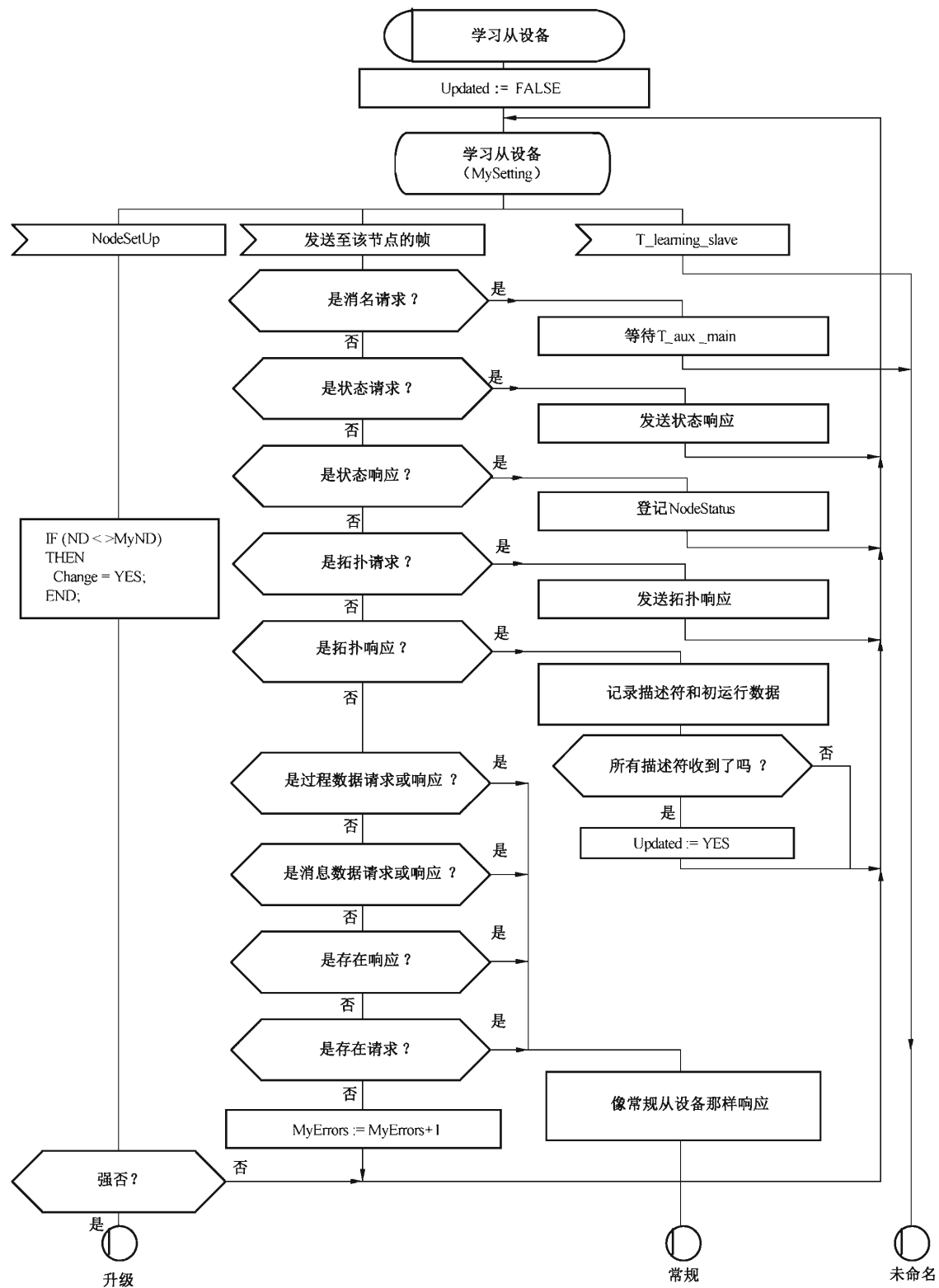


图 104 “学习从设备”宏

注：在学习阶段，总线监视通过拓扑响应帧进行，而不管是否为末端节点。

5.5.4.9.4 “常规从设备”宏

如图 105 所示，常规从设备(REGULAR_SLAVE)状态是节点的常规运行状态。在此状态节点收

发过程数据和消息数据,并通过在其响应帧中设置标志位指示事件。节点通过两个定时器监视末端节点的活动。

在“常规从设备”状态,节点应期待:

- a) 节点描述符改变:
 - 若已被提升为强节点,则进入“命名主设备”状态;
 - 否则,置位其 C 位。
- b) 用于其监视的每一个末端节点的超时 T_{bus_check} ,进入“未命名从设备”状态,保持 T_{await_naming} 的起始值。
- c) 存在请求帧,广播存在响应帧(若不是末端节点,则节点应忽略存在请求帧)。
- d) 存在响应帧,重新启动相关定时器 T_{bus_check} 。
- e) 状态请求帧,向主设备发送状态响应帧。
- f) 消名请求帧:
 - 等待 T_{aux_main} (以允许所有节点接收 3 个消名请求帧之一);
 - 将定时器 T_{await_naming} 的阈值设置为最大值;
 - 进入“未命名从设备”状态。
- g) 没有过程数据的过程数据请求帧:
 - 若节点被更新,则发送过程数据响应帧,从其源端口读取数据;
 - 否则,发送空的过程数据响应帧。
- h) 含有过程数据的过程数据请求帧(可选):
 - 若 Updated 是 TRUE,则将这些数据写入管理主设备数据专用的宿端口,且从其源端口读取数据,发送过程数据响应帧;
 - 否则,忽略帧并发送空的过程数据响应帧。
- i) 过程数据响应帧:
 - 若 Updated 是 TRUE,则将这些数据写入与源地址相符的宿端口;
 - 否则,忽略帧。
- j) 消息数据请求帧:
 - 若发送队列为空,则发送寻址主设备的空的消息数据响应帧($link_data_size = 0$);
 - 否则发送带有从其发送队列中抽取的消息数据包的消息数据响应帧。
- k) 消息数据响应帧,若其接收队列有空间,则将传入的消息数据存储在接受队列中,否则忽略帧(见 5.6.3)。
- l) 拓扑请求帧或拓扑响应帧,将 Updated 设置为 FALSE,与在“学习从设备”状态下相同响应,并进入“学习从设备”状态。
- m) 其他,错误计数器递增,并返回“学习从设备”状态。

注 1: 定时器 T_{bus_check} 可由一个定时器和计数器实现。

注 2: 没有与“常规从设备”状态相关联的、每次进入常规从设备状态时重启的定时器,因为该功能由 T_{bus_check} 执行。

注 3: 也可由 T_{bus_check} 定时器独立地对每个末端节点进行监视。

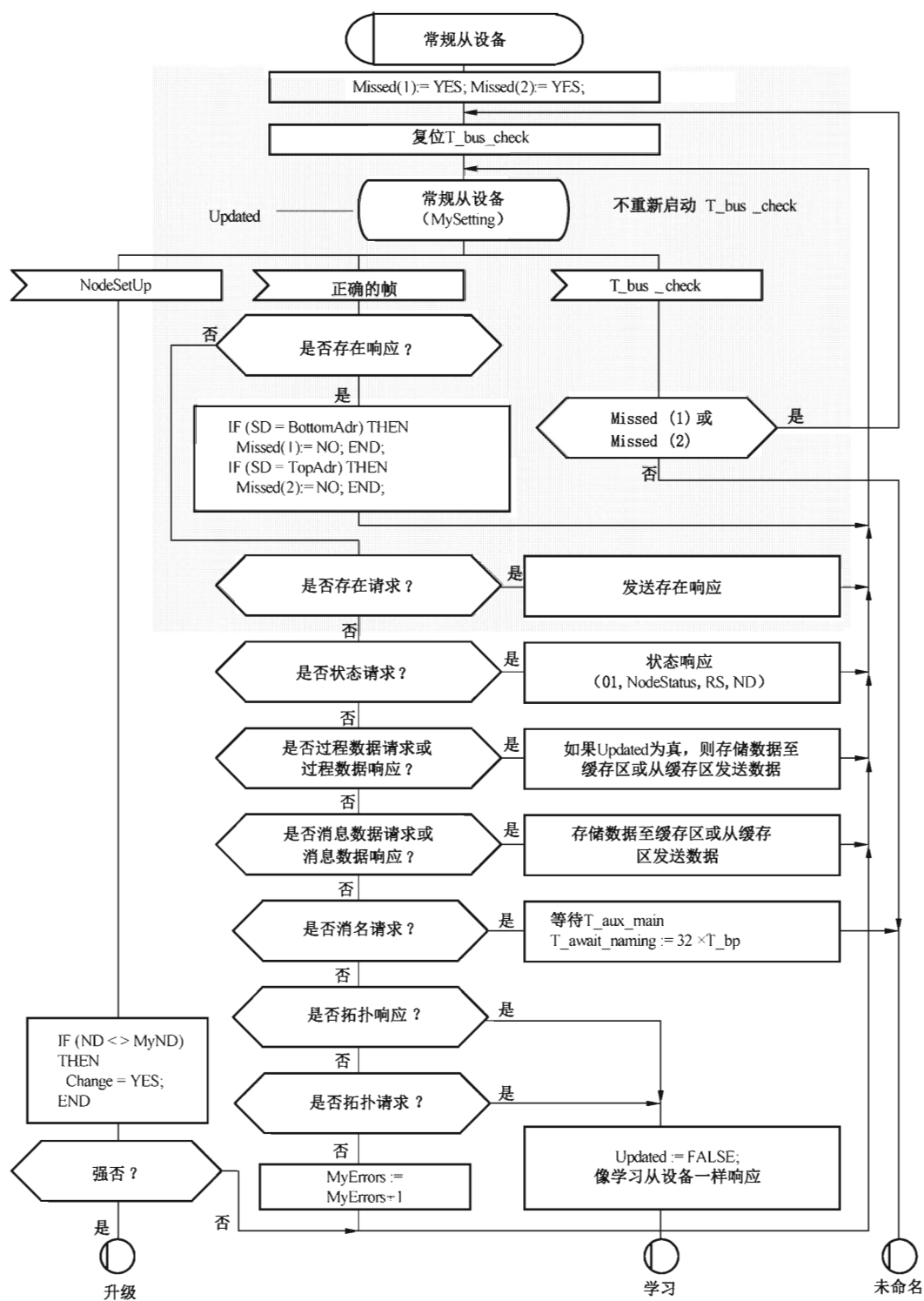


图 105 “常规从设备”宏

5.5.4.10 超时

表 18 中列出了推荐的超时值(±20%)。

表 18 时间常数值

时间常数名	值	用途
T _{await_naming}	1) $T_{\text{await_min}} = 1.0 \text{ ms} + T_{\text{switch}}$	主设备重命名其组成
	2) $((63 - \text{MyAdr}) + 0.5) \times T_{\text{bp}}$	主设备方向 1 上的节点
	3) $(\text{MyAdtr} - 1) \times T_{\text{bp}}$	主设备方向 2 上的节点
	4) $T_{\text{await_max}} = 32 \times T_{\text{bp}}$	已初始化或明确消名的节点
T _{aux_main}	1.0 ms	从辅助通道切换到主通道(或相反)的延时
T _{await_response}	1.756 ms	主设备等待从帧的时间。此时间考虑了最长可能帧,因为 HDLC 控制器只能指示帧结束
T _{bp}	25.0 ms	基本周期(常规运行)
T _{naming_master}	15.0 ms	命名周期(初运行)
T _{named_slave}	15.0 ms	在命名期间,主设备监视
T _{learning_slave}	15.0 ms	在学习期间,主设备监视
T _{bus_check} (1) T _{bus_check} (2)	$6.5 \times T_{\text{bp}}$	在常规运行期间,末端节点监视
T _{detecting}	$2 \times T_{\text{naming_master}}$	在初运行期间,检测请求帧(如有)之间间隔
	$2 \times T_{\text{bp}}$	在常规运行期间,存在请求帧或检测请求帧(如有)之间间隔
T _{detecting_response}	1.047 ms	末端节点等待检测响应帧的时间
T _{new_inaug}	$n \times T_{\text{bp}}$	连续两次初运行之间的最小时间,n 由应用设置
MAXLOST	50	末端节点在时间 $T_{\text{detecting}} \times \text{MAX-LOST}$ 之后,将假定不存在其他组成
T _{switch}	10.0 ms	闭合或断开继电器的缺省延时

5.6 链路层接口

5.6.1 链路层分层

链路层接口提供 3 种服务,如图 106 所示:

- a) 链路过程数据接口(LPI),在第 6 章规定,用于变量服务。本部分仅规定 WTB 专有参数。
- b) 链路消息数据接口(LMI),在第 6 章规定,用于消息传送服务。本部分仅规定 WTB 专有参数。
- c) 链路监视接口(LSI),允许配置链路层及监视总线,是 WTB 专有的并在本部分规定。

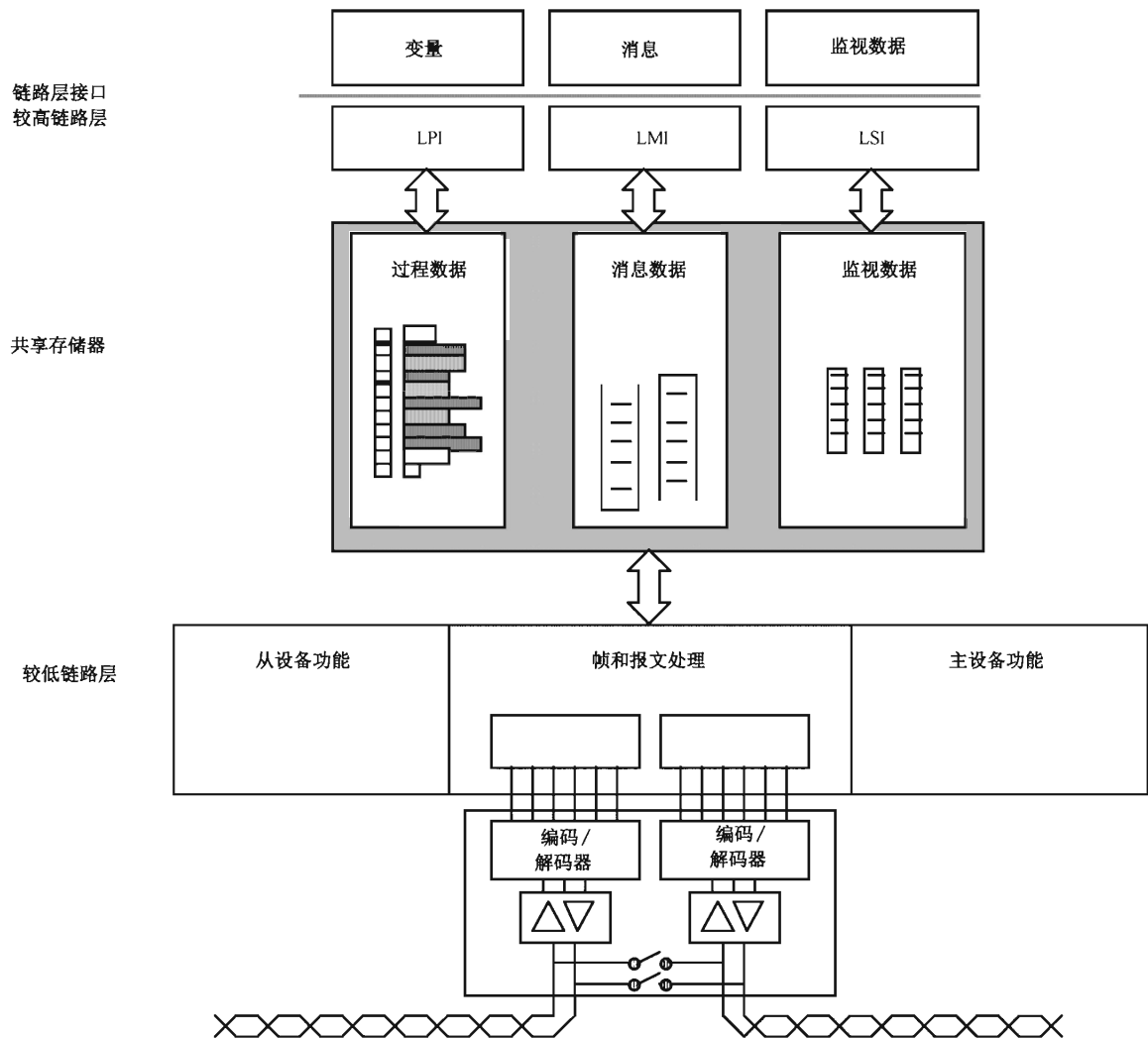


图 106 链路层分层结构

5.6.2 链路过程数据接口

5.6.2.1 概述

链路层和其上层之间用于过程数据的接口是一个称作通信存储器的共享存储器。通信存储器能被总线和应用同时访问。

通信存储器由一组端口组成,端口正好容纳一帧准备发送或接收的过程数据。

在节点内,每个端口由通信存储器标识符和 12 位端口地址标识。

端口的实现不在本部分范围之内。

第 6 章“实时协议”规定了端口的访问。

5.6.2.2 WTB 特性

WTB 通信存储器应支持最多 64 个端口,每个端口最大 1 024 位。

在所有情况:

- 每个节点应有一个广播其过程数据的源端口,端口地址的高 6 位为“000000”B、低 6 位为该节点地址;

——每个节点应有一个接收总线上来自其他可能节点的过程数据的宿端口,端口地址的高 6 位为“000000”B、低 6 位为源节点地址。

在主设备在其过程数据请求帧中包含有过程数据、仅用于被轮询从节点的应用中:

——主设备应有一个向各从节点发送过程数据的源端口,端口地址的高 6 位为“000010”B、低 6 位为目的节点地址;

——每个从设备应有一个接收主设备过程数据的宿端口,端口地址的高 6 位为“000010”B、低 6 位为其节点地址。

节点不应接收来自于未包含在已接收拓扑中,或在拓扑中接收的描述符不能解释的节点的过程数据。

节点可接收来自于节点类型已知,但节点版本与已知不同的另一个节点的过程数据,但应按两个节点版本中的较低版本解码数据。

节点没有接收到包含其新的节点密钥的拓扑时,不会改变其过程数据响应帧的格式。

宜保留过程数据的前两个八位位组用于标识帧类型(Node_Type + Node_Version = Node_Key),以作为进一步的保护。

5.6.3 链路消息数据接口

5.6.3.1 概述

在第 6 章中规定的链路消息数据接口(LMI)提供发送消息数据帧,以及恢复接收到的消息数据帧的服务。而且支持发送确认和接收指示服务。

链路消息数据接口为构建所有上层协议提供基本服务:

- a) 网络层通过网络和索引功能提供路由服务;
- b) 传送协议提供半双工端到端的消息控制;
- c) 会话层配对消息,以提供远程过程调用;
- d) 表示层统一数据表示;
- e) 应用接口提供客户和服务接口。

5.6.3.2 数据包长度

空数据包的链路数据长度(link_data_size)字段应为 0。

链路数据长度字段不应大于 128。

5.6.3.3 协议类型

用于实时协议的协议类型(Protocol_Type)应由置为“00xxx111”B(消息数据响应)的链路控制(link_control)字段指示。

5.6.3.4 消息传送协议

在其连接请求帧中,WTB 节点不应声明大于 124 个八位位组的数据包长度。

当响应连接请求帧时,节点应在其连接响应帧中规定数据包长度为 124 个八位位组或建议的较小包长度。

5.6.4 链路管理接口

5.6.4.1 概述

链路管理接口专属于 WTB。

链路管理接口为链路层的配置和检查以及事件报告提供通用服务。
以下各条并不隐含特定实现。允许任何提供相同语义的接口。
以下接口过程的参数格式没有规定。但第 8 章提出了一种消息格式,宜作为参数格式。

5.6.4.2 接口过程

5.6.4.2.1 概述

该接口的过程以 ls_t(link supervision, WTB)作为前缀。

5.6.4.2.2 类型 LS_T_RESULT

过程的返回结果具有 LS_T_RESULT 类型,见表 19。

表 19 LS_T_RESULT 类型

常数	代码	含义
L_OK	0	成功
L_BUSY	1	稍后再试
L_CALLING_SEQUENCE	2	命令序列错误
L_MISSING_UDF	3	用户定义的功能未知
L_CONFIGURATION_INVALID	4	拓扑或节点列表无效

5.6.4.2.3 常数 LS_T_STATE

下列常数指示节点当前所处主要状态,见表 20。

表 20 LS_T_STATE 常数主要状态

常数	代码	含义
LS_INITIALIZED	0	节点处于未配置状态
LS_CONFIGURED	1	节点处于已配置状态
LS_READY_TO_NAME	2	节点处于命名主设备状态
LS_READY_TO_BE_NAMED	3	节点处于未命名从设备状态
LS_INHIBITED	4	节点初运行被禁止
LS_REGULAR_STRONG	5	节点处于常规主设备(强节点)或示教主设备状态
LS_REGULAR_SLAVE	6	节点处于常规从设备或节点处于学习从状态
LS_REGULAR_WEAK	7	节点处于常规主设备(弱节点)或示教主设备状态

注：节点不能指示其处于休眠状态。

5.6.4.3 报告

5.6.4.3.1 过程 ls_t_Report

过程 ls_t_Report 见表 21。

表 21 过程 ls_t_Report

动作	向用户报告链路层的变化。 此过程由链路层调用且应事先预订(参见 ls_t_Configure)		
语法	Typedef LS_T_RESULT (* ls_t_Report)(ls_report)		
输入	ls_report	LR_REPORT 码之一	

5.6.4.3.2 常数 LR_REPORT

报告代码应如表 22 取值。

表 22 报告代码取值

常数	代码	含义
LR_CONFIGURED	16	链路层已配置
LR_STRONG	17	节点当前是运行主设备
LR_SLAVE	18	节点当前是运行从设备
LR_PROMOTED	19	节点从弱主提升为强主
LR_NAMING_SUCCESSFUL	20	主设备指示初运行结束
LR_NAMED	21	节点是已命名从节点
LR_WEAK	22	主设备转为弱主
LR_REMOVED	23	节点从配置中移除
LR_DEMOTED	24	弱主检测到强主
LR_DISCONNEXION	25	节点被断开
LR_INHIBITED	26	初运行被禁止
LR_INCLUDED	27	包含在组成中
LR_LENGTHENING	28	主设备检测到列车延伸
LR_DISRUPTION	29	节点检测到末端节点丢失
LR_MASTER_CONFLICT	30	强主检测到另一个强主
LR_NAMING_FAILED	31	命名时失败
LR_NEW_TOPOGRAPHY	32	接收到新拓扑
LR_NODE_STATUS	33	已改变的节点状态
LR_POLL_LIST_OVF	34	部分可运行
LR_ALLOWED	35	允许初运行

5.6.4.4 初始化服务

过程 ls_t_Init 见表 23。

表 23 过程 ls_t_Init

动作	初始化链路层且将变量设置为预定义的值。 调用此过程后,链路层应准备接收命令。此过程应在硬件复位后仅被调用一次。 此过程与实现相关
语法	LS_T_RESULT ls_t_Init (void);

5.6.4.5 复位服务

过程 ls_t_Reset 见表 24。

表 24 过程 ls_t_Reset

动作	将链路层复位为预定义值。 调用此过程后,链路层应处于空闲状态,准备好接收命令
语法	LS_T_RESULT ls_t_Reset (void);

5.6.4.6 配置服务

5.6.4.6.1 过程 ls_t_Configure

过程 ls_t_Configure 见表 25。

表 25 过程 ls_t_Configure

动作	配置链路层。 调用此过程后,节点应准备好开始通信	
语法	LS_T_RESULT ls_t_Configure (Type_Configuration * p_configuration);	
输入	p_configuration	指向下列配置数据结构的指针

5.6.4.6.2 数据结构 Type_NodeKey

Type_NodeKey 数据结构应包含表 26 中的元素。

表 26 Type_NodeKey 数据结构元素

属性	类型	含义
node_type	UNSIGNED8	由应用指示的节点的类型
node_version	UNSIGNED8	由应用指示的节点版本

注: Type_NodeKey 是与 Node_Key(见 5.5.2.2)结构相对应的 C 类型。

5.6.4.6.3 数据结构 Type_NodeDescriptor

Type_NodeDescriptor 数据结构应包含表 27 中的元素。

表 27 Type_NodeDescriptor 数据结构元素

属性	类型	含义
node_frame_size	UNSIGNED8	以八位位组计数的过程数据帧长度
node_period	UNSIGNED8	以基本周期的 2 ⁿ 倍数计数的节点周期的值 (node_period 是 n 的值)
node_key	Type_NodeKey	见此数据类型

注：Type_NodeDescriptor 是与 Node_Descriptor 结构(见 5.5.2.2)相对应的 C 类型。

5.6.4.6.4 数据结构 Type_Configuration

Type_Configuration 数据结构应包含表 28 中的元素。

表 28 Type_Configuration 数据结构元素

属性	类型	含义
transmission_rate	UNSIGNED16	传输速率,单位:kbit/s,缺省值:1 000 kbit/s
basic_period	UNSIGNED16	基本周期,单位:ms,缺省值:25.0 ms
fritting_disabled	UNSIGNED16	若加电清除禁止,为 1; 缺省值:0
node_descriptor	Type_NodeDescriptor	见 5.6.4.6.3
poll_md_when_idle	UNSIGNED8	若后台扫描使能,为 1;缺省值:0(见 5.4.3.4)
sink_port_count	UNSIGNED16	最大宿端口数,缺省值:22
source_port_count	UNSIGNED16	最大源端口数,缺省值:1
port_size	UNSIGNED8	端口最大长度(以八位位组计),缺省值:128
p_traffic_store	WORD32	指向通信存储器的指针,缺省值:NULL
ls_t_report	WORD32	报告用回调功能,缺省值:NULL
max_number_nodes	UNSIGNED8	其初运行数据将被存储的最大节点数,缺省值:0
inaug_data_max_size	UNSIGNED8(≤124)	应用定义的待发送的初运行数据的最大值(以八位位组计),缺省值:0
s_inaug_data_size	UNSIGNED8(≤124)	应用定义的待发送的初运行数据的实际值(以八位位组计),缺省值:0
p_inaug_data_list	WORD32	指向从此拷贝初运行数据的数据区的指针,缺省值:NULL

5.6.4.6.5 Type_Inauguration_Data

Type_Inauguration_Data 数据结构应包含表 29 中的元素。

表 29 Type_Inauguration_Data 数据结构元素

属性	类型	含义
inaug_data_max_size	UNSIGNED8(≤124)	应用定义的存储的初运行数据最大值(以八位位组计),缺省值:0
nr_descriptors	UNSIGNED8	其初运行数据被存储的节点数,缺省值:0=无效
node_descriptions	ARRAY[nr_descriptors] OF	应用为每个 WTB 节点定义的初运行数据列表,组成如下:
node_type	WORD8	Node_Key 第一部分
node_version	WORD8	Node_Key 第二部分
sam	BOOLEAN1	若与主设备朝向相同,则为 1
rsv1	WORD1(=0)	保留,=0
node_address	UNSIGNED6	从其接收到初运行数据的节点地址
inauguration_data_size	UNSIGNED8	初运行数据长度(≤124 个八位位组)
inauguration_data	ARRAY [inaug_data_len] OF WORD8	应用定义的初运行数据

node_descriptions 应在初运行开始前初始化。在初运行结束时,表 node_descriptions 包含 nr_descriptors 行,其中每个节点一行。

5.6.4.7 设置从设备服务

过程 ls_t_SetSlave 见表 30。

表 30 过程 ls_t_SetSlave

动作	阻止节点成为主设备
语法	LS_T_RESULT ls_t_SetSlave (void);

5.6.4.8 设置弱主服务

过程 ls_t_SetWeak 见表 31。

表 31 过程 ls_t_SetWeak

动作	使能节点成为弱主
语法	LS_T_RESULT ls_t_SetWeak (void);

5.6.4.9 设置强主服务

过程 ls_t_SetStrong 见表 32。

表 32 过程 ls_t_SetStrong

动作	命令节点成为强主
语法	LS_T_RESULT ls_t_SetStrong (void);

注：该命令引发初运行。

5.6.4.10 开始命名服务

过程 ls_t_StartNaming 见表 33。

表 33 过程 ls_t_StartNaming

动作	命令节点开始初运行
语法	LS_T_RESULT ls_t_StartNaming (void);

5.6.4.11 移除服务

过程 ls_t_Remove 见表 34。

表 34 过程 ls_t_Remove

动作	命令节点将自己从配置中移除并进入非激活状态
语法	LS_T_RESULT ls_t_Remove (void);

5.6.4.12 禁止总线延伸服务

过程 ls_t_Inhibit 见表 35。

表 35 过程 ls_t_Inhibit

动作	若检测到附加节点,禁止总线延伸
语法	LS_T_RESULT ls_t_Inhibit (void);

5.6.4.13 允许总线延伸服务

过程 ls_t_Allow 见表 36。

表 36 过程 ls_t_Allow

动作	若检测到附加节点,使能总线延伸
语法	LS_T_RESULT ls_t_Allow (void);

5.6.4.14 设置休眠服务

过程 ls_t_SetSleep 见表 37。

表 37 过程 ls_t_SetSleep

动作	使节点指示休眠请求
语法	LS_T_RESULT ls_t_SetSleep (void);

5.6.4.15 取消休眠服务

过程 ls_t_CancelSleep 见表 38。

表 38 过程 ls_t_CancelSleep

动作	使节点取消休眠请求
语法	LS_T_RESULT ls_t_CancelSleep (void);

5.6.4.16 获取状态服务

5.6.4.16.1 过程 ls_t_GetStatus

过程 ls_t_GetStatus 见表 39。

表 39 过程 ls_t_GetStatus

动作	获取物理层和链路层的状态	
语法	LS_T_RESULT ls_t_GetStatus (Type_WTBStatus * p_status) ;	
输入	p_status	指向 WTB_Status 数据结构的指针

5.6.4.16.2 数据结构 Type_Node_Status

Type_Node_Status 见表 40。

表 40 Type_Node_Status

属性	类型	含义
node_report	BITSET8	与节点报告(见 5.5.2.3)相关的 C 声明
user_report	BITSET8	与用户报告(见 5.5.2.4)相关的 C 声明

5.6.4.16.3 数据结构 Type_WTBStatus

Type_WTBStatus 见表 41。

表 41 Type_WTBStatus

属性	类型	含义
wtb_hardware_id	UNSIGNED8	硬件标识符
wtb_software_id	UNSIGNED8	链路层软件版本标识符
hardware_state	ENUM8	0:LS_OK 正确运行 1:LS_FAIL 硬件故障
link_layer_state	LS_T_STATE	见类型定义
net_inhibit	ENUM8	1:某些节点禁止初运行
node_address	UNSIGNED8	初运行指定的节点地址
node_orient	UNSIGNED8	相对于主设备的节点朝向： 0:L_UNKNOWN 1:L_SAM 2:L_INVERSE
node_strength	UNSIGNED8	节点强度 0:L_UNDEFINED 1:L_SLAVE 2:L_STRONG 3:L_WEAK
node_descriptor	Type_NodeDescriptor	见类型定义
node_status	Type_Node_Status	见类型定义

5.6.4.17 获取 WTB 节点服务

5.6.4.17.1 数据结构 Type_NodeList

Type_NodeList 数据结构应包含表 42 中的属性。

表 42 Type_NodeList

属性	类型	含义
nr_nodes	UNSIGNED8	组成中节点数
bottom_node	UNSIGNED8	主设备方向 1 上的末端节点地址 高 2 位为 0
top_node	UNSIGNED8	主设备方向 2 上的末端节点地址 高 2 位为 0
node_status_list	ARRAY [MAX_NODES] OF	节点状态列表,从底节点开始,按节点所在位置的顺序,到顶节点结束,包括:
node_status	Type_Node_Status	见类型定义

5.6.4.17.2 过程 ls_t_GetWTBNodes

过程 ls_t_GetWTBNodes 见表 43。

表 43 过程 ls_t_GetWTBNodes

动作	读拓扑中所有节点的节点报告和用户报告列表	
语法	LS_T_RESULT ls_t_GetWTBNodes (Type_NodeList * p_nodes);	
输入	p_nodes	指向节点列表的指针

5.6.4.18 获取拓扑服务

5.6.4.18.1 过程 ls_t_GetTopography

过程 ls_t_GetTopography 见表 44。

表 44 过程 ls_t_GetTopography

动作	允许应用读取在常规运行开始之前分发的拓扑	
语法	LS_T_RESULT ls_t_GetTopography (Type_Topography * p_topography);	
输入	p_topography	指向存放拓扑区域的指针
结果	若拓扑无效,返回 L_CONFIGURATION_INVALID	

5.6.4.18.2 数据结构 Type_Topography

拓扑的数据结构应包含表 45 中的元素。

表 45 Type_Topography

属性	类型	含义
node_address	UNSIGNED8	高 2 位为 0,低 6 位是连接到本站的节点地址
node_orient	UNSIGNED8	相对于主设备的节点朝向: 0:L_UNKNOWN 1:L_SAME 2:L_INVERSE
topo_counter	UNSIGNED8	低 6 位拷贝 6 位节点拓扑计数器,高 2 位为 0
individual_period	UNSIGNED8	主设备分配给节点的周期,以 ms 表示的基本周期的 2 的幂次倍

表 45（续）

属性	类型	含义
is_strong	UNSIGNED8	1:强主控制的总线 0:弱主控制的总线
number_of_nodes	UNSIGNED8	根据初运行结果的节点数目
bottom_address	UNSIGNED8	主设备方向 1 上的末端节点地址,高 2 位为 0
top_address	UNSIGNED8	主设备方向 2 上的末端节点地址,高 2 位为 0
inauguration_data	Type_Inauguration_Data	见类型定义

5.6.4.19 改变节点描述符服务

过程 ls_t_ChgNodeDesc 见表 46。

表 46 过程 ls_t_ChgNodeDesc

动作	给链路层提供一个新的描述符。在常规运行期间调用此过程将导致通信中断及新的拓扑发布	
语法	LS_T_RESULT Type_NodeDescriptor *	ls_t_ChgNodeDesc (node_descriptor);
输入	node_descriptor	指向 Node_Descriptor 数据结构的指针

5.6.4.20 改变用户报告服务

过程 ls_t_ChguserReport 见表 47。

表 47 过程 ls_t_ChguserReport

动作	允许应用修改用户报告	
语法	LS_T_RESULT UNSIGNED8 UNSIGNED8	ls_t_ChgUserReport (set_mask, clear_mask);
输入	set_mask	在用户报告中将掩码置位的位设置为 1
	clear_mask	在用户报告中将掩码置位的位设置为 1

5.6.4.21 改变初运行数据服务

过程 ls_t_ChgInauguration_Data 见表 48。

表 48 过程 ls_t_ChgInauguration_Data

动作	允许应用修改此节点的初运行数据	
语法	LS_T_RESULT UNSIGNED8 void *	ls_t_ChgInauguration_Data (inaug_data_size, p_inauguration);
输入	inaug_data_size	以八位位组计数的初运行数据长度(≤124)
	p_inauguration	用户定义的初运行数据

5.6.4.22 获取统计服务

5.6.4.22.1 过程 ls_t_GetStatistics

过程 ls_t_GetStatistics 见表 49。

表 49 过程 ls_t_GetStatistics

动作	提供使用和出错的统计信息	
语法	LS_T_RESULT Type_LLStatisticData *	ls_t_GetStatistics (p_statistic_data);
输入	p_statistic_data	指向统计数据结构的指针(见 5.6.4.22.3)

5.6.4.22.2 数据结构 Type_LineStatus

Type_LineStatus 数据结构应包含表 50 中的元素。

表 50 Type_LineStatus

属性	类型	含义
transmitted_count	UNSIGNED32	由该节点发送的帧数
received_count	UNSIGNED32	由该节点接收到的没有错误的帧数
errors_count	UNSIGNED16	接收到的错误帧数
timeouts_out	UNSIGNED16	等待预期响应时超时数
注：计数器达到最大值时循环计数，其初值未规定。		

5.6.4.22.3 数据结构 Type_LLStatisticData

Type_LLStatisticData 数据结构应包含表 51 中的元素。

表 51 Type_LLStatisticData

属性	类型	含义
basic_period_count	UNSIGNED32	每个基本周期时递增
inauguration_count	UNSIGNED16	每次新的初运行时递增
topography_count	UNSIGNED16	每次新的拓扑时递增
transmitted_md_count	UNSIGNED32	每发送一个消息数据响应帧时递增
received_md_count	UNSIGNED32	每接收一个消息数据响应帧时递增
line_status_a1	Type_LineStatus	见类型
line_status_a2	Type_LineStatus	见类型
line_status_b1	Type_LineStatus	见类型
line_status_b2	Type_LineStatus	见类型
line_switch_count	UNSIGNED32	每次切换到冗余线路时递增
注：计数器达到最大值时循环计数,其初值未规定。		

5.6.4.23 获取初运行数据服务

获取初运行数据服务见表 52。

表 52 获取初运行数据服务

动作	返回初运行数据指针	
语法	LS_T_RESULT Void * *	ls_t_GetInaugData (p_inaug_data_list);
输出	p_inaug_data_list	指向所有已命名节点的初运行数据的指针

6 实时协议

6.1 概述

6.1.1 本章内容

本章适用于使用 WTB 和/或 MVB 和/或遵循相同工作原理的其他总线的列车通信网络(TCN)。
本章规定了列车通信网络的一个组成部分:实时协议。实时协议提供编组内或编组间应用之间的通信,如图 107 所示。

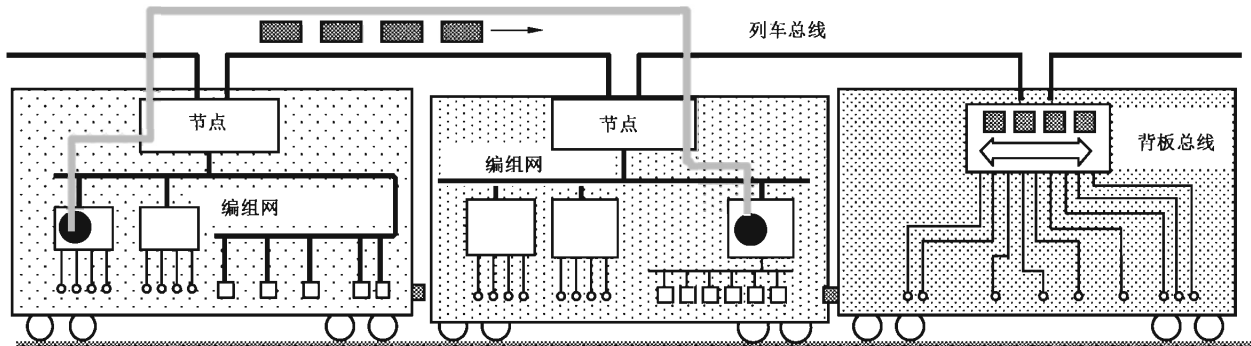


图 107 列车通信网络结构

本章规定了提供给应用的两种主要通信服务：

- a) 变量传送：具有确定传输延时的短数据传送，包括：
 - 过程数据链路层接口 (LPI)；
 - 变量传送应用层接口 (AVI)。
- b) 消息传送：可能较长但不频繁的数据项传送，如有必要数据项可拆分成小数据包并按需发送，包括：
 - 消息数据链路层接口 (LMI)。
 - 用于路由网络中数据包的网络层路由。
 - 提供流量控制和差错恢复的传输层：
 - 点到点传送；
 - 或多播消息传送 (可选)。
 - 配对呼叫消息和应答消息的会话层。
 - 消息传送应用层接口 (AMI)。

本章也规定了数据表示方式 (对于变量传送和消息传送)。

6.1.2 本章结构

本章结构与 OSI 通信模型类似，如图 108 所示。

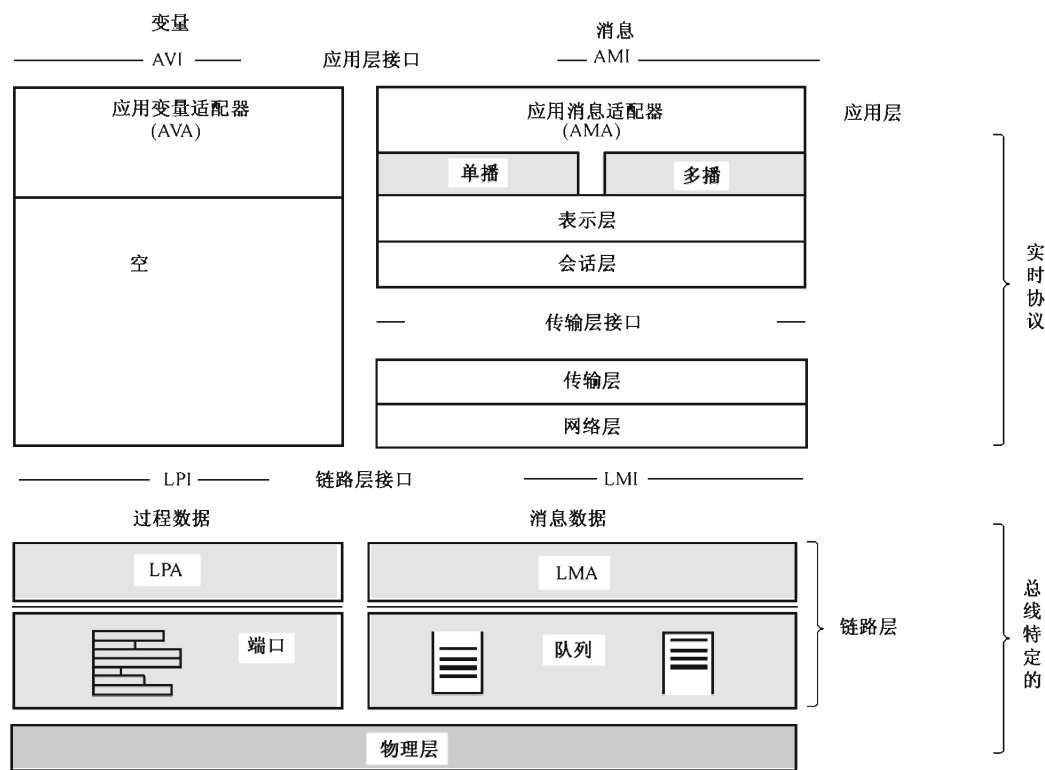


图 108 实时协议分层

6.2 变量传送服务和协议

6.2.1 概述

服务和协议分成低层接口和高层接口：

- a) 低层接口即链路层接口，规定了期望来自总线的服务；
- b) 高层接口即应用层接口，规定了提供给应用的服务。

6.2.2 过程数据链路层接口

6.2.2.1 目的

链路层过程数据接口 (LPI) 定义了由总线提供给上层协议的过程数据服务。

LPI 定义了端口初始化、整个数据集引入端口及从端口移除和与传送整个数据集相关的同步原语。

应用通常不直接访问 LPI，同步除外 (用于通知数据集的接收和发送)。

本接口不规定底层通信。端口之间的数据传输，包括总线主的轮询策略，由链路层和物理层实现。

注：单个过程变量在 LPI 层是不可见的。

6.2.2.2 数据集

6.2.2.2.1 端口和通信存储器

链路层应提供一些端口用于过程数据通信。

端口是一种共享内存结构，能被应用和网络同时访问。

端口是一种非排队的数据结构，即对端口写入新值将覆盖原值、而对端口的读操作不改变端口中的数据。

链路层和应用应能坚固地访问端口,即在一次不可分割的操作中完成对端口中所有数据的读/写。

同一链路层的端口属于同一通信存储器。

在通信存储器中,端口应由其端口地址标识。

在设备中,通信存储器应由其通信存储器标识符标识。

6.2.2.2.2 数据集的坚固性

每一个端口应只有一个数据集。

数据集应只由一个发布者应用产生。

总线上一个给定的端口地址应只有一个源端口,但可有不定数量的宿端口。

不同设备的链路层应在限定时间内发送源端口的内容到订阅同一端口地址的所有宿端口,并提供所发送数据集的坚固性。

注:不强制要求总线保证不同数据集之间的坚固性。

6.2.2.2.3 差错处理

数据集中未定义字段应以“1”覆盖。

若链路层不能保证数据集的坚固性,例如检测到有传输错误发生或其发布者应用不能提供正确或及时的数据,则链路层应以“0”覆盖整个端口。

注:由于以全“0”或全“1”覆盖过程变量的值可能产生一个合法的值,因此在可能引起问题的地方使用同一数据集的校验变量作为有效性指示。

6.2.2.2.4 刷新监视

每一个宿端口及每一个订阅的数据集都应有一个关联的刷新定时器。该定时器指示自从总线将新值写入端口后经过的时间。

该刷新定时器应与数据集内容一起在一次不可分割的操作中获取。

刷新定时器的分辨率不应大于 16 ms。

刷新定时器计时范围不应小于 4 s,达到最大定时值后应停止运行。

注 1:刷新定时器不考虑发布者应用何时将过程变量插入到该端口。源端口时间监视由应用完成,可通过校验变量处理。

注 2:变量强制不改写定时器。

6.2.2.2.5 数据集的同步

广播发送给定数据集可用于同步应用。

6.2.2.2.6 数据集的轮询

数据集轮询相关过程分别是编组网链路层和列车总线链路层的一部分,不在本部分中描述。

6.2.2.2.7 数据集标识符

6.2.2.2.7.1 数据集、端口和逻辑地址

在设备内,数据集应由其所在的通信存储器及其端口地址标识。

通过总线发送时,数据集应由该总线上其过程数据帧的逻辑地址标识。

过程数据帧的逻辑地址应与数据集所在通信存储器的端口地址相同。

6.2.2.2.7.2 数据集标识符格式

在设备内,数据集应由其数据集标识符(DS_Name)标识,见表 53。

表 53 数据集标识符格式

定义	数据集类型
语法	<pre>typedef struct /* 大开端表示 */ { unsigned traffic_store_id :4; /* DS_NAME 第一部分 */ unsigned port_address :12; /* DS_NAME 第二部分 */ } DS_NAME;</pre>

DS_Name 能方便地表示为一个 16 位字，见图 109。

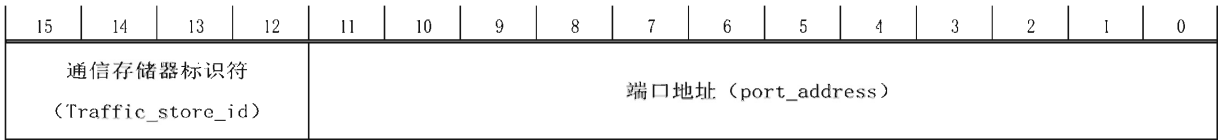


图 109 DS_Name 标识符格式

6.2.2.2.7.3 通信存储器标识符

通信存储器标识符(traffic_store_id)应选中设备内一个通信存储器。
所支持的通信存储器最大数量应为 16。

- 注 1：通信存储器标识符不隐含所访问总线的类型(MVB、WTB 或其他总线)，但是若隐含(例如 WTB 通信存储器可总为“1”)则可简化实现。
- 注 2：通信存储器标识符可与相应总线的总线标识符(Bus_Id)相同。然而，有的通信存储器可能没有与之相连的总线，例如用于任务之间通信的通信存储器。

6.2.2.2.8 端口地址

端口地址(port_address)应标识由通信存储器标识符选中的通信存储器 4 096 个端口中的一个端口。

注：实际可能的端口数量取决于所连总线的类型。

示例：在 MVB 上，每个设备可有最多 4 096 个端口，每个端口最大 256 位。

6.2.2.3 链路过程数据接口原语

6.2.2.3.1 概述

链路过程数据接口(LPI)应为数据集访问提供原语，如图 110 所示和表 54 所列，这些原语在以下各条中规定。

以下各条不隐含特定实现。任何提供相同语法的接口都是允许的。

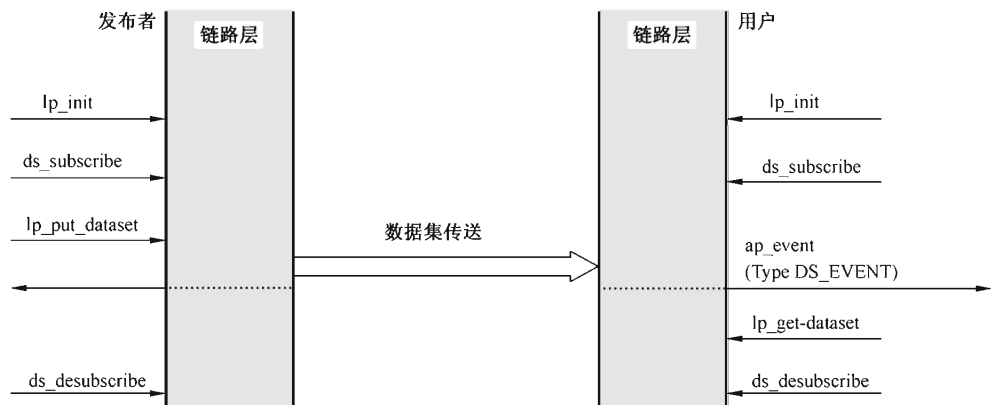


图 110 LPI 原语交换

表 54 LPI 原语

原语名	含义
lp_init	初始化通信存储器
lp_put_dataset	插入一个待发送数据集
lp_gut_dataset	获取一个接收到的数据集
ds_subscribe	订阅一个用于同步的数据集
ap_event	发送或接收时同步
ds_desubscribe	解除订阅同步数据集

注 1：应用可不使用上述原语而直接访问通信存储器结构以提高访问速度。
注 2：通信处理器能使用相同的原语从总线侧访问通信存储器。
注 3：原语不立即触发总线通信，只访问通信存储器。

6.2.2.3.2 类型“LP_RESULT”

类型“LP_RESULT”见表 55。

表 55 类型“LP_RESULT”

定义	返回值类型为 LP_RESULT 的 LPI 过程应编码如下：
语法	<pre>Typedefenum { LP_OK =0, /* 正常结束 */ LP_PRT_PASSIVE =1, /* 警告:数据集非激活 */ LP_ERROR =2, /* 未定义的错误 */ LP_CONFIG =3, /* 配置错误 */ LP_MEMORY =4, /* 内存不足 */ LP_UNKNOWN_TS =5, /* 未知的通信存储器 */ LP_RANGE =6, /* 内存地址错误 */ LP_DATA_TYPE =7 /* 不支持的数据类型 */ } LP_RESULT;</pre>

6.2.2.3.3 过程“lp_init”

过程“lp_init”见表 56。

表 56 过程“lp_init”

定义	生成通信存储器,设置订阅清单,建立源端口和宿端口并将其初始化为预设值	
语法	LP_RESULT lp_init (ENUM8 ts_id, void * p_descriptor);	
输入	ts_id	通信存储器标识符(0~15)
	p_descriptor	与实现相关的数据结构
返回	任意 LP_RESULT	

6.2.2.3.4 过程“lp_put_dataset”

过程“lp_put_dataset”见表 57。

表 57 过程“lp_put_dataset”

定义	从应用拷贝数据集到通信存储器中的一个端口	
语法	LP_RESULT lp_put_dataset (DS_NAME * dataset, void * p_value);	
输入	dataset	将发布的数据集的 DS_NAME
	p_value	指向从中拷贝数据集值的应用内存区域的指针
返回	任意 LP_RESULT	
用法	通信存储器中数据集的前值被覆盖	

6.2.2.3.5 过程“lp_get_dataset”

过程“lp_get_dataset”见表 58。

表 58 过程“lp_get_dataset”

定义	从端口拷贝数据集及其刷新定时器到应用	
语法	LP_RESULT lp_get_dataset (DS_NAME * dataset, void * p_value, void * p_fresh);	
输入	dataset	将接收的数据集的 DS_NAME
返回	任意 LP_RESULT	
输出	p_value	指向数据集值将拷贝至的应用内存地址的指针
	p_fresh	指向刷新定时器将拷贝至的应用内存地址的指针

6.2.2.3.6 过程“ds_subscribe”

过程“ds_subscribe”见表 59。

表 59 过程“ds_subscribe”

定义	订阅一个数据集用于发送或接收,并指示当指定的数据集发送或接收时调用的指示过程	
语法	LP_RESULT ds_subscribe (DS_NAME * dataset, DS_EVENT event_cnf, UNSIGNED16 instance);	
输入	dataset	将纳入订阅的数据集的 DS_NAME
	event_cnf	订阅的过程
	instance	标识订阅应用事例并在 ds_event 过程中返回的 16 位索引号
返回	任意 LP_RESULT	
用法	1. 此过程可在实现限制范围内多次调用,以用于不同的数据集和订阅过程。 2. 同一数据集只能订阅一次	

6.2.2.3.7 类型“DS_EVENT”

类型“DS_EVENT”见表 60。

表 60 类型“DS_EVENT”

定义	发送或接收数据集后,链路层应调用订阅此数据集的过程,且其类型应是 DS_EVENT	
语法	<pre>typedef void (* DS_EVENT) (UNSIGNED16 instance);</pre>	
输入	instance	标识订阅此事件的应用事例的 16 位索引号
用法	1. 此过程已预先由“ds_subscribe”订阅。 2. 不标识引起该事件的数据集,但事例参数能用于此目的	

6.2.2.3.8 过程“ds_desubscribe”

过程“ds_desubscribe”见表 61。

表 61 过程“ds_desubscribe”

定义	从订阅中移除一个数据集	
语法	<pre>LP_RESULT ds_desubscribe (DS_NAME * dataset;);</pre>	
输入	dataset	将从订阅中移除的数据集的 DS_NAME
返回	任意 LP_RESULT	

6.2.3 过程变量应用层接口

6.2.3.1 目的

应用变量接口(AVI)定义了提供给应用的变量传送服务。
该接口的原语只访问通信存储器的端口,不触发总线上的通信。
假定发布者将变量写入端口将导致在有限时间后同一变量写入各订阅者相应端口。

6.2.3.2 过程变量

6.2.3.2.1 过程变量的传送和储存

过程变量作为数据集的一部分传送。
属于同一个数据集的所有过程变量应作为坚固集传送和储存。

6.2.3.2.2 刷新监视

过程变量应与其数据集的刷新定时器一起在一次不可分割的操作中获取。

6.2.3.2.3 同步

应用可通过在 LPI 上的数据集传送同步。

6.2.3.2.4 校验变量

为了评估其有效性,每一个变量可与属于同一数据集的另一个变量相关联,该关联变量称作校验变量。

校验变量和过程变量应在一次不可分割的操作中存储和获取。

同一校验变量可用于多个过程变量。

校验变量可位于数据集中的任意位置并可覆盖一个过程变量。

如使用,校验变量应采用 ANTIVALENT2 格式,取值如下:

- a) “00”B:被保护的变量错误或可疑;
- b) “01”B:被保护的变量假定正确;
- c) “10”B:被保护的变量已强制为强制值;
- d) “11”B:被保护的变量未定义。

注 1: 当不特别指定过程变量或校验变量时,统称为变量。

注 2: 过程变量和校验变量在数据集中的位置由应用分配。

注 3: 期望总线或发布者以“0”覆盖数据集中可疑字段。这导致校验变量值置为“00”B,从而使应用可检测到错误。

注 4: 预留给未来扩展的字段宜覆写为全“1”。这导致校验变量值置为“11”B,从而使未来设备在接收老设备数据时不将填充的“1”误认为有效数据。

注 5: 期望应用处理可能出现的两种情况,即因通信问题产生的全“0”和无效化数据的全“1”。

示例:图 111 显示了同一数据集中的过程变量及其相关的校验变量。

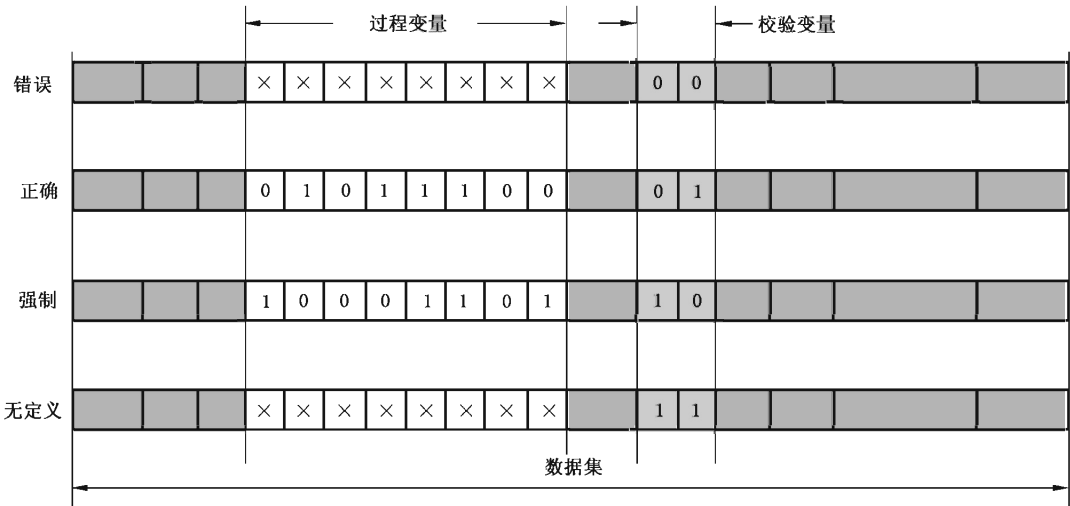


图 111 校验变量

6.2.3.2.5 过程变量标识符

6.2.3.2.5.1 变量和数据集标识符

在设备内,过程变量应由其所在数据集(DS_Name)和其在该数据集中位偏置(Var_Offset)标识。

通过总线传送时,过程变量应由其逻辑地址和其在被传送数据集中位偏置标识。

6.2.3.2.5.2 过程变量标识符格式

在设备内,每一个过程变量应由一个唯一的标识符(称作其过程变量标识符,PV_Name)标识。过程变量标识符由以下元素组成:

- a) 通信存储器标识符(Traffic_Store_Id);
- b) 端口地址(Port_Address);
- c) 变量偏置(Var_Offset);
- d) 变量长度(Var_Size);
- e) 变量类型(Var_Type);
- f) 校验偏置(Chk_Offset)。

注: 由于不同设备通信存储器标识符可能不同,因而同一总线上同一变量的 PV_Name 可能随设备不同而不同。

6.2.3.2.5.3 通信存储器标识符

通信存储器标识符应标识设备内 16 个通信存储器中的一个。

6.2.3.2.5.4 端口地址

端口地址应标识通信存储器内 4096 个端口中的一个。

6.2.3.2.5.5 变量位偏置

若数据集包含一个无符号整数,则其最低位的偏置为 0。
变量位偏置应定义该过程变量值所占据区域起始位置相对于数据集起始位置的位偏置。
过程变量应位于变量位偏置乘以其长度的位置处。
注: 对齐是为了适应不能访问跨越字边界的数据结构的编译器。

6.2.3.2.5.6 变量类型和变量长度

变量类型和变量长度应唯一标识过程变量的格式,变量类型指示类型(见 6.4),变量长度指示长度。
变量类型和变量长度应如表 62 所示编码。

表 62 过程变量标识符中变量类型(Var_Type)和变量长度(Var_Size)的编码

变量长度	变量类型	数据类型
0	0	BOOLEAN1
	1	ANTIVALENT2
	2	BCD4 or ENUM4
	3	reserved
	4	BITSET8
	5	UNSIGNED8 or ENUM8
	6	INTEGER8
	7	CHARACTER8 (ARRAY [0..0] OF WORD8)
1	4	BITSET16
	5	UNSIGNED16 or ENUM16
	6	INTEGER16

表 62 (续)

变量长度	变量类型	数据类型
1	8	BIPOLAR2.16 (±200%)
	9	UNIPOLAR2.16 (+400%)
	10	BIPOLAR4.16(±800%)
2	3	REAL32
	4	BITSET32
	5	UNSIGNED32 or ENUM32
	6	INTEGER32
3	2	TIMEDATE48
4	4	BITSET64
	5	UNSIGNED64
	6	INTEGER64
<i>n</i> −1	7	ARRAY OF WORD8 (奇数个八位位组)
	15	ARRAY OF WORD8 (偶数个八位位组)
	13	ARRAY OF UNSIGNED16 (<i>n</i> = 以 WORD16 计数的数组大小)
	14	ARRAY OF INTEGER16
	11	ARRAY OF UNSIGNED32 (<i>n</i> = 以 WORD16 计数的数组大小)
	12	ARRAY OF INTEGER32

变量长度在原始型和构造型中的解释不同：

- 在原始型中,变量长度指示使用的 16 位字个数；
- 在构造型中,变量长度指示 16 位字个数减 1。

注 1: 在原始型中,Var_Size=0 表示长度小于一个 16 位字,Var_Size=1 表示长度为一个 16 位字。

注 2: 在构造型中,Var_Size=0 表示长度为一个 16 位字。

注 3: 系数“*n*”是 WORD16 的个数。一个变量的最大长度为 128 个八位位组(64×16=1 024 位),Var_Size = “3F”H。

注 4: 表 62 中未列出的编码保留。

注 5: 尽管{Traffic_Store, Port_Address, Var_Offset}三元组唯一地标识了一个过程变量,但是 PV_Name 中包含有类型、长度信息便于在网络和应用数据类型之间快速转换。

6.2.3.2.5.7 校验偏置

校验偏置应定义与过程变量相关联的校验变量相对于数据集起始点的位置。

当过程变量没有相关联的校验变量时,则应使用对应于数据集中最右端位置(“0FFF”H)的校验偏置。

6.2.3.3 应用变量传送接口原语

6.2.3.3.1 概述

应用变量传送接口原语分成 3 组：

- a) 单个变量访问；
- b) 集合访问；
- c) 群集访问。

以下各条不隐含特定实现。任何提供相同语法的接口都是允许的。

6.2.3.3.2 类型“AP_RESULT”

类型“AP_RESULT”见表 63。

表 63 类型“AP_RESULT”

定义	AVI 过程的返回值应编码如下：
语法	<pre>Typedef enum { AP_OK =0, /* 正常结束 */ AP_PRT_PASSIVE =1, /* 警告:数据集未激活 */ AP_ERROR =2, /* 一般性错误 */ AP_CONFIG =3, /* 配置错误发生 */ AP_MEMORY =4, /* 内存不足 */ AP_UNKNOWN_TS =5, /* 未知的通信存储器 */ AP_RANGE =6, /* 内存地址错误 */ AP_DATA_TYPE =7 /* 不支持的数据类型 */ } AP_RESULT;</pre>

注：这些常量的编码与具有类似名称的 LPI 常量相同。

6.2.3.3.3 配置参数

配置参数见表 64。

表 64 配置参数

参数	取值范围	说明
AP_TS_ID_MAX	0~15	实现中支持的通信存储器最大数量 = AP_TS_ID_MAX + 1

6.2.3.3.4 变量初始化

过程变量的初始化由针对所有数据集的数据集初始化机制执行的,该机制依赖于应用。

注：期望链路层缺省初始化所有数据集为“0”。

6.2.3.3.5 单个变量访问原语

6.2.3.3.5.1 概述

变量传送应用层(AVI)应为单个变量访问提供以下原语,如图 112 所示,且在以下各条规定：

- a) ap_put_variable;
- b) ap_get_variable;
- c) ap_force_variable;

- d) ap_unforce_variable;
- e) ap_unforce_all。

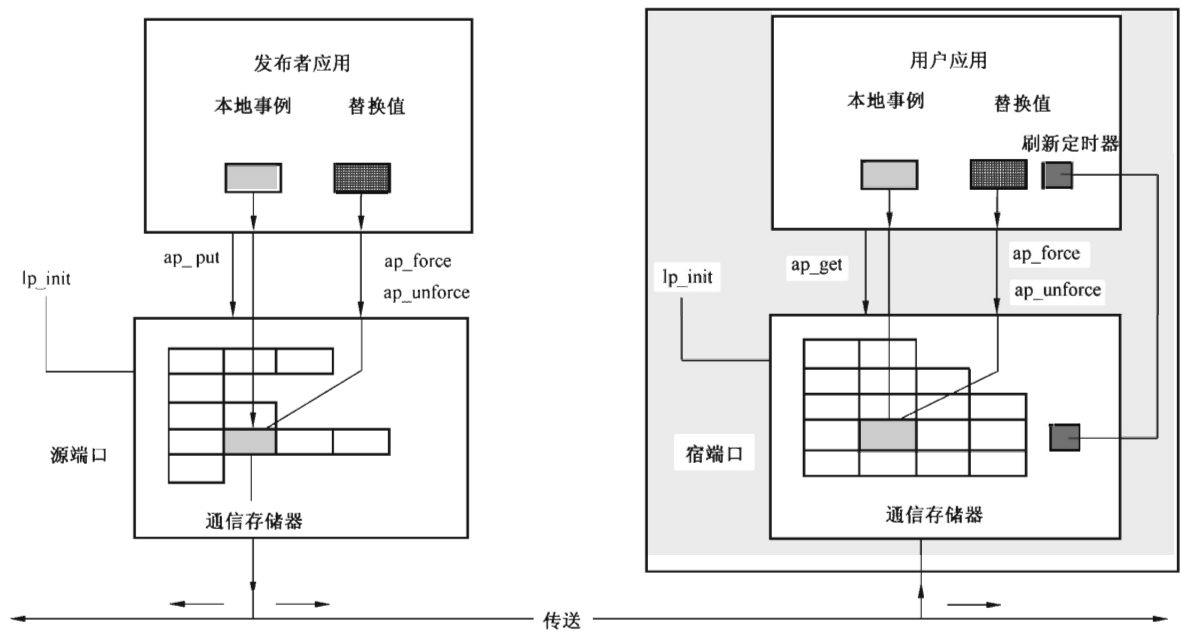


图 112 单个变量访问

6.2.3.3.5.2 类型 PV_NAME

对于单个变量访问,变量位偏置和校验偏置应包括 Var_Octet_Offset 和 Var_Bit_Number 两个字段。

Var_Octet_Offset 是相对于数据集起始点的八位位组偏置,第一个发送或存储的八位位组序号为 0。

多个八位位组组成的变量中,Var_Bit_Number 的值总为 0。变量长度小于一个八位位组时,Var_Bit_Number 是变量为在八位位组中右对齐而右移的位数。Var_Bit_Number 与八位位组中的位偏置不相同。过程变量定义见表 65。

表 65 过程变量类型定义

定义	单个过程变量类型			
语法	typedef struct /* 大开端表示 */			
	{			
	unsigned traffic_store_id	:4;	/* DS_NAME 第一部分 */	
	unsigned port_address	:12;	/* DS_NAME 第二部分 */	
	unsigned var_size	:6;	/* 如表 19 给定 */	
	unsigned var_octet_offset	:7;	/* 第一个八位位组偏置为 0 */	
	unsigned var_bit_number	:3;	/* 从右边开始计数的位数 */	
	unsigned var_type	:6;	/* 如表 19 给定 */	
	unsigned chk_octet_offset	:7;	/* 第一个八位位组偏置为 0 */	
	unsigned chk_bit_number	:3;	/* 从右边开始计数的位数 */	
	} PV_NAME;			

PV_NAME 可方便地编码,见图 113。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
通信存储器标识符 (Traffic_storc_id)				端口地址 (port_address)											
变量长度 (var_size)						变量八位位组偏置 (var_octet_offset)						变量位号 (var_bit_number)			
变量类型 (var_type)						校验八位位组偏置 (chk_octet_offset)						校验位号 (chk_bit_number)			

图 113 过程变量结构

注: var_bit_number 的引入是为了加速单个变量的访问,特别是避免加 8 或减 8 运算,从而利用处理器移位指令的优势。这种分解只有在所有数据类型都对齐时才有可能,例如,一个 ANTIVALENT2 的数据不可位于奇偏置。
示例:下列内存映射表示一个 PV_NAME,见图 114。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	0	0	1	1	0	1	1	1	0	1	0
0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0

图 114 过程变量示例

该 PV_NAME 标识了一个过程变量,该过程变量:
——位于通信存储器 3,端口地址为“1BA”H(=442),位偏置为“1F”H(31×8=248)位;
——其类型为 INTEGER8(var_size=0×WORD16,var_type=6);
——其关联的两位校验变量位于八位位组偏置 0、位号 4,即位于数据集的第 3 和第 4 位(位偏置为 2 和 3)。若位号为奇数,则没有关联的校验变量。

6.2.3.3.5.3 过程 ap_put_variable

过程 ap_put_variable 见表 66。

表 66 过程 ap_put_variable

定义	从应用内存地址空间拷贝单个过程变量及其校验变量到通信存储器	
语法	AP_RESULT ap_put_variable (PV_NAME * ts_variable, void * p_value, void * p_check);	
输入	ts_variable	过程变量的 PV_NAME
	p_value	指向从中拷贝发布值的应用内存位置的指针
	p_check	指向从中拷贝关联校验变量的应用内存位置的指针

表 66（续）

返回	任意 AP_RESULT
用法	a) 若过程变量已被强制,则 ap_put_variable 无效。 b) 过程变量的前值被覆盖。 c) 同一个数据集的其他数据不受影响,但不保证其坚固性

6.2.3.3.5.4 过程 ap_get_variable

过程 ap_get_variable 见表 67。

表 67 过程 ap_get_variable

定义	从通信存储器拷贝一个过程变量及其关联的校验变量和刷新定时器到应用	
语法	AP_RESULT ap_get_variable (PV_NAME * ts_variable, void * p_value, void * p_check, void * p_fresh);	
输入	ts_variable	过程变量的 PV_NAME
	p_value	指向存放接收值的应用内存位置的指针
	p_check	指向存放关联校验变量的应用内存位置的指针
	p_fresh	指向存放关联刷新定时器的应用内存位置的指针
返回	任意 AP_RESULT	
用法	a) 本原语可与一个源端口或宿端口一起使用,允许同一设备上的订阅者成为发布者。 b) 若过程变量已被强制,则获取强制值	

6.2.3.3.5.5 过程 ap_force_variable

过程 ap_force_variable 见表 68。

表 68 过程 ap_force_variable

定义	强制端口中单个过程变量为给定值;设置其关联校验变量为“10”B	
语法	AP_RESULT ap_force_variable (PV_NAME * ts_variable, void * p_value);	
输入	ts_variable	过程变量的 PV_NAME
	p_value	指向从中拷贝强制值的应用内存位置的指针
返回	任意 AP_RESULT	
用法	期望替换值的类型与类型 PV_NAME 兼容	

6.2.3.3.5.6 过程 ap_unforce_variable

过程 ap_unforce_variable 见表 69。

表 69 过程 ap_unforce_variable

定义	结束变量强制,并恢复对其正常的总线访问;不修改关联校验变量	
语法	AP_RESULT ap_unforce_variable (PV_NAME * ts_variable);	
输入	ts_variable	过程变量的 PV_NAME
返回	任意 AP_RESULT	

6.2.3.3.5.7 过程 ap_unforce_all

过程 ap_unforce_all 见表 70。

表 70 过程 ap_unforce_all

定义	结束通信存储器所有变量替换,不修改关联校验变量	
语法	AP_RESULT ap_unforce_all (ENUM8 ts_id);	
输入	ts_id	Traffic_Store_Id (0...15)
返回	任意 AP_RESULT	

6.2.3.3.6 集合访问过程

6.2.3.3.6.1 集合访问模式

集合由属于同一数据集的一组变量(过程变量和校验变量)组成,并作为一个整体对待以保持其坚固性和刷新信息。

变量传送应用层接口(AVI)应为集合访问提供以下原语,如图 115 所示,且在以下各条规定:

- a) ap_put_set;
- b) ap_get_set。

注:集合访问是坚固的,即所有变量在一次不可分割的操作中拷贝或都不拷贝。

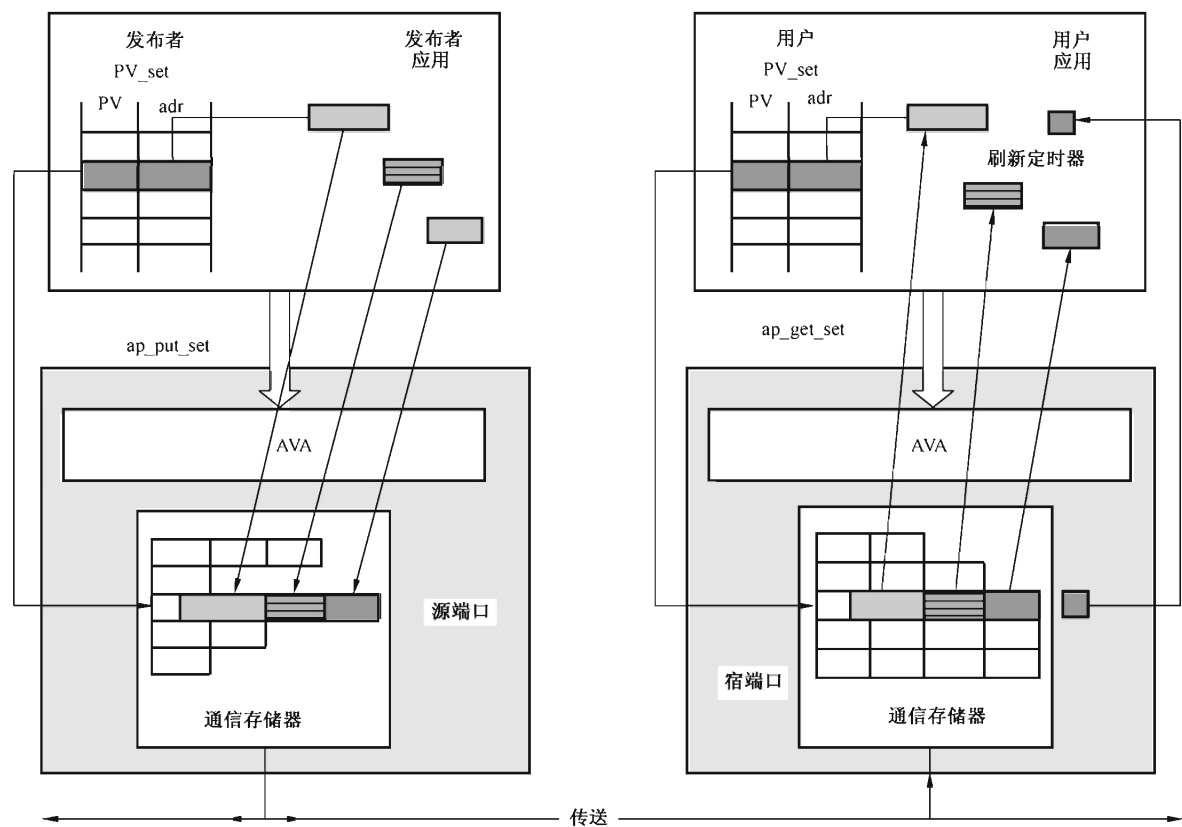


图 115 集合访问

6.2.3.3.6.2 类型 PV_SET

类型 PV_SET 见表 71。

表 71 类型 PV_SET

定义	PV_SET 标识属于同一数据集的一组变量,包括每个变量拷入(或拷出)的内存地址以及整个数据集的刷新定时器	
语法	typedef struct	PV_LIST
	{ void * p_variable; UNSIGNED8 derived_type; UNSIGNED8 array_count; UNSIGNED8 octet_offset; UNSIGNED8 bit_number; };	
	typedef	PV_SET
	{ struct PV_LIST * p_pv_list; UNSIGNED16 c_pv_list; UNSIGNED16 * p_freshtime; DS_NAME dataset; };	

表 71（续）

元素	p_variable	变量的内存地址
	derived_type	由 Var_Type 和 Var_Size 衍生成的数据类型,与实现相关
	array_count	数组中元素的个数
	octet_offset	变量的八位位组偏置
	bit_number	小于 1 个八位位组的过程变量或校验变量的位号(见 PV_Name 定义)
	p_pv_list	指向 PV_List 的指针
	c_pv_list	PV_List 中变量个数
	p_freshtime	刷新定时器的内存地址(过程 ap_put_set 不使用)
	dataset	DS_Name(整个集合)
用法	a) 过程变量和校验变量等同处理,因为 PV_SET 中所有变量是坚固的。因此 Var_Offset 和 Chk_Offset 之间没有区别。 b) 为了加快处理速度 Var_Offset(或 Chk_Offset)被分成八位位组偏置和位偏置。同样的原因,类型和长度各占一个八位位组,而不像在类型 PV_Name 中占用 6 位。 c) 尽管发布者集合中不使用刷新计数器,但是订阅者集合和发布者集合仍采用相同格式。 d) 为了提高访问效率,PV_SET 中也可包含对通信存储器内部数据结构的直接引用	

注：出于效率考虑,PV_SET 不包括每一个变量的完整 PV_NAME。尤其是,由于同一个校验变量可保护多个变量,校验变量以常规 ANTIVALENT2 类型出现。

6.2.3.3.6.3 过程 ap_put_set

过程 ap_put_set 见表 72。

表 72 过程 ap_put_set

定义	在一次不可分割的操作中,从应用内存地址空间拷贝属于同一集合的过程变量到端口	
语法	AP_RESULT ap_put_set (PV_SET * pv_set);	
输入	pv_set	指向 PV_List 的指针
返回	任意 AP_RESULT	

6.2.3.3.6.4 过程 ap_get_set

过程 ap_get_set 见表 73。

表 73 过程 ap_get_set

定义	在一次不可分割的操作中,从端口拷贝属于同一集合的过程变量到应用内存地址空间	
语法	AP_RESULT ap_get_set (PV_SET * pv_set);	
输入	pv_set	指向 PV_List 的指针
返回	任意 AP_RESULT	

6.2.3.3.7 群集访问过程

6.2.3.3.7.1 群集访问模式

群集是分散于多个数据集和多个通信存储器的变量组。
变量传送应用层接口(AVI)应为群集访问提供以下原语,如图 116 所示,且在以下各条规定:
——ap_put_cluster;
——ap_get_cluster。

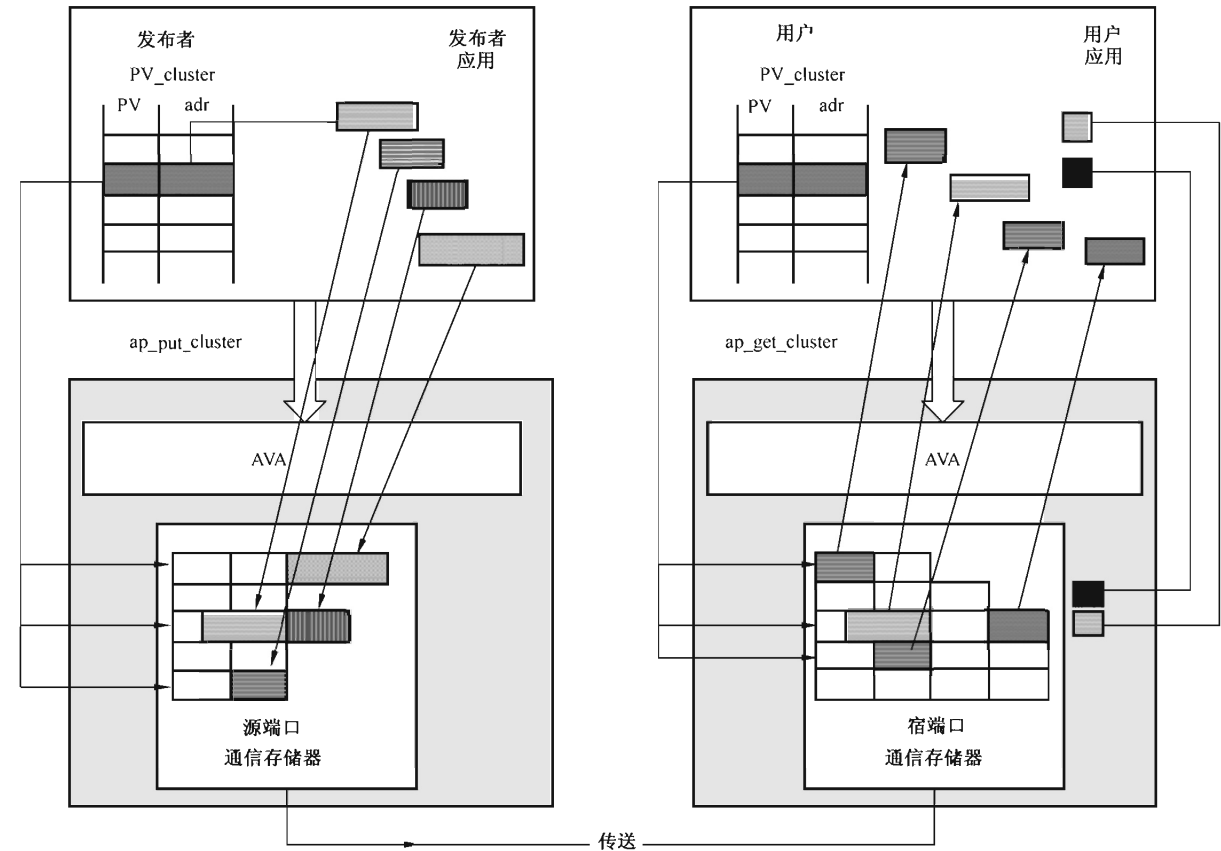


图 116 群集访问

注 1: 群集访问只保证群集中单个 PV_SET 的坚固性,而不保证整个群集的坚固性。
注 2: 不保证在不同 PV_SET 中出现的同一数据集的变量之间的坚固性。

6.2.3.3.7.2 类型 PV_CLUSTER

PV_Cluster 标识一组由通信存储器排序的 PV_SET,见表 74。

表 74 类型 PV_CLUSTER

定义	PV_Cluster 类型	
语法	<pre>typedef struct PV_CLUSTER { UNSIGNED8 ts_id; UNSIGNED8 c_pv_set; struct PV_SET * p_pv_set [c_pv_set]; };</pre>	
元素	ts_id	通信存储器的 Traffic_Store_Id(0...15)
	c_pv_set	群集中 PV_Set 的数量
	p_pv_set	指向 PV_SET 的指针数组[0...c_pv_set-1]
用法	a) 每一个通信存储器有一个群集列表。 b) 尽管发布者群集列表中不使用刷新计数器,发布者群集列表和订阅者群集列表仍采用相同格式。 c) 为了提高访问效率,PV_Cluster 中也可包含对通信存储器内部数据结构的直接引用	

6.2.3.3.7.3 过程 ap_put_cluster

过程 ap_put_cluster 见表 75。

表 75 过程 ap_put_cluster

定义	从应用拷贝一个变量群集到通信存储器。属于同一 PV_SET 的变量坚固地拷贝	
语法	<pre>AP_RESULT ap_put_cluster (PV_CLUSTER * pv_cluster);</pre>	
输入	pv_cluster	指向发布者群集列表的指针
返回	任意 AP_RESULT	

6.2.3.3.7.4 过程 ap_get_cluster

过程 ap_get_cluster 见表 76。

表 76 过程 ap_get_cluster

定义	从通信存储器拷贝过程变量群集到本地订阅者事例。属于同一 PV_SET 的变量坚固地拷贝	
语法	AP_RESULT ap_get_cluster (PV_CLUSTER * pv_cluster);	
输入	pv_cluster	指向订阅者群集列表的指针
返回	任意 AP_RESULT	

6.3 消息传送服务和协议

6.3.1 概述

- 消息传送服务和协议分为低层接口和高层接口：
- a) 低层接口即链路层接口，规定了期望总线提供的服务；
 - b) 高层接口即应用层接口，规定了提供给应用的服务。

6.3.2 基准站

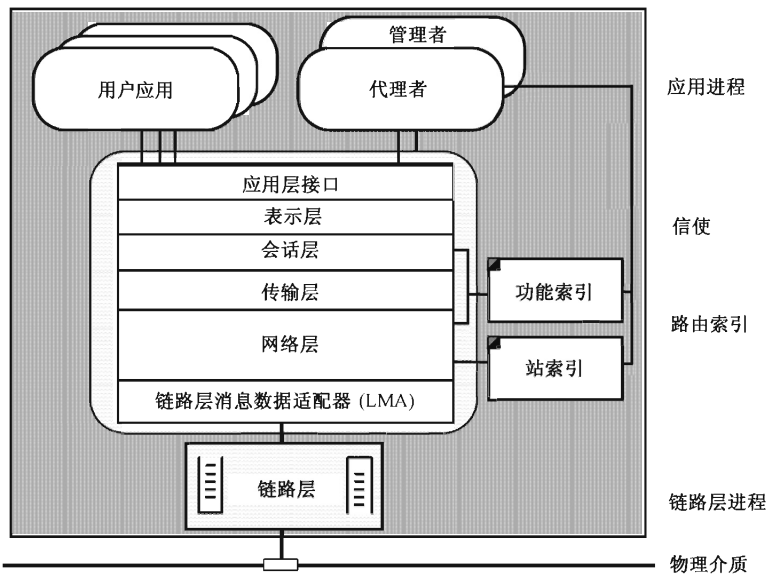
6.3.2.1 总则

- 提供消息传送服务的 TCN 站应包含下列要素：
- a) 至少一条总线连接，其可通过 LPI 访问并实现链路层进程；
 - b) 一个称作信使的协议机，实现网络层、传输层、对话层、表示层和应用层；
 - c) 一个或多个应用进程，其中一个应用进程作为网络管理代理者。
- 管理者站应提供管理者应用进程。
- 注：期望代理者提供的服务最小集在第 8 章中规定。

6.3.2.2 终端站

只连接一条总线的站，或终端站，应只有一个链路层。

示例：终端站的参考结构如图 117 所示。



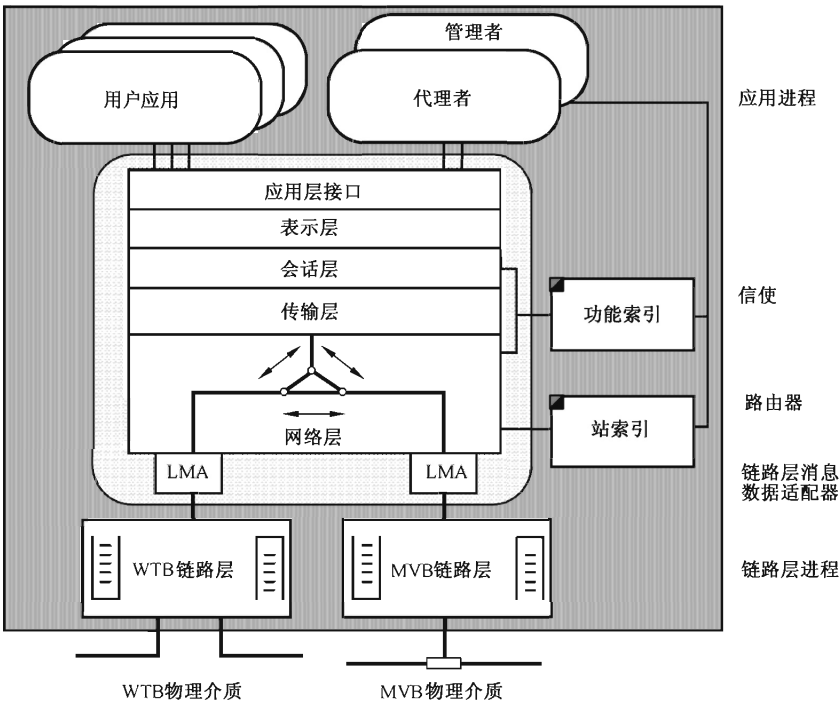
注：终端站的网络层只访问自身的传输层和链路层。

图 117 终端站

6.3.2.3 路由器站

连接到多条总线的站，或路由器站，每一条总线应有一个链路层。且各条总线共享相同的实时协议。

示例：图 118 表示一个连接到 MVB 链路层和 WTB 链路层的路由器站。



注：路由器站的网络层能将包从一条总线路由到另一条总线。

图 118 WTB 和 MVB 之间的路由器站

6.3.2.4 网关节

连接到多条总线的站,或网关节,每一条总线应有一个链路层。编组网协议与 WTB 实时协议不同。网关节应将编组网协议适配到 WTB 实时协议。

示例:图 119 表示一个连接到编组网链路层和 WTB 链路层的网关节。

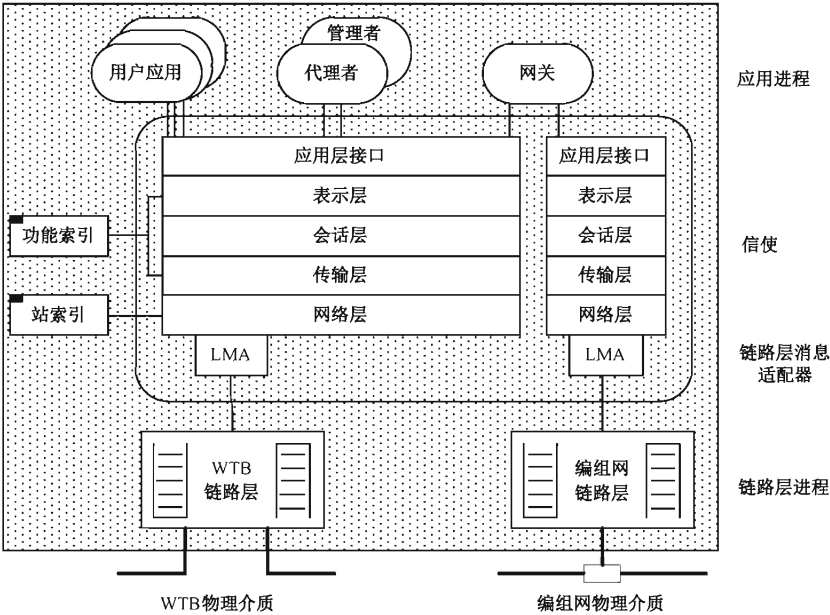


图 119 WTB 和编组网之间的网关节

6.3.2.5 站标识符

站应由站标识符(Station_Id)标识。

由于每个站只可能有一个代理者和一个管理者,因而站标识符也标识代理者。

注:站标识符不必与设备地址相同。路由器站在其联挂的每一条总线上都有一个设备地址,但只有一个站标识符。

站标识符可通过网络管理修改,而设备地址通常是由硬连线决定的。

6.3.2.6 总线标识符

站应通过总线标识符(Bus_Id)标识每一个与其连接的链路层,即每条总线(编组网或列车总线)。

每个站最多连接 16 条总线。

注:总线标识符并不隐含所联挂总线的类型信息,但是一直将 Bus_Id=1 指定给列车总线是有益的。

6.3.2.7 链路地址

站应通过链路地址(Link_Address)标识其能通过某一条总线访问的每一个设备。

链路地址应由总线标识符(Bus_Id)和设备的设备地址(Device_Address)串接组成。

注:设备地址的长度与总线类型相关。

6.3.3 消息包处理

6.3.3.1 概述

通信协议栈内的数据包处理是一个实现问题。在消息服务中,接口过程通过指针指定消息数据包。实现需要使用该指针通过索引或者拷贝的方式来传递数据包。没有规定数据包结构。

为了便于可移植及解释接口过程,本条定义了一些数据包处理过程。下列各条不隐含特定实现。任何提供相同语法的接口都被允许。

接口不必对外开放,且其不是一致性测试的内容。

6.3.3.2 包池

为保证站内访问的一致性,传输层、网络层和链路层使用的所有数据包具有相同格式。

为通过索引而不是拷贝的方式传送数据包,应以包池形式进行数据包的动态存储管理。最初,以一些空包建立包池。用户可从池中申请包并在使用完之后归还。从长远来看,包进出池的净流量宜为 0。

每个站有多个包池,一般每个链路层一对。包所属的池是包的属主。

6.3.3.3 类型“MD_PACKET”

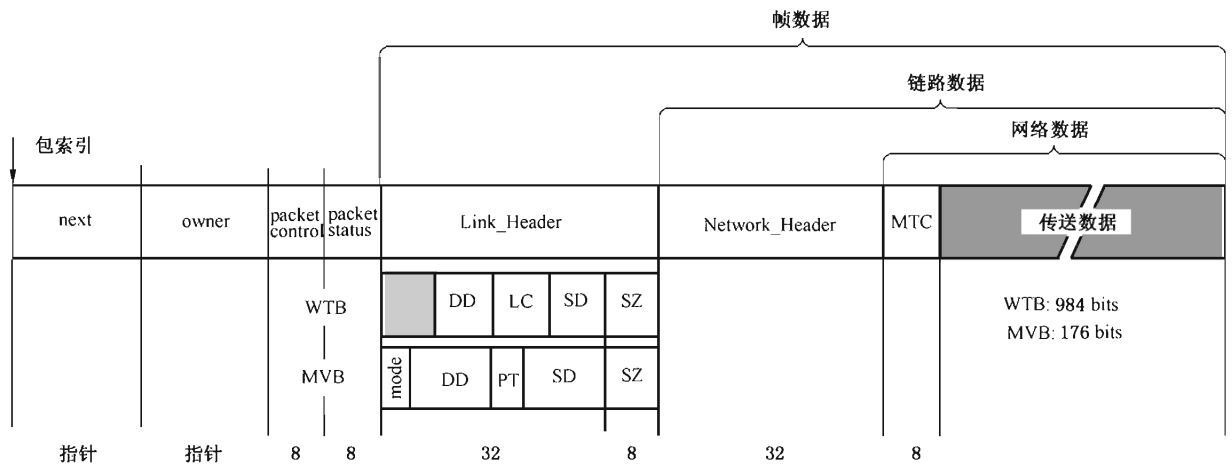
包由包描述符标识。包描述符指向将发送的包的数据以及用于包管理的一些字段。

包通过包指针引用。包指针对该包唯一,且指向 MD_PACKET 类型的包描述符,见表 77。

表 77 类型“MD_PACKET”

定义	此类型定义数据包格式
语法	typedef void * MD_PACKET;
用法	未规定 MD_PACKET 的格式

示例:图 120 是 MD_PACKET 的一个示例。



说明:

图中的数据包以下列字段开始:

- “下一包”(next):数据包第 1 个字段预留作指向另一数据包的指针。可将包链接成序列或队列。
- “属主”(owner):数据包第 2 个字段标识包的属主(池)。此字段用于处置使用过的数据包。
- “包控制”(packet control):包含链路层可读但不可修改的管理信息。
- “包状态”(packet status):由链路层修改且可被其他层读取。

包的剩余部分包含总线上实际收发的帧:

- “链路层报头”(Link_Header)对不同的总线(MVB、WTB 或其他总线)有不同的格式。链路层报头包含源设备地址、目的设备地址和其他总线专属控制信息。
- 仅链路层处理链路层报头字段。网络层不分析链路层报头,只作为参数接收。
- “长度”(size)字段表示链路层数据(Link_Data)的长度,不包括长度字段本身。

包的状态可以是:

- MD_PENDING:本包标记为用于发送;
- MD_FLUSHED:本包已从队列中清除;
- MD_SENT:本包已发送,且可回收。

图 120 包格式

6.3.3.4 过程类型“MD_GET_PACKET”

过程类型“MD_GET_PACKET”见表 78。

表 78 过程类型“MD_GET_PACKET”

定义	从池中获得一个新包,将包的属主字段设置为该池值	
语法	<pre>typedef void (* MD_GET_PACKET) (void * * pool, MD_PACKET * * packet);</pre>	
输入	pool	标识从中提取该包的包池
输出	packet	指向存储包的数据结构的指针
用法	此过程类型与能直接调用的 LM_GET_PACK 过程类型兼容	

6.3.3.5 过程类型“MD_PUT_PACKET”

过程类型“MD_PUT_PACKET”见表 79。

表 79 过程类型“MD_PUT_PACKET”

定义	将不再使用的单个包返回到由包属主字段指定的包池中	
语法	<pre>typedef void (* MD_PUT_PACKET) (MD_PACKET * packet);</pre>	
输入	packet	指向容纳该包的数据结构的指针。属主池在包中标记
用法	此过程类型与能直接调用的 LM_SEND_CONFIRM 过程类型兼容	

6.3.4 消息链路层

6.3.4.1 目的

消息服务在不同总线(WTB、MVB 或其他总线,包括并行总线、串行链路、存储器邮箱或传感器总线)上提供。

这些总线的链路层提供一套本条规定的基本服务。

一个设备的链路层与另一设备的链路层协作,以在位于同一总线上的源设备和目的设备间交换数据包,如图 121 所示。

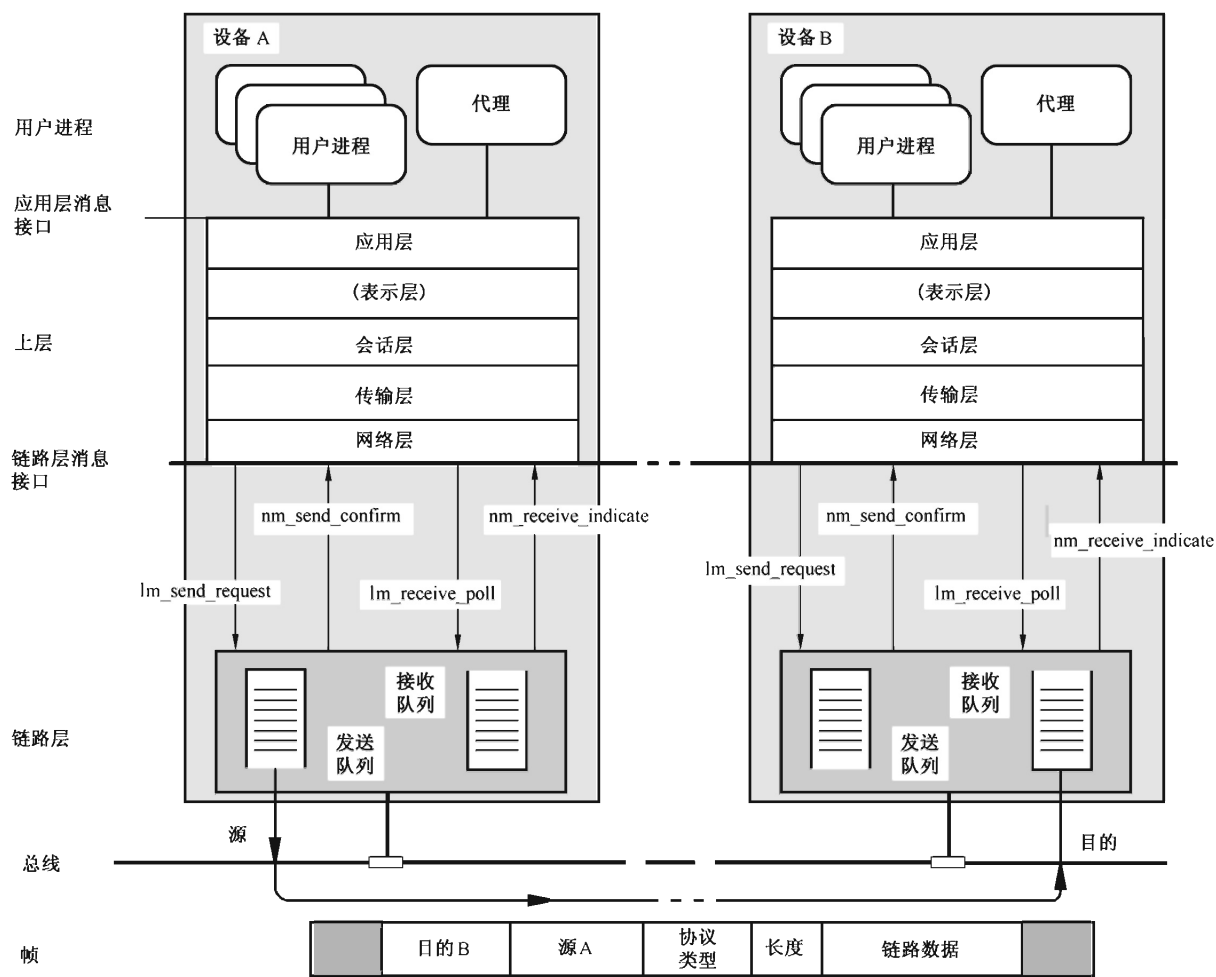


图 121 链路层数据传送

链路层进程执行总线上实际的传送。这可独立于应用及信使运行。

链路层的实现未作规定。因此,链路层服务以通用形式规定,这种形式期望存在于连接到信使的每条总线上。

6.3.4.2 链路层结构

链路层应提供一对队列:发送队列(Send_Queue)和接收队列(Receive_Queue)。发送包时网络层将包放在发送队列中准备发送;接收包时网络层从接收队列中获取从总线上接收到的包。

(路由器)站可有多条链路层,连接的每条总线都有一个链路层,如图 122 所示。

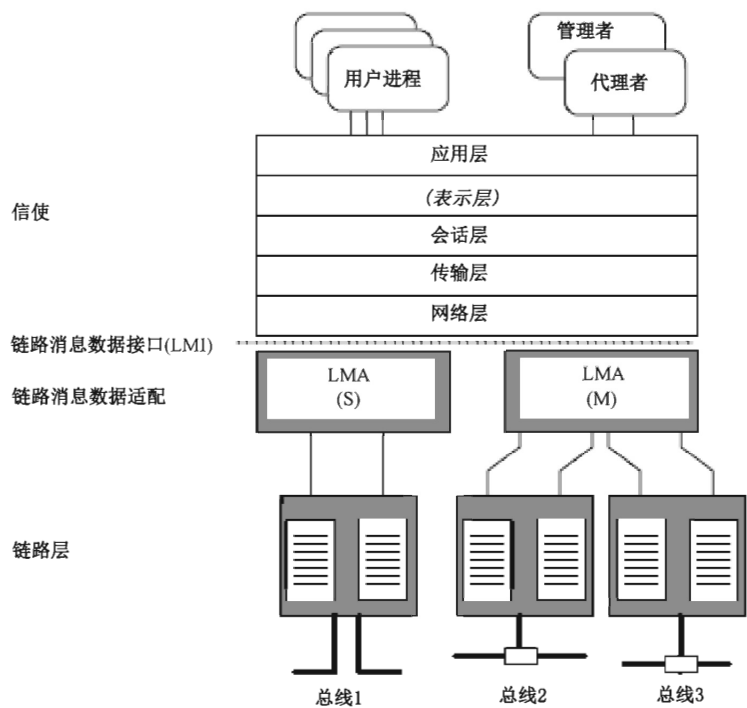


图 122 链路消息数据接口(LMI)

链路层实现的差异隐藏在链路消息数据适配(LMA)中。

注：一个 LMA 可只支持一条总线(S 型)，也可支持多条总线(M 型)。

6.3.4.3 链路层特点

6.3.4.3.1 设备地址

每一个设备应由其设备地址在总线上唯一标识。

设备地址 0 应标识本地链路层，且不应分配给特定设备。

最高的设备地址(如 8 位设备地址中“1111 1111”B)应表示对总线上所有设备的广播，且不应分配给特定设备。

链路层应在其发送的所有包中加入其设备地址作为源设备(Source_Device)，但不应使用总线专属的广播地址作为源设备。

链路层应在其发送的所有包中加入远程设备的设备地址或总线专属的广播地址作为目的设备(Destination_Device)。

注 1：设备地址的格式依赖于总线类型(WTB、MVB 或其他总线)。

注 2：连接到路由器设备的每一条总线都有一个设备地址。

6.3.4.3.2 协议类型

消息链路层应有一对队列(可消耗的缓冲区)。

若同一设备支持其他协议，则每一帧中的协议类型(Protocol_Type, PT)字段应选择用于实时协议的发送队列和接收队列。

注：协议类型字段的作用与 GB/T 9387 中链路服务访问点的作用类似。

6.3.4.3.3 优先级

链路层不区分优先级。

6.3.4.3.4 清除

节点的链路层应具有清除所有包的措施。

注：在节点接收到新的拓扑时清除是必要的。

6.3.4.3.5 包寿命

链路层应具有将发送队列或接收队列中包的寿命限制在包寿命期限(PACK_LIFE_TIME)内的措施。

包寿命期限应为 5.0 s。

注：寿命限制可由队列超时(队列清除)或队列中单个包失效实现。

6.3.4.3.6 链路层协议

链路层协议应是无连接的,即只使用数据报文。

链路层不应自动重发丢失帧。

链路层应为网络管理记录传送错误。

6.3.4.4 编组网链路层(以 MVB 为例)

编组网可由不同总线实现,MVB(见 GB/T 28029.9)作为一个特例以供参考。

常规运行期间编组网设备地址不应改变。

若 MVB 用作编组网,下述补充约定适用:

- 12 位设备地址应标识源设备和目的设备。
- 模式(Mode)字段“0001”B 应指定单播传送,“1111”B 应指定广播传送。在广播传送中,不关心设备地址。
- 若指定最高设备地址(“111111111111”B),则应选中广播传送模式。
- 4 位协议类型字段“1000”B 应标识 TCN 实时协议。

示例:MVB 消息数据帧格式,如图 123 所示。

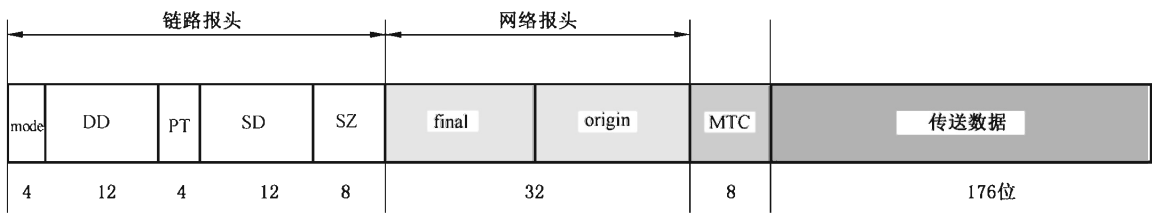


图 123 MVB 消息数据帧示例

6.3.4.5 列车总线的链路层

6.3.4.5.1 概述

列车总线可由不同的总线来实现,WTB 作为一个特例以供参考。列车总线有可变组成(如 WTB)和固定组成(如 MVB)的区别。

6.3.4.5.2 可变组成的列车总线

在可变组成的列车总线中,节点地址可动态改变,链路层应通知网络层变化并给网络层提供一个 6 位的拓扑计数器(Topo_Counter),每次组成变化时该计数器累加(模 64)。

注 1: 通过 WTB 中规定的链路层管理服务,列车总线可提供附加信息,如拓扑。尽管消息服务只关心拓扑计数器,应用需要拓扑信息以在编组类型功能中选择正确的节点地址。从编组到节点地址的映射是一个应用问题。

注 2: 由于应用在读取旧的拓扑后可能不清楚已发布了新的拓扑,因此使用拓扑计数器以验证拓扑信息。

使用 WTB 作为列车总线时,下述补充约定适用:

- WTB 节点使用初运行分配的 8 位节点地址寻址节点;
- WTB 使用目的设备 =“11111111”B 作为广播地址;
- 实时协议的链路层协议类型以链路控制字段=“x0xxx111”B(消息数据)标识。

示例: WTB 帧格式,如图 124 所示。

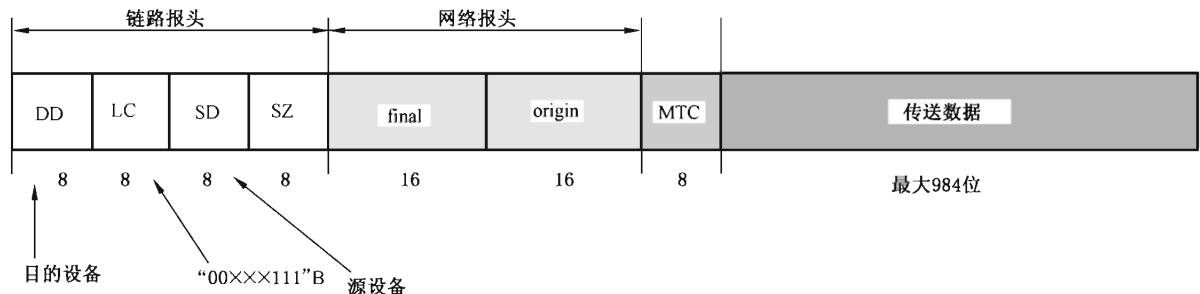


图 124 WTB 消息数据帧示例

6.3.4.6 消息数据链路层接口

6.3.4.6.1 概述

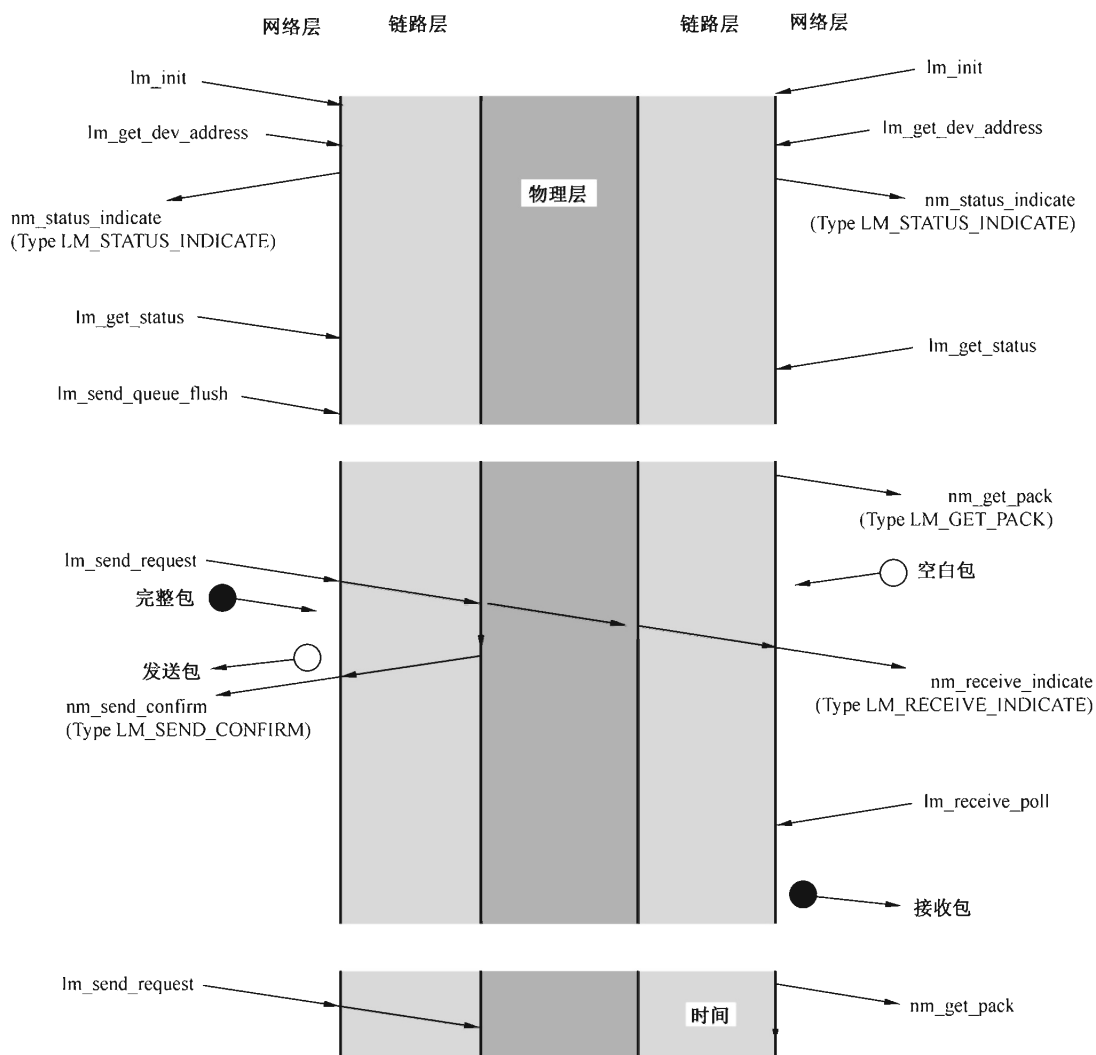
消息数据链路层接口(LMI)定义链路层提供给网络层的服务。

一致性测试不覆盖该接口。该接口在以下各条中规定,以方便移植。

以下各条不隐含特定实现。任何提供相同语义的接口都被允许。

6.3.4.6.2 LMI 原语

消息数据链路层接口(LMI)应提供图 125 所示原语,这些原语在表 80 中列出并在后续各条中规定。



注：LMI 过程前缀为 lm_，链路层类型前缀为 LM_，网络层原语前缀为 nm_。

图 125 LMI 原语

链路层调用网络层的指示过程(前缀 nm_)，这些指示过程事先订阅且具有链路层定义的类型(前缀 LM_)。

表 80 LMI 原语

名称	含义
MD_RESULT	过程的结果
lm_sent_request	请求发送一个包
LM_SEND_CONFIRM	指示包已发送
LM_GET_PACKET	获得一个空包
LM_RECEIVE_INDICATE	指示包已接收到
lm_receive_poll	轮询已接收的包
LM_STATUS_INDICATE	指示状态变化

表 80 (续)

名称	含义
lm_get_status	获取链路层状态
lm_send_queue_flush	清除发送队列
lm_init	初始化链路层
lm_get_dev_address	读设备地址

6.3.4.6.3 类型“MD_RESULT”

类型“MD_RESULT”见表 81。

表 81 类型“MD_RESULT”

定义	带返回值的 LMI 过程应编码如下：
语法	<pre>typedef enum { MD_OK 0, /* 正确执行 MD_READY 0, /* 就绪 MD_REJECT 1, /* 不接受(队列已满或空) MD_INAUGURATION 2 /* 初运行—可能不一致 } MD_RESULT;</pre>

6.3.4.6.4 过程“lm_send_request”

过程“lm_send_request”见表 82。

表 82 过程“lm_send_request”

定义	在包中加入链路报头并将包插入发送队列。 若链路层接收包用于发送,则设置其状态为 MD_PENDING	
语法	<pre>MD_RESULT lm_send_request (ENUM8 bus_id, UNSIGNED32 source, UNSIGNED32 destination, MD_PACKET * packet);</pre>	
输入	bus_id	从 16 个不同链路层中选择一个
	source	源设备设备地址。若“源”为 0,则链路层将自身设备地址填入包
	destination	目的设备设备地址或广播地址(若使用最大设备地址)
	packet	指向包包含的数据结构的指针。包长度和状态信息含在包中
返回	任意 MD_RESULT	

6.3.4.6.5 过程类型“LM_SEND_CONFIRM”

过程类型“LM_SEND_CONFIRM”见表 83。

表 83 过程类型“LM_SEND_CONFIRM”

定义	该类型过程由链路层调用,以在包发送完成或不再需要时将包返回到属主池	
语法	<pre>typedef void (* LM_SEND_CONFIRM) (MD_PACKET * packet);</pre>	
输入	packet	指向包含已发送或已清除包的数据结构的指针。包长度和状态信息含在包中
用法	a) 实际过程“nm_get_pack”(类型 LM_GET_PACK)已事先在过程“lm_init”中订阅。 b) 在调用“nm_sent_confirm”之前,期望链路层设置包状态为 MD_SENT(成功发送)或 MD_FLUSHED(队列已清除)	

6.3.4.6.6 过程类型“LM_GET_PACK”

过程类型“LM_GET_PACK”见表 84。

表 84 过程类型“LM_GET_PACK”

定义	调用该类型的过程以从属主池中请求一个空闲的包。若成功,该过程提供一个包,该包的属主字段设置成属主的参数值	
语法	<pre>typedef (* LM_GET_PACK) (void * * owner, MD_PACKET * * packet);</pre>	
输入	owner	标识从中获取包的池
	packet	指向存放包的数据结构的指针
用法	a) 实际过程“nm_get_pack”(类型 LM_GET_PACK)已事先在过程“lm_init”中订阅。 b) 链路层应仅指定在初始化时分配的池作为属主池	

6.3.4.6.7 过程类型“LM_RECEIVE_INDICATE”

过程类型“LM_RECEIVE_INDICATE”见表 85。

表 85 过程类型“LM_RECEIVE_INDICATE”

定义	在接收到包时,链路层应调用该类型过程	
语法	<pre>typedef void (* LM_RECEIVE_INDICATE) (ENUM8 bus_id);</pre>	
输入	bus_id	总线标识符(0~15)
用法	a) 实际过程“nm_receive_indicate”(类型 LM_RECEIVE_INDICATE)已事先在过程“lm_init”中订阅。 b) 若采用轮询(“lm_receive_poll”),本过程可为 NULL。 c) 本指示过程从中断服务子程序中调用,但允许调用内核,如用于唤醒信使	

6.3.4.6.8 过程“lm_receive_poll”

过程“lm_receive_poll”见表 86。

表 86 过程“lm_receive_poll”

定义	从接收队列中读一个包并将包的索引传递给网络层	
语法	<pre>MD_RESULT lm_receive_poll (ENUM8 bus_id, unsigned * source, unsigned * destination, MD_PACKET * * packet);</pre>	
输入	bus_id	该链路层的总线标识符(0~15)
返回	任意 MD_RESULT	
输出	source	源设备设备地址
	destination	目的设备设备地址或广播地址
	packet	指向包含已接收包的数据结构的指针,或 NULL(若队列为空);包长度信息含在包中
用法	a) 本过程调用后,包描述符可再次使用。 b) 若本过程的结果是 MD_OK,则接收队列空出一个位置,且本过程其他参数有意义。 c) 若接收队列为空时调用本过程,则返回结果为 MD_REJECT,packet 值为 NULL	

6.3.4.6.9 过程类型“LM_STATUS_INDICATE”

过程类型“LM_STATUS_INDICATE”见表 87。

表 87 过程类型“LM_STATUS_INDICATE”

定义	发生例外时,链路层应调用此类型过程以指示例外	
语法	typedef void (* LM_STATUS_INDICATE) (ENUM8 bus_id, MD_RESULT status);	
输入	bus_id	总线标识符(0~15)
	status	MD_OK 常规运行 MD_REJECT 总线不可用 MD_INAUGURATION 列车总线正在初始化
用法	a) 实际过程“nm_status_indicate”(类型 LM_STATUS_INDICATE)已事先在过程“lm_init”中订阅。 b) status(类型 MD_RESULT)是一个输入参数	

6.3.4.6.10 过程“lm_get_status”

过程“lm_get_status”见表 88。

表 88 过程“lm_get_status”

语法	MD_RESULT lm_get_status (ENUM8 bus_id, BITSET8 selector, BITSET8 reset, BITSET8 * status);	
输入	bus_id	总线标识符(0~15)
	selector	选择在 status 中返回的关注位。 一位表示一段信息,该位由链路层置位、由服务用户清除。 定义了以下 3 位: MD_RECEIVE_ACTIVE =1 从总线接收到正确帧时置位 MD_SEND_ACTIVE =2 发送帧后置位 MD_OVERFLOW =4 因无空闲包导致接收帧丢失时置位
	reset	选择将被清除的位
返回	任意 MD_RESULT	
输出	status	返回所选状态位的值
用法	读编组网链路层状态时,期望网络层复位 MD_SEND_ACTIVE 位	

6.3.4.6.11 过程“lm_send_queue_flush”

过程“lm_send_queue_flush”见表 89。

表 89 过程“lm_send_queue_flush”

定义	清除链路层发送队列,并设置包状态为 MD_FLUSHED。 为每一个已由 lm_send_request 插入但尚未发送的包调用一次 nm_send_confirm	
语法	<pre>MD_RESULT lm_send_queue_flush (ENUM8 bus_id /* 选项 */);</pre>	
输入	bus_id	总线标识符(0~15)
返回	任意 MD_RESULT	

6.3.4.6.12 过程“lm_init”

过程“lm_init”见表 90。

表 90 过程“lm_init”

定义	初始化链路层,清除发送队列,并从网络层订阅指示过程	
语法	<pre>MD_RESULT lm_init (ENUM8 bus_id, LM_RECEIVE_INDICATE nm_receive_indicate, LM_GET_PACKET nm_get_pack, void * * owner, LM_SEND_CONFIRM nm_send_confirm, LM_STATUS_INDICATE nm_status_indicate);</pre>	
输入	bus_id	总线标识符(0~15)
	nm_receive_indicate	网络层过程,链路层每次接收到包时调用一次
	get_pack	包池过程,链路层每次需要空闲包时调用一次
	owner	链路层用来获取包的包池
	put_pack	链路层调用处理包的包池过程
	nm_status_indicate	链路层调用指示例外(如初运行)的网络层过程
返回	任意 MD_RESULT	

6.3.4.6.13 过程“lm_get_dev_address”

过程“lm_get_dev_address”见表 91。

表 91 过程“lm_get_dev_address”

定义	读与特定链路层相对应的自身设备地址	
语法	MD_RESULT ENUM8 UNSIGNED * lm_get_dev_address (bus_id, device_address);	
输入	bus_id	总线标识符(0~15)
返回	任意 MD_RESULT	
输出	device_address	本链路层上该设备的设备地址
用法	在列车总线的链路层,获取设备地址并由此推知节点地址	

6.3.5 消息网络层

6.3.5.1 目的

- 网络层中继包,如图 126 所示:
- 从站的传输层到链路层(出境包);
 - 从链路层之一到传输层(入境包);
 - 在路由器节点中从一个链路层到另一个链路层(中转包)。

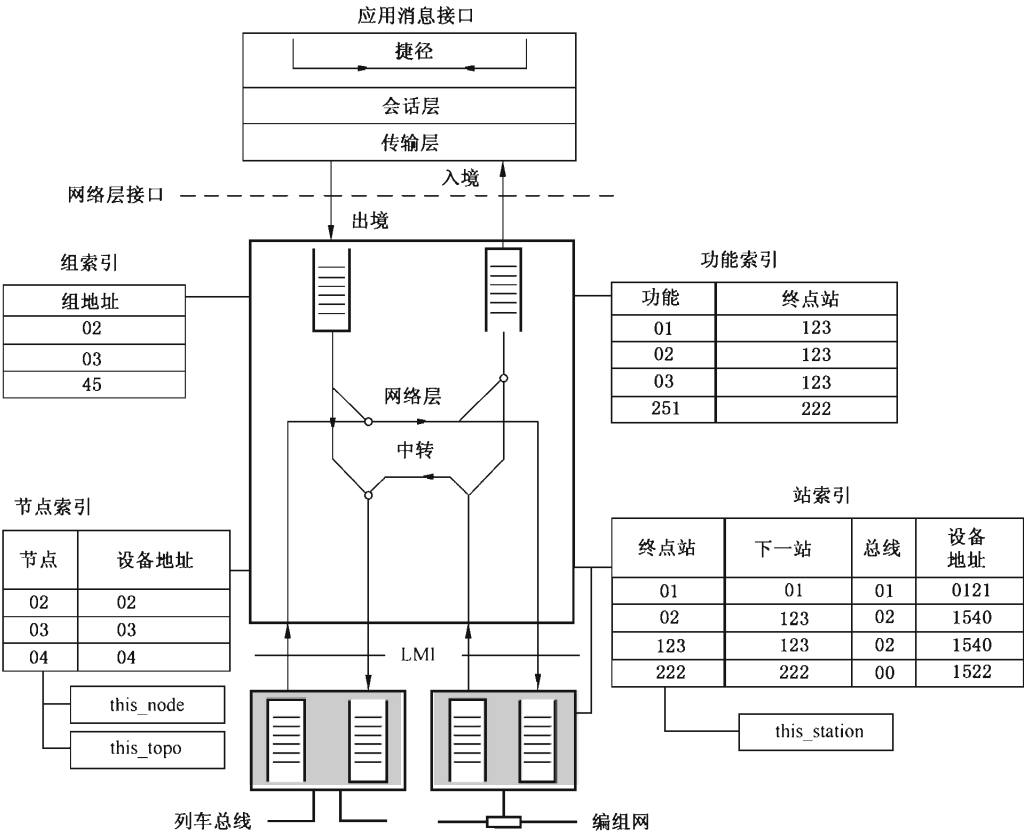


图 126 一个节点的网络层

网络层将包从起点站路由到终点站。

为此,网络层使用以下几种索引提供的映射方式:

- a) 站索引;
- b) 功能索引;
- c) 组索引;
- d) 节点索引。

网络层是无连接的。

6.3.5.2 索引

6.3.5.2.1 站索引(可选)

网络层应将站标识符映射到该站的链路层地址,反之亦然。

在终点站中,该映射可通过将站标识符扩展一个固定的前导字段以获得设备地址方式实现(简单映射)。

在路由器站上,该映射应通过表 92 所示的站索引实现。

表 92 站索引

名称	含 义
Station_Id	站标识符(站索引关键字)
Next_Station_Id	能到达的下一个站的站标识符
Bus_Id	通过其能到达下一个站的链路层
Device_Address	能到达下一个站的总线上设备地址

注 1: 站索引允许建立出境包的链路报头,但也用于识别入境包的源链路层地址。因为传输层不宜处理任意长度的特定总线地址,而只处理站标识符。

注 2: 当该站可直接到达时,Station_Id 和 Next_Station_Id 相同;当包在两个编组网之间或在编组网和传感器总线之间路由时二者有明显区别。

注 3: 站索引访问原语在应用层接口(ALI)中定义。

6.3.5.2.2 功能索引

网络层应实现将功能标识符映射到站标识符的功能索引,功能索引结构见表 93。

表 93 功能索引

名称	含 义
Function_Id	功能标识符(功能索引关键字)
Station_Id	站标识符(站索引关键字)

注 1: 功能索引属于网络层,但能被除链路层之外的所有其他层访问。

注 2: 功能索引访问原语在应用层接口(ALI)中定义。

6.3.5.2.3 组索引

参与多播通信的站的网络层应通过组索引指示本站属于哪个组,组索引结构见表 94。

表 94 组索引

名称	含 义
Membership	组列表

注：组索引访问原语在应用层接口 (ALI) 中定义。

6.3.5.2.4 节点索引(可选)

节点的网络层应将节点地址映射到列车总线上的设备地址。

当 WTB 用作列车总线时,应使用简单映射,即在 6 位节点地址前添加两个“0”以形成 8 位 WTB 设备地址。节点索引是只读的,因为其内容由初运行确定。

在简单映射不适用的固定列车组成中,应使用节点索引以将节点地址映射到设备地址,节点索引结构见表 95。

表 95 节点索引

名称	含 义
Node_Address	节点地址(节点索引关键字)
Bus_Id	与列车总线相对应的链路层
Device_Address	总线相关的设备地址

注：节点索引访问原语在应用层接口 (ALI) 中定义。

6.3.5.3 网络层常数和变量

Station_Id 或 Next_Station 的特定值见表 96。

表 96 Station_Id 或 Next_Station 的特定值

Station_Id	含 义
AM_SAME_STATION	0(常数)
AM_UNKNOWN	255(常数)
this_station	本站 8 位站标识符。若站没有分配标识符,则 this_station=AM_UNKNOWN
final_station	终点网络地址中指示的站
origin_station	起始网络地址中指示的站
next_station	网络层通过其发送或接收包的站

节点的特定值(值均为 UNSIGNED6 类型)见表 97。

表 97 节点的特定值

Node	含 义
AM_SAME_NODE	0(常数)
this_node	本节点 6 位节点地址
AM_ANY_TOPO	0(常数)任意拓朴计数器的值
origin_node	起始网络地址中的 6 位节点地址

表 97（续）

Node	含 义
final_node	终点网络地址中的 6 位节点地址
this_topo	本站拓扑计数器的值,由 am_set_current_tc 登记
my_topo	本次会话由应用指示的拓扑计数器的值
packet_topo	包中携带的拓扑计数器的值

其他过程见表 98。

表 98 其他过程

名称	含 义
multicast	若使用组寻址,则返回 true
member(group)	若使用多播且节点属于该组,则返回 true
fundi()	返回由功能索引指示的站标识符
stadi()	返回由站索引指示的链路地址

6.3.5.4 网络地址

6.3.5.4.1 格式

网络层从其传输层或链路层之一接收每一个包的终点网络地址。
网络层使用该终点网络地址构造其转发的包的链路报头和网络报头。
网络层读取入境包的链路报头和网络报头,并将它们转换为去往传输层的起始网络地址。
图 127 显示了一种方便的终点和起始网络地址编码方式。



注：本图不是传送格式,只是接口格式。

图 127 网络地址编码

6.3.5.4.2 系统地址或用户地址(fsu/osu)

网络地址第一个八位位组的最高有效位,“fsu”或“osu”含义为:

- 若 fsu 或 osu 为 1(系统地址),则下一个八位位组标识一个站;
- 若 fsu 或 osu 为 0(用户地址),则下一个八位位组标识一个功能。

起始地址中的“osu”位应与终点地址中的“fsu”位具有相同的值。

6.3.5.4.3 组地址或单个地址(fgi/ogi)

终点网络地址第一个八位位组的次高有效位,“fgi”含义为:

- 若 fgi 为 1,则后 6 位为组地址;
- 若 fgi 为 0,则后 6 位为单个地址。

起始地址中的“ogi”位应与终点地址中的“fgi”位具有相同的值。

6.3.5.4.4 节点或组地址

6 位“origin_node”表示:

- 起始节点单个节点地址。

6 位“final_node_or_group”表示:

- 若 fgi 位指定组,则为组地址;
- 否则,为单个节点地址。

6.3.5.4.5 功能或站标识符

网络地址第二个八位位组:

- 若 fsu/osu 指定系统地址,则为站标识符;
- 若 fsu/osu 指定用户地址,则为功能标识符。

6.3.5.4.6 链路地址

第三个八位位组,即“next_station”,标识连接到同一个通过其发送或接收包的节点上的站(起始站或终点站不需要)。

网络层从其站索引获得相应链路地址(总线标识符和设备地址),或对于入境包从总线标识符和设备地址推导出下一站站标识符。

“next_station”字段也可指示自身站(AM_SAME_STATION)或未知站(AM_UNKNOWN)。

通过列车总线发送的节点没有已定义的下一站(AM_UNKNOWN),但是若节点使用节点索引则下一站是该索引项。

6.3.5.4.7 拓扑计数器

起始网络地址和终点网络地址第四个八位位组包含拓扑计数器。

该八位位组最高有效位,即“frv”或“orv”,为 0。

次高有效位(“ftv”或“otv”)指示低 6 位包含一个有效的计数值(使用组地址时除外)。

6.3.5.5 网络报头编码

6.3.5.5.1 出境包和入境包

出境包的终点网络地址和起始网络地址用于构造 MVB 的链路报头和网络报头字段(不使用拓扑

计数器),如图 128 所示。

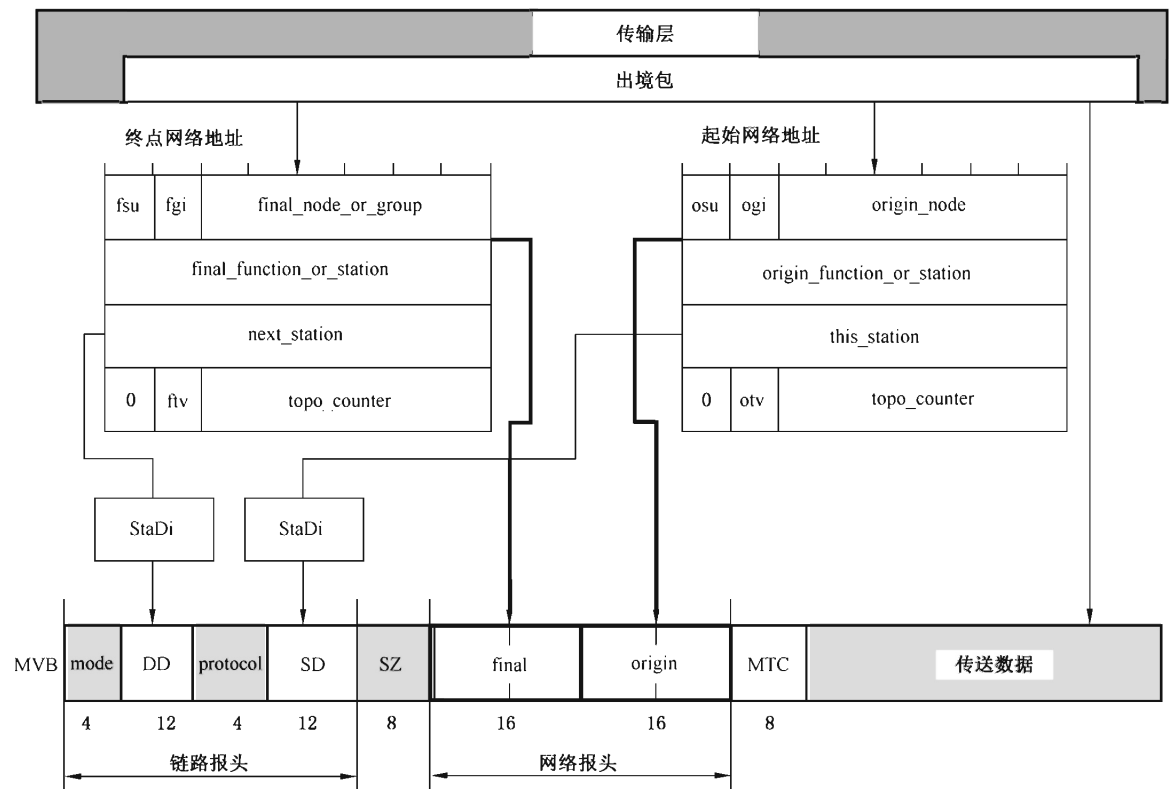


图 128 构造出境包的地址

6.3.5.5.2 列车总线上网络报头编码

列车总线上网络报头应根据以下规范编码,WTB 编码如图 129 所示。

```
Network_Header ::= RECORD
{
    fsu                                ENUM1
    {
        USER                          (0),                ——使用用户地址(功能)
        SYSTEM                        (1)                  ——使用系统地址(站)
    },
    fgi                                UNSIGNED1,            ——若接组地址则为 1;若接单个地址则为 0
    ONE_OF [fgi]
    {
        [0]    final_node              UNSIGNED6,
        [1]    final_group              UNSIGNED6
    },
    ONE_OF [fsu]
    {
        [USER] final_function            UNSIGNED8,
```


[SYSTEM] final_station

UNSIGNED8

}

osu

ENUM1,

——与 fsu 相同,否则保留

ogi

UNSIGNED1,

——与 fgi 相同(1 = 多播)

origin_node

UNSIGNED6,

——起始节点总是单个节点

ONE_OF [fsu]

{

[USER] origin_function

UNSIGNED8,

——若是用户地址

[SYSTEM] origin_station

UNSIGNED8

——若是系统地址

}

}

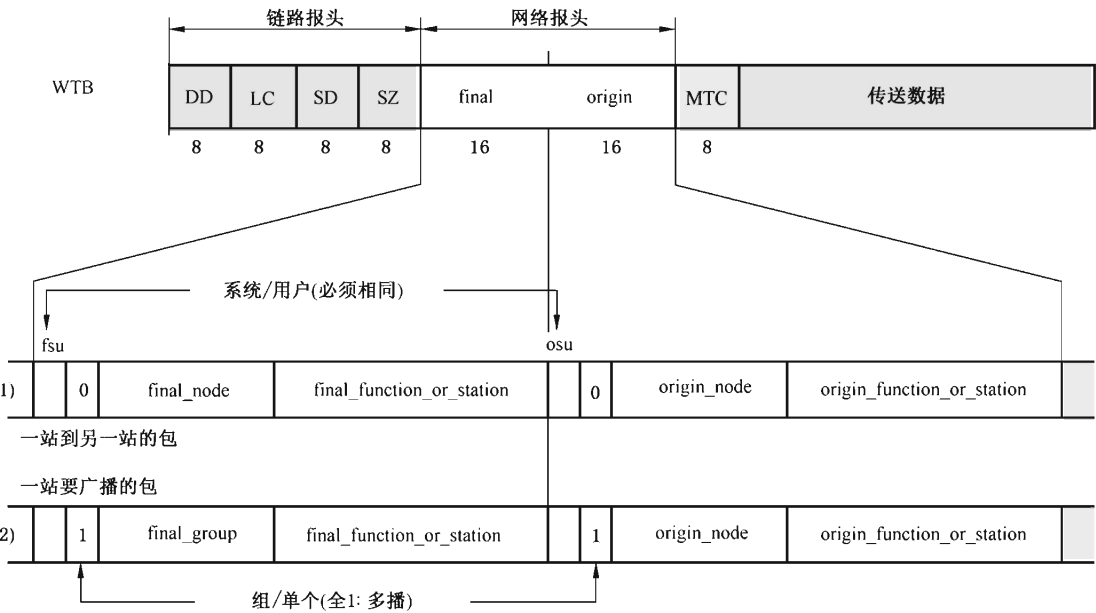


图 129 列车总线上网络地址编码

应定义起始节点和终点节点(或组),即不能为 AM_UNKNOWN。

终点地址和起始地址应使用相同的寻址类型(系统地址或用户地址),snu 位应置成相同的值。

起始地址应为单个节点地址。

若终点地址指定组(多播),则 fgi 位和 ogi 位应均为 1。

6.3.5.6 网络层路由

6.3.5.6.1 工况

表 99 列出的 9 种工况定义了一个完整的网络层行为,该网络层连接到:

- 传输层;
- 一个或多个编组网链路层;
- 一个列车总线链路层。

表 99 路由工况

工况	来自	去向	注释
1	本传输层	本传输层	在同一设备中捷径
2	本传输层	任意编组网	可能有多个编组网
3	本传输层	列车总线	站是节点
4	任意编组网	本传输层	可能有多个编组网
5	任意编组网	另一编组网	路由器功能
6	任意编组网	列车总线	路由器功能
7	列车总线	本传输层	站是节点
8	列车总线	任意编组网	路由器功能
9	列车总线	列车总线	不可预知

连接到编组网的终端站只有工况 1、2 和 4。
连接到列车总线的终端站只有工况 1、3 和 7。
连接到两个编组网的路由器站只有工况 1、2、4 和 5。

6.3.5.6.2 返回路径检查

网络层应保证属于同一呼叫消息及其相应应答消息的所有包采用相同路径。
网络层应对接收到的每一个包检查是否能回送包到其起始地址，若不能则网络层不应转发该包。
假定接收包已将起始地址和终点地址、源地址和目的地址交换，并通过检查下一站在这种情况下中是否有定义而完成上述检查。
初始化时，当呼叫站在站索引中没有登记项时，“下一站”字段可能未定义。
若从其站索引之外的一个站接收到系统消息，则网络层应假定起始地址字段中的起始站具有对应于源设备的链路地址，并将其送入站索引。在这种情况下用户消息将被拒绝。
注：若必要，隐式项可由管理消息后续校正。

6.3.5.6.3 拓扑一致性

为防止消息传送期间列车总线组态变化，通过编组网发往或来自节点的数据包包含拓扑计数器。
网络层应在其变量 this_topo 中保存拓扑计数器最新值。
注 1：this_topo 的值由 AMI 服务“am_set_current_tc”传送到网络层。
注 2：若站是节点，则每次初运行发生时 this_topo 递增。
注 3：在发出呼叫时，应用指示 my_topo 的值。若应用不知道拓扑或不关心一致性，则设置 my_topo = AM_ANY_TOPO。
传输层将 my_topo 作为网络地址的一部分传送给网络层。
节点网络层应将 this_topo 和传入包中 packet_topo 的值做比较。
当节点从其编组网接收包时：
——若起始地址中的 packet_topo 的值等于其 this_topo 的值或为 AM_ANY_TOPO，则节点应在起始地址中插入自身节点地址(this_node)，并在列车总线上转发该包；
——若起始地址中的 packet_topo 的值既不等于其 this_topo 的值、也不等于 AM_ANY_TOPO，则节点应取消传送，如 6.3.5.6.4 所述。
当节点接收来自列车总线上另一个节点的包时：

- 若包指向其站或功能之一,则节点应将 this_topo 作为 packet_topo 插入到终点地址;
- 终点站检查节点发送的 packet_topo 是否与 this_topo 相同,否则终点站应取消传送,如 6.3.5.6.4 所述。

6.3.5.6.4 取消传送

为取消传送,网络层应向其传输层转发一个破坏包,传输层向包起始节点发送解联请求,解联原因是 AM_INAUG_ERR。

初运行期间,节点应通过解联请求响应任何来自该编组网且寻址另一编组网的包,解联原因是 AM_INAUG_ERR。

当接收到解联原因是 AM_INAUG_ERR 的解联请求后,设备应取消其在列车总线上所有进行中的连接。

注:无连接的网络层不对协议起作用,因此这一任务由传输层完成。这一情形只在初运行出错时会发生。

6.3.5.6.5 路由算法

路由器应按如下规定转发包:

- 表 100 为出境包转发;
- 表 101 为来自编组网的入境包和中转包转发;
- 表 102 为来自列车总线的入境包和中转包转发。

表 100 来自传输层的包的路由

来自	去往	条件
本传输层	任意	传输层在网络地址中将下一站和拓扑计数器告知网络层。 传输层不检查出境包的拓扑计数器
(1)	传输层	单播:不可能,因为当通信方位于同一个站时会话层将取网络捷径。 多播:本站传送的广播包可能作为从链路层环回的结果寻址到驻留本站的功能,当(member AND (fundi(function_id) = this_station))时
(2)	编组网 链路层	(final_node = AM_SAME_NODE) AND (next_station <> AM_SAME_STATION) AND (next_station <> this_station) AND (Link_Address of next_station = defined)
(3)	列车总线 链路层	(next_station = AM_SAME_STATION) OR ((next_station = this_station) AND (final_node <> AM_SAME_NODE)) 这一情形只在节点上存在。 通过指定(next_station = AM_SAME_STATION),传输层指示通过列车总线转发该包。 传输层检查是否有列车总线连接(例如,无正在进行的初运行)

表 101 来自编组网的包的路由

来自	去往	条件
编组网	任意	<p>对于所有包,网络层通过分析终点网络地址确定下一站:</p> <pre> IF (final_node <> AM_SAME_NODE) THEN next_station = fundi(AM_ROUTER_FCT) ELSE IF User THEN next_station = fundi(function_id) ELSE next_station = final_station END next_station = stadi(station_id) </pre>
(4)	传输层	<p>其他条件未覆盖的所有情况。</p> <p>包不发送到列车总线,没有站接收该包。</p> <p>网络层将源设备链路地址告知下一站传输层。</p> <p>若通信方不存在,则期望传输层发送解联请求帧</p>
(5)	另一编组网	<pre> ((final_node = AM_SAME_NODE) OR (final_node = this_node)) AND ((packet_topo = this_topo) OR (packet_topo = AM_ANY_TOPO)) AND ((final_station <> this_station) AND (final_station <> AM_SAME_NODE) AND (final_station <> AM_UNKNOWN)) </pre> <p>若(final_node<>AM_SAME_NODE),则即使包不通过列车总线发送,也要检查其拓扑</p>
(6)	列车总线	<pre> ((final_node <> this_node) AND (final_node <> AM_SAME_NODE) AND ((packet_topo = this_topo) OR (packet_topo = AM_ANY_TOPO))) OR member </pre> <p>只有包的 packet_topo 正确或使用多播,才在通过列车总线发送包</p>

表 102 来自列车总线的包的路由

来自	去往	条件
列车总线		<p>与来自编组网的包的主要差别在于拓扑计数器的处理。</p> <p>在来自列车总线的包中,节点将其 this_topo 插入 final_node 的位置,从而终点站能检查</p>
(7)	传输层	<pre> ((final_node = this_node) OR (member)) AND ((final_station = this_station) OR (final_station = AM_SAME_STATION) OR (final_station = AM_UNKNOWN)) </pre> <p>包拟传送给该节点且没有其他节点接收。</p> <p>传输层判定通信方是否存在</p>

表 102 (续)

来自	去往	条件
(8)	另一编组网	((final_node = this_node) OR (member)) AND ((final_station <> this_station) AND (final_station <> AM_SAME_STATION) AND (final_station <> AM_UNKNOWN)) 存在连接到该节点的站接收包
(9)	列车总线	这种工况不可预知

6.3.5.7 网络层接口

网络层到传输层的接口未规定,该接口不开放。
应用程序可通过应用消息接口建立和查阅索引,理论上索引是网络层的一部分。

6.3.6 消息传输层

6.3.6.1 用途

本条描述两个传送协议：
——用于点对点通信的消息传送协议(MTP)；
——(可选的)用于多播通信的多播协议(MCP)。

6.3.6.2 消息传送协议

传输层将消息从一个生产者传送到一个消费者,且提供以下服务：

- a) 长消息分段成用于传送的固定长度包；
- b) 通过滑动窗口协议的端到端的流量控制和差错恢复；
- c) 取消传送。

MTP 在一条消息期间只在一个方向上打开连接。
MTP 支持在任意两个应用进程之间同时传送多条消息。
MTP 支持同一个应用同时送传多条消息。每一个连接由会话层引用参数 session_ref 区分。
但是,MTP 在相同两个生产者和消费者之间同一方向同时仅支持一个连接。

注：术语“生产者”和“消费者”是指传输层上消息的发送方和接收方。术语“起始”和“终点”是指网络上终端通信方。术语“源”和“目的”是指同一总线上通信双方的设备地址。

协议在每个站中由信使执行,理论上信使作为独立进程与应用程序并行运行。
呼叫者通过应用层接口向信使发送呼叫消息。
信使的会话层打开连接(并在以后关闭连接)。
传输层将消息分成足够小到可装入总线帧的数据包序列,并发送包给网络层。
网络层查询其功能索引和站索引以翻译包的地址,并转发包给链路层。
远端信使将完整消息到达指示给应答者,然后应答者能以应答消息反向应答。
生产者和消费者的传输层执行的消息传送协议执行流量控制和差错恢复,以避免包丢失或复制。
若两个应用驻留在同一站上,则会话层取网络捷径并不分段消息。例如,用户应用可通过发送消息

给本地代理者请求其服务,而不访问网络。

每个站上的信使执行传送协议。生产者处的信使将消息分成数据包,消费者处的信使通过控制包确认。

6.3.6.3 包交换

6.3.6.3.1 概述

消息传送应分成三个阶段:

- a) 建立连接;
- b) 已确认的数据传送;
- c) 解连。

示例: 图 130 显示了窗口大小为 1 的简单包交换。

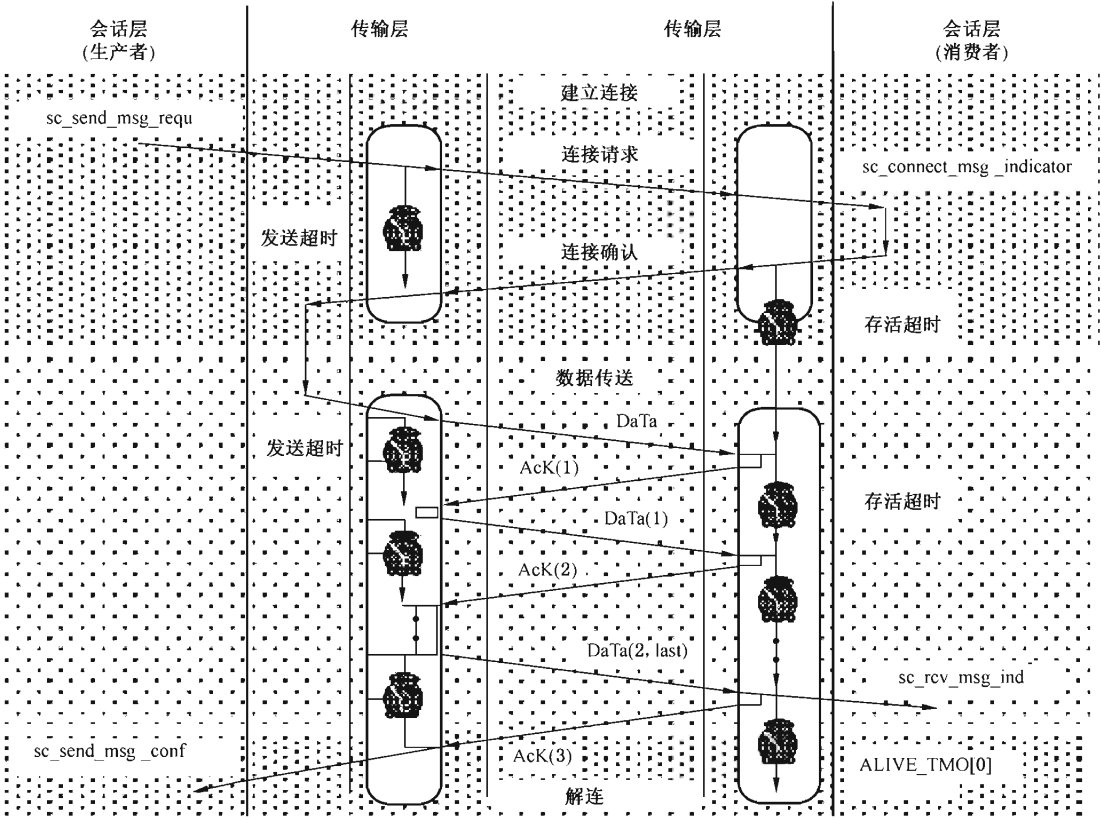


图 130 传送包交换

6.3.6.3.2 建立连接

生产者通过发送连接请求包请求消费者接收消息。该包包含消息总长度、包长度、窗口大小和由传输层提供的连接参考。在需要重复请求时,该连接参考唯一标识连接请求。

消费者若接收消息,则用连接确认应答连接请求;若拒绝消息,则用解联请求应答。

窗口大小在连接建立期间协商:生产者建议一个信用值,消费者以其接收的信用值应答,消费者应答的信用值不可大于生产者建议的信用值。

相同规则适用于包长度。连接请求包是网络中长度最短的包,其长度由 MVB 能发送的 256 位决定。

消费者返回的信用值和包长度对同一消息所有后续包保持不变。

生产者可在连接请求包中插入最多 14 个八位位组的数据。

小于或等于 14 个八位位组的消息可整体传送给消费者而不需要数据包。然后,连接确认包确认整个消息的接收。

若连接请求未确认或被取消,则程序应在解联之前尝试最多 3 次重传。

6.3.6.3.3 已确认的数据传送

生产者向消费者发送消息的各数据包。在没有接收到确认之前可发送最多 $AcpCredit$ (已接收的信用值)个包。

消费者应使用一个确认包指示其期望的下一个包的编号,以肯定确认接收到数据包。

消费者也可批量确认,即通过只确认具有最大序列号的包的方式同时确认多个包。

生产者应能处理批量确认。

当包停止被确认时,传输层应在解联之前重发包,最多 3 次。

消费者可通知生产者接收到错序包。此时,消费者应发送否定确认包,指示需要从哪个包开始重传。

6.3.6.3.4 解联

若没有显式取消传送,则解联是隐式的。但是,若丢失确认,连接不会立即断开,后续包仍然可到达。

当生产者或消费者发出解联请求包时,解联是显式的。

接收到解联请求的任意一方(以 AM_REM_CANC 原因响应解联请求或连接请求除外)以解联确认包应答。

该规则的一个例外:路由器站若不能正确转发包,例如由于组成改变,则能发送解联请求。

注:由于网络层不对协议起作用,该包被转发到路由器的传输层,由该传输层发出解联请求。

6.3.6.4 连接参考

每一个连接请求应由连接参考参数标识。

连接参考应为一个 16 位数,启动时设置为任意值(例如,取自实时时钟)。

该站每一次新的传出连接时,传输层应将连接参考的值加 1。

呼叫应用应使用连接参考配对呼叫-应答。

注 1: 连接参考区分重复的连接请求(若发生传送错误)和新的不同请求。

注 2: 连接参考在状态转换表中称作 $conn_ref$ 。

6.3.6.5 传输层包

6.3.6.5.1 包类型

传输层应处理 7 种包类型,如图 131 所示,并在以下各条规定。

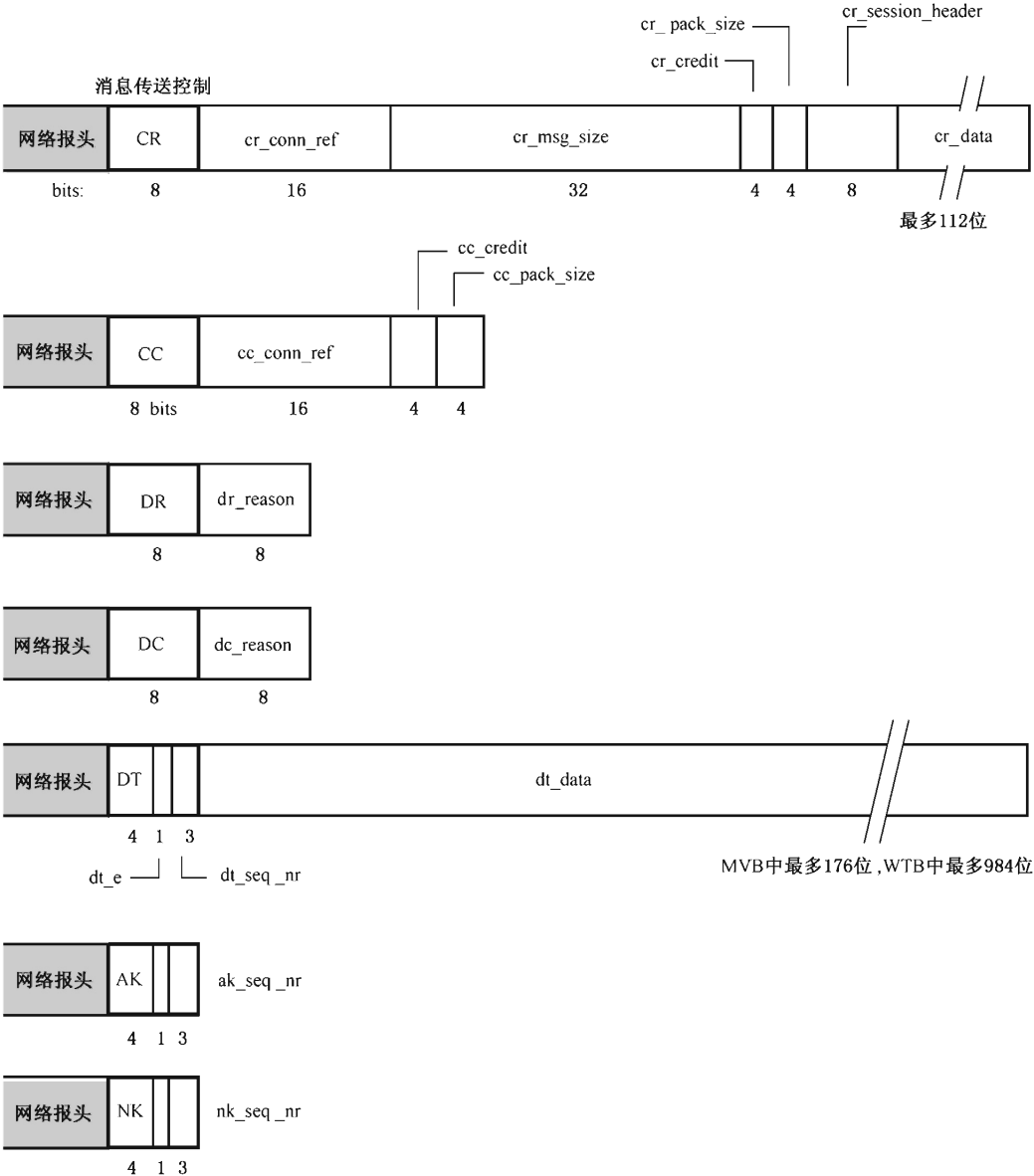


图 131 包格式(传输层正文)

注 1: 网络报头在网络层规范中规定。

注 2: 链路报头在链路层规范中规定(与总线相关)。

6.3.6.5.2 消息传送控制编码

消息传送控制字段应按表 103 规定编码。

表 103 消息传送控制字段编码

包类型	包	缩写	编码							
控制 (单播)	连接请求 (Connect_Request)	CR	cl	0	0	0	0	0	0	0
	连接确认 (Connect_Confirm)	CC	cl	1	0	0	0	0	0	1
	解联请求 (Disconnect_Request)	DR	cl	co	0	0	0	0	1	0
	解联确认 (Disconnect_Confirm)	DC	cl	co	0	0	0	0	1	1
多播	广播连接 (Broadcast_Connect)	BC	1	0	0	0	1	0	bc_rept	
	广播数据 (Broadcast_Data)	BD	1	0	0	0	1	1	0	0
	广播重复 (Broadcast_Repeat)	BR	1	0	0	0	1	1	0	1
	广播停止 (Broadcast_Stop)	BS	1	0	0	0	1	1	1	0
信息	数据 (Data)	DT	cl	0	0	1	dt_e	dt_seq_nr		
	肯定确认 (Acknowledgement)	AK	cl	1	1	0	ak_e	ak_seq_nr		
	否定确认 (Negative Acknowledgement)	NK	cl	1	1	1	nk_e	nk_seq_nr		

控制字段的第 1 位(cl)应区分呼叫报文(1)和应答报文(0)。

第 2 位(co)应区分起始作为消费者(1)或生产者(0)。

在消息的最后一个包中,dt_e 位(传送结束)应置“1”。

未定义的组合保留。

注：理论上 CL 是会话层的一部分,但也用于传输层。

6.3.6.5.3 Am_Result 编码

传送结果应在包中由 Am_Result 类型字段编码：

Am_Result ::= ENUM8

{		
AM_OK	(0),	——成功结束
AM_FAILURE	(1),	——未规定的失效
AM_BUS_ERR	(2),	——无总线传送可能

AM_REM_CONN_OVF	(3),	—— 过多传入连接
AM_CONN_TMO_ERR	(4),	—— 连接请求未回答
AM_SEND_TMO_ERR	(5),	—— 发送超时 SEND_TMO 到期
AM_REPLY_TMO_ERR	(6),	—— 未收到应答
AM_ALIVE_TMO_ERR	(7),	—— 存活超时 ALIVE_TMO 到期
AM_NO_LOC_MEM_ERR	(8),	—— 内存或定时器不足
AM_NO_REM_MEM_ERR	(9),	—— 合作方的内存或定时器不足
AM_REM_CANC_ERR	(10),	—— 被合作方取消
AM_ALREADY_USED	(11),	—— 已完成相同操作
AM_ADDR_FMT_ERR	(12),	—— 地址格式错误
AM_NO_REPLY_EXP_ERR	(13),	—— 非期望应答
AM_NR_OF_CALLS_OVF	(14),	—— 请求呼叫过多
AM_REPLY_LEN_OVF	(15),	—— 应答消息过长
AM_DUPL_LINK_ERR	(16),	—— 重复的会话错误
AM_MY_DEV_UNKNOWN_ERR	(17),	—— 本设备未知
AM_NO_READY_INST_ERR	(18),	—— 无就绪的应答者事例
AM_NR_OF_INST_OVF	(19),	—— 应答者事例过多
AM_CALL_LEN_OVF	(20),	—— 呼叫消息过长
AM_UNKNOWN_DEST_ERR	(21),	—— 合作方设备未知
AM_INAUG_ERR	(22),	—— 发生了列车初运行
AM_TRY_LATER_ERR	(23),	—— (仅内部使用)
AM_FIN_NOT_REG_ERR	(24),	—— 终点地址未注册
AM_GW_FIN_NOT_REG_ERR	(25),	—— 终点地址未在路由器中注册
AM_GW_ORI_REG_ERR	(26),	—— 起始地址未在路由器中注册
AM_MAX_ERR	(31)	—— 最高的系统代码
}		—— 用户代码大于 31

注：枚举类型 Am_Result 对应参数 AM_RESULT, AM_RESULT 由应用层过程、会话层过程和传输层过程返回。

AM_RESULT 是 C 语法的参数类型, Am_Result 定义了用于传送的该类型编码。

6.3.6.5.4 包编码

包格式应如下规定,见图 132~图 142。

Message_Packet ::= RECORD

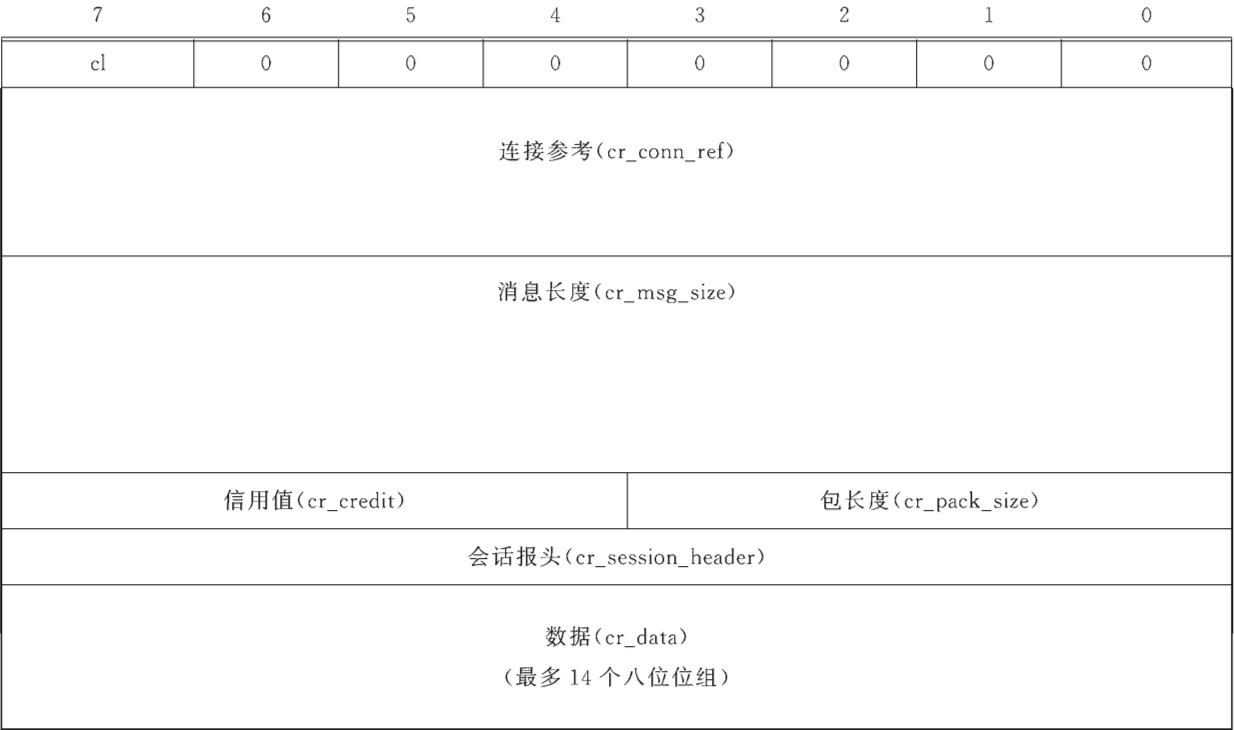
{		
call_not_reply	ENUM1	—— 第一位区分呼叫和应答
{		
REPLY_MSG	(0),	—— 应答消息
CALL_MSG	(1)	—— 呼叫消息
},		
consumer_not_producer	ENUM1	—— 第二位区分消费者和生产者
{		
CONSUMER	(0),	—— 来自消费者
PRODUCER	(1)	—— 来自生产者
},		
packet_kind	ENUM2	—— 第三位和第四位区分包的种类

```

{
    CONTROL      (0),          ——控制消息
    DATA        (1),          ——数据传送
    ACK          (2),          ——肯定确认
    NAK          (3),          ——否定确认
},
ONE_OF [packet_type]
{
    [CONTROL]    Control_Packet, ——控制
    [DATA]       Data_Packet,    ——数据传送
    [ACK]        Ack_Packet,     ——肯定确认
    [NAK]        Nak_Packet      ——否定确认
}
}

Control_Packet ::= RECORD
{
    command ::= ENUM4          ——MTC 八位位组的后 4 位表示命令
    {
        CR      (0),          ——连接请求
        CC      (1),          ——连接确认
        DR      (2),          ——解联请求
        DC      (3),          ——解联确认
        BC1     (8),          ——广播请求第一次尝试(可选)
        BC2     (9),          ——广播请求第二次尝试(可选)
        BC3     (10),         ——广播请求第三次试探(可选)
        BS      (12),         ——广播停止(可选)
        BR      (13),         ——广播重复(可选)
        BD      (14),         ——广播数据(可选)
    },
    ONE_OF [command]          ——取决于 MTC
    {
        [CR]    Connect_Request,
        [CC]    Connect_Confirm,
        [DR]    Disconnect_Request,
        [DC]    Disconnect_Confirm,
        [BC1]   Broadcast_Connect, ——见 6.3.7.5
        [BC2]   Broadcast_Connect, ——见 6.3.7.5
        [BC3]   Broadcast_Connect, ——见 6.3.7.5
        [BR]    Broadcast_Repeat, ——见 6.3.7.5
        [BS]    Broadcast_Stop,   ——见 6.3.7.5
        [BD]    Broadcast_Data    ——见 6.3.7.5
    }
}

```



```
Connect_Request ::= RECORD
{
    cr_conn_ref      UNSIGNED16,      ——16 位连接参考
    cr_msg_size      UNSIGNED32,      ——以八位位组计的消息总长度,最大 4 GB
    cr_credit        UNSIGNED4,       ——建议的信用值,“0000”B=0(停止通信),“0111”B=7(最大值)
    cr_pack_size     ENUM4            ——建议的包长度
    {
        MVB_PACKET   (0),            ——传送数据=22 个八位位组/包(MVB)
        WTB_PACKET   (1)            ——传送数据=123 个八位位组/包(WTB)
    },
    cr_session_header Am_Result,      ——在请求应答的呼叫中 AM_OK
    cr_data           ARRAY[14]OF WORD8 ——最多 14 个八位位组的传送数据
}
```

图 132 连接请求

7	6	5	4	3	2	1	0
cl	1	0	0	0	0	0	1
连接参考(cc_conn_ref)							
信用值(cc_credit)				包长度(cc_pack_size)			

Connect_Confirm ::= RECORD

{

cc_conn_ref UNSIGNED16, ——与连接请求中接收到的值相同

cc_credit UNSIGNED4, ——接收的信用值,“0000”B=0(停止通信),“0111”B=7(最大值)

cc_pack_size ENUM4 ——接收的包长度

{

 MVB_PACKET (0), ——传送数据=22个八位位组/包(MVB)

 WTB_PACKET (1) ——传送数据=123个八位位组/包(WTB)

 ——其他值保留

},

}

图 133 连接确认

7	6	5	4	3	2	1	0
cl	co	0	0	0	0	1	0
解联原因(dr_reason)							

Disconnect_Request ::= RECORD

{

dr_reason Am_Result ——解联原因

}

图 134 解联请求

7	6	5	4	3	2	1	0
cl	co	0	0	0	0	1	1
解联原因(dr_reason)							

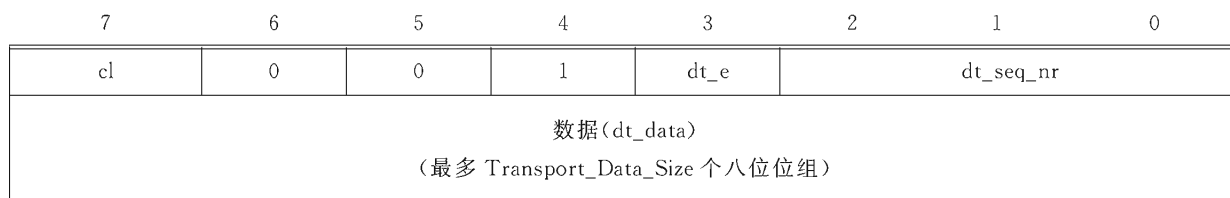
Disconnect_confirm ::= RECORD

{

dc_reason Am_Result ——若解联成功,则原因为 AM_OK

}

图 135 解联确认



Data_Packet::=RECORD	——从 MTC 八位位组后 4 位开始
{	
dt_e	BOOLEAN1, ——1=消息结束(不是会话结束)
dt_seq_nr	UNSIGNED3, ——生产者所见的该包序号
dt_data	ARRAY[transport_data_size]OF WORD8
	——消息尾部可能除外
}	

图 136 数据包



Ack_Packet::=RECORD	——MTC 八位位组后 4 位
{	
ak_e	ENUM1(=0), ——总为 FALSE(0)
ak_seq_nr	UNSIGNED3 ——消费者期待的下一包序号
}	

图 137 肯定确认包



Nak_Packet::=RECORD	——MTC 八位位组后 4 位
{	
nk_e	BOOLEAN1, ——总为 FALSE(0)
nk_seq_nr	UNSIGNED3 ——消费者期待的下一包序号
}	

图 138 否定确认包

下列包只在多播中使用(可选):

7	6	5	4	3	2	1	0
1	0	0	0	1	0	bc_rept	
连接参考(bc_run_nr)							
消息长度(bc_msg_size)							
用户数据(bc_data) (最多 18 个八位位组)							

Broadcast_Connect::=RECORD

——在此为 6.3.7.5 规定

{

——BC1,BC2 和 BC3 由 bc_rept 区分

bc_run_nr UNSIGNED16,

——标识此消息的连接参考

bc_msg_size UNSIGNED16,

——以八位位组计的消息总长度

bc_data ARRAY[18]OF WORD8

——最多 18 个八位位组的用户数据

}

图 139 广播连接(BC1、BC2、BC3)

7	6	5	4	3	2	1	0
1	0	0	0	1	1	0	0
连接参考(bd_run_nr)							
数据偏置(bd_offset)							
用户数据(bd_data) (最多 18 个八位位组)							

Broadcast_Data::=RECORD

——在此为 6.3.7.5 规定

{

bd_run_nr UNSIGNED16,

——标识此消息的连接参考

bd_offset UNSIGNED16,

——消息中以八位位组计的 bd_data 偏置

bd_data ARRAY[18]OF WORD8

——最多 18 个八位位组的用户数据

}

图 140 广播数据

7	6	5	4	3	2	1	0
1	0	0	0	1	1	0	0
连接参考(br_run_nr)							
数据偏置(br_offset)							

Broadcast_Repeat::=RECORD

——在此为 6.3.7.5 规定

{

br_run_nr UNSIGNED16,

——标识此消息的连接参考

br_offset UNSIGNED16

——丢失的用户数据以八位位组计的偏置

}

图 141 广播重复

7	6	5	4	3	2	1	0
1	co	0	0	1	1	1	0
连接参考(bs_run_nr)							

Broadcast_Stop ::= RECORD

——在此为 6.3.7.5 规定

{

bs_run_nr UNSIGNED16 ——标识此消息的连接参考

}

图 142 广播停止(BSC、BSO)

6.3.6.6 状态转换图

6.3.6.6.1 状态

消息传送协议应为每一个生产者或消费者实现一个 MTP 状态机,状态机状态如表 104 所列。

表 104 MTP 状态

状态名	发生于	简述
DISC(解联)	生产者/消费者	已解联
SETUP(建立)	生产者	等待连接确认(CC)包
SEND(发送)	生产者	开放以发送一条消息
SEND_CANC(取消发送)	生产者	已发送解联请求(DR)包,等待解联确认(DC)包或解联请求(DR)包
CLOSED(关闭)	生产者	已收到解联请求(DR)包
LISTEN(侦听)	消费者	等待确认连接确认(CC)包
RCV_CANC(取消接收)	消费者	已发送解联请求(DR)包,等待解联确认(DC)包或解联请求(DR)包
RECEIVE(接收)	消费者	开放接收一条消息且所有已接收包已确认
PEND_ACK(悬挂确认)	消费者	开放接收一条消息且并非所有已接收包已确认
FROZEN(冻结)	消费者	已接收具有带结束标志的数据包[DT(EOT)]包的完整消息
LISTEN_FROZE(冻结侦听)	消费者	已接收具有连接请求(CR)包的完整消息

传输层事例(生产者或消费者)的状态及其转换如图 143 所示,并在以下各条中规定。

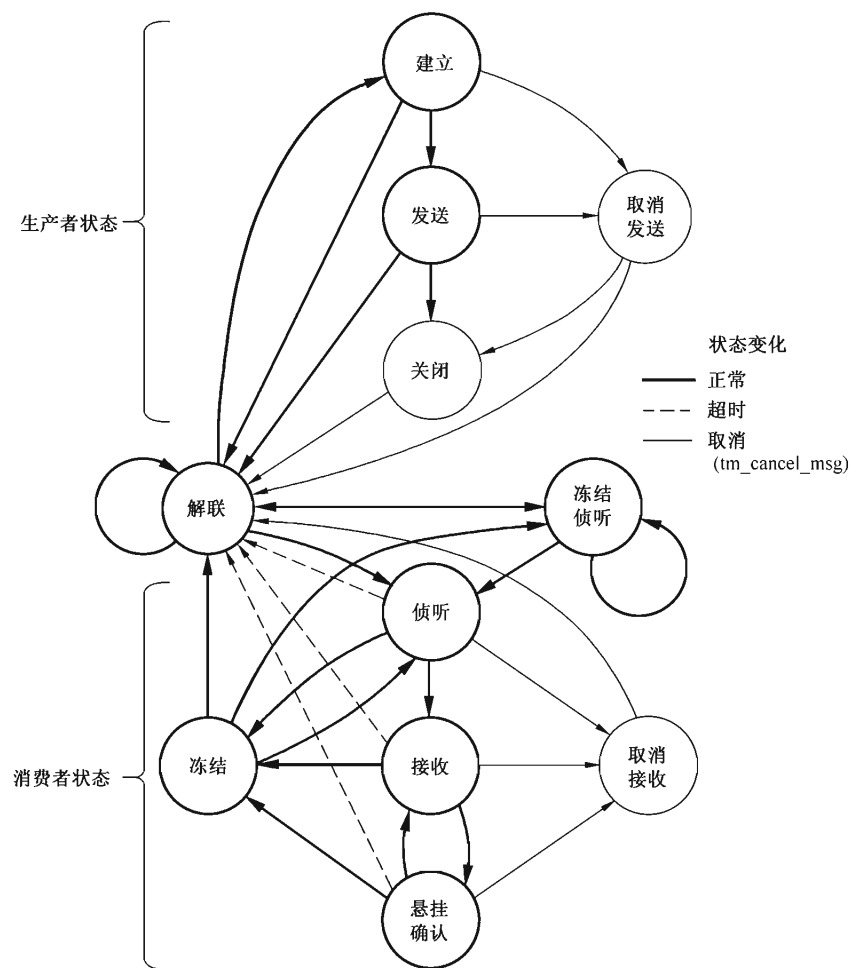


图 143 MTP 状态转换图

6.3.6.6.2 传入事件

由网络、用户或超时产生的传入事件如表 105 所列,可引起一个 MTP 状态转换到另一 MTP 状态。

表 105 MTP 传入事件

事件名	接口	简述
tm_send_req	来自用户	请求发送消息,用户传递消息缓冲区给信使
tm_cancel_req	来自用户	取消传入/传出消息的传送,若成功取消,信使返回消息缓冲区给用户
rcv_CR	来自网络	接收到连接请求(CR)包
rcv_CC	来自网络	接收到连接确认(CC)包
rcv_DT	来自网络	接收到数据(DT)包
rcv_AK	来自网络	接收到肯定确认(AK)包
rcv_NK	来自网络	接收到否定确认(NK)包
rcv_DR	来自网络	接收到解联请求(DR)包
rcv_DC	来自网络	接收到解联确认(DC)包
TMO	内部	超时到期(每次只一个运行)

6.3.6.6.3 传出事件

由 MTP 状态机产生的传出事件如表 106 所列,可引起一个 MTP 状态转换到另一 MTP 状态机的另一状态。

表 106 MTP 传出事件

事件名	接口	简述
tm_connect_ind	给用户的指示	用户应接受或拒绝传入连接请求;用户若接收消息,则传递消息缓冲区给信使
tm_receive_ind	给用户的指示	已接收完整消息或接收出错,信使返回消息缓冲区给用户
tm_send_cnf	给用户的确认	已发送完整消息或发送出错,信使返回消息缓冲区给用户
send_CR	给网络	发送一个连接请求(CR)包
send_CC	给网络	发送一个连接确认(CC)包
send_DT	给网络	发送一个数据(DT)包
send_AK	给网络	发送一个肯定确认(AK)包
send_NK	给网络	发送一个否定确认(NK)包
send_DR	给网络	发送一个解联请求(DR)包
send_DC	给网络	发送一个解联确认(DC)包

6.3.6.6.4 包内控制参数

交换的包应包含表 107 所列控制参数。

表 107 MTP 控制参数

事件	字段	字段简述
send_CR (fields) rcv_CR (fields)	CR_conn_ref CR_credit CC_pack_size	连接参考 建议的信用值 建议的包长度代码
send_CC(fields) rcv_CC(fields)	CC_conn_ref CC_credit CC_pack_size	连接参考 接收的信用值 接收的包长度代码
send_DT (fields) rcv_DT (fields)	DT_seq_nr DT_eot	包编号 消息传送结束标志
send_NK(fields) rcv_NK(fields)	NK_seq_nr	预期的下一包编号
send_AK (fields) rcv_AK (fields)	AK_seq_nr	预期的下一包编号
send_DR (fields) rcv_DR (fields)	DR_reason	解联原因,Am_Result 出错代码

对于发送事件(sent_xx)为控制字段规定实际参数,但对于接收事件(rcv_xx)则在动作中引用字段名。

6.3.6.6.5 辅助变量

状态转换图依赖于表 108 所列辅助变量减少状态数。有些变量只存在于生产者或消费者且属于一个连接,而全局变量为所有连接共同使用。

表 108 MTP 辅助变量

变量名	应用环境	变量简述
my_credit	全局	本方已接受的信用值的最大值
run_nr	全局	下一空闲的连接参考
my_pack_size	全局	本方已接受的包长度代码的最大值
cancelled	生产者/消费者	该消息已被用户取消
credit	生产者/消费者	该连接已接受的信用值
expected	生产者/消费者	发送(生产者)或接收(消费者)的下界
conn_ref	生产者/消费者	该连接的连接参考
rep_cnt	生产者/消费者	重复计数
new_cnt	消费者	未确认的包计数
next_send	生产者	下一发送包编号
send_not_yet	生产者	发送窗上界
eot	生产者	已发送完整消息
size	生产者	该连接可接受的包长度
error	生产者	发送确认中报告的 AM_xx 出错代码

6.3.6.6.6 活动和超时

每一个连接应有其自有定时器用于超时监视。

定时器应由动作“restart_tmo”和“reset_tmo”操作,且在定时到期时应产生 TMO 内部事件。

状态转换由 3 个超时控制:

- SEND_TMO:生产者处发送超时;
- ACK_TMO :消费者处确认超时;
- ALIVE_TMO:消费者处存活超时。

若生产者在包发送超时时限内没有接收到该包的确认包,则生产者应重发该包。

若在 MAX_REP_CNT 次发送尝试后生产者仍没有接收到该包的确认包,则生产者应解联。

重发计数 MAX_REP_CNT 的最大值应为 3。

当信用值耗尽或消息已完整接收时,消费者应立即确认。

信用值 AM_MAX_CREDIT 的最大值应为 7。

否则,消费者应通过确认超时延迟确认,以一次确认多个包。

发送超时的值是一个配置参数。

发送超时应大于最差情况网络传送延迟时间(NET_DELAY)与站中包处理时间(PROC_DELAY)的的和的两倍再加上确认超时时间,如图 144 所示。

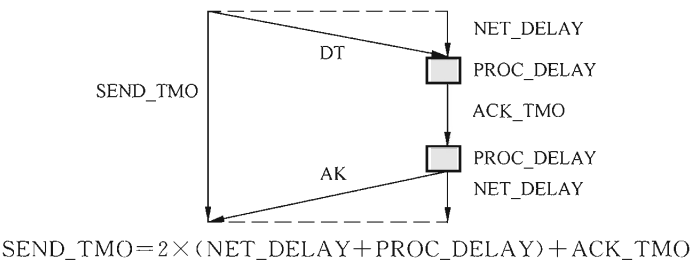


图 144 发送超时 SEND_TMO

包的消费者应期待存活超时时限内的下一包(序号相同或不同)。若此时间内没有收到包,则消费者应解联会话。

存活超时的值是一个配置参数。

存活超时的值应至少是最大重发次数乘以发送超时时间加上包在链路队列中寿命时间,如图 145 所示。

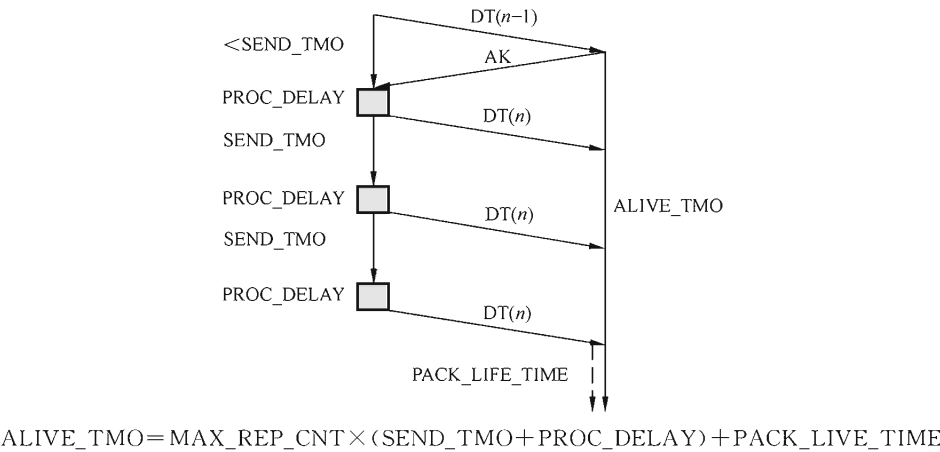


图 145 存活超时 ALIVE_TMO

最差应用情况下的缺省值总结在表 109 中。

表 109 MTP 超时(最差情况)

	网络延时	处理延时	包寿命时间	确认超时	发送超时	存活超时
同一总线	0.5 s	64 ms	5.0 s	0.25 s	1.4 s	9.5 s
通过列车总线的 编组网到编组网	9.6 s	64 ms	5.0 s	2.5 s	21.0 s	71.0 s

注：包寿命时间(PACK_LIFE_TIME)在相应的链路层规定。

6.3.6.6.7 隐式动作

下列状态转换中的动作描述可能包括表 110 所列隐式动作。

表 110 隱式動作

参考号	隐式动作
(1)	若总线上没有能发送的连接请求(CR)包,则指示 AM_BUS_ERR 状态
(2)	对包序号进行模 8 操作,比较考虑循环
(3)	在该情况中 cancelled 总为真

6.3.6.6.8 复合动作

表 111 所列动作在多个状态转换中执行。

表 111 复合动作

[illegible]

6.3.6.6.9 生产者状态事件表

一旦事件发生,生产者应从当前状态转换到下一状态并执行表 112 所规定的动作。

表 112 生产者状态及状态转换

当前状态	事件	动作	下一状态 (若不同于当前)
(any)	tm_cancel_req	cancelled := TRUE;	
DISC	tm_send_req	update(eot); send_CR(run_nr, AM_MAX_CREDIT, my_pack_size); conn_ref := run_nr; run_nr := run_nr + 1; restart_tmo(SEND_TMO); expected := 0; rep_cnt := 0; cancelled := FALSE;	SETUP
SETUP	rcv_DR	close_send(DR_reason);	DISC
	rcv_CC AND (conn_ref = CC_conn_ref)	IF (eot) THEN close_send(AM_OK); ELSE credit := CC_credit; size := decode(CC_pack_size); next_send := 0; send_not_yet := credit; send_data_or_cancel; END;	DISC SEND 或 SEND_CANC
	TMO AND (rep_cnt = MAX_REP_CNT)	close_send(AM_CONN_TMO_ERR); (1)	DISC
	TMO AND (rep_cnt < MAX_REP_CNT)	send_CR(conn_ref, AM_MAX_CREDIT, my_pack_size); rep_cnt := rep_cnt + 1; restart_tmo(SEND_TMO);	
SEND	rcv_DR	send_DC; error := DR_reason; restart_tmo(ALIVE_TMO);	CLOSED
	rcv_AK AND (expected < AK_seq_nr <= send_not_yet) (2)	expected := AK_seq_nr; send_not_yet := expected + credit; (2) IF (NOT eot) THEN send_data_or_cancel; ELSIF (expected = next_send) THEN close_send(AM_OK); ELSE REPEAT send_DT(expected, eot); expected := expected + 1; (2) UNTIL (expected = next_send); restart_tmo(SEND_TMO); rep_cnt := rep_cnt + 1; END;	SEND 或 SEND_CANC DISC SEND

表 112 (续)

当前状态	事件	动作	下一状态 (若不同于当前)
SEND	TMO AND (rep_cnt < MAX_REP_CNT)	local variable: seq_nr; seq_nr := expected; WHILE (seq_nr < (next_send-1)) DO (2) send_DT(seq_nr, FALSE); END; send_DT(next_send-1, eot); (2) rep_cnt := rep_cnt + 1; restart_tmo(SEND_TMO);	
	TMO AND (rep_cnt = MAX_REP_CNT)	close_send(AM_SEND_TMO_ERR);	DISC
	rcv_NK AND (expected < NK_seq_nr <= send_not_yet) (2)	expected := NK_seq_nr; send_not_yet := expected + credit; (2) IF (NOT eot) THEN IF cancelled THEN send_DR(AM_REM_CANC_ERR); restart_tmo(SEND_TMO); ELSE REPEAT update(eot); send_DT(next_send, eot); (2) next_send := next_send + 1; UNTIL (eot OR (next_send = send_not_yet)); restart_tmo(SEND_TMO); END;	SEND_CANC
SEND_CANC	rcv_DC	close_send; (3)	DISC
	rcv_DR	restart_tmo(ALIVE_TMO); (3)	CLOSED
	TMO AND (rep_cnt = MAX_REP_CNT)	close_send; (3)	DISC
	TMO AND (rep_cnt < MAX_REP_CNT)	send_DR(AM_REM_CANC_ERR); rep_cnt := rep_cnt + 1; restart_tmo(SEND_TMO);	
CLOSED	TMO	close_send(error);	DISC

6.3.6.6.10 消费者状态事件表

一旦事件发生,消费者应从当前状态转换到下一状态并执行表 113 所规定的动作。

表 113 消费者状态及状态转换

当前状态	事件	动作	下一状态 (若不同于当前)
(any)	tm_cancel_req	cancelled := TRUE;	
DISC	rcv_DR	send_DC(dc_reason);	
DISC 或 FROZEN	rcv_DR	Cancelled := FALSE; new_cnt := 0; expected := 0; tm_connect_ind(VAR err); IF (err = AM_OK) THEN conn_ref := CR_conn_ref; credit := min(my_credit, CR_credit); send_CC(conn_ref, credit, min(CR_pack_size, my_pack_size)); IF (eot) THEN close_rcv(AM_OK);	LISTEN_FROZEN
LISTEN_FROZEN	rcv_CR AND (CR_conn_ref < > conn_ref)	ELSE restart_tmo(ALIVE_TMO); END; ELSE send_DR(err); END;	LISTEN DISC
LISTEN_FROZEN	rcv_CR AND (CR_conn_ref = conn_ref)	send_CC (conn_ref, credit, min(CR_pack_size, my_pack_size));	
FROZEN 或 LISTEN_FROZEN	TMO		DISC
LISTEN	rcv_CR AND (CR_conn_ref = conn_ref)	send_CC(conn_ref, credit, min(CR_pack_size, my_pack_size)); restart_two_(ALIVE_TMO);	
	rcv_CR AND (CR_conn_ref < > conn_ref)	close_rcv(AM_FAILURE);	DISC
	TMO	close_rcv(AM_ALIVE_TMO_ERR);	DISC

表 113 (续)

当前状态	事件	动作	下一状态 (若不同于当前)
LISTEN 或 RECEIVE 或 PEND_ACK	rcv_DR	send_DC(dc_reason); close_rcv(DR_reason);	DISC
	rcv_DT AND cancelled	send_DR(AM_REM_CANC_ERR); rep_cnt := 0; restart_tmo(SEND_TMO);	RCV_CANC
	rcv_DT AND NOT cancelled AND (DT_seq_nr = expected)	expected := expected + 1; (2) new_cnt := new_cnt + 1; IF (DT_eom) THEN send_AK(expected); close_rcv(AM_OK); ELSIF (new_cnt = credit) THEN send_AK(expected); new_cnt := 0; restart_tmo(ALIVE_TMO); ELSIF (state = RECEIVE) OR (state = LISTEN) THEN restart_tmo(ACK_TMO); END;	FROZEN
			RECEIVE
	rcv_DT AND NOT cancelled AND (DT_seq_nr < expected)	send_NK(expected); new_cnt := 0; restart_tmo(ALIVE_TMO);	PEND_ACK
RECEIVE	TMO	close_rcv(AM_ALIVE_TMO_ERR);	DISC
PEND_ACK	TMO	send_AK(expected); new_cnt := 0; restart_tmo(ALIVE_TMO);	RECEIVE
FROZEN	rcv_DT	send_AK(expected);	
RCV_CANC	TMO AND (rep_cnt < MAX_ REP_CNT)	send_DR(AM_REM_CANC_ERR); restart_tmo(SEND_TMO); rep_cnt := rep_cnt + 1;	
	TMO AND (rep_cnt = MAX_ REP_CNT)	close_rcv(AM_FAILURE); (3)	DISC
	rcv_DR OR rcv_DC	close_rcv(AM_FAILURE); (3)	DISC

6.3.6.7 传送消息接口

6.3.6.7.1 概述

传送消息接口提供传输层到会话层的服务。
此接口可属于设备内部接口。一致性测试不包含此接口。
引入以下规定以提高可移植性。以下各条并不隐含特定实现。允许任何提供相同语义的接口。

6.3.6.7.2 传输层数据交换

图 146 显示了传输层和会话层之间的交互。箭头指示参数传送方向。

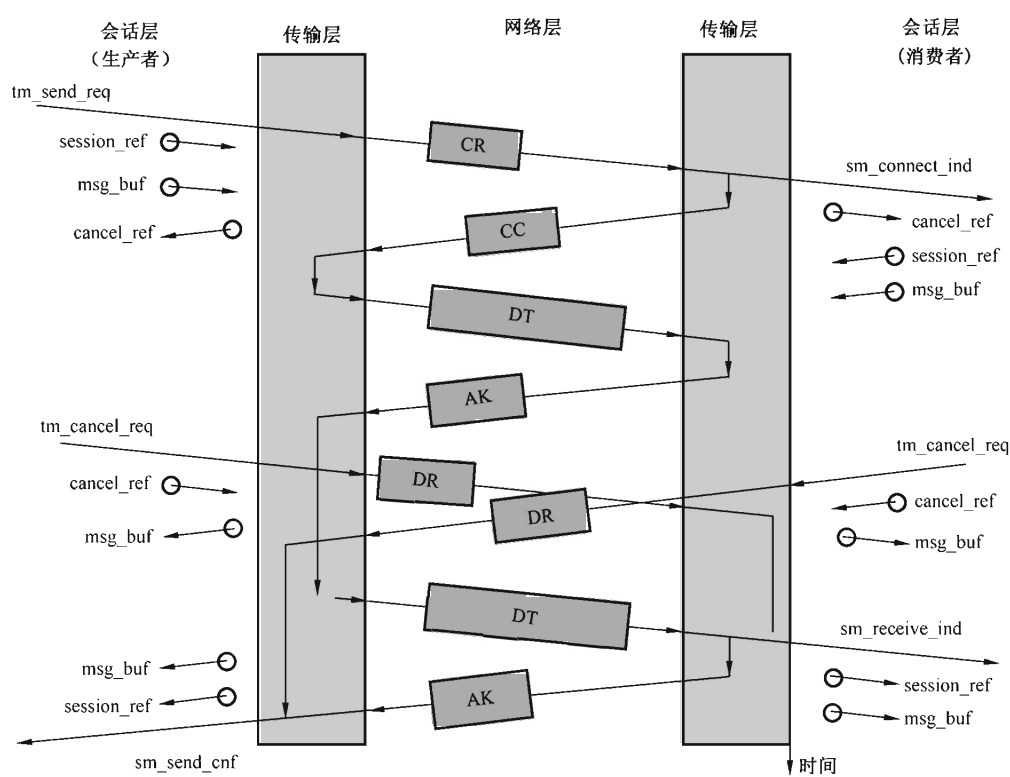


图 146 传送接口

生产者通过调用 `tm_send_req` 发送包含在 `msg_buf` 中的消息, `tm_send_req` 通过会话参考“`session_ref`”标识连接。

当传送完成(成功或失败)时,传输层调用 `sm_send_cnf`(具有相同 `session_ref`)释放消息缓冲区。

在消费者侧,传输层调用 `sm_connect_ind` 通知连接请求的会话层。会话层若接受连接请求,则为传输层提供缓冲区 `msg_buf` 以放置消息。

当消息全部接收(或被另一方取消)时,传输层调用 `sm_received_ind`。此调用包含“`session_ref`”。

任意一方通过调用 `tm_cancel_req` 取消消息传送。

TMI 原语前缀为 `tm_`。

传输层调用事先订阅且具有由传输层定义的类型(前缀 `TM_`)的会话层过程(前缀 `sm_`)。

TMI 由表 114 所列常量、类型和过程定义,且在以下各条中规定。

表 114 TMI 原语

名称	含 义
常量	
TM_CALLER_USER	用户是呼叫者
TM_REPLIER_USER	用户是应答者
类型	
TM_MSG_DESCR	消息描述符
TM_CONV_ID	会话标识符
初始化过程	
tm_define_user	通告用户
生产者过程	
tm_send_req	发送消息
TM_SEND_CNF	确认发送,指示过程,sm_send_cnf 类型
sm_send_cnf	当消息到达(或取消)时调用
消费者过程	
TM_CONNECT_IND	sm_connect_ind 类型
sm_connect_ind	指出连接请求到达
TM_RECEIVE_IND	sm_receive_ind 类型
sm_receive_ind	当消息完全接收时调用
消费者和生产者过程	
tm_cancel_req	取消消息

6.3.6.7.3 类型“TM_MSG_DESCR”

类型“TM_MSG_DESCR”见表 115。

表 115 类型“TM_MSG_DESCR”

定义	消息类型
语法	Typedef struct {} TM_MSG_DESCR;
用法	隐藏结构,用作清除锁定数据结构的句柄

6.3.6.7.4 类型“TM_CONV_ID”

类型“TM_CONV_ID”见表 116。

表 116 类型“TM_CONV_ID”

定义	Conversation_ID 类型	
语法	typedef struct	str_conv_id
		{
	UNSIGNED8	my_fct_or_station;
	UNSIGNED8	node;
	UNSIGNED8	function_or_station;
	UNSIGNED8	next_station
		} TM_CONV_ID;

6.3.6.7.5 过程“tm_define_user”

过程“tm_define_user”见表 117。

表 117 过程“tm_define_user”

定义	为标识为呼叫者或应答者的用户订阅指示过程	
语法	AM_RESULT	tm_define_user
		(
	UNSIGNED	who,
	TM_CONNECT_IND	sm_connect_ind,
	TM_RECEIVE_IND	sm_receive_ind,
	TM_SEND_CNF	sm_send_cnf
);
输入	who	传输层用户, TM_CALLER_USER 或 TM_REPLIER_USER
	sm_connect_ind	当连接请求包到达时消费者调用的过程。 见 TM_CONNECT_IND 类型定义
	sm_receive_ind	当消息已全部收到或已由生产者取消或出错时消费者调用的过程。 见 TM_RECEIVE_IND 类型定义
	sm_send_cnf	当消息已被消费者全部收到或已被消费者取消或出错时生产者调用的过程。 见 TM_SEND_CNF 类型定义
返回		任意 AM_RESULT

6.3.6.7.6 过程“tm_send_req”

过程“tm_send_req”见表 118。

表 118 过程“tm_send_req”

定义	请求消息传送	
语法	<div>AM_RESULT</div> <div>tm_send_req</div> <div>(</div> <div>UNSigned</div> <div>session_user,</div> <div>TM_CONV_ID *</div> <div>conversation,</div> <div>void *</div> <div>msg_addr,</div> <div>UNSigned32</div> <div>msg_len,</div> <div>void *</div> <div>hdr_addr,</div> <div>UNSigned</div> <div>hdr_len,</div> <div>void *</div> <div>session_ref,</div> <div>TM_MSG_DESCR **</div> <div>cancel_ref,</div> <div>UNSigned8</div> <div>my_topo</div> <div>);</div>	
输入	session_user	TM_CALLER_USER(呼叫消息)或 TM_REPLIER_USER(应答消息)
	conversation	Conversation_Id(呼叫者和应答者地址的连接)
	msg_addr	指向消息正文的指针
	msg_len	消息正文的总长度
	hdr_addr	指向 Session_Header 的指针
	hdr_len	Session_Header 的总长度
	session_ref	链接 tm_send_req 及相应 sm_send_cnf 的会话参考
	my_topo	应用提供的 Topo_Counter 值,若未知则为 0(= pass-all)
返回		任意 AM_RESULT
输出	cancel_ref	传输层已锁定数据结构的句柄
用法	<div>a) 用于待发送消息的缓冲区和用于放置其 Session_Header 的缓冲区是分开的。</div> <div>b) “session_ref”由传输层用户提供,且当调用 sm_send_cnf 时由传输层返回。</div> <div>c) “cancel_ref”允许释放已锁定数据结构。在取消操作成功或 sm_send_cnf 返回后,“cancel_ref”不再有效</div>	

6.3.6.7.7 类型“TM_SEND_CNF”

类型“TM_SEND_CNF”见表 119。

表 119 类型“TM_SEND_CNF”

定义	当消息已全部接收或已由消费者取消或报告出错时,传输层应调用此类型的会话层过程 sm_send_cnf	
语法	<pre>typedef void (* TM_SEND_CNF) (void * session_ref, UNSIGNED8 my_topo, AM_RESULT status);</pre>	
输入	session_ref	将 sm_send_cnf 与先前 tm_send_req 连接的会话参考
	my_topo	若结果为 AM_OK,则节点 Topo_Counter 当前有效
	status	若消息已被消费者确认则为 AM_OK,否则给出错误代码
用法	<p>a) 对每一个 tm_send_req,传输层调用 TM_SEND_CNF 类型的会话层过程 sm_send_cnf,以通知生产者其消息已被消费者接收或取消或报告错误。</p> <p>b) 若连接已被生产者成功取消,则不调用 sm_send_cnf。</p> <p>c) status 是输入参数</p>	

6.3.6.7.8 类型“TM_CONNECT_IND”

类型“TM_CONNECT_IND”见表 120。

表 120 类型“TM_CONNECT_IND”

定义	<p>当接收到连接请求包时,消费者侧的传输层应调用 TM_CONNECT_IND 类型的过程“sm_connect_ind”。</p> <p>若“sm_connect_ind”接收消息,则期待返回结果 AM_OK 并为该消息提供缓冲区。</p> <p>若“sm_connect_ind”返回结果不同于 AM_OK,则传输层应拒绝连接并以“sm_connect_ind”返回结果作为“reason”参数发送解联请求</p>	
语法	<pre>typedef void (* TM_CONNECT_IND) (TM_CONV_ID * conversation, /* in */ void * * msg_addr, /* out */ UNSIGNED32 msg_len, /* in */ void * * hdr_addr, /* out */ UNSIGNED8 * hdr_len, /* out */ TM_MSG_DESCR * cancel_ref, /* in */ void * * session_ref, /* out */ UNSIGNED8 my_topo, /* in */ AM_RESULT status, /* out */);</pre>	
输入	conversation	Conversation_Id(呼叫者和应答者地址的连接)
	msg_len	会话层提供的消息缓冲区的长度
	cancel_ref	传输层已锁定数据结构的句柄
	my_topo	接收到连接请求时节点 Topo_Counter 值

表 120 (续)

输出	msg_addr	指向会话层提供的消息正文的指针
	hdr_addr	指向消息的 Session_Header 的指针
	hdr_len	Session_Header 的总长度
	session_ref	会话层提供的用于标识会话层已锁定数据结构。链接 sm_connect_ind 和相应 sm_receive_ind
	status	若会话层接受连接则为 AM_OK, 否则为拒绝原因(AM_RESULT)
用法	期望会话层在 tm_define_user 过程中订阅 TM_CONNECT_IND 类型的 sm_connect_ind 过程	

6.3.6.7.9 类型“TM_RECEIVE_IND”

类型“TM_RECEIVE_IND”见表 121。

表 121 类型“TM_RECEIVE_IND”

定义	当完整接收到传入消息时, 传输层应调用 TM_RECEIVE_IND 类型的 sm_receive_ind 过程	
语法	typedef void (* TM_RECEIVE_IND) (void * session_ref, AM_RESULT status);	
输入	session_ref	将 sm_receive_ind 和先前 sm_connect_ind 链接的参考
	status	当消息被成功接收到并放入缓冲区时为 AM_OK, 否则为另一 AM_RESULT 错误码
用法	1. 期望会话层在 tm_define_user 过程中提供 TM_RECEIVE_IND 类型的 sm_receive_ind 过程。 2. 期望“sm_receive_ind”将会话层在“sm_connect_ind”中提供的缓冲区返回给会话层	

6.3.6.7.10 过程“tm_cancel_req”

过程“tm_cancel_req”见表 122。

表 122 过程“tm_cancel_req”

定义	当被生产者或消费者调用时取消连接	
语法	AM_RESULT tm_cancel_req (TM_MSG_DESCR * cancel_ref);	
输入	cancel_ref	传输层已锁定数据结构的句柄
返回		若对进行中的消息施加取消则为 AM_OK, 否则为 AM_FAILURE
用法	1. 若结果为 AM_OK, 则传输层已发送解联请求包, 且没有“sm_send_cnf”或“sm_receive_ind”未被调用。 2. 若消息已完整发送并确认, 则该过程无效。 3. “cancel_ref”允许释放已锁定数据结构。在取消操作成功后, 其不再有定义	

6.3.7 多播传送协议(可选)

6.3.7.1 多播包交换

6.3.7.1.1 概述

多播传送应分为三个阶段：

- a) 建立连接；
- b) 已确认的数据传送；
- c) 解联。

图 147 显示了无包丢失的简单包交换。

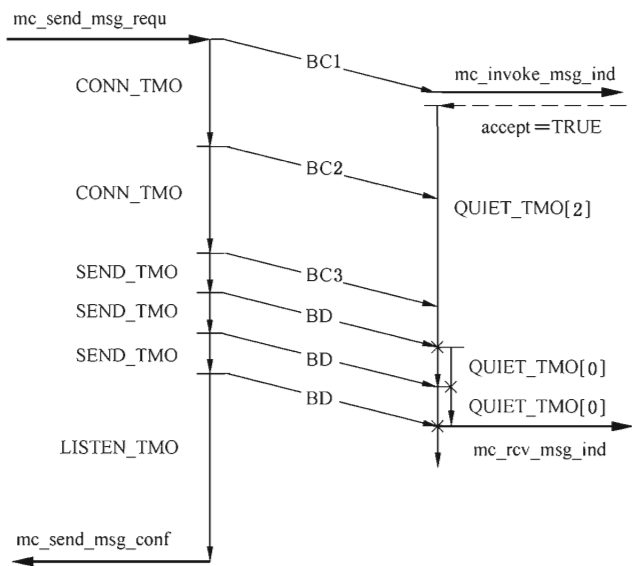


图 147 无重传的多播消息

6.3.7.1.2 建立连接

消息的第一个包为广播连接(Broadcast_Connect,BC)包。该包指示消息总长度,并包含第一个数据段。

当 BC 包到达时,消费者检查是否有应用就绪以接收消息。若无,则丢失该包。否则,打开连接并将第一个数据段拷贝到应用的消息缓冲区。

生产者总是重复发送 BC 包三次,在重发之间有一个长延时,即连接超时(CONN_TMO)时长。

消费者足以获得这三个拷贝中的一个。

BC 包还包含一个重发倒计数值,该值允许消费者调节其超时。

消费者不能请求重发 BC。

若消息短到可装入 BC 包,则发生图 148 所示情况：

- 在发送最后一个 BC 之后,生产者信使立即确认发送完成；
 - 消费者信使通知应用,并启动超时(ALIVE_TMO[])以防止由于重复 BC 导致的消息重复。
- 该超时依赖于 BC 重发计数。接收到最后一个 BC 则停止超时并关闭连接。

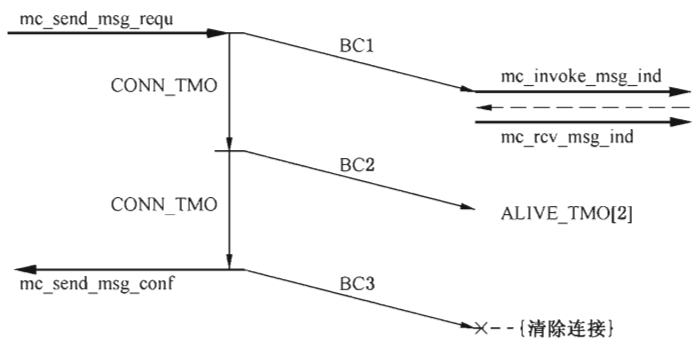


图 148 无 BD 包且无丢失的短多播消息

6.3.7.1.3 数据传送

若用 BC 未发完消息，则生产者以小于 CONN_TMO 的发送超时 (SEND_TMO) 的时间间隔开始发送广播数据 (Broadcast_Data, BD) 包。

BD 包包含指示数据段在整个消息中位置的 16 位偏置。

期待数据的消费者启动空闲超时 (QUIET_TMO) 以监视活动。若该超时到期或消费者检测到偏置高于期望值的包，则消费者认为已丢失一些 BD 包，并因而发布广播重复 (Broadcast_Repeat, BR) 包请求重传。BR 包包含请求重传起始地址的 16 偏置。

在发送 BD 包时，生产者将接收到的 BR 包过滤，并在插入传送暂停 (正常的 SEND_TMO 加上 PAUSE_TMO) 后开始重传。

在发送 BR 包时，消费者启动重复超时 (REPEAT_TMO) 以监视 BR 效果。

只有当重复超时当前未运行时消费者才可发送 BR 包，否则重复超时可能过期并导致 BR 重传，而在第二个超时时将丢弃消息并关闭连接 (此时已丢失三个包)。

接收到所期望的或较早的 BD 则重启超时 QUIET_TMO[0]。

示例：图 149 显示了多包丢失的情况。

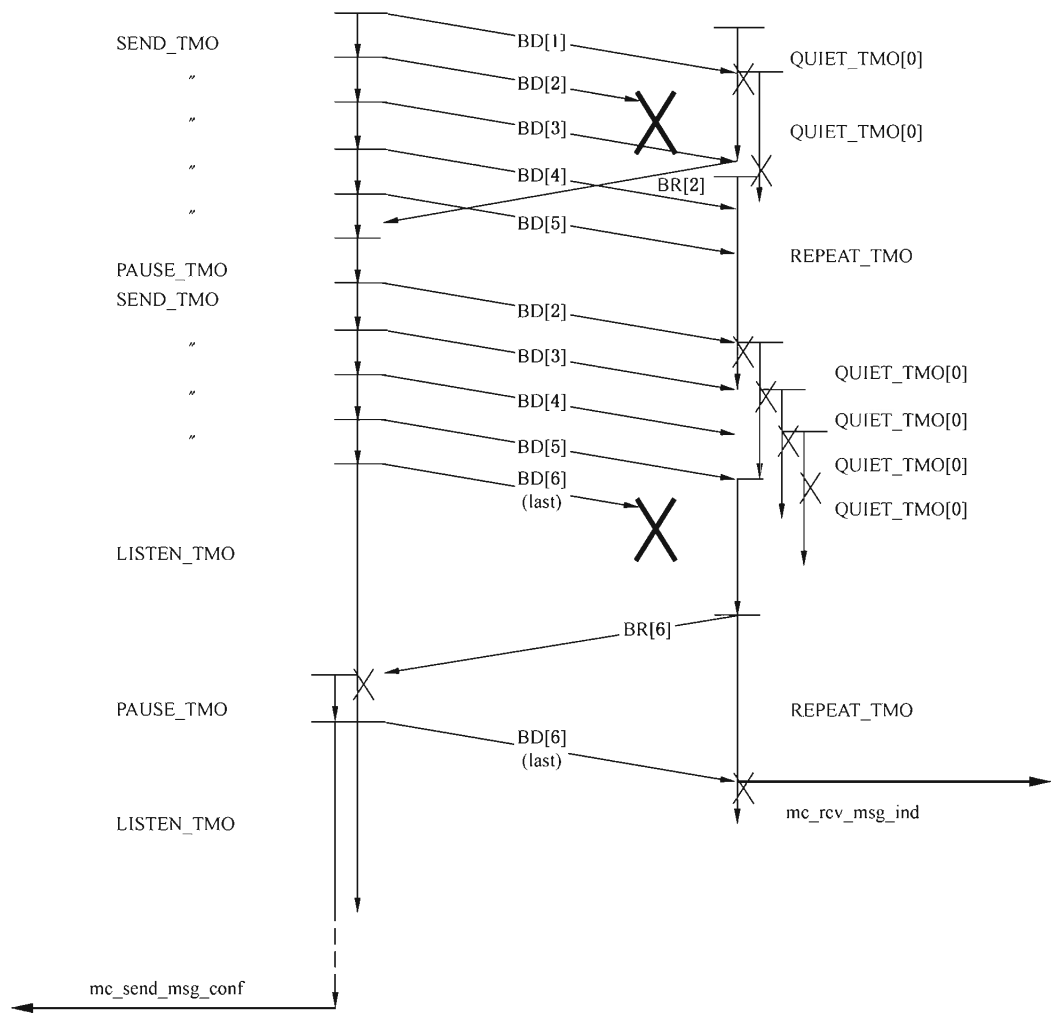


图 149 发生了多包丢失的交换

6.3.7.1.4 解联

在最后一个 BD 包之后,传递消息给消费者应用,并关闭连接。
在最后一个 BD 包之后,生产者在关闭和确认传送完成前的侦听超时(LISTEN_TMO)期间等待 BR 包。

6.3.7.1.5 取消传送

若消费者取消请求,则丢弃接收到的消息,而总线上不可见。但是若生产者取消消息多播,则应发送一个特定的包,即停止广播(Broadcast_Stop,BS)包,以禁止所有消费者超时和大量重复 BR 包的产生。

6.3.7.1.6 重传算法

多个消费者可能发布相同的 BR 包,但这些包将以不同延时到达生产者。也可能到达的 BR 包不指示增长的丢失偏置(这是点对点传送的情况)。该问题不能通过在开始重传前插入长延时解决,因为双方的超时互相依赖。

因此在生产者处应有某种 BR 过滤器机制。使用的过滤器可如下描述:
——在接收到 BR 包之后,具有相同重传偏置的另一个传入 BR 包将在跳过超时(SKIP_TMO)期

间丢弃；

- 只有数量限制为 REP_LIMIT、具有不同的重传偏置的 BR 包才保证在固定长度 REP_RANGE 的滑动偏置窗口中被接受；
- 对任意窗口超出此限制的更多 BR 包可能被丢弃或不丢弃。

传入的 BR 包根据以下算法处置：生产者对每个消息维持一个请求重传的登记表。该登记表由 REP_LIMIT 个记录项组成，在开始传送时记录项全空。占用的记录项包含重传偏置（作为关键字）和其自有定时器，在接收到相应 BR 包时该计数器以 SKIP_TMO 载入。对传入的 BR 包，检查登记表以寻找重传请求偏置是否已登记。若存在具有相同偏置的记录项，则只要该记录项的定时器仍在运行中无论运行到何时都丢弃 BR 包，否则接受 BR 包。若请求偏置尚未登记，则接受 BR 包并插入一个新的记录项，直到表满为止。当表满时，是否接受 BR 取决于是否存在可释放以再用的记录项。若新的 BR 包的偏置或其余记录项之一偏置落在窗口之外（该窗口的下界为最低登记偏置），则具有最低偏置的记录项被覆盖。否则，因为在同一窗口中已接受 REP_LIMIT 个 BR 请求而丢弃该 BR 包。当登记表满时，该表一直保持满（直到连接关闭）。

依此算法，若消息只有一个消费者，则对任何窗口所有超过限制数量 REP_LIMIT 的 BR 包将被丢弃（若消费者多于一个，则这样的 BR 包仍可能接受）。

6.3.7.1.7 跨初运行的消息一致性

MC 消息的消费者保存传入 BC 包网络报头中的 Final_Node（终点节点）。一旦打开接收连接，若后续包寻址到相同的终点节点则都只有接受，否则连接以出错关闭。路由节点保证转发到车辆总线（VB）的包中的 Final_Node 字段随每次初运行变化（Final_Node 字段包含初运行计数器）。这防止了始于不同节点的包被组装成一个无效的、混合的消息（一个节点的地址可被初运行指定给另一个节点，且两个节点可能碰巧使用相同的连接参考）。

若 MC 消息的生产者位于 VB 的末端设备，则不会感知到中间的初运行，并继续发送 BD 包直到消息全部传送。

6.3.7.2 连接标识

对每一个传入包，获取包所属连接。在接收端，连接由自身 Final_Function（终点功能）、Origin_Function（起始功能）、Origin_Node（起始节点）和 Origin_Station（起始站）组成的 32 位结构标识。

在发送端，连接仅由两个涉及的功能标识，因为传入 BR 包可能包含任意起始节点（有多个消费者）。

具有相同连接标识的消息按顺序发送。

6.3.7.3 连接参考

每一个包包含一个作为消息标识的 16 位连接参考（Connection_Reference）。该连接参考由生产者对每一个传出消息进行递增指定，并随机初始化。

6.3.7.4 多播消息传送控制编码

多播协议与 MTP 协议使用相同的网络层，因此包具有相同的链路报头（Link_Header）和网络报头（Network_Header）。由 6.3.6.5.2 中规定的消息传送控制区分这两个协议。

6.3.7.5 多播包编码

多播协议把包分为图 150 所示的格式。

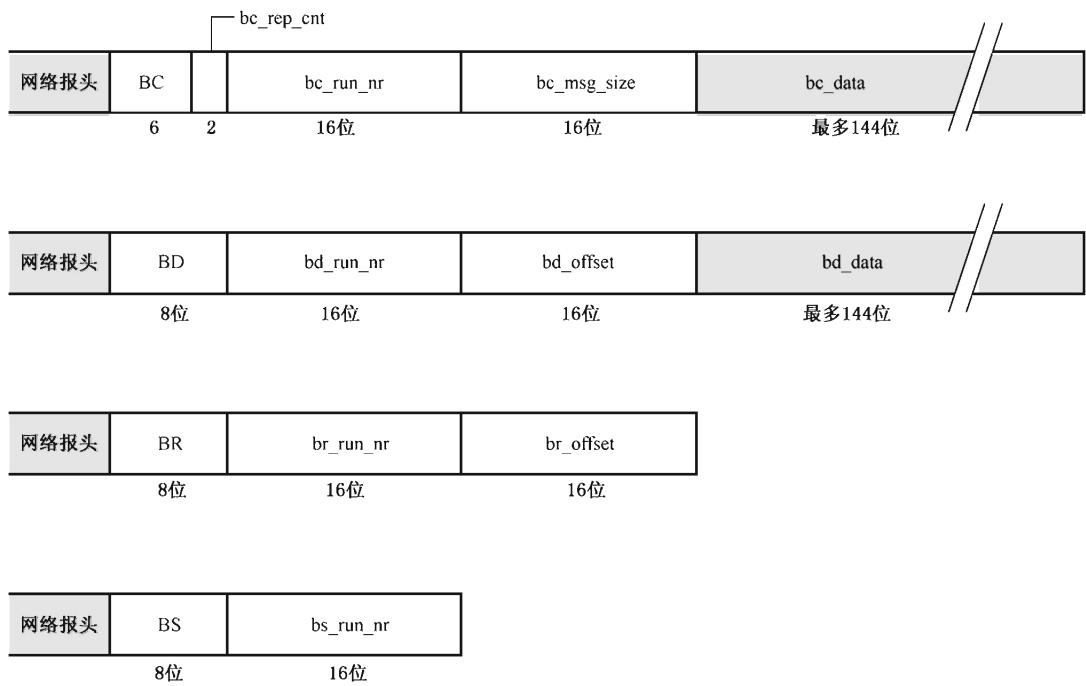


图 150 包格式

应依照 6.3.6.5.4 中的规定编码多播包。

6.3.7.6 多播传送协议定义

6.3.7.6.1 概述

多播协议工作在一个无连接、不可靠的网络上,该网络维持包序并限制包寿命时间为 PACK_LIFE_TIME。

为每一个消息传送建立连接。消息的生产方称为生产者,接收方称为消费者。在消费者侧每一个连接需要一个定时器,而生产者应为登记表每一个记录项有另一个定时器以实现 SKIP_TMO。

6.3.7.6.2 状态和状态转换图

图 151 所示的多播协议机应具有表 123 所列状态。

表 123 MCP 机状态

状态名	发生于	简述
IDLE(空闲)	生产者/消费者	已解联
SETUP(设置)	生产者	发送广播连接(BC)包
SEND(发送)	生产者	周期性发送广播数据(BD)包
LISTEN(侦听)	生产者	已发送完整消息,等待广播重复(BR)包
INSERT_PAUSE(插入暂停)	生产者	在发送下一广播数据(BD)包前,请求附加延时
PAUSE(暂停)	消费者	附加延时运行中
RECEIVE(接收)	消费者	接收到广播连接(BC)包并期待广播数据(BD)包
FROZEN(冻结)	消费者	接收到一些非最后的广播连接(BC)包,并无期待的广播数据(BD)包

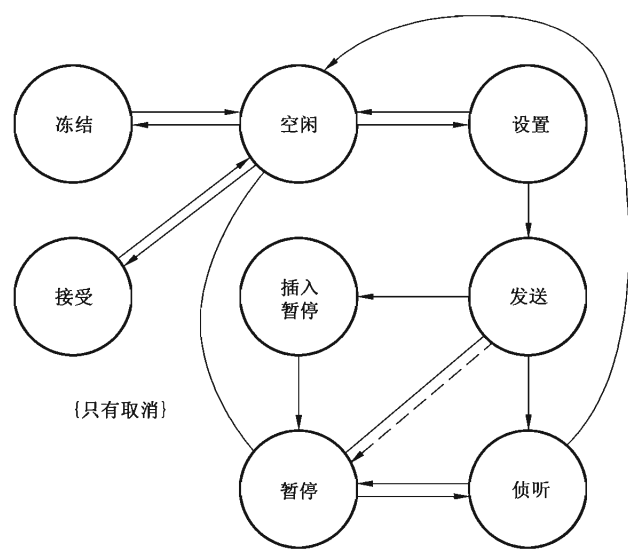


图 151 协议机状态

这些状态之间的转换取决于表 124 中定义的传入事件并产生表 125 中定义的传出事件。

表 124 传入事件

事件名	发生于	接口	事件简述
mc_send_msg_requ	生产者	来自用户	请求发送消息,用户将消息缓冲区传递给信使
mc_cancel_msg	生产者/消费者	来自用户	取消传入或传出消息传送,若取消操作成功,则信使将消息缓冲区返回给用户
rcv_BC	消费者	来自网络	接收到 BC 包
rcv_BD	消费者	来自网络	接收到 BD 包
rcv_BS	消费者	来自网络	接收到 BS 包
rcv_BR	生产者	来自网络	接收到 BR 包
skipped(reg_entry)	生产者	内部	登记表记录项 reg_entry 的 SKIP_TMO 期满
TMO	消费者	内部	任何其他超时期满

表 125 传出事件

事件名	发生于	接口	事件简述
mc_invoke_msg_requ	消费者	到用户	传入的连接请求被用户接受或拒绝,若用户接收消息,则将消息缓冲区传递给信使
mc_rcv_msg_ind	消费者	到用户	接收到完整消息或出错,信使将消息缓冲区返回给用户
mc _msg_conf	生产者	到用户	发送出全部消息或出错,信使将消息缓冲区返回给用户
send_BC	生产者	到网络	传送 BC 包
send _BD	生产者	到网络	传送 BD 包
send _BS	生产者	到网络	传送 BS 包
send _BR	消费者	到网络	传送 BR 包

6.3.7.6.3 数据包中的控制字段

交换的包包含表 126 所述控制字段。

表 126 包中控制字段

事件名	字段名	字段简述
xxx_BC(BC_run_nr,	连接参考(Connect_Reference)
	BC_rep_cnt,	挂起的 BC 重传计数
	BC_msg_size)	消息总长度
xxx_BD(BC_run_nr,	Connect_Reference
	BD_offset)	数据段在消息中的偏置
xxx_BS(BS_run_nr)	Connect_Reference
xxx_BR(BR_run_nr,	Connect_Reference
	BR_offset)	丢失的偏置
注：缩写 xxx 代表“发送”(send)或“接收”(rcv)。对 xxx = send 的事件,为控制字段规定实际参数;而对 xxx = rcv 的事件,在动作中引用字段名。		

6.3.7.6.4 辅助变量

状态转换图依赖辅助变量减少状态数。一些变量仅存在于生产者或消费者并属于一个连接,而其他变量是全局的,即意味着为所有连接共同使用。应实现表 127 所列辅助变量。

表 127 辅助变量

变量名	应用环境	变量简述
run_nr	全局	下一空闲的连接参考
cancelled	生产者/消费者	该消息已被用户取消
timer	生产者/消费者	除 SKIP_TMO 外所有超时的定时器
msg_size	生产者/消费者	消息总长度
offset	生产者/消费者	下一数据段的偏置
conn_run_nr	生产者/消费者	该连接的连接参考
rep_cnt	生产者/消费者	重复计数(BR 或 BC)
my_node	消费者	该连接的终点节点
reg_table[REP_LIMIT]. offset	生产者	登记表,重传偏置
reg_table[REP_LIMIT]. skip_timer	生产者	登记表,SKIP_TMO 超时
reg_table[REP_LIMIT]. timer_running	生产者	登记表,布尔型:“skip_timer”正在运行
table_is_full	生产者	布尔型:登记表满
window_is_full	生产者	布尔型:所有已登记偏置在大小为 REP_RANGE 的相同窗口内
next_entry	生产者	登记表中下一空闲项(索引),或者登记表满则为最低偏置项
注：定时器只能被启动或停止(不可改变),且在超时产生一个事件。		

6.3.7.6.5 常量

表 128 列出了多播协议机使用的常量。

表 128 MCP 常量

常量名	常量值	常量简述
REP_LIMIT	6	登记表长度(记录项数)
MAX_TRIALS	3	尝试次数(传送 BC 包),容许丢失一个包
BD_DATA_SIZE	18(对 MVB 帧)	BD 包中八位位组数据数
PROC_DELAY	1×64 (ms)	事件处理时间(轮询周期)
NET_DELAY	8×64 (ms)	期望传送延时(端对端)
REP_RANGE	LISTEN_TMO/SEND_TMO×BD_DATA_SIZE	
注: 可进行重传的偏置范围定义了窗口长度 REP_RANGE。		

超时应如表 129 规定。

表 129 MCP 超时

超时名	超时值
生产者侧	
CONN_TMO	4×64 (ms)
SEND_TMO	1×64 (ms)
PAUSE_TMO	1×64 (ms)
SEND_MAX_TMO	SEND_TMO + PROC_DELAY + PAUSE_TMO
SKIP_TMO	REPEAT_TMO-NET_DELAY-PROC_DELAY
LISTEN_TMO	(QUIET_TMO[0] + PROC_DELAY) + (MAX_TRIALS-2) × (REPEAT_TMO + PROC_DELAY) + 2 × NET_DELAY + PROC_DELAY-SEND_TMO
消费者侧	
QUIET_TMO [MAX _ TRIALS]	SEND_MAX_TMO + PROC_DELAY + NET_DELAY + ix × (CONN_TMO + PROC_DELAY)
ALIVE_TMO [MAX _ TRIALS]	NET_DELAY + PACK_LIFE_TIME + ix × (CONN_TMO + PROC_DELAY)
REPEAT_TMO	SEND_MAX_TMO + 2 × NET_DELAY + PROC_DELAY
注 1：规定所有超时和其他时间指示都以毫秒为单位。对于可自由选择的超时，规定了选定数；而对于有依赖关系的超时，给定了最小值的公式。	
注 2：符号“ix”含义为“数组索引 0~(MAX_TRIALS-1)”。	

6.3.7.6.6 复合动作

表 130 所列的复合操作在 MCP 机中多次使用。

每一个连接有用于超时监视的自有定时器。该定时器由动作重启超时(restart_tmo)和复位超时(reset_tmo)控制,并在期满时产生一个内部事件“TMO”。在生产者侧,另一个定时器用于每个 SKIP_

TMO,其通过“restart_skip”或“reset_skip”控制并在期满时产生一个内部事件“skipped(reg_entry)”。属于一个连接的所有定时器使用解联复位。

表 130 MCP 复合操作

动作名	动作简述	下一状态
restart_tmo(xxx_TMO)	停止 timer,再以超时 xxx_TMO 启动	
reset_tmo	停止 timer	
restart_skip(reg_entry)	以超时 SKIP_TMO 启动 skip_timer,并对项 reg_table[reg_entry]设置 timer_running = TRUE	
reset_skip(reg_entry)	停止 skip_timer,并对项 reg_table[reg_entry]设置 timer_running = FALSE	
close_send	<pre> IF (NOT cancelled) THEN mc_send_msg_conf; END; reset_tmo; FOR ix := 0 TO (REP_LIMIT-1) DO reset_skip(ix); END; </pre>	IDLE
close_rcv(error)	<pre> IF (NOT cancelled) THEN mc_rcv_msg_ind(error); END; reset_tmo; </pre>	IDLE
accept_BC	<pre> cancelled := FALSE; my_node := final_node; offset := 0; mc_invoke_msg_ind(VARerr); IF (err = AM_OK) THEN conn_run_nr := BC_run_nr; msg_size := BC_msg_size; update(offset); IF (offset = msg_size) THEN IF (NOT cancelled) THEN mc_rcv_msg_ind(AM_OK); END; IF (BC_rep_cnt = 0) THEN ELSE restart_tmo(ALIVE_TMO[BC_rep_cnt]); END; ELSE rep_cnt := 0; restart_tmo(QUIET_TMO[BC_rep_cnt]); END; END; END; </pre>	<div>IDLE</div> <div>FROZEN</div> <div>RECEIVE</div>

check_BR 完成 BR 包的过滤,并返回一个布尔数指示是否接受 BR。除了由 init_BR_filter 完成的初始化外,所有只在生产者出现的辅助变量由 check_BR 访问。包的过滤在表 131 中规定。

表 131 BR 包的过滤

动作名	动作规定
Init_BR_filter	<pre> table_is_full := FALSE; window_is_full := FALSE; next_entry := 0; FOR ix := 0 TO (REP_LIMIT-1) DO reg_table[ix].timer_running := FALSE; END;</pre>
Check_BR (VAR accept)	<pre> temporary variables: ix, max_offset; accept := FALSE; IF ((BR_run_nr < conn_run_nr) OR (BR_offset >= offset)) THEN RETURN; END; IF (any entry ix registered with reg_table[ix].offset = BR_offset) THEN IF (NOT (reg_table[ix].timer_running)) THEN accept := TRUE; END; ELSIF (NOT (table_is_full)) THEN ix := next_entry; INC(next_entry); IF (next_entry = REP_LIMIT) THEN table_is_full := TRUE; END; accept := TRUE; ELSIF ((BR_offset-reg_table[next_entry].offset >= REP_RANGE) OR (NOT (window_is_full))) THEN ix := next_entry; accept := TRUE; END; IF (accept) THEN offset := BR_offset; restart_skip(ix); IF (table_is_full) THEN (* 更新 next_entry 作为具有最低登记偏置记录项; *) (* 更新 window_is_full (窗口满标志); *) next_entry := 0; max_offset := reg_table[0].offset; FOR ix := 1 TO (REP_LIMIT-1) DO IF (reg_table[ix].offset < reg_table[next_entry].offset) THEN next_entry := ix; ELSIF (reg_table[next_entry].offset) THEN max_offset := reg_table[ix].offset; END; END; window_is_full := (max_offset-reg_table[next_entry].offset) < REP_RANGE; END; END;</pre>

6.3.7.6.7 生产者状态事件表

表 132 规定了生产者的行为。

表 132 MCP 生产者状态事件表

当前状态	事件	动作	下一状态
任意	mc_cancel_msg	cancelled := TRUE;	
	skipped(reg_entry)	reg_table[reg_rentry].timer_running := FALSE;	
IDLE	mc_send_msg_requ	conn_run_nr := run_nr; INC(run_nr); cancelled := FALSE; offset := 0; msg_size := requ_msg_size; rep_cnt := MAX_TRIALS-1; init_BR_filter; send_BC(conn_run_nr, rep_cnt, msg_size); update(offset); restart_tmo(CONN_TMO);	SETUP
SETUP	TMO	IF (cancelled) THEN send_BS(conn_cun_nr); close_send; ELSE DEC(rep_cnt); send_BS(conn_cun_nr, rep_cnt, msg_size); IF (rep_cnt > 0) THEN restart_tmo(CONN_TMO); ELSIF (offset = msg_size) THEN close_send; ELSE restart_tmo(SEND_TMO); END; END;	IDLE IDLE SEND
SEND	rcv_BR	check_BR(VAR accept); IF (accept) THEN END;	INSERT_PAUSE
	TMO AND cancelled	restart_tmo(PAUSE_TMO);	PAUSE
SEND 或 PAUSE	TMO AND NOT cancelled	send_BD(conn_run_nr, offset); update(offset); IF (offset = msg_size) THEN restart_tmo(LISTEN_TMO); ELSE restart_tmo(SEND_TMO); END;	LISTEN SEND

表 132 (续)

当前状态	事件	动作	下一状态
PAUSE	TMO AND cancelled	send_BS(conn_run_nr); close_send;	IDLE
PAUSE 或 INSERT_PAUSE	rcv_BR	check_BR(VAR accept);	
INSERT_PAUSE	TMO	restart_tmo(PAUSE_TMO);	PAUSE
LISTEN	TMO	close_end;	IDLE
	rcv_BR	check_BR(VAR accept); IF (accept) THEN restart_tmo(PAUSE_TMO); END;	PAUSE

6.3.7.6.8 消费者状态事件表

一旦某个事件发生,消费者应从当前状态转换到下一状态,并执行表 133 规定的动作。

表 133 消费者状态事件表

[illegible]

表 133 (续)

当前状态	事件	动作	下一状态
RECEIVE	TMO AND (rep_cnt<(MAX_TRIALS-1))	IF (cancelled) THEN reset_tmo; ELSE send_BR(conn_cun_nr, offset); INC(rep_cnt); restart_tmo(REPEAT_TMO); END;	IDLE
	TMO AND (rep_cnt=(MAX_TRIALS-1))	close_rcv(AM_REPEAT_TMO_ERR);	IDLE
	rcv_BC AND (BC_run_nr = conn_run_nr) AND (my_node=final_node)	(* 无动作 *)	
	(rcv_BS OR rcv_BD) AND ((BC_run_nr<) conn_run_nr) OR (my_node<)final_node))	close_rcv(AM_FAILURE);	IDLE
	rcv_BC AND ((BC_run_nr<) conn_run_nr) OR (my_node<) final_node))	close_rcv(AM_FAILURE); accept_BC;	IDLE IDLE 或 FROZEN 或 RECEIVE
	rcv_BC AND ((BC_run_nr = conn_run_nr) OR (my_node= final_node))	close_rcv(AM_REM_CANC_ERR);	IDLE
FROZEN	TMO OR rcv_BS	reset_tmo;	IDLE
	rcv_BC AND (BC_run_nr = conn_run_nr) AND (my_node= final_node)	IF (BC_rep_cnt = 0) THEN reset_tmo; END;	IDLE
	rcv_BC AND ((BC_run_nr<) conn_run_nr) OR (my_node<) final_node))	reset_tmo; accept_BC;	IDLE IDLE 或 FROZEN 或 RECEIVE

6.3.8 消息会话层

6.3.8.1 目的

会话层配对两个消息：一个呼叫消息(Call_Message)和一个应答消息(Reply_Message)。
呼叫消息由呼叫者发往应答者，应答消息从应答者发往呼叫者。
这对消息允许实现图 152 所示的远程过程调用。

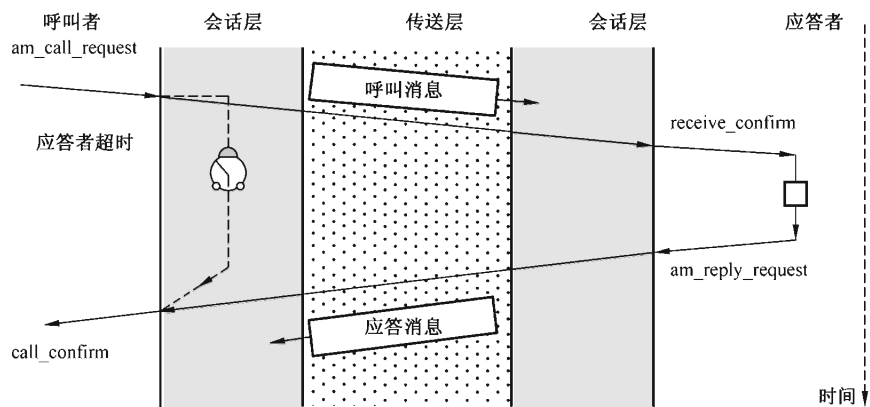


图 152 会话层传送

会话层使用传输层服务以传送每一个消息。

6.3.8.2 会话标识符

会话层应通过一个会话标识符(Conversation_Id)唯一标识通信合作方,会话标识符由以下连接组成:

- 远程应用的网络地址(Network_Address);
- 本地应用的功能标识符(Function_Id)。

只要另一个具有相同会话标识符的会话正在运行,则会话层应拒绝使用该会话标识符的通信请求。会话层应保留呼叫者的完整地址,以转发相应应答消息(Reply_Message)。

注: 会话标识符包含向网络层转发包和标识来自网络层的包所必须的所有信息。

6.3.8.3 取捷径

当通信合作方驻留在同一个站时,会话层应取网络捷径,即直接转发消息而不涉及传输层。

6.3.8.4 拓扑计数器检查

会话层应根据以下算法检查 my_topo(由应用提供)和 this_topo(网络层保持的 Topo_Counter 的值)之间的一致性:

```
IF (my_topo = AM_ANY_TOPO) THEN
    my_topo = this_topo
ELSE
    IF (this_topo = AM_ANY_TOPO) THEN
        this_topo = my_topo
    ELSE
        IF (my_topo <> this_topo) THEN
            reject the call
```

会话层应在该会话期间使用拓扑计数器(Topo_Counter)的值。

注: 这保证了若在呼叫和应答消息之间发生初运行则会话被取消。

6.3.8.5 会话报头编码

在呼叫消息的连接请求中,会话报头(Session_Header)由一个以“0”填充的八位位组构成。所有其他组合保留,如图 153 所示。

在应答消息的连接请求中,会话报头由一个包含应答者应用提供的应答状态的八位位组构成。

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

图 153 呼叫消息中的会话报头(Am_Result 类型)

6.3.8.6 缓冲区管理

会话层应提供两种缓冲区管理:

- a) 由应用分配和回收的静态缓冲区;
- b) 由会话分配和回收的动态缓冲区。

6.3.8.7 会话层接口

会话层接口和应用层接口相同,因为表示层没有协议。

6.3.9 消息表示层

消息表示层没有协议。

消息应按 ARRAY OF WORD8 格式、内存地址升序传送。

消息报头和参数应遵守与过程变量(Process_Variables)相同的数据表示规则(例如,所有数据首先传送最高有效的八位位组)。

协议中使用的数据类型和为应用推荐的数据类型列于 6.4 中。

6.3.10 消息应用层

6.3.10.1 目的

应用消息接口(Application_Message_Interface, AMI)允许应用通过网络发送和接收消息。AMI 提供呼叫/应答服务,以及初始化、缓冲区管理和多播服务。

AMI 定义为一组直接访问会话层的过程(表示层和应用层无协议)。

6.3.10.2 应用消息接口

6.3.10.2.1 AMI 原语

应用层接口应实现图 154 所示原语,这些原语列于表 134 中并在以下各条中规定。

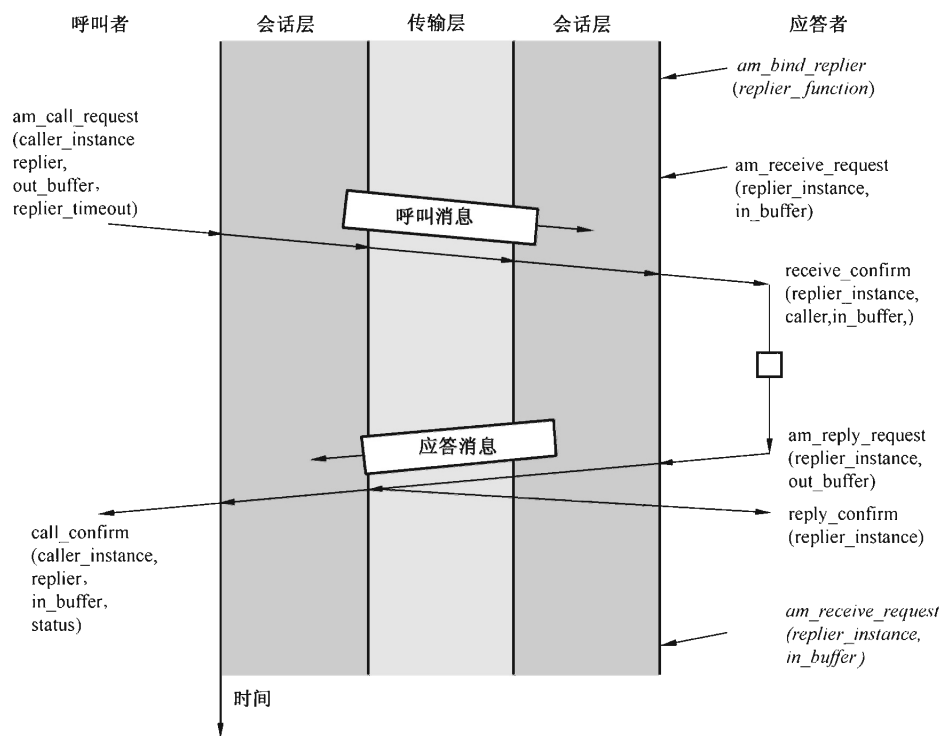


图 154 应用消息接口

注 1：AMI 对象前缀为 am_或 AM_（对于应用消息），属于呼叫者或应答者事例的对象不加前缀。

注 2：以下缩写使用于名称中：

- REM: 远程，合作方设备报告的失效；
- LOC: 本地，设备自身报告的失效；
- OVF: 溢出；
- TMO: 超时。

表 134 AMI 原语

名称	含 义
常量和类型	
AM_RESULT	过程的结果, 与 Am_Result 定义相同
AM_ADDRESS	远程实体的网络地址
初始化	
am_init	初始化信使
am_announce_device	配置设备
am_show_busses	列出所连接链路层的总线标识符
am_set_current_tc	向信使指示当前拓扑计数器
站索引接口	
AM_STADI_ENTRY	站索引项
am_stadi_write	写站索引
am_stadi_read	读站索引

表 134 (续)

名称	含 义
功能索引接口	
AM_DIR_ENTRY	功能索引项
am_clear_dir	初始化功能索引
am_insert_dir_entries	记录功能列表的站标识符
am_remove_dir_entries	删除功能列表
am_get_dir_entry	获取给定功能的站标识符
组索引接口	
AM_GROUP	组定义
am_clear_groups	清除组索引
am_insert_member	将节点纳入组索引中
am_remove_member	从组索引中移除节点
am_member	组索引中的成员
呼叫者接口	
am_call_request	呼叫者发送完整消息
AM_CALL_CONFIRM	当应答到达时所调用过程的类型
am_call_cancel	取消会话并丢弃应答消息
应答者接口	
am_bind_replier	向会话层通告应答者事例
am_unbind_replier	取消以上通告
am_receive_request	通告事例已为下一个呼叫就绪
AM_RECEIVE_CONFIRM	当呼叫完成时所调用过程的类型
am_reply_request	被应答者事例调用以回送应答消息
AM_REPLY_CONFIRM	当应答发送时所调用过程的类型
am_receive_cancel	取消已就绪或使用中的应答者事例
缓冲区控制	
am_buffer_free	回收动态消息缓冲区

注：接口过程是非阻塞的，且任务调度不受限制。

6.3.10.2.2 AM_RESULT 定义

AM_RESULT 定义见表 135。

表 135 AM_RESULT 定义

定义	返回值类型为 AM_RESULT 的过程应编码返回值如下：
语法	<pre>typedef enum { AM_OK =0, /* 成功终止 AM_FAILURE =1, /* 未规定的失效 AM_BUS_ERR =2, /* 无总线传送可能 AM_REM_CONN_OVF =3, /* 传入连接过多 AM_CONN_TMO_ERR =4, /* 未回答连接请求 AM_SEND_TMO_ERR =5, /* 发送超时(连接正常) AM_REPLY_TMO_ERR =6, /* 未收到应答 AM_ALIVE_TMO_ERR =7, /* 未收到完整消息 AM_NO_LOC_MEM_ERR =8, /* 存储器或定时器不足 AM_NO_REM_MEM_ERR =9, /* (合作方)存储器或定时器不足 AM_REM_CANC_ERR =10, /* 被合作方取消 AM_ALREADY_USED =11, /* 相同操作已执行 AM_ADDR_FMT_ERR =12, /* 地址格式错误 AM_NO_REPLY_EXP_ERR =13, /* 非预期应答 AM_NR_OF_CALLS_OVF =14, /* 请求的呼叫过多 AM_REPLY_LEN_OVF =15, /* 应答消息过长 AM_DUPL_LINK_ERR =16, /* 复制会话出错 AM_MY_DEV_UNKNOWN_ERR=17, /* 自身设备未知或无效 AM_NO_READY_INST_ERR =18, /* 没有就绪的应答者事例 AM_NR_OF_INST_OVF =19, /* 应答者事例过多 AM_CALL_LEN_OVF =20, /* 呼叫消息过长 AM_UNKNOWN_DEST_ERR =21, /* 合作方设备未知 AM_INAUG_ERR =22, /* 已发生列车初运行 AM_TRY_LATER_ERR =23, /* (仅在内部使用) AM_FIN_NOT_REG_ERR =24, /* 未记录终点 AM_GW_FIN_NOT_REG_ERR=25, /* 路由器未记录终点 AM_GW_ORI_REG_ERR =26, /* 路由器未记录起始 AM_MAX_ERR =31 /* 最高的系统代码 /* 用户定义的代码高于 31 } AM_RESULT;</pre>

若 AMI 过程返回应用相关的用户代码作为结果,则该值应大于 AM_MAX_ERR 且小于 256。

注：AM_RESULT 使用与包中传送的 Am_Result 字段相同的编码方式。

6.3.10.2.3 地址常量

列于表 136 的常量是保留的标识符。

表 136 地址常量

常量	代码	含义
AM_SAME_STATION	0	本站,与站标识符无关
AM_UNKNOWN	255	未知的站标识符
AM_MAX_BUSSES	1~16	实现支持的链路层最大数量
AM_ROUTER_FCT	251	路由器的功能标识符
AM_AGENT_FCT	253	代理者的功能标识符
AM_MANAGER_FCT	254	管理者的功能标识符
AM_SAME_NODE	0	同一节点的通信,不通过列车总线
AM_SYSTEM_ADDR	128	节点地址的位 0 指示系统地址
AM_ANY_TOPO	0	拓扑计数器未知

6.3.10.2.4 类型“AM_ADDRESS”

呼叫者或应答者应通过其 AM_ADDRESS 类型的应用地址标识另一合作方。类型“AM_ADDRESS”见表 137。

表 137 类型“AM_ADDRESS”

定义	(呼叫者或应答者)应用地址的类型	
语法	<pre>typedef struct AM_ADDRESS ——大开端表示 { UNSIGNED snu :1, /* bit 7 */ UNSIGNED gni :1, /* bit 6 */ UNSIGNED node_or_group :6, UNSIGNED func_or_stat :8, UNSIGNED next_station :8, UNSIGNED topo_rsv :1, /* bit 7 */ UNSIGNED topo_valid :1, /* bit 6 */ UNSIGNED topo_counter :6 } AM_ADDRESS;</pre>	
元素	snu	(系统,非用户) 位 7 = 0,指示用户地址; 位 7 = 1,指示系统地址
	gni	(组,非单个) 位 6 = 0,指示单个(节点)地址; 位 6 = 1,指示组地址
	node_or_group	若 gni = 0,则位 0~5 指定一个节点地址; 若 gni = 1,则位 0~5 指定一个组地址
	func_or_stat	若 snu = 0,则为功能标识符; 若 snu = 1,则为站标识符
	next_station	下一站标识符
	topo_rsv	保留,一直为 0
	topo_valid	(拓扑有效)指示其后的 topo_counter 是否有效
	topo_counter	拓扑计数器或 AM_ANY_TOPO

注：如下规定,在呼叫者侧和应答者侧,字段含义不同。

应用地址一种方便的编码方式如图 155 所示。

7	6	5	4	3	2	1	0
系统/用户 (snu)	组/单个 (gni)	节点/组地址(node_or_group)					
功能/站标识符(func_or_stat)							
下一站站标识符(next_station)							
保留 (trv)	拓扑有效 (tpv)	拓扑计数器(topo_counter)					

注：AM_ADDRESS 是接口格式，Am_Address 是传送格式。

图 155 AM_ADDRESS 编码

6.3.10.3 呼叫者侧

6.3.10.3.1 自身标识

呼叫者应通过其功能标识符(Function_Id)标识自己。

注：呼叫者在“am_call_request”中标识自己。

6.3.10.3.2 呼叫者事例

由于呼叫者在接收到应答之前能建立多个呼叫，变量“caller_ref”应链接“am_call_request”与相应的“call_confirm”。

6.3.10.3.3 系统或用户

若其他非管理者的功能使用系统地址产生呼叫，则不应执行该呼叫，且应在“call_confirm”中报告地址错误。

注：仅当通信不通过列车总线传递(节点 = AM_SAME_NODE)时，任意功能可通过指定下一站的方式通过其用户地址调用代理者功能或管理者功能。

6.3.10.3.4 组或单个(gni)

若呼叫者设置“gni”位为“0”，则应使用单播协议，且随后六位应为节点地址。

若呼叫者设置“gni”位为“1”，则应使用多播协议，且随后六位应为组地址。

注：多播协议使用相同的地址格式。

6.3.10.3.5 节点或组(node_or_group)

若呼叫者指定(Node_Address<>AM_SAME_NODE)，则呼叫应被转发到列车总线节点。

注：即使应答者的节点地址和呼叫者的节点地址相同，消息也转发到检查拓扑计数器并通过编组网返回消息的节点。

6.3.10.3.6 站或功能(func_or_stat)

任意 Function_Id 可与一个用户地址一起使用。

管理者可在 System_Address 中指定站未知(Station_Id = AM_UNKNOWN),但是,若“next_station”未知(next_station = AM_UNKNOWN),则不应执行呼叫且应在“call_confirm”中报告地址错误。

注 1: 这允许管理者访问在初始化时具有未知站标识符的站。

注 2: 当呼叫通过列车总线发送时,0 或 255 的 Station_Id 寻址远端节点,而与其站标识符无关。

6.3.10.3.7 下一站(Next_Station)

Next_Station 指定下一次消息应转发到的链路地址。Next_Station 也能指定终点站或一个路由器站。应按如下方法计算:

- a) 若“Next_Station_Id <> AM_UNKNOWN”,则应使用 Next_Station_Id 作为项从站索引中取出 Link_Address。
- b) 若“Next_Station_Id = AM_UNKNOWN”,则应使用以下缺省值作为项从站索引中取出 Link_Address:
 - 若消息被发送到列车总线“Node_Address <> AM_SAME_NODE”或(多播),则应使用路由器功能(AM_ROUTER_FCT)作为项从功能索引取出 Next_Station_Id。
 - 若消息没有被发送到列车总线“Node_Address = AM_SAME_NODE”:
 - ◆ 若是系统地址,则 Next_Station_Id 应等于 Station_Id。
 - ◆ 若是用户地址,则应使用功能标识符作为项从功能索引中获取 Next_Station_Id。
- c) 若“Next_Station_Id = AM_SAME_STATION”或“Next_Station_Id = this_station”,信使应转发呼叫消息给本地应答者(若存在)。

若站索引没有与 Next_Station_Id 相对应的链路地址项,则应产生地址错误。

注: 若呼叫者驻留在该节点上,则“next_station = AM_SAME_STATION”。

6.3.10.3.8 拓扑计数器(Topo_Counter)

该八位位组的位 7(最高有效位)应为 0。

该八位位组的位 6 应为 0。

位 0~5 应包含一个范围在 1~63 的有效拓扑计数器。

否则,该八位位组所有位应为 0(AM_ANY_TOPO)。

若应用为“node <> AM_SAME_NODE”的任意呼叫指定了值 AM_ANY_TOPO,则应产生地址错误。

注: 若呼叫者忽略拓扑,则其不能通过列车总线发送点对点消息。期望首先从节点或中间应用获取拓扑。在固定的列车总线组态中,任何拓扑计数器的值都可接受。

6.3.10.3.9 呼叫者侧网络地址的使用

系统地址和用户地址模式总结在表 138 中。

表 138 系统地址和用户地址

系统地址		同一节点	其他节点
Next_Station =	Station_Id =	Link_Address =	Link_Address =
AM_SAME_STATION	AM_SAME_STATION	取捷径到自身站	Node_Address (若 this_station 不是一个节点则出错)
	⟨⟩ AM_SAME_STATION and ⟨⟩ AM_UNKNOWN	错误	
	AM_UNKNOWN	取捷径到自身站	
⟨⟩ AM_SAME_STATION and ⟨⟩ AM_UNKNOWN	AM_SAME_STATION	stadi(next_station)	stadi(next_station) (代理者位于远端站)
	⟨⟩ AM_SAME_STATION and ⟨⟩ AM_UNKNOWN		
	AM_UNKNOWN		
AM_UNKNOWN	AM_SAME_STATION	取捷径到自身站	stadi(fundi(AM_ROUTER_FCT)) (代理者位于远端节点)
	⟨⟩ AM_SAME_STATION and ⟨⟩ AM_UNKNOWN	stadi(station_Id)	stadi(fundi(AM_ROUTER_FCT)) (代理者位于远端站)
	AM_UNKNOWN	错误	stadi(fundi(AM_ROUTER_FCT)) (代理者位于远端节点)
用户地址		同一节点	其他节点
Next_Station =	Function_Id =	Link_Address =	Link_Address =
AM_SAME_STATION	任意	取捷径到自身站	列车总线, Node_Address (若站不是节点则错误)
⟨⟩ AM_SAME_STATION and ⟨⟩ AM_UNKNOWN	任意	stadi(next_station)	stadi(fundi(AM_ROUTER_FCT))
AM_UNKNOWN	任意	stadi(fundi(Function_Id)) (若功能未登记则错误)	stadi(fundi(AM_ROUTER_FCT)) 或(列车总线, Node_Address)

6.3.10.4 应答者侧

6.3.10.4.1 应答者事例

应答者进程是应用进程。多个应答者事例可并行服务同一个功能。呼叫者不能指定哪一个事例服务呼叫。

每一个应答者功能应在该功能能调用过程“am_receive_request”接收传入呼叫、调用过程“am_reply_request”应答接收到的呼叫之前绑定。

当等待呼叫消息时或在应答消息传送期间,应答者进程不阻塞,而是当已接收到呼叫消息或已完成

应答消息传送时通知应答者进程。

规定绑定为通知而调用的确认过程,因此对同一应答者功能的所有事例确认过程都相同。

应答者事例应在能发布另一“am_receive_request”之前应答或取消每一个接收到的呼叫。每一个尚未确认的请求也能被取消。已被成功取消的请求将不被确认。

6.3.10.4.2 应答者标识

由于一个功能能被多个事例执行,变量“replier_ref”应将“am_receive_request”与相应的 receive_confirm 链接、“am_repy_request”与相应的 reply_confirm 链接。

应答者事例应通过其功能标识符(Function_Id)和外部引用(External_Reference)标识。

注:会话层为应答消息保留在呼叫消息中接收到的应答者的全地址。会话层把外部引用作为会话标识符的一部分保留。

6.3.10.4.3 系统或用户

若接收到带系统地址的消息,则“snu”位应置 1;且此时应调用代理者功能,呼叫者隐含为管理者。

若接收到带用户地址的消息,则“snu”位应置 0。

注:可由通过用户地址的其他功能寻址代理者,但只能来自连接到同一个节点上的站。

6.3.10.4.4 组或单个

“gni”位应置 1 以表示已通过多播地址接收到消息;若已通过单播地址接收到消息,则应置 0。

注:这允许不区分单播或多播协议的呼叫应答者。

6.3.10.4.5 节点或组

无论使用单个地址或组地址,随后 6 位应指示呼叫者的节点地址,或若呼叫者在其应答者地址中指定了 AM_SAME_NODE 则为 AM_SAME_NODE。

注:若呼叫者指定了节点地址,则即使消息不通过列车总线传送,该地址也传送给应答者。

6.3.10.4.6 下一站

Next_Station 应为接收呼叫所通过站的站标识符,或若终点站为节点自身则 Next_Station 应为 AM_SAME_STATION。

6.3.10.4.7 拓扑计数器

若消息已通过节点转发,则应答者应接收所连接节点的拓扑计数器;否则,该字段置为 AM_ANY_TOPO。

注:应答者负责检查呼叫消息的拓扑计数器值是否与“my_topo”值匹配。

6.3.10.5 初始化

6.3.10.5.1 概述

消息服务由下列过程在不同层初始化。

以下各条不隐含特定实现。任何提供相同语义的接口都被允许。

6.3.10.5.2 过程“am_init”

过程“am_init”见表 139。

表 139 过程“am_init”

定义	初始化信使并调用 am_clear_dir 初始化索引	
语法	AM_RESULT am_init (void);	
结果	AM_RESULT	AM_OK 或 AM_FAILURE
用法	应在系统初始化时调用任何其他 am_xxx 过程之前调用该过程	

6.3.10.5.3 过程“am_announce_device”

过程“am_announce_device”见表 140。

表 140 过程“am_announce_device”

定义	声明设备的配置	
语法	AM_RESULT am_announce_device (UNSIGNED16 max_call_number, UNSIGNED16 max_inst_number, UNSIGNED16 default_reply_timeout, UNSIGNED8 my_credit);	
输入	max_call_number	该设备上并发呼叫数
	max_inst_number	该设备上任意应答者并发事例数(缺省值为 3)
	default_reply_timeout	缺省的呼叫请求应答超时
	my_credit	对所有终结于该设备的连接的最大(可接收)信用值,且截断到 AM_MAX_CREDIT
返回	任意 AM_RESULT	
用法	该过程直接从链路层获得节点地址	

6.3.10.5.4 过程“am_show_busses”

过程“am_show_busses”见表 141。

表 141 过程“am_show_busses”

定义	获取连接到该站的链路层(总线)的数量,并列出其总线标识符	
语法	AM_RESULT am_show_busses (UNSIGNED8 * nr_of_busses, UNSIGNED8 link_id_list[AM_MAX_BUSSES]);	
返回	任意 AM_RESULT	
输出	nr_of_busses	已连接的链路层数(也是 link_id_list 中元素数)
	link_id_list	链路层的列表,每项至少包含总线标识符

6.3.10.5.5 过程“am set current tc”

过程“am set current tc”见表 142。

表 142 过程“am_set_current_tc”

定义	为网络层设置拓扑计数器当前值为 this_topo	
语法	<pre>AM_RESULT am_set_current_tc (UNSIGNED8 this_topo);</pre>	
输入	this_topo	拓扑计数器,或当拓扑计数器未知时为 AM_ANY_TOPO
返回	若“AM_ANY_TOPO \leq this_topo $<$ 63”,则为 AM_OK;否则,为 AM_FAILURE	
用法	<p>a) 期望(呼叫者或应答者)应用获得拓扑计数器的当前值并将其复制到“my_topo”变量中。随不同应用,即使在同一节点内,该值通常不同,由于初运行能在任意时间发生且不期望应用被通知每次改变。</p> <p>b) 未规定应用接收拓扑计数器的途径:可通过管理消息,通过直接访问节点链路层,通过周期性变量等。</p> <p>c) 若 this_topo 不等于 AM_ANY_TOPO,信使将拒绝后续拓扑计数器值不等于该值或不等于 AM_ANY_TOPO 的呼叫,并在“call_confirm”中报以结果 AM_INAUG_ERR。</p> <p>d) this_topo 的初始值为 AM_ANY_TOPO</p>	

6.3.10.6 站索引接口

6.3.10.6.1 概述

站索引是可选的。简单系统可以固定方式完成映射。在使用站索引时,可通过以下过程完成。以下各条不隐含特定实现。任何提供相同语义的接口都被允许。

6.3.10.6.2 类型“AM STADI ENTRY”

类型“AM STADI ENTRY”见表 143。

表 143 类型“AM STADI ENTRY”

定义	站索引项的类型	
语法	<pre>typedef struct { UNSIGNED8 station; UNSIGNED8 next_station; ENUM8 bus_id; UNSIGNED64 device_adr; } AM_STADI_ENTRY;</pre>	
元素	station	站标识符(获取 next_station 的关键字)
	next_station	下一站站标识符,对直接可达的站,下一站站标识符等于站标识符
	bus_id	总线标识符
	device_adr	设备地址(与总线相关)

6.3.10.6.3 过程“am_stadi_write”

过程“am_stadi_write”见表 144。

表 144 过程“am_stadi_write”

定义	在站索引中插入一定数量的项,每个项进行有效性和一致性检查,不符合要求者被拒绝	
语法	AM_RESULT am_stadi_write (Const entries[], AM_STADI_ENTRY nr_of_entries UNSIGNED8);	
输入	entries	新的站索引项列表
	nr_of_entries	项中元素数
返回	AM_OK:所有项成功写入站索引。 AM_FAILURE:列表中任意项未通过检查。此时,仅未通过检查的项被拒绝	

6.3.10.6.4 过程“am_stadi_read”

过程“am_stadi_read”见表 145。

表 145 过程“am_stadi_read”

定义	从站索引中读取一定数量的项	
语法	AM_RESULT am_stadi_read (AM_STADI_ENTRY entries[], UNSIGNED8 nr_of_entries);	
输入	entries[].station	拟读取的项
	nr_of_entries	项数
返回	AM_OK 或 AM_FAILURE	
输出	entries[].next_station	字段 entries[].next_station 是输出信息

6.3.10.7 功能索引接口

6.3.10.7.1 概述

以下各条不隐含特定实现。任何提供相同语义的接口都被允许。

6.3.10.7.2 类型“AM_DIR_ENTRY”

类型“AM_DIR_ENTRY”见表 146。

表 146 类型“AM_DIR_ENTRY”

定义	功能索引项的类型	
语法	typedef struct	AM_DIR_ENTRY
		{
	UNSIGNED8	function;
	UNSIGNED8	station;
		} AM_DIR_ENTRY;

6.3.10.7.3 过程“am_clear_dir”

过程“am_clear_dir”见表 147。

表 147 过程“am_clear_dir”

定义	设置索引中所有功能的站标识符为 AM_UNKNOWN	
语法	AM_RESULT am_clear_dir(void);	
输出	AM_OK 或 AM_FAILURE	

6.3.10.7.4 过程“am_insert_dir_entries”

过程“am_insert_dir_entries”见表 148。

表 148 过程“am_insert_dir_entries”

定义	对列于 function_list 列表第一个元素的每一个功能标识符,插入站标识符	
语法	AM_RESULT	am_insert_dir_entries
		(
	AM_DIR_ENTRY *	function_list,
	UNSIGNED8	number_of_entries
);
输入	function_list	功能索引列表
	number_of_entries	该列表中元素数
返回	AM_OK 或 AM_FAILURE	

6.3.10.7.5 过程“am_remove_dir_entries”

过程“am_remove_dir_entries”见表 149。

表 149 过程“am_remove_dir_entries”

定义	对列于 function_list 列表第一个元素的每一个功能标识符,置站标识符为 AM_UNKNOWN	
语法	AM_RESULT am_remove_dir_entries (AM_DIR_ENTRY * function_list, UNSIGNED8 number_of_entries);	
输入	function_list	功能索引列表
	number_of_entries	该列表中元素数
返回	AM_OK 或 AM_FAILURE	

6.3.10.7.6 过程“am_get_dir_entry”

过程“am_get_dir_entry”见表 150。

表 150 过程“am_get_dir_entry”

定义	从功能索引中为给定功能标识符读取站标识符	
语法	AM_RESULT am_get_dir_entry (UNSIGNED8 function, UNSIGNED8 * station);	
输入	function	功能标识符(关键字)
输出	station	执行功能的站标识符,或若功能没有指定站则为 AM_UNKNOWN
返回	AM_OK 或 AM_FAILURE	

6.3.10.8 组索引接口

6.3.10.8.1 概述

以下各条不隐含特定实现。任何提供相同语义的接口都被允许。

6.3.10.8.2 类型“AM_GROUP”

类型“AM_GROUP”见表 151。

表 151 类型“AM_GROUP”

定义	组索引项的类型
语法	typedef UNSIGNED8 AM_GROUP;
用法	组以一个 6 位地址标识,该地址用一个八位位组表示,其高 2 位忽略

6.3.10.8.3 过程“am_clear_group”

过程“am_clear_group”见表 152。

表 152 过程“am_clear_group”

定义	清除组索引中所有项	
语法	AM_RESULT	am_clear_group(void);
输出	AM_OK 或 AM_FAILURE	

6.3.10.8.4 过程“am_insert_member”

过程“am_insert_member”见表 153。

表 153 过程“am_insert_member”

定义	将该站标识符作为一个成员注册到组索引中	
语法	AM_RESULT am_insert_member (AM_GROUP group);	
输入	group	该站拟归属的组
返回	AM_OK 或 AM_FAILURE	

6.3.10.8.5 过程“am_remove_member”

过程“am_remove_member”见表 154。

表 154 过程“am_remove_member”

定义	将该站标识符作为组成员移除	
语法	AM_RESULT am_remove_member (AM_GROUP group);	
输入	group	该站拟从中移除的组
返回	AM_OK 或 AM_FAILURE	

6.3.10.8.6 过程“am_member”

过程“am_member”见表 155。

表 155 过程“am_member”

定义	检查指定的站标识符是否是组成员	
语法	AM_RESULT am_member (AM_GROUP group);	
输入	group	该站标识符归属的组
返回	若该站标识符是组成员则为 AM_OK;否则,为 AM_FAILURE	

6.3.10.9 呼叫者应用接口

6.3.10.9.1 概述

以下各条不隐含特定实现。任何提供相同语义的接口都被允许。

6.3.10.9.2 过程“am_call_request”

过程“am_call_request”见表 156。

表 156 过程“am_call_request”

定义	请求发送呼叫消息,为接收应答消息设置数据结构并订阅指示性过程	
语法	void am_call_request (UNSIGNED8 caller_function, const AM_ADDRESS * replier, void * out_msg_adr, UNSIGNED32 out_msg_size, void * in_msg_adr, UNSIGNED32 in_msg_size, UNSIGNED16 reply_timeout, AM_CALL_CONFIRM call_confirm, void * caller_ref);	
输入	caller_function	呼叫者的功能标识符。 若应答者地址是系统地址则应使用 AM_MANAGER_FCT
	replier	应答者功能或站的应用地址
	out_msg_adr	指向拟传送的呼叫消息的指针
	out_msg_size	以八位位组计的呼叫消息总长度
	in_msg_adr	指向放置应答消息的缓冲区的指针
	in_msg_size	以八位位组计的已接收的应答消息最大总长度
	reply_timeout	64ms 倍数的超时值,用于呼叫消息传送后的应答
	call_confirm	指向呼叫确认过程。除非“am_call_request”被 am_call_cancel 成功地取消,否则将调用“call_confirm”
	caller_ref	“call_confirm”返回的呼叫外部引用。能被呼叫者任意使用

表 156 (续)

返回	该过程不返回值,因为结果由“call_confirm”提供
用法	a) 在调用“am_call_request”之前,应调用过程 am_init 和 am_announce_device。 b) 若 in_msg_adr 为 NULL(空),则信使为应答消息分配缓冲区。呼叫者负责在使用后调用 am_buffer_free 返回该缓冲区。 c) 若已建立具有相同呼叫者和应答者地址(且方向相同)的会话,则该呼叫被拒绝。 d) 若应答者的站标识符是 AM_UNKNOWN,则应为应答者功能定义一个功能索引项。 e) 若呼叫者和应答者在同一个站内,则没有总线通信发生。 f) 若功能标识符不等于 254(管理者),则拒绝带系统地址的呼叫。 g) 呼叫者不可修改消息缓冲区,直到“call_confirm”被调用。 h) 若应答超时为 0,则期待使用 am_announce_device 规定的缺省应答超时

6.3.10.9.3 类型“AM_CALL_CONFIRM”

类型“AM_CALL_CONFIRM”见表 157。

表 157 类型“AM_CALL_CONFIRM”

定义	当请求呼叫完成返回错误状态或接收到的应答消息时,会话层应调用呼叫者过程“call_confirm”,该过程即本类型	
语法	<pre>typedef void (* AM_CALL_CONFIRM) (UNSIGNED8 caller_function, void * am_caller_ref, const AM_ADDRESS * replier, void * in_msg_adr, UNSIGNED32 in_msg_size, AM_RESULT status);</pre>	
输入	caller_function	呼叫者的功能标识符
	replier	应答者功能或站的应用地址
	am_caller_ref	在相关“am_call_request”中规定的返回值
	in_msg_adr	指向包含接收到的应答消息的缓冲区的指针。若呼叫者未向应答消息提供缓冲区且同时 in_msg_size 等于 0,则该指针为 NULL(空)
	in_msg_size	以八位位组计的应答消息总长度。若发生错误或应答者应用只以一个状态应答,则为 0
	status	给出小于 AM_MAX_ERR 的错误码,或成功时由应答者提供的状态
用法	a) 过程“call_confirm”应预先被“am_call_request”订阅。 b) 若应答者地址是系统地址,则返回 AM_MANAGER_FCT。 c) 呼叫确认隐式地将呼叫消息缓冲区返回给呼叫者。 d) 在使用后,应使用 am_buffer_free 返回先前被信使分配的应答消息缓冲区	

6.3.10.9.4 过程“am call cancel”

过程“am call cancel”见表 158。

表 158 过程“am call cancel”

定义	取消尚未确认的呼叫请求	
语法	<pre> AM_RESULT am_call_cancel (UNSIGNED8 caller_function, const AM_ADDRESS * replier); </pre>	
输入	caller_function	呼叫者的功能标识符
	replier	被呼叫的功能或站的应用地址
返回	成功取消则为 AM_OK,任何错误则为 AM_FAILURE	
用法	若返回值是 AM_OK,则不调用呼叫确认过程	

6.3.10.10 应答者应用接口

6.3.10.10.1 概述

以下各条不隐含特定实现。任何提供相同语义的接口都被允许。

6.3.10.10.2 过程“am_bind_replier”

过程“am_bind_replier”见表 159。

表 159 过程“am_bind_replier”

定义	使应答者被信使知晓,并连接其过程	
语法	<pre> AM_RESULT am_bind_replier (UNSIGNED8 replier_function, AM_RECEIVE_CONFIRM receive_confirm, AM_REPLY_CONFIRM reply_confirm); </pre>	
输入	replier_function	拟绑定的应答者的功能标识符
	receive_confirm	在接收请求完成后调用的接收确认过程
	reply_confirm	在应答完成后调用的应答确认过程
返回	<p>AM_OK = 0:绑定成功,否则 am_bind_replier 不执行任何动作。</p> <p>AM_ALREADY_USED:该应答者功能已被绑定,不修改确认过程。</p> <p>AM_NO_LOC_MEM_ERR:没有用于绑定表的存储器,没有能绑定的应答者功能。</p> <p>AM_FAILURE:绑定表满。绑定表大小由 am_announce_device 定义</p>	
用法	<p>a) 应在“am_bind_replier”之前调用“am_init”和“am_announce_device”。</p> <p>b) 每一个应答者应在其能发布任何“am_receive_request”之前绑定</p>	

6.3.10.10.3 过程“am_unbind_replier”

过程“am_unbind_replier”见表 160。

表 160 过程“am_unbind_replier”

定义	取消指定应答者的所有事例,并解除绑定	
语法	AM_RESULT am_unbind_replier (UNSIGNED8 replier_function) ;	
输入	replier_function	拟解绑的应答者的功能标识符
返回	AM_OK 或 AM_FAILURE	
用法	取消在调用 am_unbind_replier 之前已接收到但尚未应答的呼叫	

6.3.10.10.4 过程“am_receive_request”

过程“am_receive_request”见表 161。

表 161 过程“am_receive_request”

定义	应答者已准备接收传入呼叫的通知	
语法	AM_RESULT am_receive_request (UNSIGNED8 replier_function, void * in_msg_adr, UNSIGNED32 in_msg_size, void * replier_ref) ;	
输入	replier_function	期望呼叫的应答者的功能标识符。功能标识符为 253 则隐含系统地址
	in_msg_adr	指向传入应答消息缓冲区的指针。若 in_msg_adr 为 NULL,则信使为呼叫消息分配缓冲区。该缓冲区不可修改,直到 am_receive_request 被确认或取消
	in_msg_size	以八位位组计的能接收的呼叫消息最大总长度
	replier_ref	使用相关 receive_confirm 过程返回的外部引用。同时是事例引用,并区分服务于同一个应答者的应答者事例
返回	AM_OK:若请求不取消,则调用捆定的接收确认过程以传递已接收呼叫。 AM_ALREADY_USED:同一应答者事例已发布一个尚未确认、取消或应答的接收请求。 AM_FAILURE:应答者未绑定。 AM_NO_LOC_MEM_ERR:存储器不足以接收“am_receive_request”。 AM_NR_OF_INST_OVF:同时发布的“am_receive_request”的数量多于“am_announce_device”的参数“max_inst_number”的限制	
用法	该过程要求同一个应答者先前已调用过程“am_bind_replier”	

6.3.10.10.5 类型“AM_RECEIVE_CONFIRM”

类型“AM_RECEIVE_CONFIRM”见表 162。

表 162 类型“AM_RECEIVE_CONFIRM”

定义	当会话层接收呼叫消息时,应调用本类型的接收确认过程 receive_confirm	
语法	<pre>typedef void (* AM_RECEIVE_CONFIRM) (UNSIGNED8 replier_function, const AM_ADDRESS * caller, void * in_msg_adr, UNSIGNED32 in_msg_size, void * replier_ref);</pre>	
输入	replier_function	在相应“am_receive_request”中规定的应答者功能标识符
	caller	呼叫者地址
	in_msg_adr	指向容纳呼叫消息的缓冲区的指针
	in_msg_size	以八位位组计的接收到的呼叫消息总长度
	replier_ref	在相应“am_receive_request”中规定的外部引用
用法	a) 接收确认过程“receive_confirm”先前已被“am_bind_replier”订阅。 b) 若应答者事例未在“am_receive_request”中提供缓冲区,则应答缓冲区由信使提供,且应答者应在使用后通过“am_buffer_free”返回	

6.3.10.10.6 过程“am_reply_request”

过程“am_reply_request”见表 163。

表 163 过程“am_reply_request”

定义	请求发送应答消息以响应先前接收到的呼叫消息	
语法	<pre>AM_RESULT am_reply_request (UNSIGNED8 replier_function, void * out_msg_adr, UNSIGNED32 out_msg_size, void * replier_ref, AM_RESULT status);</pre>	
输入	replier_function	在相应“am_receive_request”中规定的应答者功能标识符
	replier_ref	在相应“am_receive_request”中规定的外部引用
	out_msg_adr	指向应答消息缓冲区的指针。该缓冲区在确认应答请求前不可修改。 若 out_msg_adr 为 NULL,则仅传送状态给呼叫者
	out_msg_size	以八位位组计的应答消息总长度
	status	由应答者提供的呼叫执行结果,发送给呼叫者及应答消息自身

表 163 (续)

返回	AM_OK 或 AM_FAILURE
用法	a) 每一个接收到的呼叫应以“am_reply_request”应答或以“am_receive_cancel”取消。 b) 此过程在传送应答消息之前返回。 c) 信使将该应答消息和指定状态一起传回给呼叫者。 d) 从应答者事例内部获取呼叫者地址

6.3.10.10.7 类型“AM_REPLY_CONFIRM”

类型“AM_REPLY_CONFIRM”见表 164。

表 164 类型“AM_REPLY_CONFIRM”

定义	当应答消息已全部发送并被呼叫者确认,或发生错误时,会话层调用本类型的应答确认过程“reply_confirm”	
语法	<pre>typedef void (* AM_REPLY_CONFIRM) (UNSIGNED8 replier_function, void * replier_ref);</pre>	
输入	replier_function	在相应“am_receive_request”中规定的应答者功能标识符
	replier_ref	在相应“am_receive_request”中规定的外部引用
用法	a) “reply_confirm”已被“am_bind_replier”事先订阅。 b) 此过程将应答消息缓冲区返回给应答者事例。 c) 对同一应答者事例,应答确认允许一个另外的“am_receive_request”	

6.3.10.10.8 过程“am_receive_cancel”

过程“am_receive_cancel”见表 165。

表 165 过程“am_receive_cancel”

定义	取消尚未确认的“am_receive_request”或“am_reply_request”,或通告不应答已接收的呼叫	
语法	<pre>AM_RESULT am_receive_cancel (UNSIGNED8 replier_function, void * replier_ref);</pre>	
输入	replier_function	在相应“am_receive_request”中规定的应答者功能标识符
	replier_ref	在相应“am_receive_request”中规定的外部引用
返回	AM_OK 或 AM_FAILURE	
用法	a) 不再调用已成功取消的应答请求的确认过程。 b) 用户负责辨别接收请求是否已完成(即仍接收到呼叫消息),并释放呼叫消息的动态缓冲区	

6.3.10.11 过程“am buffer free”

过程“am buffer free”见表 166。

表 166 过程“am buffer free”

定义	释放在先前被会话层分配的使用后的消息缓冲区	
语法	<pre> AM_RESULT am_buffer_free (void * in_msg_adr, UNSIGNED32 size); </pre>	
输入	in_msg_adr	指向被释放缓冲区的指针
	size	以八位位组计的该缓冲区总长度
返回	AM_OK 或 AM_FAILURE	
用法	缓冲区分配与包池管理无关	

6.3.10.12 多播应用接口

多播消息应用接口应与单播消息应用接口相同。

不期待应答者回送应答消息,但期望应答者调用“am_reply_request”释放动态缓冲区(若使用完)。但在此情况下不应产生应答消息。

6.4 传送和存储数据的表示和编码

6.4.1 目的

本条规定数据类型,并定义抽象语法符号以表示这些数据类型和用于传送或存储的编码规则。该符号基于 ASN.1(见 GB/T 16262),但包含适用于实时通信的附加结构。与之相反,设备内部接口以精确性较差的 C 语言规定。

6.4.2 数据排序

6.4.2.1 传送格式

本部分规定通过列车通信网络传送的位和字的顺序。为此,定义了一些原始类型和结构体。数据的含义不属于本部分范畴。

6.4.2.2 通信存储器格式

本部分推荐数据存入通信存储器(Traffic_Store)的格式与其通过总线传送的格式相同,均作为八位位组数组处理。

6.4.2.3 应用数据格式

本部分不规定应用数据格式。期待由接口过程完成应用数据格式和用于存储或传送的格式之间的转换。

6.4.2.4 通用规则

本部分应遵循如下通用规则：

- a) 对于 WTB 的数据结构,应从右到左、从顶到底编号,编号从 0 开始,对于 MVB 的数据结构,应
从左到右、从顶到底编号,编号从 0 开始。
- b) 存储器应作为八位位组数组对待,且应按照地址升序传送(以八位位组、32 位字等传送),与被
传送单元长度无关。第一个八位位组的八位位组偏置为 0。
- c) 数据结构内的位应以相对于该结构起始处的偏置标识。若结构包含一个无符号整数,则该整
数的最低有效位偏置为零。在读取该数据结构时此位被认为是“最右边”的一位。
- d) 为便于理解,6.4.3.*x* 中的位编号以编程者视角对齐到通信存储器数据结构。位偏置 *x* 可解释
为 2 的 *x* 次方,即 2^x 。位组中的第 *x* 位可如下计算八位位组偏置和位偏置:八位位组偏置等
于 $x/8$ 后取整,位偏置等于 x 模 8。
- e) 所有数据应首先传送最高有效八位位组(大开端)。
- f) 八位位组内的位传送顺序与总线相关,编程者不可见。例如,HDLC 协议,如 WTB,首先传送
八位位组最低有效位,而 MVB 最后传送八位位组最低有效位。
- g) 数据类型相关信息不随数据发送。在特定应用中,期待类型被定义且预先在 TCN 用户之间
达成一致。
- h) 结构体的元素(记录、序列)应按照其声明顺序传送。
- i) 数组应按照索引升序传送。多维数组按照其索引列出的顺序传送(例如,数组[行,列]逐行传
送)。
- j) 为简化实现,变量应存储在偏置为其长度整数倍的地址上(对齐)。
- k) 可变长度的数据(开式数组、记录、集合等)不应用作过程变量,但可在消息中传送。

6.4.2.5 与 ASN.1 的关系

GB/T 16262(所有部分)定义了抽象语法(ASN.1)以表示机器可读形式的数据结构。

虽然不强制传送语法,但 ASN.1 不能表示编程者在带宽或时间受限场合中经常使用的紧凑编码规
则。此外,ASN.1 不能表示不遵守其结构化方法的已有编码。

因此,本部分定义了一个基于 ASN.1 的抽象语法,该语法同时表示数据编码和数据逐位内容。

表 167 中的关键字已添加到 ASN.1 中。

表 167 与 ASN.1 的关系

关键字	关键字	关键字	关键字
ALIGN	ANTIVALENT2	ARRAY	BCD4
BIPOLAR2.16	BIPOLAR4.16	BITSET #	BITSET_L #
BOOLEAN1	BOOLEAN8	ENUM #	INDIRECT
INTEGER #	INTEGER_L #	ONE_OF	REAL32
REAL64	RECORD	SOME_OF	STOP
STRING #	TIME64	TIMEDATE48	UNICODE16
UNIPOLAR2.16	UNSIGNED #	UNSIGNED_L #	WORD #

该表示使用紧凑编码规则：

——假定目的识别所有用户定义类型；

- 使用固定长度的原始类型(在 ASN.1 中整数可是任意长度);
 - 在需要时使用专用字段明确元素长度;
 - 在语义相似但不同(例如:ONE_OF 代替 CHOICE,SOME_OF 代替 SET)处引入自有关键字以避免和 ASN.1 混淆;
 - 除了 ONE_OF 和 SOME_OF 类型由一个专用字段明确加注标签外,不使用隐式类型标签;
 - 没有选项字段(除了 SOME_OF);
 - 不对齐,尽管可规定对齐。
- 使用以下表示规则:
- 关键字(包括基本类型)和常量标识符全部大写;
 - 类型标识符首字符大写;
 - 字段标识符首字符小写。

6.4.3 原始类型的表示

6.4.3.1 布尔类型的表示

6.4.3.1.1 定义

一种具有两个特定值的原始类型:真(TRUE)和假(FALSE)。

- 注 1: 这是 ASN.1 的 BooleanType 定义。
- 注 2: 此类型用于表示二进制输入和输出(继电器、发光二极管、微动开关等)。

6.4.3.1.2 语法

BooleanType ::= BOOLEAN1

6.4.3.1.3 编码

布尔类型变量应编码成一位:

1st	解释
<div><div></div></div>	
0	FALSE
1	TRUE

6.4.3.2 反价类型的表示

6.4.3.2.1 定义

一种具有四个特定值的原始类型。

- 注 1: 这不是 ASN.1 类型。
- 注 2: 此类型变量用作其他变量或关键布尔型变量的校验变量。

6.4.3.2.2 语法

AntivalentType ::= ANTIVALENT2

6.4.3.2.3 编码

反价类型变量应作两位传送,第一位对应于变量的布尔型含义,第二位取反。

反价类型变量可为以下四个状态之一：

1	0	解释
<div>2⁺¹</div>	<div>2⁰</div>	
0	0	错误(ERROR)
0	1	FALSE
1	0	TRUE
1	1	未定义(UNDEFINED)

注：ERROR 状态和 UNDEFINED 状态可由应用解释为合法状态。

6.4.3.3 无符号整数类型的表示

6.4.3.3.1 定义

一种其特定值为全部正数和零(作为一个单值)且具有由后缀 # 定义的固定长度位的原始类型。

注：这是 ASN.1 的 IntegerType,约束为固定长度 # 且非负值。

6.4.3.3.2 语法

UnsignedType ::= UNSIGNED # , (# = 任意无符号整数)。

6.4.3.3.3 编码

6.4.3.3.3.1 概述

无符号整数应以二进制表示传送。
当携带值长度小于 UNSIGNED # 类型时,该值应右对齐且以零扩展左端。

6.4.3.3.3.2 UNSIGNED8 编码

UNSIGNED8 编码见图 156。

7	6	5	4	3	2	1	0
<div>2⁷</div>	<div>2⁶</div>	<div>2⁵</div>	<div>2⁴</div>	<div>2³</div>	<div>2²</div>	<div>2¹</div>	<div>2⁰</div>

范围:0~255。

图 156 UNSIGNED8 编码

6.4.3.3.3.3 UNSIGNED16 编码

UNSIGNED16 编码见图 157。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<div>2¹⁵</div>	<div>2¹⁴</div>	<div>2¹³</div>	<div>2¹²</div>	<div>2¹¹</div>	<div>2¹⁰</div>	<div>2⁹</div>	<div>2⁸</div>	<div>2⁷</div>	<div>2⁶</div>	<div>2⁵</div>	<div>2⁴</div>	<div>2³</div>	<div>2²</div>	<div>2¹</div>	<div>2⁰</div>

范围:0~65 535。

图 157 UNSIGNED16 编码

6.4.3.3.3.4 UNSIGNED32 编码

UNSIGNED32 编码见图 158。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

范围： $0 \sim +2^{32}-1$ 。

图 158 UNSIGNED32 编码

6.4.3.4 整数类型的表示

6.4.3.4.1 定义

一种其特定值为全部正数、负数和零(作为一个单值),且具有由后缀#定义的固定长度位的原始类型。

注:这是 ASN.1 的 integer 类型,约束为固定长度#。

6.4.3.4.2 语法

IntegerType ::= INTEGER#,(# = 任意无符号整数)。

6.4.3.4.3 编码

6.4.3.4.3.1 概述

值应以二进制补码表示。

当携带值长度小于 INTEGER#类型时,该值应右对齐且符号扩展左端(若为负,则以“1”扩展;否则以“0”扩展)。

6.4.3.4.3.2 INTEGER8 编码

INTEGER8 编码见图 159。

7	6	5	4	3	2	1	0
sign	2^6	2^5	2^4	2^3	2^2	2^1	2^0

范围： $-128 \sim +127$ 。

图 159 INTEGER8 编码

示例：“1111 1110”B = -2。

6.4.3.4.3.3 INTEGER16 编码

INTEGER16 编码见图 160。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
sign	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

范围： $-2^{15} \sim +2^{15}-1$ 。

图 160 INTEGER16 编码

6.4.3.4.3.4 INTEGER32 编码

INTEGER32 编码见图 161。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
sign	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

范围： $-2^{31} \sim +2^{31} - 1$ 。

图 161 INTEGER32 编码

6.4.3.5 枚举类型的表示

6.4.3.5.1 定义

一种其值是作为类型表示一部分的给定特异标识符,且具有由后缀 # 定义的固定长度位的原始类型。

注: 这是 ASN.1 的 ENUMERATED 类型,约束为固定长度 #。

6.4.3.5.2 语法

EnumeratedType ::= ENUM # { Enumeration }
其中:
(# = 任意无符号整数)
Enumeration ::= NamedNumber | Enumeration, NamedNumber
NamedNumber ::= identifier (UnsignedNumber) | identifier (DefinedValue)
值能以任意顺序列出。

示例:

```
Day_Of_Week_Type ::= ENUM4
{
    monday      (1),
    tuesday     (2),
    wednesday   (3),
    thursday    (4),
    friday      (5),
    saturday    (6),
    sunday      (7),
    undefined   (0)
}
```

值“2”即为“tuesday”。

6.4.3.5.3 编码

6.4.3.5.3.1 概述

ENUM # 的值应以占据相同位置的无符号整数表示。

6.4.3.5.3.2 ENUM4 编码

ENUM4 编码见图 162。

3	2	1	0
2^3	2^2	2^1	2^0

范围:0~15。

图 162 ENUM4 编码

示例：在上例中，“0001”B 即“monday”。

6.4.3.5.3.3 ENUE8 编码

ENUM8 编码见图 163。

7	6	5	4	3	2	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

范围:0~255。

图 163 ENUM8 编码

示例：在上例中，“0000 0001”B 即“monday”(将其看作 ENUM8 而不是 ENUM4)。

6.4.3.6 二进制编码的十进制类型的表示

6.4.3.6.1 定义

一个 4 位无符号整数表示一个 0~9 的十进制数。

注：此类型在 ASN.1 中不存在。

6.4.3.6.2 语法

BinaryCodedDecimalType ::= BCD4

6.4.3.6.3 编码

BCD4 应编码成一个占据相同位置的无符号整数,见图 164。

3	2	1	0
2^3	2^2	2^1	2^0

范围:0~9(其他值未定义)。

图 164 BCD4 编码

示例：“0111”B = 7。

注：可使用某些未定义的值,例如指示符号或另一个算术运算符。

6.4.3.7 单极性类型的表示

6.4.3.7.1 定义

一种原始类型,其特定值为非负数,全部数字除以 2 的固定幂,以一个范围百分数表示一个值。

注：这些类型在 ASN.1 中不存在,在 GB/T 18657.1 中表示为无符号定点数(unsigned fixed point number)。

6.4.3.7.2 语法

UnipolarType ::= UNIPOLAR2.16

注 1：点号之前的数字给出 2 的幂数,构成整数部分。

注 2：ε 因子等于字(双字节)中的 2 的最小幂值。

6.4.3.7.3 编码

6.4.3.7.3.1 概述

单极性类型变量应作为一个无符号整数传送。

6.4.3.7.3.2 UNIPOLAR2.16 编码

UNIPOLAR2.16 编码见图 165。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}
整数部分		小数部分													
范围： $0 \sim 400\% - \epsilon$ 。															

图 165 UNIPOLAR2.16 编码

6.4.3.8 双极性类型的表示

6.4.3.8.1 定义

一种原始类型，其特定值为正或负数，全部数字(包括零)除以 2 的固定幂，以一个范围百分数表示一个值。

注：这些类型在 ASN.1 中不存在，在 GB/T 18657.1 中表示为有符号定点数(signed fixed point number)。

6.4.3.8.2 语法

BipolarType ::= BIPOLAR2.16 | BIPOLAR4.16

注 1：点号之前的数字给出 2 的幂数，构成整数部分。

注 2： ϵ 因子等于字(双字节)中的 2 的最小幂值。

6.4.3.8.3 编码

6.4.3.8.3.1 BIPOLAR2.16 编码

BIPOLAR2.16 编码见图 166。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
sign	2 ⁰	2 ⁻¹	2 ⁻²	2 ⁻³	2 ⁻⁴	2 ⁻⁵	2 ⁻⁶	2 ⁻⁷	2 ⁻⁸	2 ⁻⁹	2 ⁻¹⁰	2 ⁻¹¹	2 ⁻¹²	2 ⁻¹³	2 ⁻¹⁴
整数部分		小数部分													
范围：-200%~+200%-2。															

图 166 BIPOLAR2.16 编码

6.4.3.8.3.2 BIPOLAR4.16 编码

BIPOLAR4.16 编码见图 167。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
sign	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}
整数部分				小数部分											
范围： $-800\% \sim +800\% - \epsilon$ 。															

图 167 BIPOLAR4.16 编码

6.4.3.9 实数类型的表示

6.4.3.9.1 定义

一种特定值为实数集成员的原始类型。

6.4.3.9.2 语法

RealType ::= REAL32

6.4.3.9.3 编码

该类型应编码为 IEEE 754 规定的短实数(32 位),见图 168。

注 1: 这是 ASN.1 的 RealType,约束为 IEEE 754 的短实数格式。

注 2: IEEE 754 的 64 位浮点数(REAL64)在本文中认为无用。

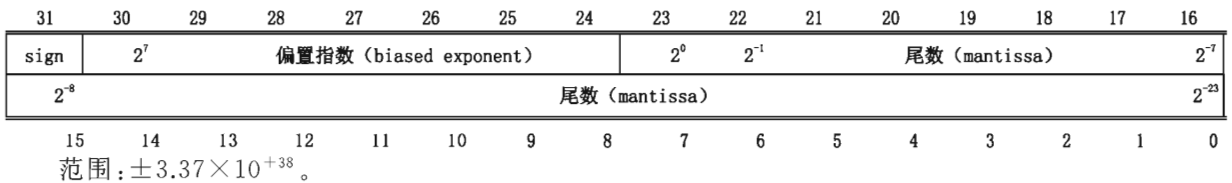


图 168 实数类型编码

6.4.3.10 字符类型的表示

6.4.3.10.1 定义

一种特定值为 GB/T 15273.1 中定义的字符集成员的原始类型。

6.4.3.10.2 语法

CharacterType ::= CHARACTER8

6.4.3.10.3 编码

字符应以一个无奇偶校验位的八位位组传送,见图 169。



图 169 字符类型编码

示例: 根据 GB/T 15273.1,“0110 0001”B = 字符“a”。

6.4.3.11 Unicode 字符类型的表示

6.4.3.11.1 定义

一种特定值为 GB/T 13000 中定义的字符集成员的原始类型。

6.4.3.11.2 语法

UnicodeType ::= UNICODE16

6.4.3.11.3 编码

Unicode 字符应以两个八位位组传送,见图 170。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

图 170 Unicode 字符类型编码

6.4.3.12 无约束类型的表示

6.4.3.12.1 定义

一种内容未定义、但长度固定的无约束类型。

6.4.3.12.2 语法

AnyType ::= WORD #, (# = 任意无符号整数)

6.4.3.12.3 编码

一个无约束类型变量没有规定的编码。

应根据占据该位置的、类型为 UNSIGNED # 的变量的 2 的幂值命名位。

注：该命名与同一字内偏置方向相反。

6.4.3.12.3.1 WORD8 编码

WORD8 编码见图 171。

7	6	5	4	3	2	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

图 171 WORD8 编码

6.4.3.12.3.2 WORD16 编码如下：

WORD16 编码见图 172。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

图 172 WORD16 编码

6.4.4 结构体

6.4.4.1 概述

定义了五种不同的结构体：

- a) 记录(RECORD):可变长度；
- b) 数组(ARRAY):固定长度或可变长度；
- c) 位集(BITSET #):固定长度；
- d) 选择(ONE_OF):可变长度；
- e) 集合(SOME_OF):可变长度。

6.4.4.2 记录类型的表示

6.4.4.2.1 定义

一种通过引用一个固定、有序的类型列表而定义的结构类型；新类型的每一个值是一个有序的值列表，值来自于每一个元素类型。

- 注 1：此类型是 ASN.1 无选项类型的 Sequence Type。
- 注 2：当定义记录(RECORD)时宜对齐，即所有元素宜位于相对该记录起始处元素偏置为元素长度的整数倍。

6.4.4.2.2 语法

RecordType ::= RECORD { ElementTypeList }

其中，

ElementTypeList ::= ElementType | ElementTypeList, ElementType

ElementType ::= identifierType | Type

记录的元素应标识以记录字段标识符，后接点号和子字段标识符。子字段标识符自身可是结构体。

示例：file.date.day。

6.4.4.2.3 编码

记录的元素应按照其声明顺序传送。

示例：一个 Date32 类型的值表示见图 173。

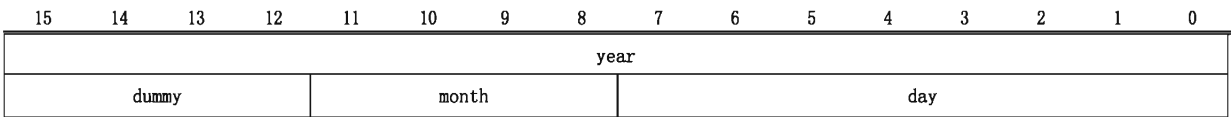


图 173 Date32 类型的值编码

Date32 ::= RECORD

{

year INTEGER16,

dummy WORD4,

month UNSIGNED4,

day UNSIGNED8

}

引入“dummy”字段使变量“day”对齐到 16 位字边界。

6.4.4.3 位集类型的表示

6.4.4.3.1 定义

一个 BOOLEAN1 的 ARRAY[#], 具有由后缀 # 定义的固定长度位。

注：该类型对应于 ASN.1 的 BITSTRING。

6.4.4.3.2 语法

BitsetType ::= BITSET # { NamedBitList }

其中：

NamedBitList ::= NamedBit | NamedBitList, NamedBit

NamedBit ::= identifier(number) | identifier(DefinedValue)

- a) 出现在“NamedBitList”中的每个“number”或“DefinedValue”的值应不同,且是位集值中特异位的偏置;
- b) 出现在“NamedBitList”中的每个“identifier”应不同;
- c) 所有元素隐含属于 BOOLEAN1 类型。DefinedValue 仅能为 TRUE(1)或 FALSE(0);
- d) 应按偏置增序声明元素;
- e) 若位集所有元素已声明,则“number”可省略。这宜是正常情况。

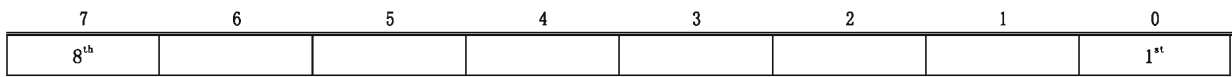
6.4.4.3.3 编码

6.4.4.3.3.1 概述

位集元素应按照声明顺序传送。

6.4.4.3.3.2 BITSET8 编码

BITSET8 编码见图 174。



范围:8 位布尔集。

图 174 BITSET8 编码

示例:

AccessType8 ::= BITSET8

```
{
  system (0),    ——位集的第一位(LSB)
  owner  (1),
  group  (2),
  world  (3)
}
```

等价于:

AccessType8 ::= BITSET8

```
{
  system,        ——位集的第一位(LSB)
  owner,
  group,
  world,
  reserved4,
  reserved5,
  reserved6,
  reserved7      ——位集的第八位或最后一位(MSB)
}
```

一个占据此空间且值为“80”H 即“system”的 UNSIGNED8 是集合唯一成员。

6.4.4.3.3.3 BITSET16 编码

BITSET16 编码见图 175。

8 th							1 st	16 th							9 th
-----------------	--	--	--	--	--	--	-----------------	------------------	--	--	--	--	--	--	-----------------

图 175 BITSET16 编码

示例：
AccessType ::= BITSET16 {system(0), owner(1), group(2), world(3)}
值“0000 0000 0000 0110”B 意为“owner”和“group”是集合成员。

6.4.4.3.3.4 BITSET32 编码

BITSET32 编码见图 176。

8 th							1 st	16 th							9 th
24 th							17 th	32 nd							25 th

图 176 BITSET32 编码

6.4.4.3.3.5 BITSET64 编码

BITSET64 编码见图 177。

8 th							1 st	16 th							9 th
24 th							17 th	32 nd							25 th
40 th							33 th	48 th							41 st
56 th							49 th	64 th							57 th

图 177 BITSET64 编码

6.4.4.4 数组类型的表示

6.4.4.4.1 定义

通过引用某种已有类型而定义的一种结构体；新类型的每一个值是零、一个或多个已有类型值的有序列表。每个值的位置由索引标识。值的数量由一个常数或嵌套结构的一个字段指示。若提供了停止元素，则可省略值的数量。

注：数组是 ASN1 的 SequenceOfType，具有由常数、专用变量或无(停止元素)指示的元素数。

6.4.4.4.2 语法

ArrayType ::= ARRAY [IndexList] OF Type
IndexList ::= Index | IndexList, Index
Index ::= number | DefinedValue | identifier | identifier UnsignedType | UnsignedType | STOP = Value
number、DefinedValue 或 identifier 指定以元素个数计的数组长度(空数组为 0)。其类型应是无符号整数。
若指示了具有已定义标识符的无符号类型，则声明了相应字段。
若标识符命名了一个数组外声明的字段，则该字段应位于相同嵌套级别的嵌入数据结构中，或是位于相同嵌套级别字段的子字段，后者应指示全路径名。
若定义了停止值以关闭开放数组，则值应与数组元素类型相同。

长度可由一个算术表达式给定。

6.4.4.4.3 编码

数组应按索引增序传送。

多维数组应按索引列出顺序传送。

注：数组[行,列]逐行传送。

八位位组数组(无约束内容,例如存储器转储)应按应用存储器的存储器地址(或索引)增序传送。

应传送数组所有元素,即使元素无意义。

示例 1：八位位组存储器转储的传送见图 178。

DumpOctetType ::= ARRAY[octet_count UNSIGNED16] OF WORD8。

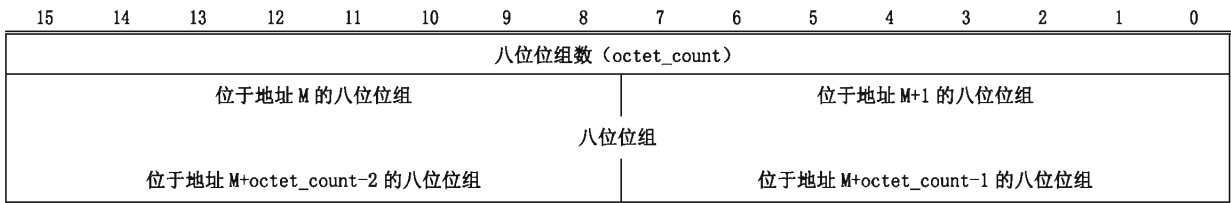


图 178 八位位组存储器转储的传送

示例 2：16 位字同一存储器转储的传送,“word_count”是上例“octet_count”值的一半,见图 179。

DumpWordType ::= ARRAY[word_count UNSIGNED16] OF WORD16。

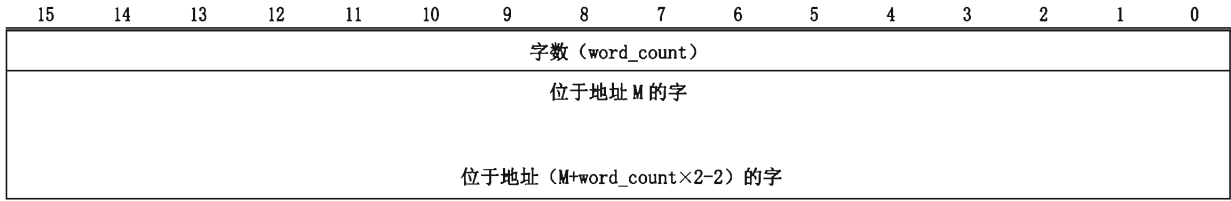


图 179 16 位字同一存储器转储的传送

示例 3：由嵌套数据结构的一个字段给定元素数量(偏置未定)：

DumpOctetType ::= ARRAY[array_count] OF WORD8。

示例 4：由嵌套结构中一个构造值的一个字段给定元素数量：

HeaderType ::= RECORD
 {
 name ARRAY [32] OF CHARACTER8,
 bodysize UNSIGNED16,
 ...U }
FrameType ::= RECORD
 {
 header HeaderType,
 body ARRAY[header.bodysize] OF CHARACTER8
 }

示例 5：字符串,其中停止字符为“space”：

ProfibusString ::= ARRAY[STOP=“20”H] OF CHARACTER8。

6.4.4.5 选择类型的表示

6.4.4.5.1 定义

通过引用一个固定的、不排序的不同类型列表而定义的一种结构体；新类型的每一个值是且仅是一

种元素类型的一个值。

注：此类型对应于 ASN1 的 ChoiceType, 但有一个专用标签。

6.4.4.5.2 语法

OneOfType ::= ONE_OF [identifier | identifierEnumeratedType] {AlternativeTypeList}

其中：

AlternativeTypeList ::= ElementType | ElementTypeList, ElementType

ElementType ::= identifier [tag] Type | [tag] Type

tag ::= UnsignedNumber | DefinedValue | identifier

若一个已命名变量用作标签, 则该变量应位于嵌入元素的结构中。

若标签变量位于与选择相同的嵌套级别上, 则应仅包含变量名。

若变量位于嵌套的另一级别上, 则应包含嵌套的同一级别的路径。

示例 1: 标签是数字(不宜使用, 因为应在别处定义该数字):

```
Commands ::= ONE_OF[choice_var ENUM8]
{
  [3] OpenSequence,
  [2] CloseSequence,
  [5] StandbySequence
}
```

示例 2: 标签是位于选择前 16 位的一个枚举类型:

```
CommandType ::= ENUM16
{
  OPEN      (3),
  CLOSE     (2),
  STANDBY   (5)
}
Commands ::= ONE_OF[choice_var CommandType]
{
  [OPEN]      OpenSequence,
  [CLOSE]     CloseSequence,
  [STANDBY]   StandbySequence
}
```

示例 3: 在嵌入结构中嵌套同级处定义标签:

```
Commands ::= ONE_OF[choice_var]
{
  [OPEN]      OpenSequence,
  [CLOSE]     CloseSequence,
  [STANDBY]   StandbySequence
}
Command_Frame ::= RECORD
{
  choice_var    CommandType,
  ...
  command      Commands,
  ...
}
```

示例 4：在位于嵌套同级处一个字段的子字段中定义标签：

```
Commands ::= ONE_OF[Command_Frame.header.choice_var]
{
  [OPEN]      OpenSequence,
  [CLOSE]     CloseSequence,
  [STANDBY]   StandbySequence
}

Command_Frame ::= RECORD
{
  header      RECORD
  {
    ...addresses ...
    choice_var CommandType
    ...
  }
  commands   Commands
}
```

注：不推荐相对路径(例如:../header)。

6.4.4.5.3 编码

应通过在传送值前先传送指示已做出何种选择的标签字段的方式编码选择类型。
传送值的长度或隐含,或在类型自身中指示。

注：ONE_OF 是只有一个元素的集合(SOME_OF)。

示例：上述 Commands 选择类型的一个特定值将按如下传送,见图 180。

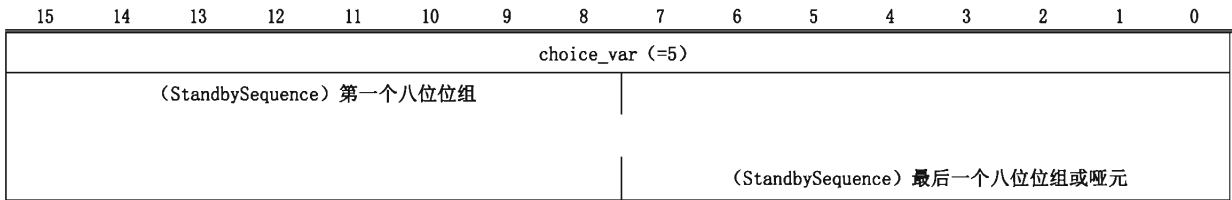


图 180 选择类型的一个特定值的传送的示例

6.4.4.6 集合类型的表示

6.4.4.6.1 定义

通过引用一个固定的、不排序的不同类型(其中部分类型可声明为选项)列表而定义的一种结构体；新类型的每一个值是一个不排序的值列表,每种传送元素类型对应一个值。

注：此类型对应于 ASN1 的 SetType,但有一个显式标签。

6.4.4.6.2 语法

```
SetType ::= SOME_OF{ ElementTypeList}
其中：
ElementTypeList ::= ElementType | ElementTypeList, ElementType
ElementType ::= [tag]NamedType
tag ::= identifier | identifierElementType | ElementType
```

若标签是一个已命名变量,则对应变量应属于同一嵌套级的一个数据结构,或属于同一嵌套级一个字段的子字段,后者应由其全路径名标识变量。

若集合类型成员数量固定,则对每一个从其类型明显了解目的的成员可省略引用名。

若选择符是枚举类型,则用以选择集合元素的枚举常量应置于括号中。

若一个位集变量用作选择符,则该变量应在嵌入数据结构中事先定义,或属于同一嵌套级一个字段的子字段,后者应由其全路径名标识变量。

示例 1: 集合值之前字段中的无符号整数作为标签:

```
MemberType ::= SOME_OF[UNSIGNED8]
{
    OPENSEQ          [3]    Type_OpenSequence,
    CLOSESEQ         [2]    Type_CloseSequence,
    STANDBY          [5]    Type_StandbySequence
}
```

示例 2: 省略引用名:

```
MemberType ::= SOME_OF[UNSIGNED8]
{
    [3]              Type_OpenSequence,
    [2]              Type_CloseSequence,
    [5]              Type_StandbySequence
}
```

示例 3: 枚举类型作为标签(宜实际应用):

```
MemberType      ENUM8
{
    OPENSEQ      (3),
    CLOSESEQ     (2),
    STANDBY      (5)
}
...
CommandsType ::= SOME_OF [MemberType]
{
    [OPENSEQ]      Type_OpenSequence,
    [CLOSESEQ]     Type_CloseSequence,
    [STANDBY]      Type_StandbySequence
}
```

示例 4: 位集用作标签:

```
MembersType      BITSET8
{
    OPENSEQ      (3),
    CLOSESEQ     (2),
    STANDBY      (5)
}
...
CommandsType ::= SOME_OF[members]
{
    [OPENSEQ]      Type_OpenSequence,
    [CLOSESEQ]     Type_CloseSequence,
    [STANDBY]      Type_StandbySequence
}
```

```
    }  
    Commands_Frame ::= RECORD  
    {  
    ...  
    members           MembersType,  
    ...  
    commands          CommandsType,  
    ...  
    }  
}
```

6.4.4.6.3 编码

集合应通过传送每个选中值的方式编码。
若包含标签,则标签应先于每个选中值,特殊标签值“FF”H(所有值)关闭已传送集。
若标签被位集代替,则位集应在集合前传送,且集合不同成员应连续传送。

示例 1: 上述 MemberType 集合的一个特定值传送见图 181。

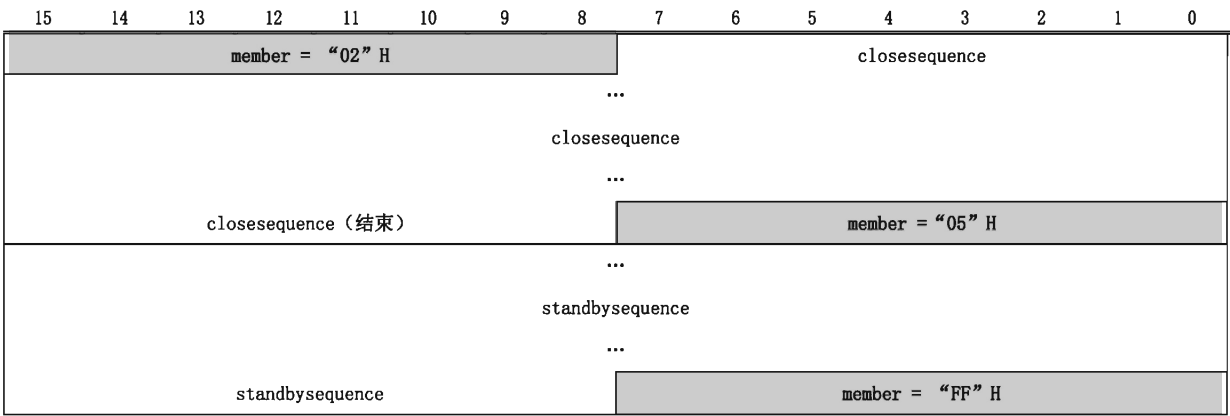


图 181 集合的一个特定值传送示例

示例 2: 若标签被位集代替,则编码见图 182。

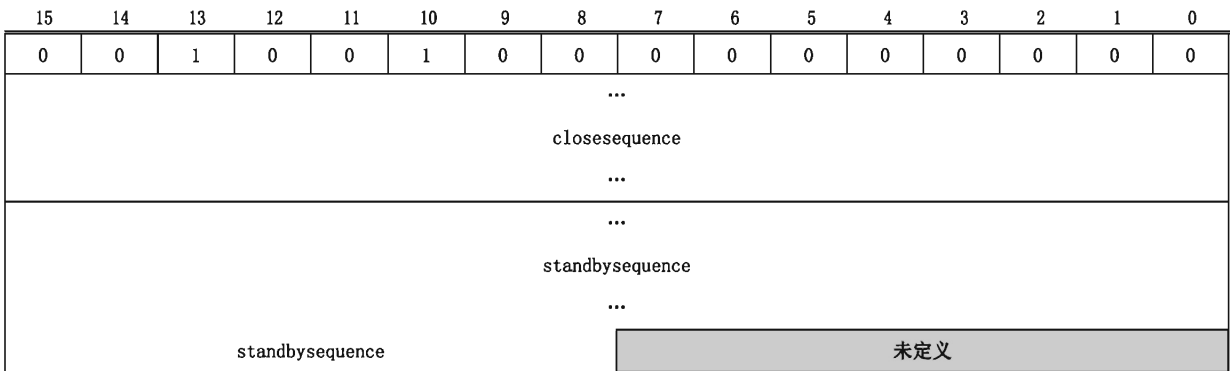


图 182 标签被位集取代的集合的一个特定值传送示例

6.4.5 对齐

在任意类型中,可能需要添加填充位以对齐下一字段到 16 位或 32 位边界(或其他边界)。为此,在类型之后使用限定符 ALIGN。填充位未定义(缺省值 0)。

示例: 以下定义了一个对齐到 32 位边界的字符数组,与“count”的值无关。

AlignedString ::= ARRAY ALIGN 32 [count] OF CHARACTER8。

6.4.6 特殊类型的表示

6.4.6.1 概述

一些结构体有特殊的类型指示符。

6.4.6.2 字符串类型的表示

STRING# 是一个 ARRAY [] OF CHARACTER8(八位字符数组),其中停止元素应为字符“00”H,字符串实际长度从有效字符数推导得出,尽管传送的字符数可能更多。

示例: STRING32 类型的文本字符串以一个 ARRAY[32 STOP=“00”H] OF CHARACTER8 表示,见图 183。



图 183 STRING32 类型的文本字符串的表示

6.4.6.3 TIMEDATE48 类型的表示

6.4.6.3.1 定义

一种结构体,以从世界标准时间 1970 年 1 月 1 日 00:00:00 开始的秒数表示绝对时间。

注: 此类型用于分发实际时间、事件打标、同步等。

6.4.6.3.2 语法

```
TimeDate48 ::= RECORD
{
  seconds      UNSIGNED32,      ——自 1970-01-01 00:00 起的秒数
  ticks        UNSIGNED16      ——不足 1 s 部分的滴答(tick)数(1 滴答 = 1/65 536 s)
}
```

6.4.6.3.3 编码

TimeDate48 的编码见图 184。

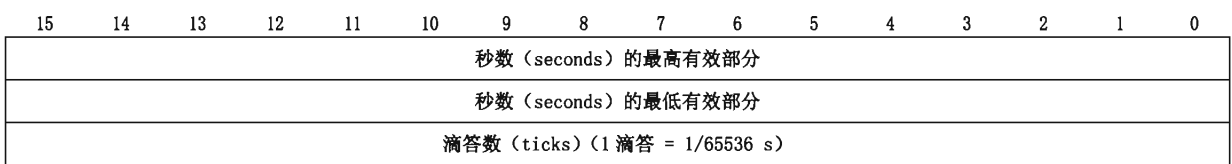


图 184 TimeDate48 的编码

表示的时间粒度是 15.3 μ s(=1/65 536 s)。

范围是 68 年。

小数部分的精度应至少 10 位。

未使用的低序位应置零。

注: TimeDate48 变量约在国际标准世界 2038 年 1 月 19 日 3:14:07 循环计数。宜在软件测试中考虑该循环计数。

6.4.6.4 TIME64 类型的表示

一个结构体,以从 1900 年 1 月 1 日 00:00 开始的秒(s)的数量来表示绝对时间(UTC)。该时间不用闰秒补偿。

Time64 类型的表示见图 185。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
秒数 (seconds) 的最高有效部分															
秒数 (seconds) 的最低有效部分															
滴答数 (ticks) (1 ticks = 1/65536 s)															
吱喳数 (chirp) (1 chirp = 0.232 μs)															

图 185 Time64 类型的表示

注 1: 此时间定义取自为分布式时钟系统定义同步协议的 Internet RFC 1305。与基于 1970 年的 UNIX 时间不同。

注 2: Time64 变量约在 2036 年循环计数。宜在软件测试中考虑该循环计数。因此也将该时间定义看成定义到 2036 年 1 月的剩余时间。

6.4.6.5 ASN.1 boolean8 类型的表示

6.4.6.5.1 定义

一种原始类型,具有两个特定值:TRUE 和 FALSE。

注: 这是 ASN.1 的 BooleanType。

6.4.6.5.2 语法

Boolean8Type ::= BOOLEAN8

6.4.6.5.3 编码

boolean8 类型的变量应编码为 8 位,“0000 0000”B 解释为 FALSE,任意其他值解释为 TRUE,见图 186。

7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	FALSE
0	0	0	0	0	0	0	1	TRUE

图 186 boolean8 类型的编码

7 应用层

7.1 过程数据编排

7.1.1 编排类型

7.1.1.1 概述

过程数据编排(PDM)将过程变量从一个通信存储器拷贝到另一个通信存储器,如图 187 所示。

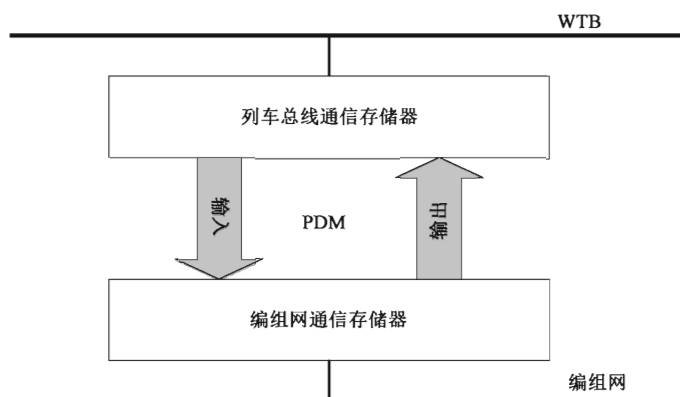


图 187 过程数据编排

定义了两类编排:

——输出编排；

——输入编排。

7.1.1.2 输出编排

输出编排即将变量从一个或多个编组网通信存储器拷贝到 WTB 通信存储器源端口。整个 WTB 端口被写入, 端口中未使用的空间以缺省值填充。输出编排可对过程变量进行处理。

输出编排根据帧类型决定输出帧的长度。

7.1.1.3 输入编排

输入编排即将变量从 WTB 通信存储器拷贝到静态配置的编组网通信存储器。输入编排可对过程变量进行处理。

7.1.2 编排模式

编组可有不同的动态运行模式。PDM 根据编组运行模式提供编排模式。每一种编排模式可有不同配置。

示例：

牵引编组,如机车,可是列车组的牵引主控车、牵引从控车或拖车。在各场景中,输入和输出不同的数据。

若 PDM 未使用专属模式, 缺省模式宜总是存在。

若编组运行模式改变,PDM 可重新配置以接收新的编排模式。

7.1.3 PDM 数据通道

由于各网关提供商实现方式可能不同,本条仅为资料性目的而不是规范性要求。

PDM 编排从源到目的的过程数据。目的总是通信存储器。源是通信存储器、缺省值缓存或未定义值缓存, PDM 数据通道见图 188。

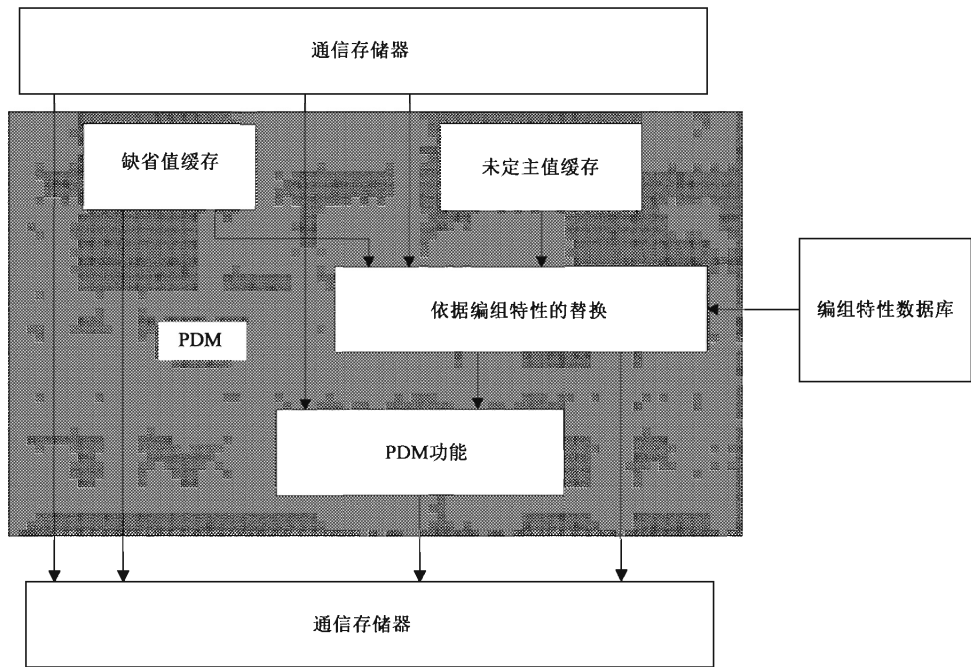


图 188 PDM 数据通道

可配置下列数据通道：

- 从通信存储器直接或通过 PDM 功能到通信存储器。这是 PDM 的基本任务。过程变量在被写入目的通信存储器之前,可由功能处理。
- 当值宜写入端口但无可传送该值的过程变量时,使用从缺省值缓存到通信存储器。缺省值不会替换无效或过时(未刷新)的过程变量。
- 编组特性相关过程数据从通信存储器直接或通过 PDM 功能到通信存储器。过程变量可与静态/动态编组特性结合。仅当编组特性存在时,可完成编排。否则,PDM 以合适数据类型的未定义值或应用定义缺省值替换过程变量。未定义值以校验变量置 11B 或者变量值置全 1 来表示。

对于编组相关过程变量,宜考虑三种情况：

- 一个变量带一个校验变量:PDM 可通过设置校验变量为 11B 的方式替换未定义值。
- 多个变量带一个校验变量:若所有变量都不支持,PDM 可通过设置校验变量为 11B 的方式替换未定义值。
- 多个变量带一个校验变量:若仅部分变量不支持,PDM 宜替换应用定义缺省值。

过程变量在写入目的通信存储器之前可由功能处理。过程变量若被未定义值替换,则作为功能参数可被忽略。

7.1.4 PDM 操作

PDM 由可配置定时器或事件(如 WTB 数据接收)激活。激活后,PDM 拷贝为激活编排类型配置的所有变量。

拷贝过程有以下四个步骤,见图 189：

- a) 所有变量按数据集依次读取；
- b) 对于已配置刷新监视的每个变量检查刷新。过旧的变量无效化(描述见后)；
- c) 对变量应用已配置的所有功能(见 7.1.5 PDM 功能)；
- d) PDM 拷贝变量、功能结果和缺省值到目的端口和通信存储器。

所有数据集的变量输入			
读出数据集的所有变量			
若是输入编排且帧类型字段正确或者不是输入编排			
则			否则 设置数据集的所有变量无效
所有数据集的变量			
若检查刷新，发现数据过旧			
则		否则	
设置变量无效			
若是输入编排且任意帧类型字段不正确			
则			否则
设置 WTB 通信存储器所有变量无效			
所有的 PDM 功能			
执行 PDM 功能			
所有数据集变量跳转			
写数据集所有变量			
(PD 变量、功能结果、缺省值)			

图 189 PDM 操作

变量或功能结果(见 7.1.5)通过图 190 所示算法无效化。

若变量或者功能结果有校验变量	
则	否则
设置校验为“00”B	设置变量值为全“0”

图 190 PDM 无效化变量或功能结果

若变量或功能结果有校验变量,则仅校验变量置为 00B。变量值不置为全 0,因为变量不能真的无效。

若变量或功能结果没有校验变量,则其值置全 0。这符合本部分规定。

注：由于改写值而得到全 0 或全 1 的过程变量可能得到一个合法值,所以在可能产生问题的地方使用同一数据集的校验变量作为有效性指示(见 6.2.2.2.3)。

7.1.5 PDM 功能

7.1.5.1 概述

除了变量拷贝以外,PDM 支持由功能处理过程变量。所有编排类型都支持功能处理。

示例：假设应用需要知道列车所有门是否已关闭。由于列车可能有 1~20 个具有门的编组,应用应能处理大范围的输入数据。使用读所有门状态并发布一个表示“所有门已关闭”的变量的 PDM 功能使得应用编程更加简单。

PDM 提供的功能通常具有式(4)所示形式：

$$y = f(x_1, x_2, \cdots, x_n) \qquad \cdots \cdots \cdots (4)$$

式中：

$x_i (i=1\sim n)$ ——输入参数；

y ——功能结果。

可有任意数量(大于 0)的参数和一个结果。功能参数可来自不同的端口和通信存储器。参数和结果由 PV_Name 描述。

PDM 提供以下标准处理功能：

——逻辑功能：

AND, AND_IGNORE_INVALID

OR, OR_IGNORE_INVALID

XOR, XOR_IGNORE_INVALID

——数值功能：

MIN, MIN_IGNORE_INVALID

MAX, MAX_IGNORE_INVALID

SUM, SUM_IGNORE_INVALID

7.1.5.2 功能处理

功能处理操作见图 191。

功能所有参数需检查有效性。过旧的变量已由拷贝进程第二步无效化。参数可有校验变量,也可没有。若功能具有多于一个参数,则可混合使用带校验变量的参数和不带校验变量的参数。

若参数无效或未定义,则参数可被忽略(功能 XXX_IGNORE_INVALID)。若无效或未定义的参数未被忽略,则设置功能结果无效。

仅处理有效参数。若所有参数无效或未定义,则功能结果置为无效。

若必要,参数类型转换成适合处理的类型。处理类型是可配置的。

若在功能评估期间发生错误,则结果置为无效。

在处理完所有参数之后,计算结果转换成 PV_Name 描述的期望功能结果。

功能结果可有校验变量,也可没有,这与参数无关。

功能的所有参数			
若变量是有效的			
则 将参数类型转为计算类型		否则 若 IGNORE_INVALID	
将功能应用到参数并计算功能结果		则 忽略参数	否则 设置功能结果无效 返回功能结果
若所有变量是无效的			
则 设置功能结果无效		否则	
返回功能结果			
将计算类型转为结果类型			
若函数结果有校验变量			
则 设置校验为“01” B			否则
返回功能结果			

图 191 PDM 操作

变量的有效性由图 192 所示算法检查。

若变量有校验			
则		否则	
若校验为“10”B 或者“01”B		若变量值是全“0”或全“1”	
则	否则	则	否则
变量是有效的	变量是无效的	变量是无效的	变量是有效的

图 192 PDM 有效性检查

7.1.5.3 逻辑功能:AND、OR 和 XOR

对于这些功能,参数类型是 BOOLEAN、ANTIVALENT 或 BITSET。若参数类型是 BITSET,应同时指定位组中位的位置。在一个功能调用中,参数类型可混合使用。

功能结果值的类型是 BOOLEAN 或 ANTIVALENT。此外,可指定变量是直接使用还是在使用前取反。

7.1.5.4 数值功能:MIN、MAX 和 SUM

对于这些功能,参数类型是最大长度 32 位的 INTEGER、UNSIGNED、REAL 或 FRACTIONAL。在一个功能调用中,不同长度的 INTEGER 和 UNSIGNED 类型可混合使用。处理后,结果类型转换(通过 cast)成目的变量的类型。没有范围检查,注意溢出。

7.2 WTB 线路故障位置检测

7.2.1 概述

诊断计算机可监视网关节点状态字的 WTB 线路扰动位。若诊断计算机检测到线路上稳定扰动,则可请求 TCN 网络管理(TNM)的线路故障位置检测(Line Fault Location Detection,LFLD)。TNM 与 WTB 链路层链路管理协作检测故障位置。在 LFLD 进程完成后,诊断计算机可读取 TNM 的结果。

扰动线路合适端接的线路故障可能导致信号反射,从而影响该线路上多个节点的通信。LFLD 进程在中间节点处断开扰动线路的中继。这将产生可检查的 WTB 端接段。

7.2.2 体系结构

LFLD 体系结构见图 193。

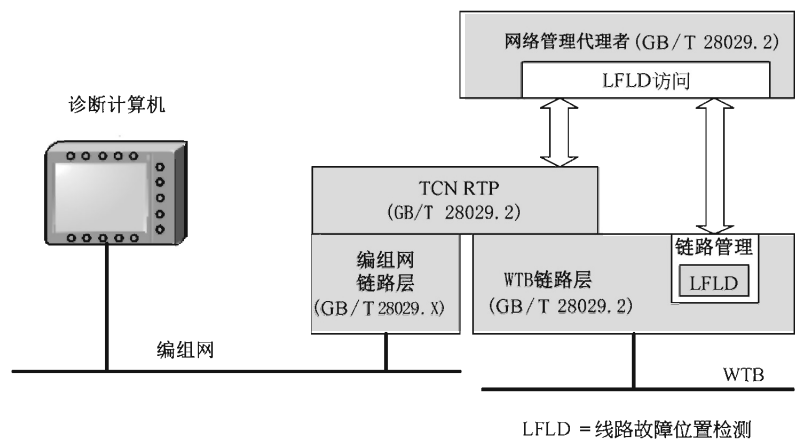


图 193 LFLD 体系结构

WTB 线路故障位置检测(LFLD)是在链路管理中实现的 WTB 链路层功能。在 LFLD 进程期间，链路管理使用 TNM 与其他 WTB 节点通信。

TCN 网络管理提供新的 WTB 链路服务——LFLD(见 8.4.3.6),并支持诊断计算机和 WTB 链路管理所需的子命令。

诊断计算机可经 MVB 通过 TCN 消息数据访问 TNM 服务 LFLD,以:

- 启动 WTB 末端节点上的 LFLD;
- 获取 LFLD 结果;
- 取消 LFLD。

WTB 链路管理者经 WTB 通过 TCN 消息数据访问 TNM 服务 LFLD,以:

- 启动和停止另一 WTB 末端节点上的 LFLD 进程;
- 启动和停止 WTB 中间节点上的 LFLD 进程。

7.2.3 协议概述

第一步:

诊断计算机请求一个末端节点 LFLD,并使之成为测试节点(Testing Node,TN),见图 194。测试节点的 WTB 链路管理启动检测进程。

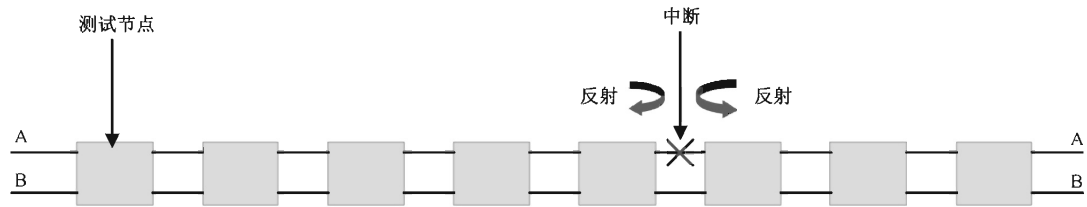


图 194 诊断计算机请求一个末端节点 LFLD 并使之成为测试节点

第二步:

测试节点请求下一节点(即分段节点,SegmentingNode,SN)断开线路开关,见图 195。若测试节点在 T_2 时间内从分段节点接收到的帧中未发现线路故障,则认为该线路段工作正常。然后,测试节点请求分段节点闭合线路开关,并继续测试分段节点后的下一个节点。若测试节点未从分段节点发出的帧中发现线路故障,则此进程持续。

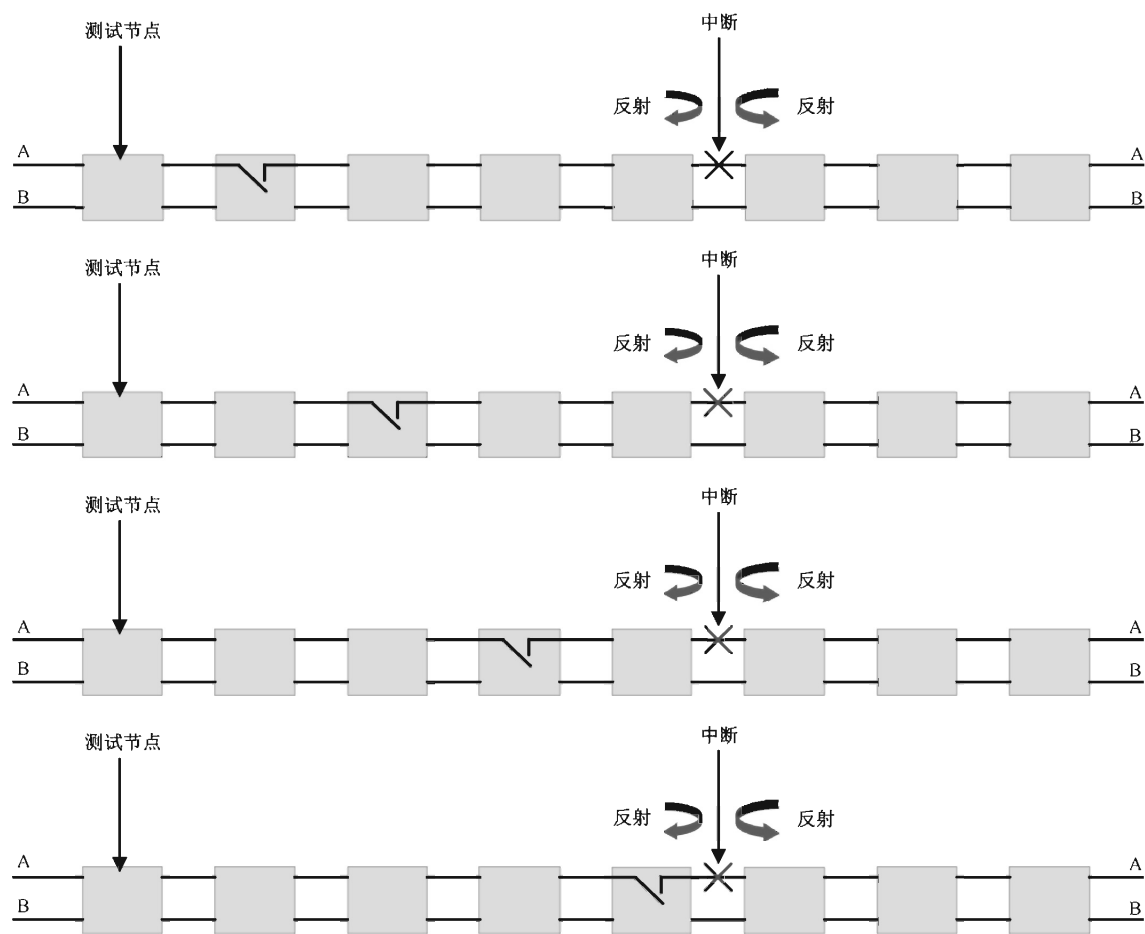


图 195 测试节点请求分段节点断开线路开关及分段节点下移

第三步：
此时，测试节点不再接收到来自分段节点的无线路故障的帧，见图 196。这表明在最新测试段中存在线路故障。

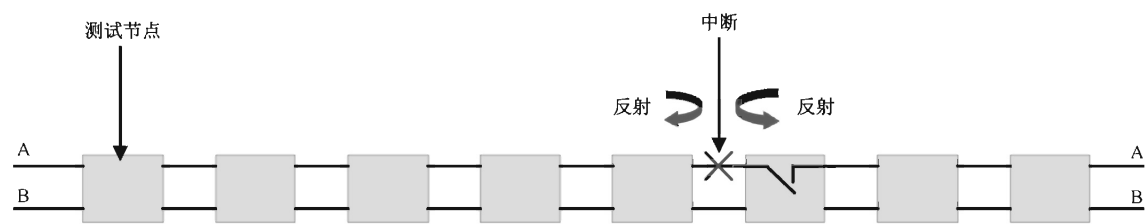


图 196 测试节点不再接收到来自分段节点的无线路故障的帧

7.2.4 LFLD 顺序

图 197 显示了 LFLD 进程步骤：

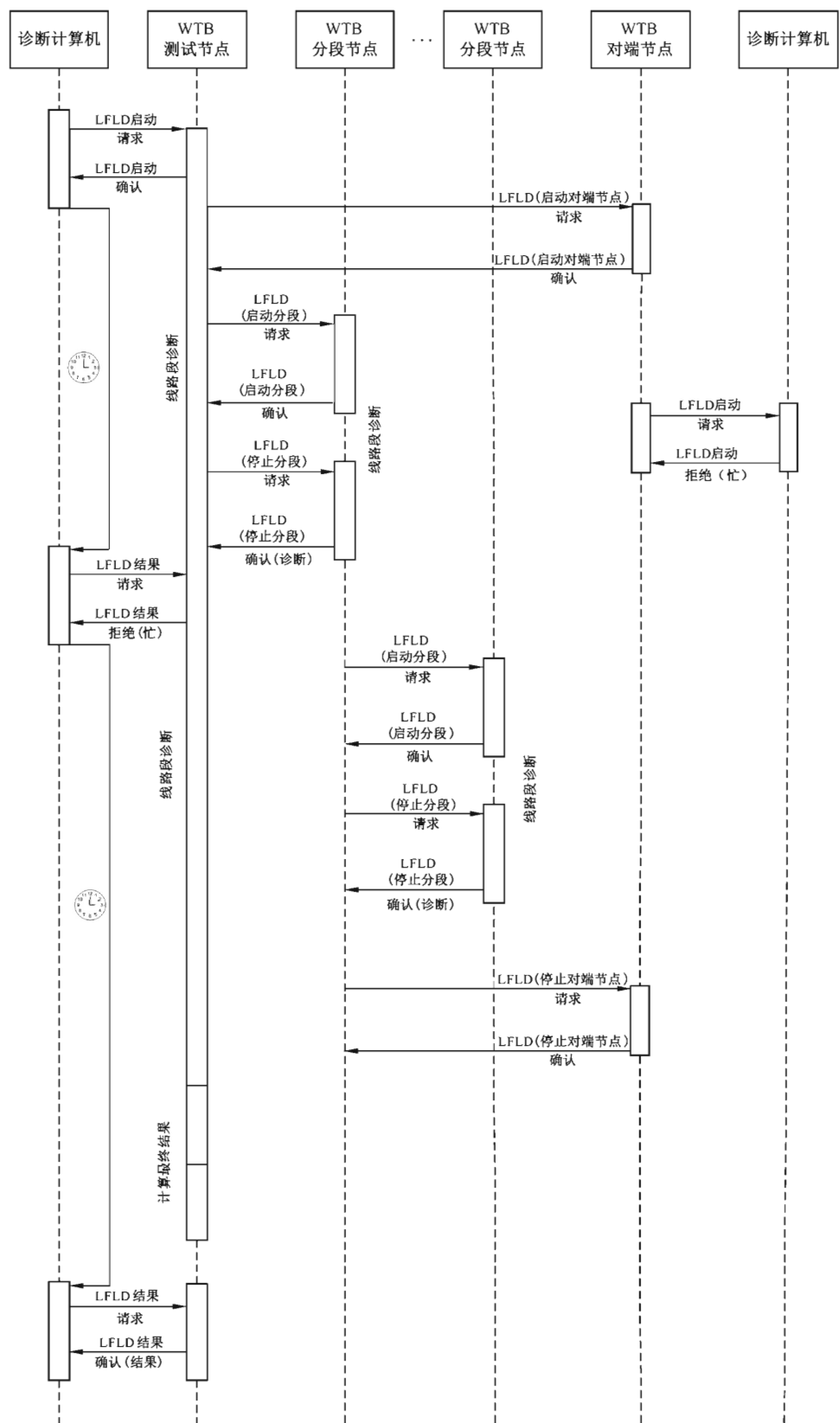


图 197 LFLD 顺序

a) 诊断计算机通过 TNM 消息启动 LFLD 进程。

- b) 测试节点通过 TNM 消息通知对端节点 (opposite end node, OE) 已启动 LFLD 进程。这应阻止对端节点也启动 LFLD 进程。
- c) 当测试节点接收到来自对端节点的应答消息时,选择第一个分段节点。
- d) 测试节点请求分段节点分段总线 (断开中断继电器),并启动线路诊断。
- e) 当测试节点接收到来自分段节点的应答消息时,监视从测试节点到分段节点之间总线段是否可接收来自分段节点或前一节点的无线路扰动的过程数据响应帧。
- f) 分段节点监视从分段节点到对端节点之间的总线段是否可接收来自对端节点的无线路扰动的过程数据响应帧。
- g) 在最长可能特征周期 (128×25 ms) 之后,测试节点请求分段节点停止分段总线 (闭合中断继电器) 并报告分段节点是否可无线路扰动地接收对端节点。
- h) 测试节点选择下一分段节点,并重复步骤 d)~g),直到检测出故障位置或到达对端节点。
- i) 当到达对端节点时,测试节点通知对端节点 LFLD 进程已完成。
- j) 诊断计算机可从测试节点获取 LFLD 结果。若 LFLD 进程仍在继续,则测试节点拒绝请求并报错误消息。诊断计算机应轮询测试节点直到结果可用。
- k) 若在 LFLD 进程期间诊断计算机请求对端节点 LFLD,则对端节点拒绝请求并报错误消息。

7.2.5 末端节点状态机 (测试节点)

末端节点状态机见图 198。

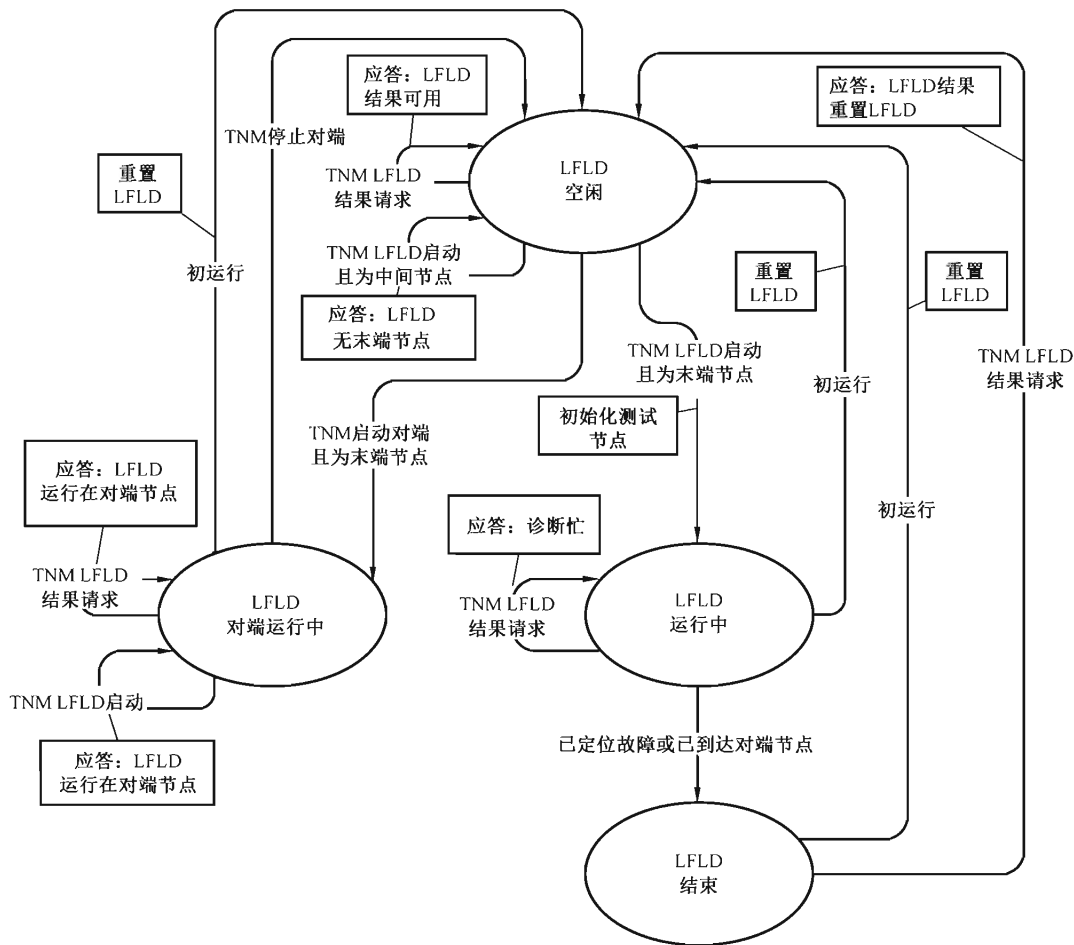


图 198 末端节点状态机

7.2.6 中间节点状态机(分段节点)

分段节点的实现是无状态的。

7.2.7 受扰线路选择

受扰线路 A 或 B 取决于每个节点的朝向。

测试节点获取其本地受扰线路。测试节点根据自身相对 WTB 主设备的朝向推断 WTB 主设备的受扰线路,并将其作为标准受扰线路。当测试节点启动和停止分段节点时,测试节点请求断开和闭合标准受扰线路。分段节点自身根据标准受扰线路和其相对于主设备的朝向推断出哪一个本地线路中继应断开或闭合。

7.2.8 位置检测

测试节点初始化测试节点及其直接邻节点组成的节点对。LFLD 进程按节点顺序执行。

示例: LFLD 进程到达分段节点 63,故障位于节点 63 和节点 1 之间,见图 199。

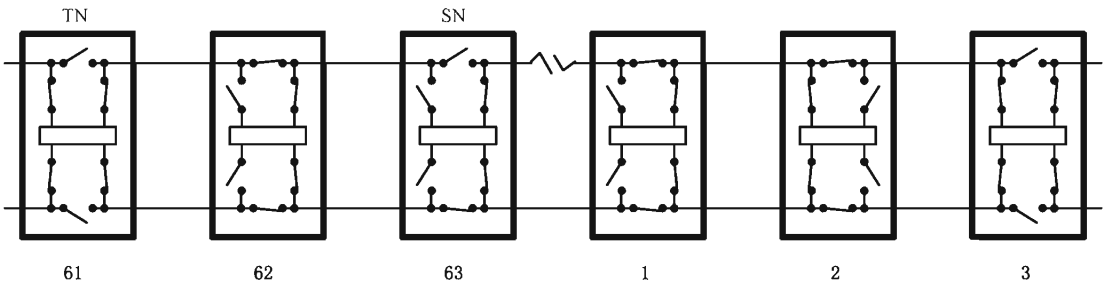


图 199 LFLD 进程,分段节点位于节点 63

在图 199 中,测试节点不能到达分段节点 63,因为相连的中继在远端闭合。测试节点仅能从前一节点收到正常的响应。所以测试节点不能区分节点 62 与 63 之间的故障和节点 63 与 1 之间的故障。

分段节点也监视中断继电器另一侧的 WTB 段,即此例中节点 63~3 之间的总线段。当测试节点请求闭合中断继电器时,分段节点向测试节点报告该段的线路质量。此例中,节点 63 检测到该段存在故障,且测试节点可假定从节点 61~63 的总线段无故障,即使测试节点不能获取节点 63 的响应。

测试节点继续下一分段节点(节点 1),如图 200 所示。

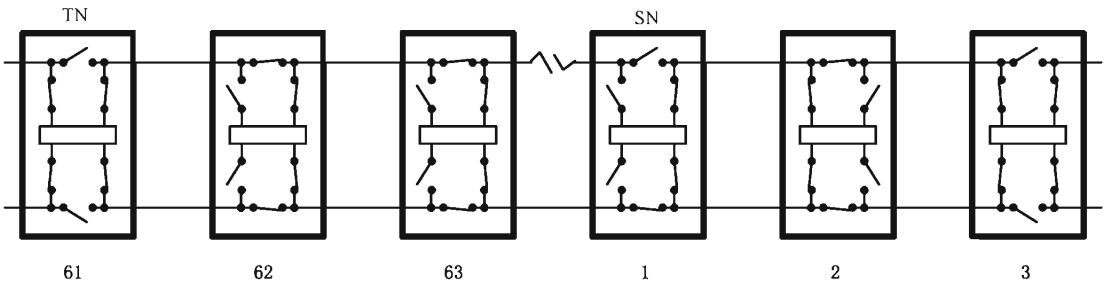


图 200 LFLD 进程,分段节点位于节点 1

分段节点(此时为节点 1)监视从节点 1~3 的总线段,检测其无故障并报告给测试节点。测试节点

可推断出故障应位于此分段节点之前,即在节点 63 和 1 之间。

当从分段节点获取报告时,测试节点决定:

- a) 若分段节点报告其无扰接收对端节点消息时,则停止 LFLD 进程,且节点对由分段节点和分段节点前一节点组成;
- b) 若测试节点无扰接收分段节点或分段节点前一节点消息时,则设置节点对为已接收节点及其后一节点。

在每一个分段节点处做此决定。

当测试节点已停止或到达对端节点时,检查是否应解决图 201 中的情况。

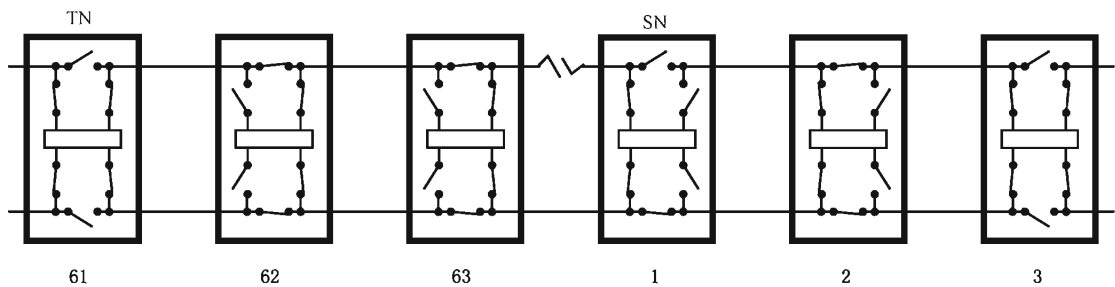


图 201 LFLD 进程,分段节点位于节点 1,在方向 1 连接

测试节点可能既不能到达节点 63,也不能到达节点 1,且分段节点不能到达节点 3。但是测试节点可推断出故障一定位于节点 63 和节点 1 之间,因为若不是这样,测试节点应可检测到从节点 61 到节点 1 之间的总线段无故障。

测试节点决定:

- 若节点对是(63,1)、测试节点位于底地址且 WTB 主设备连接继电器在方向 2 上,则更正节点对为(1,2);
- 若节点对是(1,2)、测试节点位于顶地址且 WTB 主设备连接继电器在方向 1 上,则更正节点对为(63,1)。

LFLD 进程的最终结果是指明故障位置的节点对。结果存储在列车网络管理中,并为诊断计算机所获取。

8 列车网络管理

8.1 总则

8.1.1 本章内容

列车网络管理制定了一系列服务,用于辅助列车通信网的试运行、测试、运行和维护,诸如:

- a) 站的标识和控制;
- b) 列车总线和编组网的链路层管理;
- c) 路由和拓扑的发布;
- d) 变量的远程读取和强制;
- e) 下载和上传。

所有服务可由管理者进程远程请求。

这些服务由每个站通过代理者进程执行。本章规定了本地管理对象以及与代理者之间的接口。

本章规定了在管理者与代理者之间交换用于网络管理的管理消息。

此外,本章还规定了用户自定义服务和代理者之间的接口方法。

注 1: 列车网络管理并不规定应用专属信息(如 UIC 557 中的列车诊断或 UIC 556 中的编组标识)。但是,用户应用可在常规运行期间使用网络管理服务。

注 2: 列车网络管理不考虑与应用直接相关的管理服务。尤其是,适用于设备(如空调)实际作用的虚拟设备描述不属于本章范畴。

8.1.2 本章结构

本章结构如下:

——8.1 总则;

——8.2 管理者、代理者及其接口;

——8.3 管理对象;

对每一对象,规定了:

- 对象描述;
- 访问;
- 对象提供的服务。

——8.4 服务和管理消息;

对每项服务,规定了:

- 服务描述;
- 呼叫消息和参数;
- 应答消息和参数。

——8.5 接口过程。

对每个接口,规定了本地访问和控制代理者的过程。

8.2 管理者、代理者及其接口

8.2.1 管理者和代理者

网络管理服务应在每个站中由代理者提供。

代理者应由其驻留站的站标识符(Station_Id)标识。

网络管理服务应由管理者请求。

8.2.2 管理消息协议

为实现网络管理目的,管理者和代理者使用 TCN 的消息服务,通过交换管理消息在网络上通信,如图 202 所示。

管理者应作为呼叫者,代理者应作为应答者。

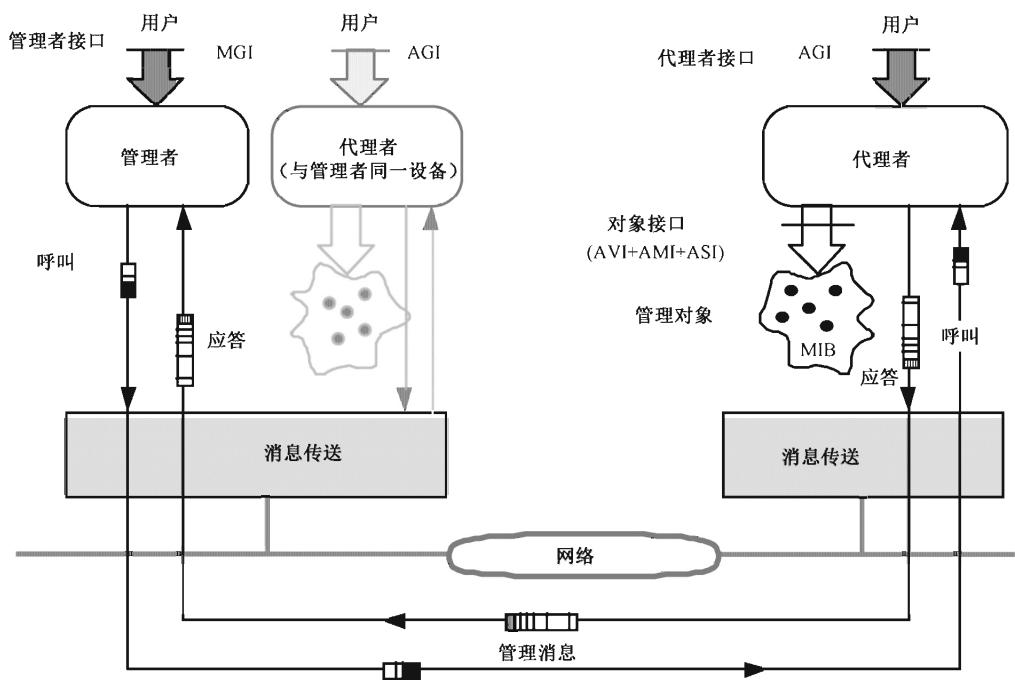


图 202 管理消息

管理者分两步访问远程对象：

- a) 管理者发出管理呼叫消息(Call_Message)；
- b) 代理者解码此消息,访问实际对象并回送携带服务结果的管理应答消息(Reply_Message)。

代理者应可通过系统地址和用户地址访问,其 Function_Id=253。
管理者应可通过系统地址和用户地址访问,其 Function_Id=254。
管理消息格式应如下列各条定义。

8.2.3 接口

8.2.3.1 对象接口

通信对象与网络通信相关,非通信对象与站的其他属性相关。

示例 1: 总线管理器配置是一个通信对象。
示例 2: 存储区或存储域、调度器、时钟、站配置和描述符都是非通信对象。

代理者应通过本部分定义的通用访问接口访问通信对象,尤其是通过：

- a) AVI(应用变量接口)用于变量；
- b) AMI(应用消息接口)用于消息；
- c) ASI(应用监视接口)用于用户进程不能访问的对象。

ASI 提供了访问位于不同通信层中对象的接口。代理者通过对象驻留层的层管理接口访问这些对象(如图 203)。该接口定义包含在规定相应层的章节——本部分和 GB/T 28029.9 的实时协议中。

注：能有效访问每层对象的实体称为层管理实体或 LME。
代理者应也能访问非通信对象。此部分接口未作规定。

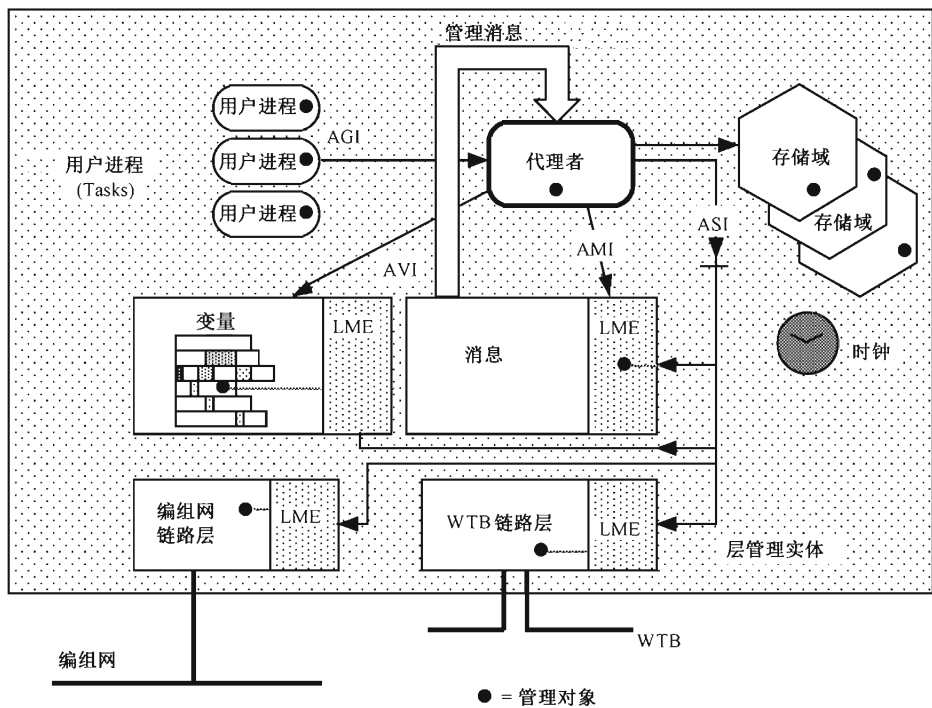


图 203 一个(网关)站中的代理者接口

8.2.3.2 管理者接口(MGI)

管理者应提供一组访问远程对象的过程,这些过程形成管理者接口(MGI),如图 191 所示。

对于该管理者的每一个服务,管理者接口(MGI)应将管理者发送的管理呼叫消息和代理者应答的管理应答消息关联起来。

MGI 过程没有单独规定,但定义了提供所有服务的两个通用过程。

管理消息的格式也应用于管理者服务过程的参数。

8.2.3.3 代理者接口(AGI)

一些管理对象需要访问代理者。当任务进行同步或报告状态改变时,需查询代理者的状态。

为此,代理者应提供一个称作代理者接口(AGI)的接口。代理者接口允许本地用户直接读取或修改某些管理对象,如 8.5.3 规定。

注:没有与代理者接口相关联的网络消息。

8.3 管理对象

8.3.1 对象属性

每个管理对象应含有四种属性:

- a) 对象标识符,标识对象的全文本字符串,最多 31 字符;
- b) 状态(只读),指示对象的状态;
- c) 控制(只写),作用于对象;
- d) 参数(读/写),包含对象的可修改部分,只要其未包含在状态或控制中。

对象的属性应由本章定义的服务或用户定义的服务处理。

每一个为对象定义的服务都应以文本信息描述,该文本信息称作服务描述符(Service_Descriptor)。

标准服务的文本信息可包含本部分引用。用户定义服务的文本信息未作规定。

8.3.2 站对象

8.3.2.1 站状态对象

每个站应实现一个站状态(Station_Status)对象,以指示站的通用特性和一些动态参数。
当站只连接一个编组网且没有连接列车总线时,站状态应为编组网设备状态的拷贝。否则,不同链路层状态应进行“或”运算以形成站状态。
站状态应具有下列格式,如图 204 所示。

Station_Status ::= RECORD
{
 station_capabilities BITSET4 ——站的基本能力
 {
 sp (3), ——特殊设备
 ba (2), ——MVB 总线管理器
 gw (1), ——网关或列车总线节点
 md (0) ——消息能力
 },
 class_specific WORD4, ——总为 0
 common_flags BITSET8
 {
 lat (7), ——未使用,=0
 lnd (6), ——链路受扰
 ssd (5), ——受控进程中系统干扰故障(如掉电)
 sdd (4), ——设备干扰:设备故障(如校验和错)
 scd (3), ——通信干扰:当该设备任意通信存储器的任意端口的宿时间监视被触发
 时置位,当所有已配置端口正常工作时复位
 frc (2), ——强制站:该设备一个端口被强制
 snr (1), ——站未就绪:如站未初始化
 ser (0) ——管理者保留的站
 }
}

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
特殊 设备 (sp)	总线管 理器 (ba)	网关 (gw)	消息 (md)					信任 线A (lat)	链路 受扰 (lnd)	系统 干扰 (ssd)	设备 干扰 (sdd)	通信 干扰 (scd)	强制 站 (frc)	站未 就绪 (snr)	管理者 保留 (ser)
站能力 (station_capabilities)				特殊类别 (class_specific)				公共标志 (common_flags)							

图 204 站状态

8.3.2.2 站控制对象

每个站应实现具有两个相关服务的站控制(Station_Control)对象:

- a) 启动站并指定配置参数,如站名;
 - b) 重启站,即停止所有任务、清除所有表格、关闭所有会话并仅重启信使和代理者。
- 若仅启动或停止应用任务,则任务对象应允许软启动。

8.3.2.3 站信息对象

每个站应实现一个只读的站信息(Station Inventory)对象,以描述站的静态特性,包括:

- a) 站标识:
 - 供应商;
 - 序列号;
 - 站标识符和站名。
- b) 站能力:
 - 软件版本;
 - 支持的服务;
 - 链路层列表。

链路层列表由一个 16 位位组的链路集(Link_Set)表示,链路集的每一位对应一个链路层。每个链路层关联一个通信存储器。链路集定义如下:

Link_Set ::= BITSET16

{		
link_layer0	(0),	—— 最先传送位
link_layer1	(1),	
link_layer2	(2),	
link_layer3	(3),	
link_layer4	(4),	
link_layer5	(5),	
link_layer6	(6),	
link_layer7	(7),	
link_layer8	(8),	
link_layer9	(9),	
link_layer10	(10),	
link_layer11	(11),	
link_layer12	(12),	
link_layer13	(13),	
link_layer14	(14),	
link_layer15	(15)	
}		

8.3.2.4 站保留对象

每个站应实现一个站保留(Station Reservation)对象,以允许管理者保留此站给其独自使用。

保留对象应是一个带超时的信号量,确保对可修改对象的排他访问。

信号量的状态应在站状态“ser”位上反映(“1”=保留,“0”=不保留)。

对保留对象的服务是“保留”和“释放”。

管理者应在被允许修改其管理对象之前保留站。

拟保留站的管理者应向代理者发送保留请求。

接受保留的代理者应设置保留对象,并登记管理者的标识。

保留对象的拥有者应以代理者接收到的 24 位呼叫者地址标识。

此外,保留呼叫提供了一个 32 位字,以为特定应用更准确地标识管理者(或网络管理者)。

若管理者保留多个站,则其应按应用地址升序保留站,以避免与另一管理者发生死锁。

若站没有被保留(SER 位未置位)或者请求由另一未登记的管理者发布,则代理者应拒绝任何修改服务呼叫。

若管理者退出服务但没有释放站,则信号量超时保证站不被阻塞。

每当代理者从其已登记的管理者接收到管理呼叫消息时,信号量超时应复位。

在下列情况下,保留信号量应清除:

- a) 列车总线初运行;
- b) 站复位;
- c) “覆盖”保留消息。

注 1: 非修改服务(只读)不涉及保留。它们可由任何管理者或其他功能呼叫。仅已登记的管理者可发布修改请求(写)。

注 2: 管理者和代理者的标识可能会受到初运行的影响。

注 3: 当站因某些原因阻塞在保留状态时,“覆盖”消息是为保留站而采取的最后手段。宜谨慎使用。

注 4: 保留不提供访问的安全性,而只能防止某些事故。如需安全访问,宜在工作站级实现口令。

8.3.3 WTB 链路对象

8.3.3.1 WTB 状态对象

连接到列车总线的每个节点应实现一个只读的 WTB 状态(WTB_Status)对象。该对象在本部分规定。

WTB 状态标识软硬件版本、定义动态和静态参数、显示 WTB 链路层的故障及统计信息并包含与该 WTB 连接相对应的节点状态字。

代理者通过 WTB 链路层监视接口 ls_t_xxx 访问 WTB 状态对象。

WTB 状态对象包括以下数据结构,由本地应用提供并在本部分定义:

- a) Node_Descriptor,定义过程数据帧的结构(见 5.5.2.2);
- b) Node_Report,BITSET8 类型,指示扰动(见 5.5.2.3);
- c) User_Report,WORD8 类型,由应用提供(见 5.5.2.4);
- d) Node_Key,包含节点类型和节点版本的记录(见 5.5.2.2);
- e) LS_T_State,ENUM8 类型,指示节点所处状态(见 5.6.4.2.3);
- f) Node_Strength,ENUM8 类型,表示节点强度(见 5.6.4.16.3)。

8.3.3.2 WTB 拓扑对象

连接到列车总线的每个节点应实现一个只读的 WTB 拓扑(WTB_Topography)对象。该对象在本部分规定。

代理者通过链路管理接口 ls_t_xxx 访问此对象。

注 1: WTB 拓扑描述当前列车总线组态、列出现有节点及其描述符并报告主设备状态。

注 2: 拓扑对象宜被在列车总线上发送消息的所有应用程序访问,因为当组成变化时拓扑确保寻址的一致性,如 6.3 描述。

注 3: 拓扑可从任何站获得。但是,只有具有列车总线连接的站可提供拓扑的当前版本。

注 4: 拓扑对象包含一个强制部分和用户提供的初运行数据。

8.3.4 变量对象

8.3.4.1 端口配置对象

每个具有过程数据能力的站应实现只读的端口配置(Port_Configuration)对象。端口配置对象指示端口数量及其地址。

端口可是固定长度(编组网)或可变长度(WTB)。

端口长度由其 F_code 定义:

```
F_Code ::= ENUM4
{
    PD16      (0),      ——16 位端口
    PD32      (1),      ——32 位端口
    PD64      (2),      ——64 位端口
    PD128     (3),      ——128 位端口
    PD256     (4),      ——256 位端口
    PD512     (5),      ——512 位端口
    PD1024    (6),      ——1024 位端口
    PDW       (7)       ——可变长度端口(WTB)
}
```

端口在总线上由其 12 位端口地址定义。在 WTB 上,仅使用低 6 位地址。F_code 和端口地址组合成一个称作 FcodeAdr 的 16 位结构:

```
FcodeAdr ::= RECORD
{
    f_code      F_Code,      ——给出端口长度的 F_code
    address     UNSIGNED12 ——逻辑地址
}
```

每个端口具有以下属性:

```
Port_Object ::= RECORD
{
    f_code      F_Code,      ——端口 F_code
    port_address UNSIGNED12, ——端口地址
    port_config BITSET4
{
    src      [0],      ——端口使能为发布者(源)
    snk      [1],      ——端口使能为用户(宿)
    twc      [2],      ——端口传送带校验和的帧
    frc      [3]       ——强制成强制值的端口
},
    port_size  UNSIGNED8 ——以八位位组表示的最大端口长度,仅用于 PDW 端口(WTB 端口)
}
```


8.3.4.2 变量对象

8.3.4.2.1 对象

每个具有过程数据能力的站应实现变量(Variables)对象,以指示值、校验变量及变量刷新。

注:端口刷新的特征周期受控于编组网上的总线配置对象或列车总线上节点的介质访问控制(MAC)。端口、变量和总线地址间的关联由应用决定。

8.3.4.2.2 访问

如同其他应用进程,代理者通过应用变量接口(AVI)访问变量。该接口允许在通信存储器中以单个或群集方式读取变量或强制变量。

8.3.4.2.3 服务

所有变量远程服务以群集(多个变量)方式执行,单个变量访问属特例。

下列服务远程访问变量:

- a) 读(read):获取变量值、其校验变量及变量刷新。
- b) 强制(force):强制变量为指定值。除非变量取消强制,否则该值不能被应用(源)或总线(宿)改写。同一数据集的其他变量可被强制或取消强制:
 - 因为强制变量不能被其生产者改写,代理者应通过 AGI 请求允许对其修改;
 - 若变量位于源端口,则总线在下一周期开始向所有宿端口广播其值;
 - 若变量位于宿端口,其强制值仅对其自身站可见。当变量被强制时,禁止总线对变量的写入,但本地刷新仍由总线控制;
 - 若站中任意变量被强制,则代理者置位其站状态和连到被强制通信存储器的编组网设备状态中的“frc”位。
- c) 取消强制(unforce):释放强制变量,使得应用可再次修改其值。
- d) 取消所有强制(unforce_all):释放通信存储器中所有变量。
- e) 读绑定(read_bindings):指示本地变量绑定的端口。
- f) 写绑定(write_bindings):将本地变量绑定到端口。

注1:用户进程可通过请求其本地代理者对其一个通信存储器执行读变量或强制变量的方式访问变量。但与直接访问相比,花费更多的时间。

注2:需区分强制与非强制变量的消费者应用可置强制变量的校验变量为“10”B。

注3:每个通信存储器宜只调用取消全部强制(Unforce_all)一次,直至站状态中的 FRC 位被复位。

8.3.4.3 端口连接

每个 MVB 二类设备应实现端口连接(Port_Attach)对象。端口连接对象定义端口和输入/输出点之间的编排。该对象仅能远程写入。

8.3.5 信使对象

8.3.5.1 信使状态

提供消息服务的每个站应实现只读的信使状态(Messenger status)对象。信使状态对象报告软件版本、配置和两种协议的使用统计:

- a) 单播协议;
- b) 多播协议(可选)。

代理者通过信使的 ASI 接口访问该对象。

8.3.5.2 信使控制

提供消息服务的每个站应实现信使控制(Messenger control)对象。信使控制对象允许消息通信参数的远程配置。

8.3.5.3 索引对象

提供消息服务的每个站应实现功能索引(Function_Directory)对象。功能索引对象为每个功能指示由哪个站执行。

具备消息能力但不使用简单路由的每个站应实现站索引(Station_Directory)对象。站索引对象将站标识符映射为下一站标识符及其物理地址(总线标识和设备地址)。

支持多播协议的每个终端站和每个路由站应实现组索引(Group_Directory)对象。组索引对象指示站所属的组。

具有消息能力但不使用简单路由的固定组态列车总线的每个节点应实现节点索引(Node_Directory)对象。节点索引对象指示给定节点地址相应的设备地址。

每种索引应有两种服务：

- a) read_xxx_directory:读整个索引；
- b) write_xxx_directory:写整个索引(事先清除或不清除)。

注 1：在终端站中站索引可被简单路由替代,此时不提供站索引对象。

注 2：代理者与在其他应用中一样通过 AMI 接口访问索引。

8.3.6 域对象

8.3.6.1 对象

具有上传或下载存储区域能力的每个站应实现域(Domain)对象。

每个站的域数量未规定。

因技术原因,域宜整体包含在同一存储器类型中。

作为一个特例,管理者可访问域的物理存储区。

注 1：域是包括代码或数据的存储区。域在不同类型的存储器(RAM、EEPROM、FLASH-EPROM)中可采用同样方式定位。

注 2：代理者通过实现特定过程访问域,该过程提供对不同类型存储器的读/写,但可能受访问权限或存储器管理的保护。

注 3：访问域需精确了解站配置。

8.3.6.2 服务

域对象服务应包括：

- a) 下载设置(准备下载、校验、启动)；
- b) 下载段；
- c) 读存储器；
- d) 写存储器。

注 1：域装载可能覆盖现有存储区,甚至通信软件和代理者本身。

注 2：为安全起见,在下载域之前,管理者先发布“reset_station”呼叫。此时,管理者宜再保留此站一次以下载。

8.3.7 任务对象

8.3.7.1 对象

能管理用户任务的每个站应实现只写的任务(Task)对象。

任务的数量未规定。

出于管理考虑,所有任务作为一个整体进行处理。

任务控制对象控制所有任务的执行。

8.3.7.2 访问

代理者访问调度表以启动和停止任务。

对任务的访问是实现相关的。

8.3.7.3 服务

规定了三种任务服务:

- a) 启动(start),启动所有任务;
- b) 停止(stop),停止所有任务;
- c) 任务列表(tasks_list),读出安装于站中的任务列表及其状态。

停止任务前,代理者应通过 AGI(见 8.2.3.3)获得任务许可,除非是“覆盖”访问。

8.3.8 时钟对象

8.3.8.1 对象

具有远程可访问时钟的每个站应实现时钟(Clock)对象。

时钟精度未作规定。

注 1: 通过管理消息设置或读时钟受限于不可预测的延时。特别是消息分发的上限不能给定,因为这与本站及他站队列中其他消息是否存在有关。

注 2: 一种更精确的时钟设置可通过让总线管理器在确定时间点发送同步变量的方式获得。这是一个实现问题。

8.3.8.2 服务

规定了两种时钟服务:

- a) 读时钟(read_clock),读当前时钟值;
- b) 设置时钟(set_clock),设置时钟值。

8.3.9 记录对象

8.3.9.1 对象

出于调试目的,站可实现记录(Journal)对象。

记录由带序列号和时间戳的用户定义项列表组成。记录期望以循环缓冲区方式实现,其中最新记录项被寄存,而较旧记录项被丢弃。

纪录可被多个管理者读取,读不是消费动作。

节点上的应用可高速并行地写入记录。为避免缓存,一个索引机制用于选择记录的有效部分。

每个任务可通过 AGI 接口将事件送入记录。

8.3.9.2 服务

读记录(read_journal):读记录的最后一项。

8.3.10 设备对象

8.3.10.1 对象

站可实现设备描述符,以描述其所支持的设备。

8.3.10.2 服务

读设备描述符(read_equipment_descriptor):获取指向设备描述符所在存储器位置的指针。

8.4 服务和管理消息

8.4.1 管理消息的表示

8.4.1.1 消息结构

每个服务应通过下列格式的呼叫/应答管理消息交换调用。

Management_Message ::= RECORD
 {
 tnm_key ENUM8 ——第一个八位位组
 {
 CALL ("02"H), ——呼叫(请求)
 REPLY ("82"H) ——应答(响应)
 },
 message ONE_OF[tnm_key] ——选择呼叫或应答
 {
 [CALL] Call_Mgt_Message, ——描述见后
 [REPLY] Reply_Mgt_Message ——描述见后
 }
 }

tnm_key 最高有效位应指示这是呼叫消息还是应答消息。

私有网管理中,呼叫消息的 tnm_key 值为 40H~7FH,应答消息的 tnm_key 值为 C0H~FFH。
tnm_key 的其他值保留给将来使用。

tnm_key 的缺省分配值是国际铁路联盟的公司代码加上 40H。

注:为提高 32 位处理器操作速度,管理消息中的字段以 32 位边界对齐。对齐要求首字段 tnm_key 位于 4 的倍数的地址。无论是发送还是接收,消息服务宜遵守此约定。

8.4.1.2 SIF_code 的表示

SIF_code 指示所请求的服务。最低有效位指示这是读服务还是写(修改)服务。读服务使用前不必保留。对于缺省的 tnm_key 对(02H/82H),定义了下列 SIF_code。

Sif_Code ::= ENUM8 ——服务选择
 {
 READ_STATION_STATUS (00),
 WRITE_STATION_CONTROL (01),
 READ_STATION_INVENTORY (02),

WRITE_RESERVATION	(03),
READ_SERVICE_DESCRIPTOR	(04),
READ_LINKS_DESCRIPTOR	(06),
WRITE_LINKS_DESCRIPTOR	(07),
READ_MVB_STATUS	(10),
WRITE_MVB_CONTROL	(11),
READ_MVB_DEVICES	(12),
WRITE_MVB_ADMINISTRATOR	(13),
READ_WTB_STATUS	(20),
WRITE_WTB_CONTROL	(21),
READ_WTB_NODES	(22),
READ_WTB_TOPOGRAPHY	(24),
WRITE_WTB_USER_REPORT _{port}	(25),
LINE_FAULT_LOCATION_DETECTION	(26),
WRITE_ATTACH_PORT	(29),
READ_PORTS_CONFIGURATION	(30),
WRITE_PORTS_CONFIGURATION	(31),
READ_VARIABLES	(32),
WRITE_FORCE_VARIABLES	(33),
WRITE_UNFORCE_VARIABLES	(35),
WRITE_UNFORCE_ALL	(37),
READ_VARIABLE_BINDINGS	(38),
WRITE_VARIABLE_BINDINGS	(39),
READ_MESSENGER_STATUS	(40),
WRITE_MESSENGER_CONTROL	(41),
READ_FUNCTION_DIRECTORY	(42),
WRITE_FUNCTION_DIRECTORY	(43),
READ_STATION_DIRECTORY	(44),
WRITE_STATION_DIRECTORY	(45),
READ_GROUP_DIRECTORY	(46),
WRITE_GROUP_DIRECTORY	(47),
READ_NODE_DIRECTORY	(48),
WRITE_NODE_DIRECTORY	(49),
READ_MEMORY	(50),
WRITE_MEMORY	(51),
WRITE_DOWNLOAD_SETUP	(53),
WRITE_DOWNLOAD_SEGMENT	(55),
READ_TASKS_STATUS	(60),
WRITE_TASKS_CONTROL	(61),
READ_CLOCK	(70),
WRITE_CLOCK	(71),
READ_JOURNAL	(80),
READ_EQUIPMENT	(82),
USER_SERVICE_0	(128),
USER_SERVICE_127	(255)——128~255 预留给用户服务
}	

注：出于完整性考虑，包含了 MVB 服务的 SIF_code。MVB 服务的描述见 GB/T 28209.9。

8.4.1.3 呼叫管理消息的表示

Call_Mgt_Message ::= RECORD

{	
sif_code	Sif_Code,
message_body	ONE_OF[sif_code]
{	
[READ_STATION_STATUS]	Call_Read_Station_Status,
[WRITE_STATION_CONTROL]	Call_Write_Station_Control,
[READ_STATION_INVENTORY]	Call_Read_Station_Inventory,
[WRITE_RESERVATION]	Call_Write_Reservation,
[READ_SERVICE_DESCRIPTOR]	Call_Read_Service_Descriptor,
[READ_LINKS_DESCRIPTOR]	Call_Read_Links_Descriptor,
[WRITE_LINKS_DESCRIPTOR]	Call_Write_Links_Descriptor,
[READ_MVB_STATUS]	Call_Read_Mvb_Status,
[WRITE_MVB_CONTROL]	Call_Write_Mvb_Control,
[READ_MVB_DEVICES]	Call_Read_Mvb_Devices,
[WRITE_MVB_ADMINISTRATOR]	Call_Write_Mvb_Administrator,
[READ_WTB_STATUS]	Call_Read_Wtb_Status,
[WRITE_WTB_CONTROL]	Call_Write_Wtb_Control,
[READ_WTB_NODES]	Call_Read_Wtb_Nodes,
[READ_WTB_TOPOGRAPHY]	Call_Read_Wtb_Topography,
[WRITE_ATTACH_PORT]	Call_Write_Attach_Port,
[READ_PORTS_CONFIGURATION]	Call_Read_Ports_Configuration,
[WRITE_PORTS_CONFIGURATION]	Call_Write_Ports_Configuration,
[READ_VARIABLES]	Call_Read_Variables,
[WRITE_FORCE_VARIABLES]	Call_Write_Force_Variables,
[WRITE_UNFORCE_VARIABLES]	Call_Write_Unforce_Variables,
[WRITE_UNFORCE_ALL]	Call_Write_Unforce_All,
[READ_VARIABLE_BINDINGS]	Call_Read_Variable_Bindings,
[WRITE_VARIABLE_BINDINGS]	Call_Write_Variable_Bindings,
[READ_MESSENGER_STATUS]	Call_Read_Messenger_Status,
[WRITE_MESSENGER_CONTROL]	Call_Write_Messenger_Control,
[READ_FUNCTION_DIRECTORY]	Call_Read_Function_Directory,
[WRITE_FUNCTION_DIRECTORY]	Call_Write_Function_Directory,
[READ_STATION_DIRECTORY]	Call_Read_Station_Directory,
[WRITE_STATION_DIRECTORY]	Call_Write_Station_Directory,
[READ_GROUP_DIRECTORY]	Call_Read_Group_Directory,
[WRITE_GROUP_DIRECTORY]	Call_Write_Group_Directory,
[READ_NODE_DIRECTORY]	Call_Read_Node_Directory,
[WRITE_NODE_DIRECTORY]	Call_Write_Node_Directory,
[READ_MEMORY]	Call_Read_Memory,
[WRITE_MEMORY]	Call_Write_Memory,
[WRITE_DOWNLOAD_SETUP]	Call_Write_Download_Setup,
[WRITE_DOWNLOAD_SEGMENT]	Call_Write_Download_Segment,
[READ_TASKS_STATUS]	Call_Read_Tasks_Status,
[WRITE_TASKS_CONTROL]	Call_Write_Tasks_Control,

——第二个八位位组是 SIF_code

[READ_CLOCK]	Call_Read_Clock,
[WRITE_CLOCK]	Call_Write_Clock,
[READ_JOURNAL]	Call_Read_Journal,
[READ_EQUIPMENT]	Call_Read_Equipment
}	
}	

8.4.1.4 应答管理消息的表示

Reply_Mgt_Message ::= RECORD

{
sif_code Sif_Code,
message_body ONE_OF[sif_code]
{

[READ_STATION_STATUS]	Reply_Read_Station_Status,
[WRITE_STATION_CONTROL]	Reply_Write_Station_Control,
[READ_STATION_INVENTORY]	Reply_Read_Station_Inventory,
[WRITE_RESERVATION]	Reply_Write_Reservation,
[READ_SERVICE_DESCRIPTOR]	Reply_Read_Service_Descriptor,
[READ_LINKS_DESCRIPTOR]	Reply_Links_Descriptor,
[WRITE_LINKS_DESCRIPTOR]	Reply_Write_Links_Descriptor,
[READ_MVB_STATUS]	Reply_Read_Mvb_Status,
[WRITE_MVB_CONTROL]	Reply_Write_Mvb_Control,
[READ_MVB_DEVICES]	Reply_Read_Mvb_Device,
[WRITE_MVB_ADMINISTRATOR]	Reply_Write_Mvb_Administrator],
[READ_WTB_STATUS]	Reply_Read_Wtb_Status,
[WRITE_WTB_CONTROL]	Reply_Write_Wtb_Control,
[READ_WTB_NODES]	Reply_Read_Wtb_Nodes,
[READ_WTB_TOPOGRAPHY]	Reply_Read_Wtb_Topography,
[WRITE_ATTACH_PORT]	Reply_Write_Attach_Port,
[READ_PORTS_CONFIGURATION]	Reply_Read_Ports_Configuration,
[WRITE_PORTS_CONFIGURATION]	Reply_Write_Ports_Configuration,
[READ_VARIABLES]	Reply_Read_Variables,
[WRITE_FORCE_VARIABLES]	Reply_Write_Force_Variables,
[WRITE_UNFORCE_VARIABLES]	Reply_Write_Unforce_Variables,
[WRITE_UNFORCE_ALL]	Reply_Write_Unforce_All,
[READ_VARIABLE_BINDINGS]	Reply_Read_Variable_Bindings,
[WRITE_VARIABLE_BINDINGS]	Reply_Write_Variable_Bindings,
[READ_MESSENGER_STATUS]	Reply_Read_Messenger_Status,
[WRITE_MESSENGER_CONTROL]	Reply_Write_Messenger_Control,
[READ_FUNCTION_DIRECTORY]	Reply_Read_Function_Directory,
[WRITE_FUNCTION_DIRECTORY]	Reply_Read_Function_Directory,
[READ_STATION_DIRECTORY]	Reply_Read_Station_Directory,
[WRITE_STATION_DIRECTORY]	Reply_Write_Station_Directory,
[READ_GROUP_DIRECTORY]	Reply_Read_Group_Directory,
[WRITE_GROUP_DIRECTORY]	Reply_Write_Group_Directory,
[READ_NODE_DIRECTORY]	Reply_Read_Node_Directory,
[WRITE_NODE_DIRECTORY]	Reply_Write_Node_Directory,

——第二个八位位组是 SIFcode

[READ_MEMORY]	Reply_Read_Memory,
[WRITE_MEMORY]	Reply_Write_Memory,
[WRITE_DOWNLOAD_SETUP]	Reply_Write_Download_Setup,
[WRITE_DOWNLOAD_SEGMENT]	Reply_Write_Download_Segment,
[READ_TASKS_STATUS]	Reply_Read_Tasks_Status,
[WRITE_TASKS_CONTROL]	Reply_Write_Tasks_Control,
[READ_CLOCK]	Reply_Read_Clock,
[WRITE_CLOCK]	Reply_Write_Clock,
[READ_JOURNAL]	Reply_Read_Journal,
[READ_EQUIPMENT]	Reply_Read_Equipment
}	
}	

8.4.1.5 状态和错误报告

代理者应通过其应答者状态报告成功或失败。应答者状态不在应答消息正文中传送,而是在会话报头中作为一个独立参数传递。

若发生通信错误,则代理者设置的结果可能被网络修改。此时,网络通过以其 Am_Result 替换代理者代码的方式返回一个通信错误。

应答者状态应作为 mm_service_conf 的一个参数返回给管理者。

状态与错误报告应组合代理者和网络的结果,按 MM_RESULT 类型规定。

MM_RESULT ::= ENUM8

{	
MM_OK	(0), ——成功结束
AM_FAILURE	(1), ——未规定的故障
AM_BUS_ERR	(2), ——没有总线传输可能
AM_REM_CONN_OVF	(3), ——过多的传入连接
AM_CONN_TMO_ERR	(4), ——连接请求无应答
AM_SEND_TMO_ERR	(5), ——SEND_TMO 超时
AM_REPLY_TMO_ERR	(6), ——没有收到应答
AM_ALIVE_TMO_ERR	(7), ——ALIVE_TMO 超时
AM_NO_LOC_MEM_ERR	(8), ——存储器或定时器不足
AM_NO_REM_MEM_ERR	(9), ——对方存储器或定时器不足
AM_REM_CANC_ERR	(10), ——对方已取消
AM_ALREADY_USED	(11), ——已完成相同操作
AM_ADDR_FMT_ERR	(12), ——地址格式错误
AM_NO_REPLY_EXP_ERR	(13), ——非期望的应答
AM_NR_OF_CALLS_OVF	(14), ——过多的呼叫请求
AM_REPLY_LEN_OVF	(15), ——应答消息过长
AM_DUPL_LINK_ERR	(16), ——重复会话错误
AM_MY_DEV_UNKNOWN_ERR	(17), ——己方设备未知
AM_NO_READY_INST_OVF	(18), ——未就绪的应答者事例
AM_NR_OF_INST_OVF	(19), ——过多的应答者事例
AM_CALL_LEN_OVF	(20), ——呼叫消息过长
AM_UNKNOWN_DEST_ERR	(21), ——对方设备未知
AM_INAUG_ERR	(22), ——列车发生初运行
AM_TRY_LATER_ERR	(23), ——(仅内部使用)
AM_FIN_NOT_REG_ERR	(24), ——未登记的终点地址
AM_GW_FIN_NOT_REG_ERR	(25), ——路由器中未登记终点地址

AM_GW_ORI_REG_ERR	(26),	——路由器中未登记起点地址
AM_SIF_NOT_SUPPORTED	(33),	——SIF_code 不支持
MM_RDONLY_ACCESS	(34),	——可只读访问
MM_CMD_NOT_EXECUTED	(35),	——服务失败
MM_DNLD_NO_FLASH	(36),	——此位置无非易失存储器
MM_DNLD_FLASH_HW_ERR	(37),	——下载时发生硬件错误
MM_BAD_CHECKSUM	(38),	——域校验和错误
MM_INT_ERROR	(39),	——内部错误
MM_ER_VERS	(40),	——版本错误
MM_BUS_HW_BAD	(41),	——链路损坏
MM_BUS_HW_NO_CONFIG	(42),	——未配置的硬件
MM_LP_ERROR	(43),	——访问通信存储器失败
MM_VERSION_CONFLICT	(44)	——版本冲突
}		

8.4.1.6 位编号和数据编码

管理消息表示为 16 位单元的结构。一个 16 位单元可有两个 8 位类型、一个 16 位类型或一个 32 位类型的一部分。图 205 映射显示了类型相关位的位编号。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNSIGNED8、BITSET8、ENUM8、WORD8								UNSIGNED8、BITSET8、ENUM8、WORD8							
2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNSIGNED16、ENUM16、WORD16															
2 ¹⁵	2 ¹⁴	2 ¹³	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BITSET16															
2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	2 ¹⁵	2 ¹⁴	2 ¹³	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNSIGNED32															
2 ³¹	2 ³⁰	2 ²⁹	2 ²⁸	2 ²⁷	2 ²⁶	2 ²⁵	2 ²⁴	2 ²³	2 ²²	2 ²¹	2 ²⁰	2 ¹⁹	2 ¹⁸	2 ¹⁷	2 ¹⁶
2 ¹⁵	2 ¹⁴	2 ¹³	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰

图 205 管理消息位编号和数据编码

8.4.2 站服务

8.4.2.1 读站状态

8.4.2.1.1 描述

本服务远程读取站状态对象。

8.4.2.1.2 呼叫读站状态

呼叫读站状态格式如图 206 所示。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 0							

图 206 呼叫读站状态

Call_Read_Station_Status ::= RECORD
{ } ———无参数

8.4.2.1.3 应答读站状态

应答读站状态格式如图 207 所示。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 0							
总线标识 (bus_id)								保留 (reserved1)							
设备地址 (device_address)															
站状态 (station_status)															

图 207 应答读站状态

Reply_Read_Station_Status ::= RECORD
{
 bus_id UNSIGNED8(0...15), ——代理者接收呼叫的链路(如 MVB、WTB)标识。
 该链路在整个管理会话过程中不可改变
 reserved1 WORD8 (=0), ——保留
 device_address WORD16, ——代理者接收呼叫的总线设备地址
 station_status Station_Status ——见定义
}

注：管理者可通过设备地址方式访问未知站。

8.4.2.2 写站控制

8.4.2.2.1 描述

本服务复位站,并指定其站名及站标识符。

8.4.2.2.2 呼叫写站控制

呼叫写站控制格式如图 208 所示。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 1							
命令 (command)							复位 (RST)	站标识 (station_id)							
站名 (station_name) :string32															
(CHARACTER8)								(CHARACTER8) 或00H							

图 208 呼叫写站控制

```
Call_Write_Station_Control ::= RECORD
{
  command      BITSET8
  {
    rst          (0)          ——1:复位站,清除表格及保留信号量,仅启动信使和代理者。
                                0:仅分配新的站名和站标识符
  },
  station_id    UNSIGNED8,    ——分配给该站的 8 位站标识符。
                                若 station_id 为 0,则站标识符不变
  station_name  STRING32      ——分配给该站的站名。若站名长度为 0,则站名不变
}
```

8.4.2.2.3 应答写站控制

应答写站控制格式如图 209 所示。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 1							
总线标识 (bus_id)								保留 (reserved1)							
设备地址 (device_address)															

图 209 应答写站控制

```
Reply_Write_Station_Control ::= RECORD      ——sif_code = 1
{
  bus_id          UNSIGNED8 (0...15),      ——代理者接收呼叫消息的链路
  reserved1       WORD8 (=0),              ——保留
  device_address  WORD16                    ——代理者接收呼叫消息的地址
}
```

8.4.2.3 读站信息

8.4.2.3.1 描述

本服务读取站信息对象。

8.4.2.3.2 呼叫读站信息

呼叫读站信息格式如图 210 所示。

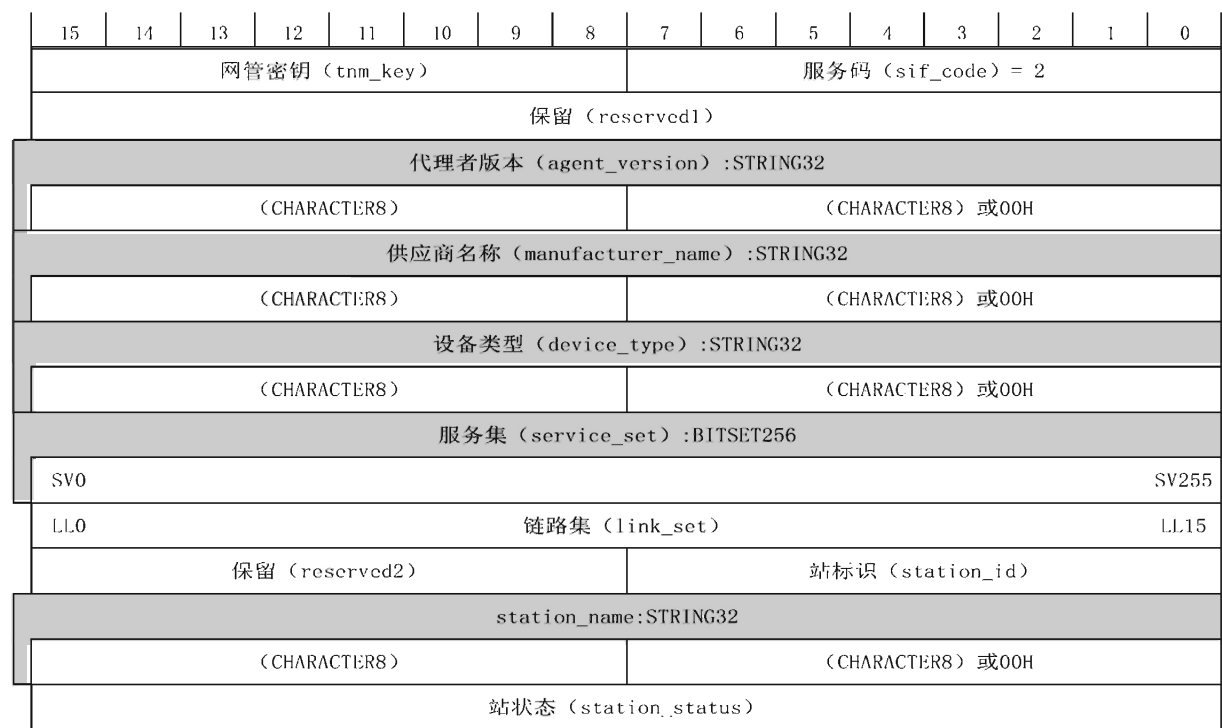
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 2							

图 210 呼叫读站信息

```
Call_Read_Station_Inventory ::= RECORD
{ }      ——无参数
```

8.4.2.3.3 应答读站信息

应答读站信息格式如图 211 所示。



```
Reply_Read_Station_Inventory ::= RECORD
{
    reserved1          WORD16 (=0),          ——用于对齐
    agent_version      STRING32,             ——代理者版本(如,引用本章)
    manufacturer_name  STRING32,             ——设备供应商名称
    device_type        STRING32,             ——设备名及序列号
    service_set        BITSET256,            ——每位对应一种服务(第一位置1代表设备支持读站状态服务)
    link_set           Link_Set,              ——每位对应一个支持的通信存储器或链路层,每一个链路层对应一个通信存储器,第一位对应通信存储器0
    reserved2          WORD8 (=0),           ——保留
    station_id         UNSIGNED,              ——本站标识符(未定义时为0或“FF”H)
    station_name       STRING32,              ——本站给定名称(初始为空)
    station_status     Station_Status         ——站状态字
}
```

注：管理者可通过站设备地址访问未知站。

图 211 应答读站信息

8.4.2.4 写站保留

8.4.2.4.1 描述

本服务访问保留对象,保留或释放该站。

保留呼叫规定了管理者(通过其应用地址)、保留超时和访问权限。保留呼叫包含一个 32 位管理者标识符,其用法与应用有关。

若站已被保留且先前执行保留的管理者与当前管理者不同时,则站应拒绝保留请求。

已保留的站应置位其站状态中的 SER 位和每个与站相连的 MVB 的设备状态中的 SER 位。
接收到当前管理者的任意服务呼叫时,保留超时应重启。

下列情况下,站应释放保留并清除 SER 位:

- a) 当接收到没有重启选项的“释放”呼叫时(正常情况);
- b) 当保留超时期满;
- c) 当发生列车初运行且管理者位于另一节点上;
- d) 当接收到“复位”命令时;
- e) 当接收到“覆盖”命令时。

在后两种情况下,站应执行应用以“站重启”(station_restart)(见 8.5.3.4)订阅的过程。

8.4.2.4.2 呼叫写站保留

呼叫写站保留格式如图 212 所示。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 3							
命令 (command)															
访问类型 (access_type)															
保留超时 (reservation_time_out)															
管理者标识 (manager_id)															

```
Call_Write_Reservation ::= RECORD
{
    command                ENUM16
    {
        RESERVE            (1),          —— 为该管理者保留的设备
        KEEPREL            (2),          —— 释放且保持变化
        STARTREL           (3)          —— 释放且重启
    },
    access_type            ENUM16
    {
        WRITEREQ           (0),          —— 请求的写访问
        OVERRIDE           (1)          —— 保留的覆盖访问
    },
    reservation_time_out   UNSIGNED16,   —— 站维持保留的时间,1 s 的整数倍数,但不大于 3 600 s
    manager_id             UNSIGNED32     —— 管理者标识符(该标识符的用法和实现有关)
}
```

图 212 呼叫写站保留

8.4.2.4.3 应答写站保留

应答写站保留格式如图 213 所示。

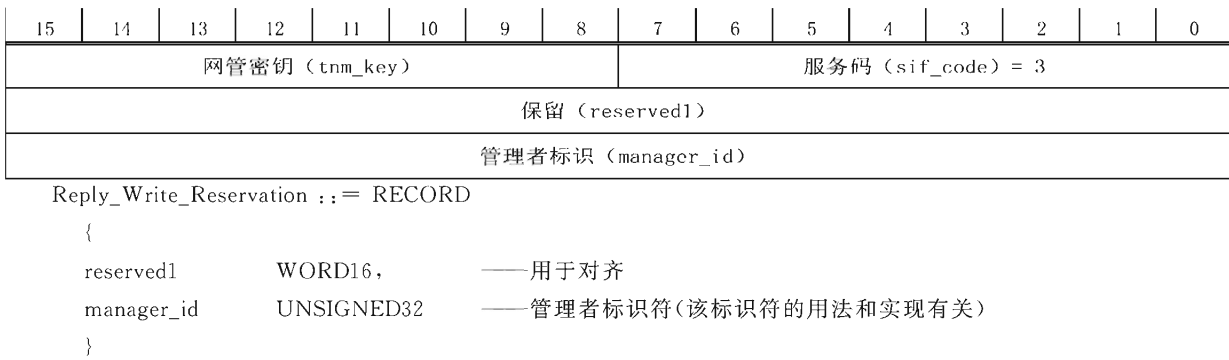


图 213 应答写站保留

8.4.2.5 读服务描述符

8.4.2.5.1 描述

本服务读取服务描述符,即定义服务或通过本服务访问的对象的描述文本。通常仅用于用户自定义服务。

8.4.2.5.2 呼叫读服务描述符

呼叫读服务描述符格式如图 214 所示。

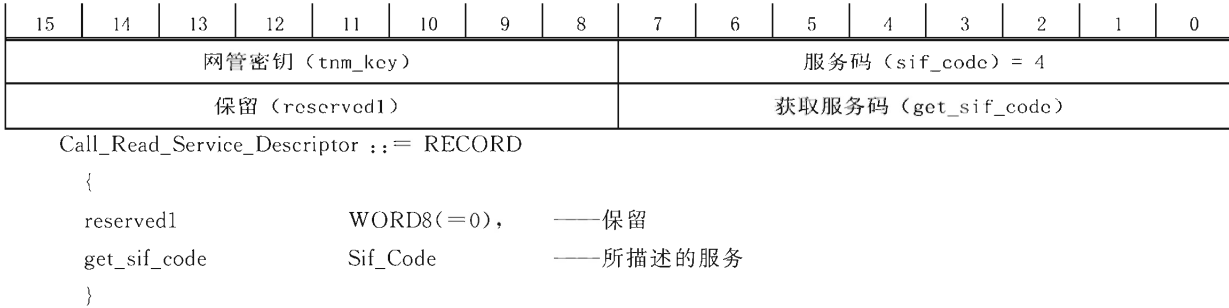


图 214 呼叫读服务描述符

8.4.2.5.3 应答读服务描述符

应答读服务描述符格式如图 215 所示。



图 215 应答读服务描述符

Reply_Read_Service_Descriptor ::= RECORD

{		
reserved1	WORD8(=0),	——保留
get_sif_code	Sif_Code,	——所描述的服务
reserved2	WORD16(=0),	——保留
string_size	UNSIGNED16,	——最多 65 535 个字符
service_description	ARRAY ALIGN16 [string_size] OF CHARACTER8	——用户定义的字符串
}		

8.4.2.6 读链路描述符

8.4.2.6.1 描述

本服务读取链路描述符,即应描述存在于站信息对象中的每个链路层的描述文本。

8.4.2.6.2 呼叫读链路描述符

呼叫读链路描述符格式如图 216 所示。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 6							

图 216 呼叫读链路描述符

Call_Read_Links_Descriptor ::= RECORD
{ } ——无参数

8.4.2.6.3 应答读链路描述符

应答读链路描述符格式如图 217 所示。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 6							
链路数 (nr_links)															
链路描述符 (link_descriptor) :ARRAY[nr_links] OF															
总线标识符 (bus_id)								链路类型 (link_type)							
链路名 (link_name) :STRING32															
								(CHARACTER8) 或00H							

图 217 应答链路描述符

Reply_Read_Links_Descriptor ::= RECORD
{
nr_links UNSIGNED16 (0...15), ——支持的链路数
links_descriptor ARRAY[nr_links] OF
 {
 bus_id UNSIGNED8 (0...15), ——链路标识符
 link_type ENUM8
 {
 LINK_UNKNOWN (0), ——未知
 LINK_MVB (1), ——MVB 总线

LINK_WTB	(2),	——WTB 总线
LINK_MBX	(3),	——内存信箱
LINK_SER	(4)	——串行链路
		——其他保留
},		
link_name	STRING32	——用字符串表示的链路名
}		
}		

8.4.2.7 写链路描述符

8.4.2.7.1 描述

本服务写入链路描述符,即应描述存在于站信息对象中的每个链路层的描述文本。

8.4.2.7.2 呼叫写链路描述符

呼叫写链路描述符格式如图 218 所示。



图 218 呼叫写链路描述符

```
Call_Write_Links_Descriptor ::= RECORD
{
  nr_links          UNSIGNED16 (0...15),      ——支持的链路数
  links_descriptor  ARRAY[nr_links] OF
  {
    bus_id          UNSIGNED8(0...15),        ——链路标识符
    link_type        ENUM8
    {
      LINK_UNKNOWN  (0),                      ——未知
      LINK_MVB      (1),                      ——MVB 总线
      LINK_WTB      (2),                      ——WTB 总线
      LINK_MBX      (3),                      ——内存信箱
      LINK_SER      (4),                      ——串行链路
      LINK_CAN      (5),                      ——CAN 总线链路
      LINK_ETH      (6),                      ——以太网链路
      ———其他保留
    },
    link_name        STRING32                  ——链路名
  }
}
```


8.4.2.7.3 应答写链路描述符

应答写链路描述符格式如图 219 所示。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 7							

图 219 应答写链路描述符

reply_Write_Links_Descriptor ::= RECORD
{ } ——— 无参数

8.4.3 WTB 链路服务

8.4.3.1 读 WTB 状态

8.4.3.1.1 描述

本服务读取节点的 WTB 链路层状态。若本站无列车总线,则返回一个错误且应答消息仅包含报头部分。

返回的数据对应于 5.6.4.16.3 和 5.6.4.22.3 中规定的数据结构 Type_WTBStatus 和 Type_LL-StatisticData。

注: 如有歧义,以 5.6.4.16.3 和 5.6.4.22.3 中的参数定义为准。

8.4.3.1.2 呼叫读 WTB 状态

呼叫读 WTB 状态格式如图 220 所示。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 20							
总线标识符 (bus_id)								保留 (reserved1)							

图 220 呼叫读 WTB 状态

Call_Read_Wtb_Status ::= RECORD
{
 bus_id UNSIGNED8 (0...15), —— 链路标识符
 reserved1 WORD8 (=0) —— 保留
}

8.4.3.1.3 应答读 WTB 状态

应答读 WTB 状态格式见图 221。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 20							
总线标识符 (bus_id)								节点地址 (node_address)							
WTB硬件标识符 (wtb_hardware_id)								WTB软件标识符 (wtb_software_id)							
网络禁止初运行 (net_inhibit)								节点地址 (node_address)							
节点朝向 (node_orient)								节点强度 (node_strength)							
节点帧长 (node_frame_size)								节点周期 (node_period)							
节点类型 (node_type)								节点版本 (node_version)							
节点报告 (node_report)								用户报告 (user_report)							
基本周期计数 (basic_period_count)															
初运行计数 (inauguration_count)															
拓扑计数 (topography_count)															
已发送消息计数 (transmitted_md_count)															
已接收消息计数 (received_md_count)															
A1线路状态 (line_status_a1) (已发送计数 (transmitted_count)) (已接收计数 (received_count)) (错误计数 (errors_count))															
A2线路状态 (line_status_a2)															
B1线路状态 (line_status_b1)															
B2线路状态 (line_status_b2)															
线路切换计数 (line_switch_count)															

图 221 应答读 WTB 状态

```
Line_Status ::= RECORD
{
    transmitted_count    UNSIGNED32,      —— 每发送一帧递增
    received_count       UNSIGNED32,      —— 每接收一帧递增
    frame_errors_count   UNSIGNED16,      —— 每错误一帧递增
    frame_timeout_count  UNSIGNED16       —— 每超时一帧递增
}

Reply_Read_Wtb_Status ::= RECORD
{
    bus_id               UNSIGNED8 (0...15), —— 链路标识符
    node_address         WORD8,             —— 节点地址(未命名节点为 127)
    wtb_hardware_id      UNSIGNED8,        —— 硬件标识符
    wtb_software_id      UNSIGNED8,        —— 链路层软件版本标识符
    hardware_state       ENUM8
    {
        LS_OK            (0),             —— WTB_LS_OK    正确操作
        LS_FALL          (1)             —— WTB_LS_FALL  硬件故障
    }
}
```

},		
link_layer_state	Ls_T_State,	——节点当前所处主要状态:
net_inhibit	ENUM8,	——1:某个节点禁止初运行
node_address	UNSIGNED8,	——分配的节点地址
node_orient	ENUM8	——节点相对于主节点的朝向
{		
L_UNKNOWN	(0),	——WTB_LS_UNKNOWN
L_SAME	(1),	——WTB_LS_SAME
L_INVERSE	(2)	——WTB_LS_INVERSE
},		
node_strength	ENUM8	——节点强度
{		
L_UNDEFINED	(0),	——未定义的节点强度
L_SLAVE	(1),	——节点仅为从
L_STRONG	(2),	——节点为强主
L_WEAK	(3)	——节点为弱主
},		
node_frame_size	UNSIGNED8,	——节点发送的过程数据帧长度
node_period	UNSIGNED8,	——期望的特征周期,以 T _{bp} 倍数计数
node_type	UNSIGNED8,	——节点能力描述符(节点密钥的一部分)
node_version	UNSIGNED8,	——节点能力描述符(节点密钥的一部分)
node_report	Node_Report,	——8 位节点报告,与状态响应帧中表示相同
user_report	User_Report,	——8 位用户报告,与状态响应帧中表示相同
——以下对应于 Type_LLStatisticData		
basic_period_count	UNSIGNED32,	——每个基本周期递增
inauguration_count	UNSIGNED16,	——每次初运行递增
topography_count	UNSIGNED16,	——每个新拓扑递增
transmitted_md_count	UNSIGNED32,	——每次发送消息数据响应帧递增
received_md_count	UNSIGNED32,	——每次接收消息数据响应帧递增
line_status_a1	Type_LineStatus,	——A1 线路统计,见类型定义
line_status_a2	Type_LineStatus,	——A2 线路统计,见类型定义
line_status_b1	Type_LineStatus,	——B1 线路统计,见类型定义
line_status_b2	Type_LineStatus,	——B2 线路统计,见类型定义
line_switch_count	UNSIGNED32	——每次线路切换递增
}		

8.4.3.2 写 WTB 控制

8.4.3.2.1 描述

设置列车总线节点中的参数。

8.4.3.2.2 呼叫写 WTB 控制

呼叫写 WTB 控制格式见图 222。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 21							
总线标识符 (bus_id)								保留 (reserved1)							
保留 (rsv1)	保留 (rsv2)	保留 (rsv3)	保留 (rsv4)	保留 (rsv5)	取消 休眠 (csl)	设置 休眠 (slp)	允许 (alw)	禁止 (inh)	移除 (rmv)	开始 命名 (snm)	设置 强主 (stm)	设置 弱主 (wkm)	设置 从设备 (slv)	配置 (cnf)	复位 节点 (rst)

图 222 呼叫写 WTB 控制

```
Call_Write_Wtb_Control ::= RECORD
{
    bus_id      UNSIGNED8 (0...15),      ——链路标识符
    reserved1   WORD8 (=0),              ——保留
    command     BITSET16
    {
        rsv1    (15),                    ——保留
        rsv2    (14),                    ——保留
        rsv3    (13),                    ——保留
        rsv4    (12),                    ——保留
        rsv5    (11),                    ——保留
        csl     (10),                    ——取消休眠
        slp     (9),                     ——设置休眠
        alw     (8),                     ——允许
        inh     (7),                     ——禁止
        rmv     (6),                     ——移除
        snm     (5),                     ——开始命名
        stm     (4),                     ——设置强主
        wkm     (3),                     ——设置弱主
        slv     (2),                     ——设置从设备
        cnf     (1),                     ——配置
        rst     (0)                      ——复位节点
    }
}
```

8.4.3.2.3 应答写 WTB 控制

应答写 WTB 控制格式见图 223。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 21							
总线标识符 (bus_id)								保留 (reserved1)							

图 223 应答写 WTB 控制

```
Reply_Write_Wtb_Control ::= RECORD
{
    bus_id      UNSIGNED8 (0...15),      ——链路标识符
    reserved1   WORD8 (=0)               ——保留
}
```

8.4.3.3 读 WTB 节点

8.4.3.3.1 描述

读取主设备已在总线上发现的节点列表及其节点状态字(该命令总是送到节点 01)。

8.4.3.3.2 呼叫读 WTB 节点

呼叫读 WTB 节点格式见图 224。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 22							
总线标识符 (bus_id)								保留 (reserved1) = 0							

图 224 呼叫读 WTB 节点

```
Call_Read_Wtb_Nodes := RECORD
{
  bus_id          UNSIGNED8 (0...15),    ——链路标识符
  reserved1       WORD8 (=0)             ——保留
}
```

8.4.3.3.3 应答读 WTB 节点

应答读 WTB 节点格式见图 225。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 22							
总线标识符 (bus_id)								节点地址 (node_address)							
保留 (reserved1)								节点数 (nr_nodes)							
底节点 (bottom_node)								顶节点 (top_node)							
节点状态列表 (node_status_list) :ARRAY[MAX_NODES] OF															
节点报告 (node_report)								用户报告 (user_report)							

图 225 应答读 WTB 节点

```
Reply_Read_Wtb_Nodes := RECORD
{
  bus_id          UNSIGNED8 (0...15),    ——链路标识符
  node_address     UNSIGNED8,             ——节点地址(未命名节点为 127)
  reserved1       WORD8 (=0),             ——保留
  nr_nodes        UNSIGNED8,             ——组成节点数
  bottom_node     UNSIGNED8,             ——最低地址节点的节点地址
  top_node        UNSIGNED8,             ——最高地址节点的节点地址
  node_status_list ARRAY [MAX_NODES] OF
    ——节点状态表,由底节点开始,以节点位置为序,至顶节点结束,包括:
    {
      node_report  Node_Report,          ——不同节点的节点报告
      user_report  User_Report           ——不同节点的用户报告
    }
}
```

8.4.3.4 读 WTB 拓扑

8.4.3.4.1 描述

读拓扑。拓扑的格式在参数列表中规定,因为拓扑信息在 WTB 上发送时、在 WTB 监视接口可用时与通过网络管理可用时都略有差异。

8.4.3.4.2 呼叫读 WTB 拓扑

呼叫读 WTB 拓扑格式见图 226。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 24							
总线标识符 (bus_id)								保留 (reserved1)							

图 226 呼叫读 WTB 拓扑

```
Call_Read_Wtb_Topography ::= RECORD
{
    bus_id          UNSIGNED8 (0...15),    ——链路标识符
    reserved1      WORD8 (=0)             ——保留
}
```

8.4.3.4.3 应答读 WTB 拓扑

应答读 WTB 拓扑格式见图 227。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 24							
总线标识符 (bus_id)								节点地址 (node_address)							
节点朝向 (node_orient)								预留 (rsv1)	预留 (rsv2)	拓扑计数器 (topo_counter)					
特征周期 (individual_period)								强主 (is_strong)							
节点数 (number_of_nodes)								底节点地址 (bottom_address)							
顶节点地址 (top_address)								保留 (reserved1)							
初运行数据最大长度 (inaug_data_max_size)								描述符数 (nr_descriptors)							
节点描述 (node_descriptions) :ARRAY[nr_descriptors] OF															
节点类型 (node_type)								节点版本 (node_version)							
同向 (sam)	保留 (rsv1)	节点地址 (node_address)						初运行数据长度 (inauguration_data_size)							
初运行数据 (inauguration_data) ARRAY ALIGNED16[inauguration_data_size] OF															
WORDS								WORDS							

图 227 应答读 WTB 拓扑

```
Reply_Read_Wtb_Topography ::= RECORD
{
    bus_id          UNSIGNED8 (0...15),    ——链路标识符
}
```

node_address	UNSIGNED8,	——该站连接的节点地址
node_orient	ENUM8,	——该节点相对于主节点的方向： 0=未知,1=相同,2=相反
rsv1	WORD1 (=0),	——保留,=0
rsv2	WORD1 (=0),	——保留,=0
topo_counter	UNSIGNED6,	——该节点中 6 位拓扑计数器
individual_period	UNSIGNED8,	——8 位整数,表示由主设备分配的特征周期,以基本 周期的 2 的幂数倍表示,单位是 ms
is_strong	ENUM8,	——=1:由强主控制的总线;=0:由弱主控制的总线
number_of_nodes	UNSIGNED8 (0...63),	——组成中节点数
bottom_address	UNSIGNED8,	——最低地址节点的节点地址
top_ address	UNSIGNED8,	——最高地址节点的节点地址
inaug_data_max_size	UNSIGNED8,	——maxinaugdatasize 的拷贝
nr_descriptors	UNSIGNED8,	——无错时等于 number_of_nodes
node_descriptions	ARRAY [nr_descriptors] OF	
{		——描述符列表,由底节点开始,按节点地址升序,由 下列元素组成
node_type	UNSIGNED8,	——节点密钥的第一部分
node_version	UNSIGNED8,	——节点密钥的第二部分
sam	BOOLEAN1,	——与主设备同向
rsv1	WORD1 (=0),	——保留,=0
node_address	UNSIGNED16,	——节点地址
inauguration_data_size	UNSIGNED8,	——后续数据长度
inauguration_data	ARRAY [inauguration_data_size] OF WORD8	——初运行数据,用户提供的结构
}		
}		

8.4.3.5 写 WTB 用户报告

8.4.3.5.1 描述

设置列车总线节点参数。

8.4.3.5.2 呼叫写 WTB 用户报告

呼叫写 WTB 用户报告格式见图 228。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm key)								服务码 (sif_code) = 25							
总线标识符 (bus_id)								ur7	Ur6	Ur5	Ur4	Ur3	Ur2	Ur1	Ur0

图 228 呼叫写 WTB 用户报告

```
Call_Write_Wtb_User_Report ::= RECORD
{
    bus_id      UNSIGNED8 (0...15),      ——链路标识符
    command    BITSET8
```

{		
ur7	(7),	——用户报告位 7
ur6	(6),	——用户报告位 6
ur5	(5),	——用户报告位 5
ur4	(4),	——用户报告位 4
ur3	(3),	——用户报告位 3
ur2	(2),	——用户报告位 2
ur1	(1),	——用户报告位 1
ur0	(0)	——用户报告位 0
}		
}		

8.4.3.5.3 应答写 WTB 用户报告

应答写 WTB 用户报告格式见图 229。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 25							
总线标识符 (bus_id)								保留 (reserved1)							

图 229 应答写 WTB 用户报告

```
Reply_Write_Wtb_User_Report ::= RECORD
{
    bus_id          UNSIGNED8 (0...15),    ——链路标识符
    reserved1      WORD8 (=0)             ——保留
}
```

8.4.3.6 线路故障位置检测

8.4.3.6.1 描述

本服务定位受扰线路故障到两个节点之间。若站无列车总线,则返回错误且应答消息仅包含报头。本服务有多个子命令。下列子命令提供给网络管理应用:

- 启动 LFLD(Line Fault Location Detection,线路故障位置检测);
- 获取 LFLD 结果;
- 取消 LFLD。

下列子命令提供给内部使用:

- 向对端节点指示 LFLD 启动;
- 向对端节点指示 LFLD 停止;
- 启动 LFLD 分段节点(中间节点);
- 停止 LFLD 分段节点(中间节点)。

为使用 LFLD 服务,宜采用多阶进程。诊断应用宜监视线路状态位以检测线路扰动。若检测到稳定的线路扰动,诊断应用宜发送 LFLD 子命令 0 给一个末端节点以启动 LFLD 进程。然后诊断应用应使用子命令 1 轮询 LFLD 结果,直到结果可用。LFLD 结果仅能获取一次。为获取新的 LFLD 结果,应再次启动 LFLD 进程。

为取消运行中的 LFLD 进程,诊断应用应向以子命令 0 并启动 LFLD 进程的末端节点发布 LFLD 子命令 2。

注 1: 若受扰线路上有多个故障,仅可检测一个。

注 2：若有节点不支持 LFLD 服务，则故障位置可能仅定位到多个节点之间。
注 3：若发生了初运行，则 LFLD 进程中止。

8.4.3.6.2 呼叫线路故障位置检测

呼叫线路故障位置检测格式见图 230。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 26							
总线标识符 (bus_id)								子命令 (sub_command)							
参数1 (par1)								参数2 (par2)							

图 230 呼叫线路故障位置检测

```
Call_Line_Fault_Location_Detection ::= RECORD
{
    bus_id            UNSIGNED8 (0...15),      ——链路标识符
    sub_command       WORD8 (0...6),           ——子命令
    par1              WORD8,                   ——LFLD 子命令参数 1
    par2              WORD8                   ——LFLD 子命令参数 2
}
```

定义了下列 LFLD 子命令：

- 子命令 = 0:启动 LFLD 进程。该子命令仅发送给一个末端节点；
par1 = 0,par2 = 0:未使用。
- 子命令 = 1:获取 LFLD 结果。该子命令仅发送给一个末端节点；
par1 = 0,par2 = 0:未使用。
- 子命令 = 2:取消 LFLD 进程。该子命令仅发送给一个末端节点；
par1 = 0,par2 = 0:未使用。
- 子命令 = 3:向对端节点指示 LFLD 启动；
par1 = 0,par2 = 0:未使用。
- 子命令 = 4:向对端节点指示 LFLD 停止；
par1 = 0,par2 = 0:未使用。
- 子命令 = 5:启动 LFLD 分段节点(中间节点)；
par1 = line WTB 总线主所见的标准受扰线路(line=0 即 A 线,line=1 即 B 线)。
par2 = oe 对端节点的节点地址 oe。
- 子命令=6:停止 LFLD 分段节点(中间节点)。
par1 = line WTB 总线主所见的标准受扰线路(line=0 即 A 线,line=1 即 B 线)。
par2 = 0 未使用。

8.4.3.6.3 应答线路故障位置检测

应答线路故障位置检测格式见图 231。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 26							
总线标识符 (bus_id)								子命令 (sub_command)							
结果 (result)								返回值1 (ret1)							
返回值2 (ret2)								保留 (reserved)							

图 231 应答线路故障位置检测

Reply_Line_Fault_Location_Detection ::= RECORD
{
 bus_id UNSIGNED8 (0...15), ——链路标识符
 sub_command WORD8 (0...6), ——子命令
 result WORD8, ——执行结果
 ret1 WORD8, ——LFLD 子命令返回值 1
 ret2 WORD8, ——LFLD 子命令返回值 2
 reserved WORD8 (=0) ——保留
}

定义了下列 LFLD 子命令应答：

- 子命令 = 0:启动 LFLD 进程;
 - result = 0 LFLD 进程已启动
 - = 1 LFLD 进程已运行
 - = 4 发送给中间节点的命令:LFLD 进程未启动
 - = 5 无线路受扰:LFLD 进程未启动
 - = 6 双线(暂时)受扰:LFLD 进程不能启动
 - ret1、ret2 = 0 未使用
- 子命令 = 1:获取 LFLD 结果;
 - result = 0 LFLD 进程未启动,无可用的结果
 - = 1 LFLD 进程正在运行,无可用的结果
 - = 2 LFLD 结果可用
 - ret1 = n1 定位线路故障的一个节点地址 n1
 - ret2 = n2 定位线路故障的另一节点地址 n2
- 子命令 = 2:取消 LFLD 进程;
 - result = 0 在该末端节点上未启动 LFLD 进程
 - = 1 LFLD 进程已取消
 - ret1、ret2 = 0 未使用
- 子命令 = 3:向对端节点指示 LFLD 启动。
 - result = 0 在对端节点上启动 LFLD 进程
 - ret1、ret2 = 0 未使用
- 子命令 = 4:向对端节点指示 LFLD 停止;
 - result = 0 在对端节点上停止 LFLD 进程
 - ret1、ret2 = 0 未使用
- 子命令 = 5:启动 LFLD 分段节点;
 - result = 0 LFLD 分段节点已启动
 - ret1、ret2 = 0 未使用

- 子命令 = 6:停止 LFLD 分段节点。
 - result = 0 LFLD 分段节点已停止
 - ret1 = oe 对端节点的节点地址 oe,若成功接收帧
 - = 127 若未从对端节点成功接收帧
 - ret2 = 0 方向 1 上连接继电器闭合
 - = 1 方向 2 上连接继电器闭合

8.4.4 变量服务

8.4.4.1 读端口配置

8.4.4.1.1 描述

获取已配置端口列表及其 F_code(隐含长度)和属性。

8.4.4.1.2 呼叫读端口配置

呼叫读端口配置格式见图 232。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 30							
总线标识符 (bus_id)								保留 (reserved1) = 0							

图 232 呼叫读端口配置

```
Call_Read_Ports_Configuration ::= RECORD
{
  bus_id      UNSIGNED8,      ——(0...15)
  reserved1   WORD8 (=0)      ——保留
}
```

8.4.4.1.3 应答读端口配置

应答读端口配置格式见图 233。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 30							
端口数 (nr_ports)															
端口列表 (ports_list) :ARRAY[nr_ports] OF															
总线标识符 (bus_id)				端口地址 (port_address)											
功能码 (f_code)				源 (src)	宿 (snk)	带校 验和 (twc)	强制 (frc)	端口长度 (port_size)							

图 233 应答读端口配置

```
Reply_Read_Ports_Configuration ::= RECORD
{
  nr_ports      UNSIGNED16,      ——该通信存储器中端口数(最大 4 096)
  ports_list    ARRAY [nr_ports] OF
  {
```

bus_id	WORD4,	——通信存储器
port_address	UNSIGNED12,	——端口地址
f_code	F_Code,	——端口功能码
port_config	BITSET4	
{		
src		——发布者端口(源)
snk		——订阅者端口(宿)
twc		——带校验和传送
frc		——强制端口
},		
port_size	UNSIGNED8	——以八位位组计数的端口最大长度(仅对 WTB)
}		
}		

8.4.4.2 写端口配置

8.4.4.2.1 描述

使能或禁止端口。

8.4.4.2.2 呼叫写端口配置

呼叫写端口配置格式见图 234。

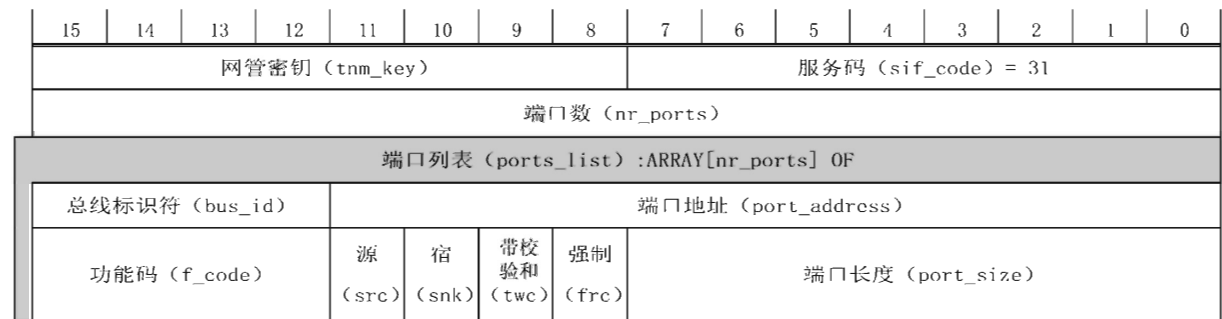


图 234 呼叫写端口配置

Call_Write_Ports_Configuration ::= RECORD		
{		
nr_ports	UNSIGNED16,	——通信存储器中端口数(最大 4 096)
ports_list	ARRAY [nr_ports] OF	
{		
bus_id	UNSIGNED8 (0...15),	——站可见的通信存储器标识符
port_address	UNSIGNED12,	——端口地址
f_code	F_Code,	——端口功能码(见端口配置对象)
port_config	BITSET4	
{		
src		——使能发布者端口(源)
snk		——使能订阅者端口(宿)
twc		——使能带校验和传送
frc		——强制端口为缺省值
},		

port_size UNSIGNED8 ——以八位位组计数的端口最大长度(仅对 WTB)

}

}

8.4.4.2.3 应答写端口配置

应答写端口配置格式见图 235。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 31							
总线标识符 (bus_id)								保留 (reserved1)							

图 235 应答写端口配置

Reply_Write_Ports_Configuration ::= RECORD

{

bus_id UNSIGNED8 (0...15), ——站可见的通信存储器标识符

reserved1 WORD8 (=0) ——保留

}

8.4.4.3 读变量

8.4.4.3.1 描述

读变量集群的值、校验变量和刷新值。

每个变量由通信存储器内其位置和其校验变量位置、以及其类型和长度标识。

变量可通过 PV_Sets 或 PV_Clusters 单个获取。

若连续变量属于同一数据集，则应坚固访问(通过 PV_Set)。

代理者应以值列表响应，其顺序应与呼叫消息中列出的变量顺序相同。

变量值应以其作为过程数据在 WTB 上传送的相同格式(即大开端格式)在管理消息中传送。

每一个变量值应从一新的字边界开始。

长度不足 16 位的变量应右对齐，如是带符号数，则符号位扩展。

长度大于 16 位但又不是 16 位整数倍的变量应右对齐，并以“0”填充至下一个 16 位字边界。

示例 1：一个 8 位整型数“0111 1111”B(+127)按“0000 0000 0111 1111”B 发送。

示例 2：一个 8 位整型数“1111 1111”B(−1)按“1111 1111 1111 1111”B 发送。

8.4.4.3.2 呼叫读变量

呼叫读变量服务格式见图 236。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 32							
变量数 (nr_vars)															
变量列表 (variables_list) :ARRAY[nr_vars] OF															
总线标识符 (bus_id)				端口地址 (port_address)											
变量长度 (var_size)								变量类型 (var_type)							
变量偏置 (var_offset)															
校验偏置 (chk_offset)															

图 236 呼叫读变量服务

Call_Read_Variables ::= RECORD
 {
 nr_vars UNSIGNED16, ——列表中变量数
 variables_list ARRAY[nr_vars] OF
 {
 bus_id UNSIGNED4, ——通信存储器标识符
 port_address WORD12, ——通信存储器端口地址
 var_size UNSIGNED8, ——简单类型:以位计的长度;
 结构类型:根据 6.2(RTP)的元素个数
 var_type UNSIGNED8, ——根据 6.2(RTP)的变量类型
 var_offset UNSIGNED16, ——变量偏置
 chk_offset UNSIGNED16 ——校验变量偏置(未使用为“FFFF”H)
 }
 }

8.4.4.3.3 应答读变量

应答读变量格式见图 237。

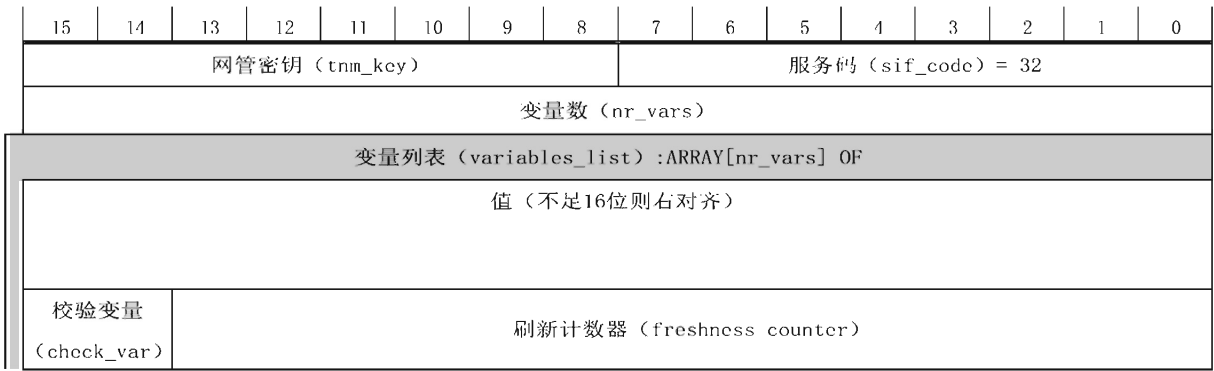


图 237 应答读变量

Reply_Read_Variables ::= RECORD
 {
 nr_vars UNSIGNED16, ——列表中变量数
 values_list ARRAY[nr_vars] OF
 {
 value ANY, ——每个变量以呼叫消息中的顺序存放在该表中
 ——值,格式由 PV_NAME 长度及类型决定,对齐到 16 位字
 边界。CHARACTER8 例外,因其解释为字符数组
 [0..0] 的特例,字符占高八位位组,低八位位组填充
 “00”H
 check_var ANTIVALENT2, ——相关联的检验变量,无规定时为“11”B
 freshness UNSIGNED14 ——以 ms 计的变量刷新值(最大 4 096 ms)
 }
 }

注：校验变量也可按任何其他(ANTIVALENT2)变量传送。

8.4.4.4 写强制变量

8.4.4.4.1 描述

强制集群变量为给定值。

强制一个变量应设置相应 MVB 通信存储器设备状态字和站状态字中的“frc”位。
校验字段应设置为校验变量规定的值。
每个变量值应从一个新的字边界开始。
长度不足 16 位字的变量应右对齐,若是有符号数则符号扩展。
长度大于 16 位但又不是 16 位整数倍的变量应右对齐,并以“0”填充至下一个 16 位字边界。
注:列车总线状态中无“frc”位,但此信息可由站状态得到。

8.4.4.4.2 呼叫写强制变量

呼叫写强制变量格式见图 238。

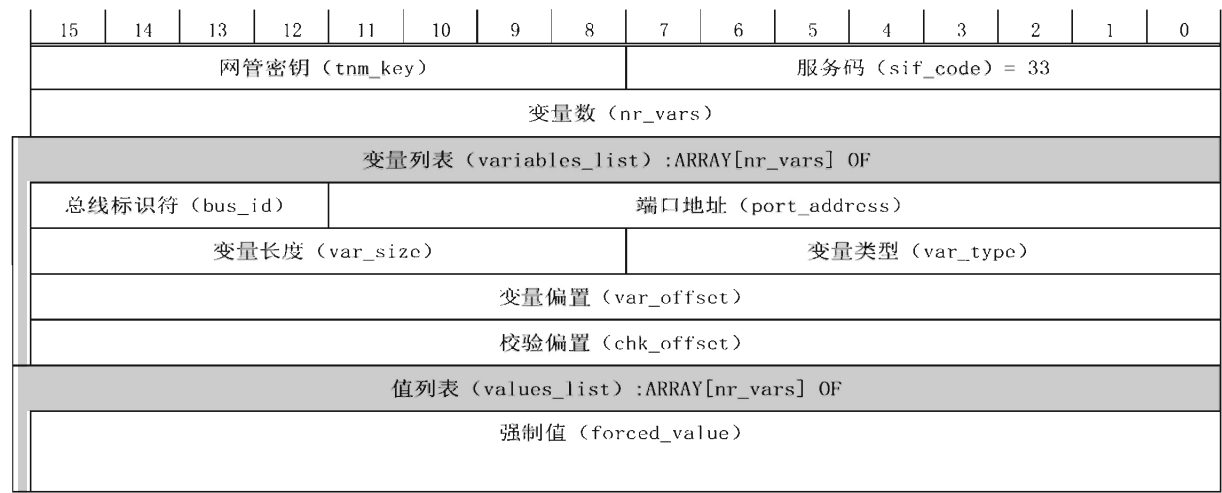


图 238 呼叫写强制变量

```
Call_Write_Force_Variables := RECORD
{
  nr_vars          UNSIGNED16,      —— 列表中变量数
  variables_list   ARRAY [nr_vars] OF
  {
    bus_id         UNSIGNED4,      —— 通信存储器标识符
    port_address   WORD12,        —— 通信存储器端口地址
    var_size       UNSIGNED8,      —— 简单类型:长度以位计;构造类型:长度以 6.2(RTP)的元素
                                   个数计
    var_type       UNSIGNED8,      —— 依照 6.2(RTP)定义的变量类型
    var_offset     UNSIGNED16,     —— 变量偏置
    chk_offset     UNSIGNED16     —— 检验变量偏置(未使用时为“FFFF”H)
  },
  values_list     ARRAY ALIGN16[nr_vars] OF
  {
    forced_value   ANY            —— PV_NAME 给定格式的变量值,作为过程数据通过总线发送
  }
}
```

8.4.4.4.3 应答写强制变量

应答写强制变量格式见图 239。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 33							

图 239 应答写强制变量

Reply_Write_Force_Variables ::= RECORD
{ }
——无参数

8.4.4.5 写解除强制变量

8.4.4.5.1 描述

对所有列出的变量解除强制。如本站所有变量已成功解除强制,则本服务应复位相应链路层状态的“frc”位和站状态的“frc”位。

8.4.4.5.2 呼叫写解除强制变量

呼叫写解除强制变量格式见图 240。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 35							
变量数 (nr_vars)															
变量列表 (variables_list) :ARRAY[nr_vars] OF															
总线标识符 (bus_id)				端口地址 (port_address)											
变量长度 (var_size)								变量类型 (var_type)							
变量偏置 (var_offset)															
校验偏置 (chk_offset)															

图 240 呼叫写解除强制变量

Call_Write_Unforce_Variables ::= RECORD
{
 nr_vars UNSIGNED8, ——列表中变量数
 variables_list ARRAY [nr_vars] OF
 {
 bus_id UNSIGNED4, ——通信存储器标识符
 port_address WORD12, ——通信存储器端口地址
 var_size UNSIGNED8, ——简单类型:长度以位计;构造类型:长度以 6.2(RTP)的元素个数计
 var_type UNSIGNED8, ——依照 6.2(RTP)定义的变量类型
 var_offset UNSIGNED16, ——变量偏置
 chk_offset UNSIGNED16 ——校验变量偏置(未使用时为“FFFF”H)
 }
}

8.4.4.5.3 应答写解除强制变量

应答写解除强制变量格式见图 241。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 35							

图 241 应答写解除强制变量

Reply_Write_Unforce_Variables ::= RECORD
{ } ——— 无参数

8.4.4.6 写全部变量解除强制

8.4.4.6.1 描述

对特定通信存储器中所有变量解除强制。

一旦所有通信存储器被解除强制,本服务如成功则清除通信存储器对应设备状态字和站状态中的“frc”位。

8.4.4.6.2 呼叫写全部变量解除强制

呼叫写全部变量解除强制格式见图 242。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 37							
ts0								链路集 (link_set)							
								ts15							

图 242 呼叫写全部变量解除强制

Call_Write_Unforce_All ::= RECORD
{
link_set Link_Set ——— 指示待解除强制的通信存储器的位组,偏移量 0 的位表示 ts0
}
}

8.4.4.6.3 应答写全部变量解除强制

应答写全部变量解除强制格式见图 243。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 37							

图 243 应答写全部变量解除强制

Reply_Write_Unforce_All ::= RECORD
{ } ——— 无参数

8.4.4.7 读变量绑定

8.4.4.7.1 描述

读取站支持的所有变量的绑定。顺序任意。

注：本服务支持内部设备变量与网络变量编排。

8.4.4.7.2 呼叫读变量绑定

呼叫读变量绑定格式见图 244。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 38							

图 244 呼叫读变量绑定

Call_Read_Variable_Bindings ::= RECORD
{ } ——— 无参数

8.4.4.7.3 应答读变量绑定

应答读变量绑定格式见图 245。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 38							
变量数 (nr_vars)															
变量列表 (variables_list) :ARRAY[nr_vars] OF															
变量名 (variable_name) :STRING32															
(CHARACTER8)								(CHARACTER8) 或00H							
变量属性 (var_properties)								特征周期 (individual_period)							
标准类型 (standard_type)															
总线标识符 (bus_id)				端口地址 (port_address)											
变量长度 (var_size)								变量类型 (var_type)							
变量偏置 (var_offset)															
校验偏置 (chk_offset)															

图 245 应答读变量绑定

Reply_Read_Variables_Bindings ::= RECORD
{
 nr_vars UNSIGNED16, —— 列表中变量数
 variables_list ARRAY[nr_vars]OF
 {
 variables_name STRING32, —— 变量本地名
 var_properties BITSET8
 {
 bnd (7), —— 1: 变量绑定, 0: 无动作
 phl (6), —— 1: 物理地址(存储器), 0: 逻辑地址(端口)
 reg (1), —— 1: 常规变量, 0: 维护变量
 imp (0) —— 1: 输入变量, 0: 输出变量
 },
 individual_period UNSIGNED8, —— 变量特征周期, 1 ms 的 2 的幂次倍(例如: 4 = 16 ms)
 standard_type ENUM16, —— 应用定义的标准类型
 bus_id UNSIGNED4, —— 绑定变量的通信存储器
 port_address WORD12, —— 绑定变量的端口地址
 var_size UNSIGNED8, —— 简单类型: 长度以位计; 构造类型: 长度以 6.2(RTP)的元
 素个数计
 var_type UNSIGNED8, —— 依照 6.2(RTP)定义的变量类型
}

var_offset

UNSigned16,

——数据集内变量偏置

chk_offset

UNSigned16

——校验变量偏置(未使用时为“FFFF”H)

}

}

8.4.4.8 写变量绑定

8.4.4.8.1 描述

绑定或解除绑定一些变量到特定通信存储器或端口地址。

注：本服务支持 ROSIN 配置。

8.4.4.8.2 呼叫写变量绑定

呼叫写变量绑定格式见图 246。

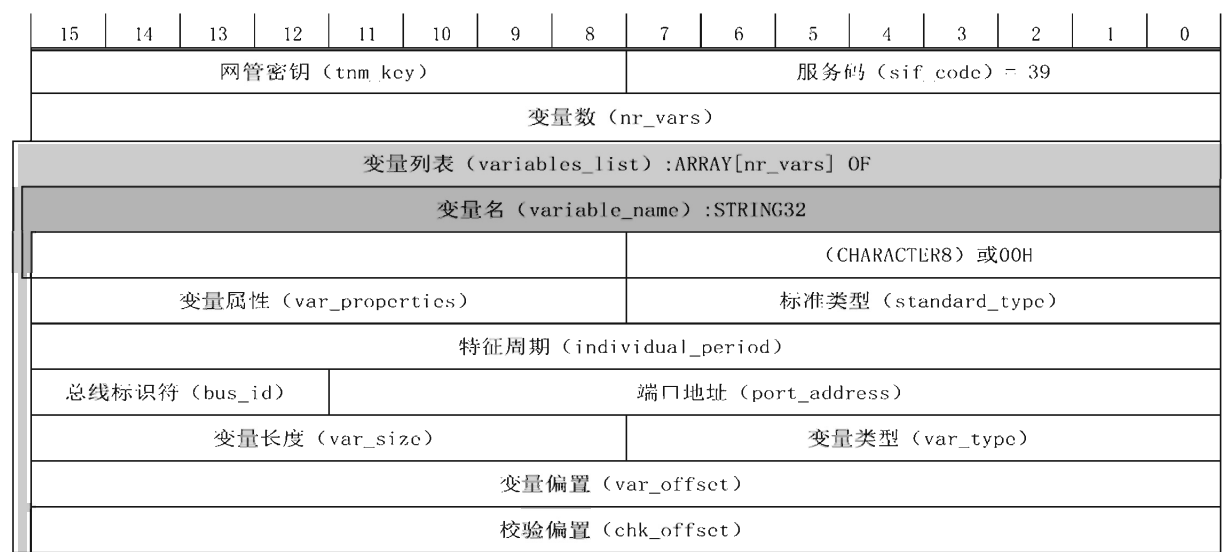


图 246 呼叫写变量绑定

Call_Write_Variable_Bindings ::= RECORD

{

nr_vars

UNSigned16

——拟绑定的变量数

variables_list

ARRAY[nr_vars]OF

{

variable_name

STRING32,

——拟绑定或解除绑定变量的本地名

var_properties

BITSET8

{

bnd

(7),

——1:变量绑定,0:无操作

pbl

(6),

——1:物理地址(存储器),0:逻辑地址(端口)

reg

(1),

——1:常规变量,0:维护变量

imp

(0)

——1:输入变量 0:输出变量

}

standard_type

ENUM8,

——应用定义的标准类型

individual_period

UNSigned16,

——变量特征周期,以 ms 为单位

standard_type

ENUM8,

——类型的应用代码

bus_id

UNSigned4,

——绑定或解除绑定变量的通信存储器

}

309

port_address	WORD12,	——绑定变量的端口(若端口为 0 则解除绑定)
var_size	UNSIGNED8,	——简单类型:长度以位计;构造类型:长度以 6.2(RTP)的元素 个数计
var_type	UNSIGNED8,	——依照 6.2(RTP)定义的变量类型
var_offset	UNSIGNED16,	——变量偏置
chk_offset	UNSIGNED16	——校验变量偏置(未使用时为“FFFF”H)
}		
}		

8.4.4.8.3 应答写变量绑定

应答写变量绑定格式见图 247。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 39							

图 247 应答写变量绑定

Reply_Write_Variable_Bindings ::= RECORD
{ } ——无参数

8.4.4.9 写端口连接

8.4.4.9.1 描述

连接通信存储器的端口到特定的输入或输出(2 类站)。

8.4.4.9.2 呼叫写端口连接

呼叫写端口连接格式见图 248。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 29							
端口数 (nr_ports)															
端口指针列表 (port_point_list) :ARRAY[nr_ports] OF															
总线标识符 (bus_id)				端口地址 (port_address)											
指针 (point)															
滤波 (filter)															
增益 (gain)															
偏置 (offset)															

图 248 呼叫写端口连接

Call_Write_Attach_Port ::= RECORD
{
nr_ports UNSIGNED16, ——列表中端口数
port_point_list ARRAY [nr_ports] OF
 {
 ds_name RECORD
 {
 bus_id UNSIGNED4(0...15), ——总线标识符(通信存储器),缺省为 0

port_address

WORD12

——12 位端口地址(应能被 2 整除)

}

port_descripor

RECORD

——指针描述符,包含:

{

point

UNSIGNED16,

——I/O 指针 16 位标识符

filter

UNSIGNED16,

——以 10 ms 计的滤波时间常数(不使用时为 0)

gain

UNSIGNED16,

——模拟量增益值

offset

INTEGER16

——模拟量偏置值

}

}

}

8.4.4.9.3 应答写端口连接

应答写端口连接格式见图 249。

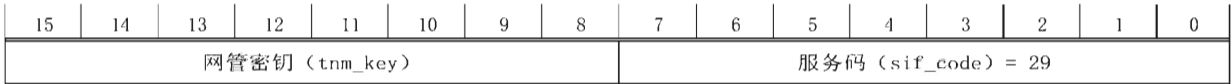


图 249 应答写端口连接

Reply_Write_Attach_Port ::= RECORD

{}

——无参数

8.4.5 信使服务

8.4.5.1 读信使状态

8.4.5.1.1 描述

读取信使状态及其统计计数器。

8.4.5.1.2 呼叫读信使状态

呼叫读信使状态格式见图 250。

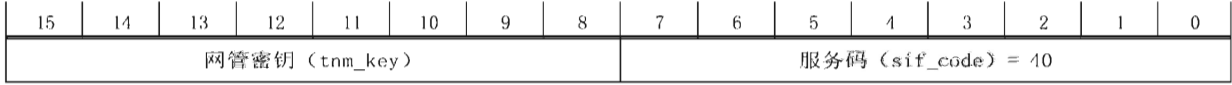


图 250 呼叫读信使状态

Call_Read_Messenger_Status ::= RECORD

{}

——无参数

8.4.5.1.3 应答读信使状态

应答读信使状态格式见图 251。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 10							
保留 (reserved1)															
信使名 (messenger_name) :STRING32															
								(CHARACTER8) 或00H							
发送超时 (sent_time_out)								存活超时 (alive_time_out)							
确认超时 (ack_time_out)								信用 (credit)							
保留 (reserved2)								包长 (packet_size)							
事例数 (instances)								多播窗口 (multicast_window)							
发送消息数 (messages_sent)															
接收消息数 (messages_received)															
发送包数 (packets_sent)															
重发包数 (packets_retries)															
重发多播数 (multicast_retries)															

图 251 应答读信使状态

Reply_Read_Messenger_Status ::= RECORD

{

 reserved1 WORD16(=0), ——保留

 messenger_name STRING32, ——信使软件版本,宜采用格式:xxxx-Vz.z-dd.mm.yy

 send_time_out UNSIGNED8, ——生产者重试前的超时,以 64 ms 计

 alive_time_out UNSIGNED8, ——消费者解联前的超时,以 s 计

 ack_time_out UNSIGNED8, ——应答者确认所有接收数据包前的超时,以 64 ms 计

 credit UNSIGNED8, ——生产者收到应答前可发送的数据包数

 reserved2 WORD8(=0), ——保留

 packet_size UNSIGNED8, ——以八位位组计的包长度

 instances UNSIGNED8, ——每个应答者支持的事例数

 multicast_window UNSIGNED8, ——多播机制的窗口长度。若为 0,则不支持多播

 messenger_sent UNSIGNED32, ——用于统计本站发送消息计数(循环计数)

 messenger_received UNSIGNED32, ——用于统计本站接收消息计数(循环计数)

 packet_sent UNSIGNED32, ——用于统计本站发送包计数(循环计数)

 packet_retries UNSIGNED32, ——用于统计单播协议重发包计数(循环计数)

 multicast_retries UNSIGNED32 ——用于统计多播协议重发包计数(循环计数)

}

8.4.5.2 写信使控制

8.4.5.2.1 描述

设置信使参数。

8.4.5.2.2 呼叫写信使控制

呼叫写信使控制格式见图 252。

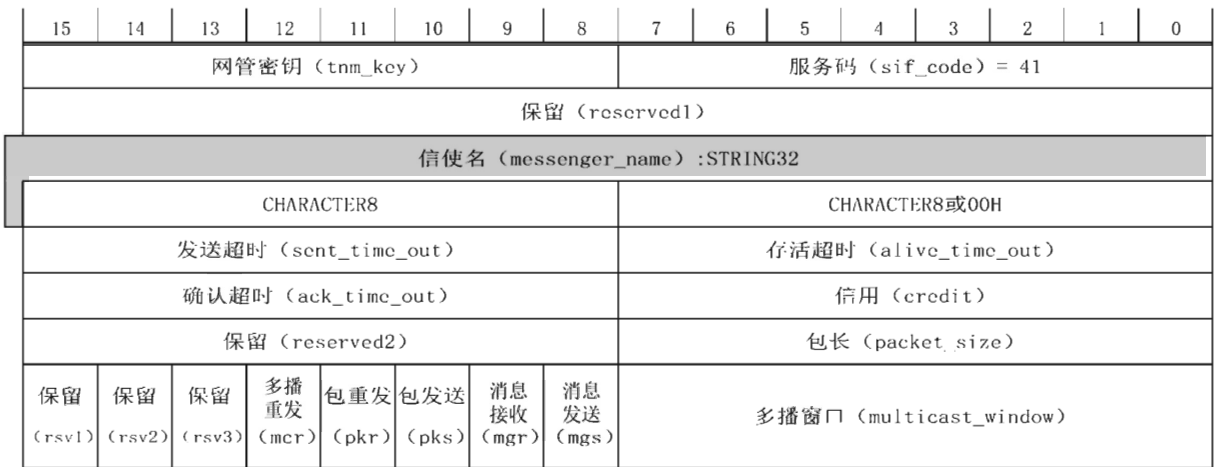


图 252 呼叫写信使控制

```
Call_Write_Messenger_Control ::= RECORD
{
    reserved1          WORD16(=0),      ——保留
    messenger_name     STRING32,         ——信使软件版本,宜采用格式:xxxx-Vz.z-dd.mm.yy
    send_time_out      UNSIGNED8,        ——生产者重试前的超时,以 64 ms 计
    alive_time_out     UNSIGNED8,        ——消费者解联前的超时,以 s 计
    ack_time_out       UNSIGNED8,        ——应答者确认所有接收数据包前的超时,以 64 ms 为单位计
    credit             UNSIGNED8,        ——生产者收到应答前可发送的数据包数
    reserved2         WORD8(=0),         ——保留
    packet_size        UNSIGNED8,        ——以八位位组计的包长度
    clear_counter      BITSET8           ——清除下列计数器:
    {
        rsv1,          ——保留
        rsv2,          ——保留
        rsv3,          ——保留
        mcr,           ——多播重发计数器
        pkr,           ——包重发计数器
        pks,           ——包发送计数器
        mgr,           ——消息接收计数器
        mgs,           ——消息发送计数器
    },
    multicast_windows  UNSIGNED8         ——多播机制的窗口长度。若为 0,则不支持多播
}
```

8.4.5.2.3 应答写信使控制

应答写信使控制格式见图 253。



图 253 应答写信使控制

```
Reply_Write_Messenger_Control ::= RECORD
```

{} ——— 无参数

8.4.5.3 读功能索引

8.4.5.3.1 描述

读站中功能索引。

8.4.5.3.2 呼叫读功能索引

呼叫读功能索引格式见图 254。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 42							

图 254 呼叫读功能索引

Call_Read_Function_Directory ::= RECORD
{}

8.4.5.3.3 应答读功能索引

应答读功能索引格式见图 255。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 42							
保留 (reserved1)								功能数 (nr_functions)							
功能列表 (function_list) :ARRAY[nr_functions] OF															
功能标识符 (function_id)								站标识符 (station_id)							

图 255 应答读功能索引

Reply_Read_Function_Directory ::= RECORD
{
 reserved1 WORD8 (=0), ——保留
 nr_functions UNSIGNED8, ——列表中项数
 function_list ARRAY [nr_functions] OF
 {
 function_id UNSIGNED8, ——功能标识符
 station_id UNSIGNED8 ——相应站标识符
 }
}

8.4.5.4 写功能索引

8.4.5.4.1 描述

写站中功能索引。

8.4.5.4.2 呼叫写功能索引

呼叫写功能索引格式见图 256。

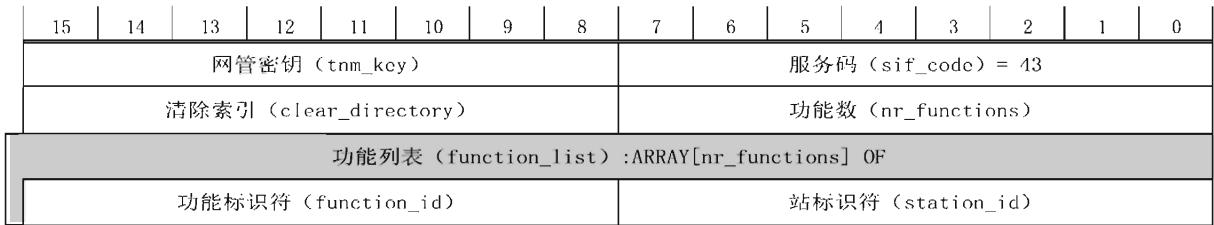


图 256 呼叫写功能索引

```
Call_Write_Function_Directory ::= RECORD
{
  clear_directory      ENUM8
  {
    REPLACE           (0),           ——不清除,只替代项
    CLEARFIRST        (1)           ——插入前清除索引
  },
  nr_functions         UNSIGNED8,     ——列表中项数
  function_list        ARRAY [nr_functions] OF
  {
    function_id       UNSIGNED8,     ——功能标识符
    station_id        UNSIGNED8      ——相应站标识符
  }
}
```

8.4.5.4.3 应答写功能索引

应答写功能索引格式见图 257。

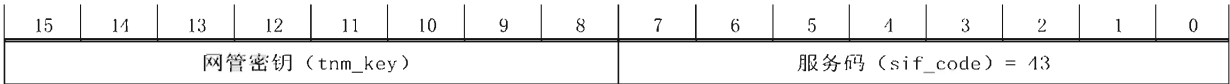


图 257 应答写功能索引

```
Reply_Write_Function_Directory ::= RECORD
{ }           ——无参数
```

8.4.5.5 读站索引

8.4.5.5.1 描述

读站中站索引(若存在)。

8.4.5.5.2 呼叫读站索引

呼叫读站索引格式见图 258。

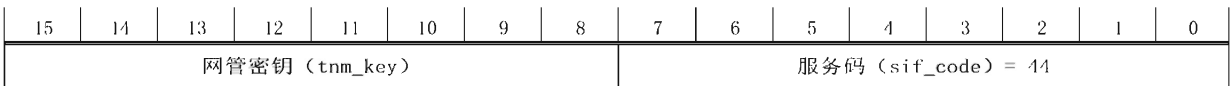


图 258 呼叫读站索引

```
Call_Read_Station_Directory ::= RECORD
```

{ } ——— 无参数

8.4.5.5.3 应答读站索引

应答读站索引格式见图 259。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 44							
保留 (reserved1)								站数 (nr_stations)							
站列表 (station_list) :ARRAY[nr_stations] OF															
站标识符 (station_id)								下一站标识符 (next_station_id)							
总线标识符 (bus_id)								保留 (reserved2)							
设备地址 (device_address)															

图 259 应答读站索引

```
Reply_Read_Station_Directory ::= RECORD
{
  reserved1      WORD8 (=0),      ——— 保留
  nr_stations    UNSIGNED8,      ——— 列表中站数
  station_list   ARRAY [nr_stations] OF
  {
    station_id    UNSIGNED8,      ——— 站标识符
    next_station_id  UNSIGNED8,    ——— 下一站站标识符
    bus_id        UNSIGNED8 (0...15), ——— 下一站所在链路标识符
    reserved2     WORD8 (=0),      ——— 保留
    device_address UNSIGNED16      ——— 携带下一站的设备地址
  }
}
```

8.4.5.6 写站索引

8.4.5.6.1 描述

写站中站索引(若存在)。

8.4.5.6.2 呼叫写站索引

呼叫写站索引格式见图 260。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 45							
清除索引 (clear_directory)								站数 (nr_stations)							
站列表 (station_list) :ARRAY[nr_stations] OF															
站标识符 (station_id)								下一站标识符 (next_station_id)							
总线标识符 (bus_id)								保留 (reserved1)							
设备地址 (device_address)															

图 260 呼叫写站索引

```
Call_Write_Station_Directory ::= RECORD
{
  clear_directory      ENUM8
  {
    REPLACE      (0),          —— 不清除,只替代项
    CLEARFIRST   (1)          —— 插入前清除索引
  },
  nr_stations         UNSIGNED8,      —— 列表中站数
  station_list        ARRAY [nr_sations] OF
  {
    station_id      UNSIGNED8,      —— 站标识符
    next_station_id UNSIGNED8,      —— 下一站站标识符
    bus_id          UNSIGNED8 (0...15), —— 下一站所在链路标识符
    reserved1       WORD8 (=0),     —— 保留
    device_address  UNSIGNED16      —— 携带下一站的设备地址
  }
}
```

8.4.5.6.3 应答写站索引

应答写站索引格式见图 261。

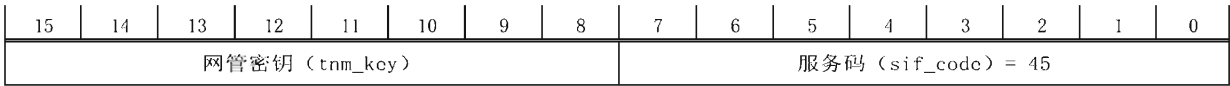


图 261 应答写站索引

```
Reply_Write_Station_Directory ::= RECORD
{ }          —— 无参数
```

8.4.5.7 读组索引

8.4.5.7.1 描述

读站中组索引(若存在)。

8.4.5.7.2 呼叫读组索引

呼叫读组索引格式见图 262。

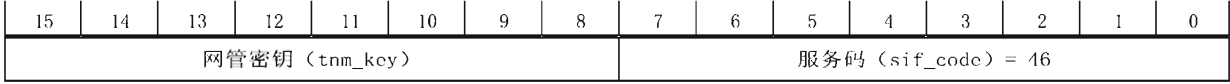


图 262 呼叫读组索引

```
Call_Read_Group_Directory ::= RECORD
{ }          —— 无参数
```

8.4.5.7.3 应答读组索引

应答读组索引格式见图 263。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 46							
g7	g6	g5	g4	g3	g2	g1	g0	g15	g14	g13	g12	g11	g10	g9	g8
g23															g24
g39															g40
g55								g63							g56

图 263 应答读组索引

Reply_Read_Group_Directory ::= RECORD
{
 group_list BITSET64 ——若站属于可能的 64 个组之一则置位相应位。组 0 偏置为 0
}

8.4.5.8 写组索引

8.4.5.8.1 描述

写站中组索引(若存在)。

8.4.5.8.2 呼叫写组索引

呼叫写组索引格式见图 264。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 47							
g7	g6	g5	g4	g3	g2	g1	g0	g15	g14	g13	g12	g11	g10	g9	g8
g23															g24
g39															g40
g55								g63							g56

图 264 呼叫写组索引

Call_Write_Group_Directory ::= RECORD
{
 group_list BITSET64 ——每位代表站可能属于的 64 个组之一。位 0 标识组 0
}

8.4.5.8.3 应答写组索引

应答写组索引格式见图 265。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 47							

图 265 应答写组索引

Reply_Write_Group_Directory ::= RECORD
{ } ——无参数

8.4.5.9 读节点索引

8.4.5.9.1 描述

读节点中节点索引(若存在)。

8.4.5.9.2 呼叫读节点索引

呼叫读节点索引格式见图 266。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 48							

图 266 呼叫读节点索引

Call_Read_Node_Directory ::= RECORD
{ } ———无参数

8.4.5.9.3 应答读节点索引

应答读节点索引格式见图 267。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 48							
保留 (reserved1)								节点数 (nr_nodes)							
节点列表 (nodes_list) :ARRAY[nr_nodes] OF															
节点地址 (node_address)								保留 (reserved2)							
设备地址 (device_address)															

图 267 应答读节点索引

Reply_Read_Node_Directory ::= RECORD
{
 reserved1 WORD8 (=0), ——保留
 nr_nodes UNSIGNED8, ——列表中节点数
 nodes_list ARRAY [nr_nodes] OF
 {
 node_address UNSIGNED8, ——8 位节点地址
 reserved2 WORD8 (=0), ——保留
 device_address UNSIGNED16 ——节点设备地址
 }
}

8.4.5.10 写节点索引

8.4.5.10.1 描述

写节点中节点索引(若存在)。

8.4.5.10.2 呼叫写节点索引

呼叫写节点索引格式见图 268。

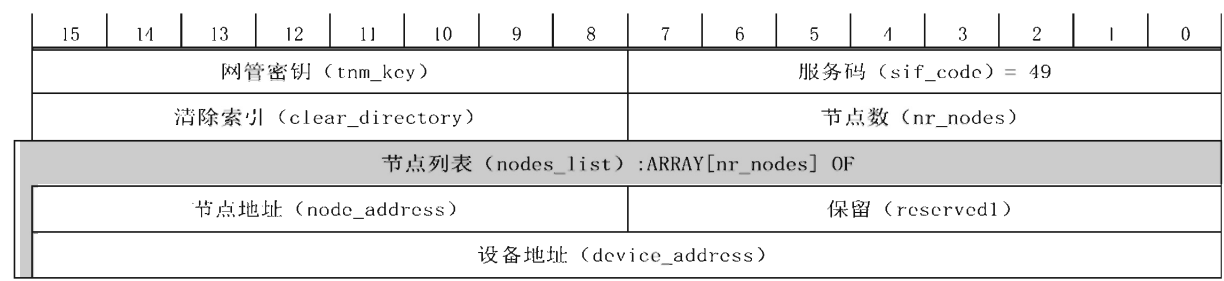


图 268 呼叫写节点索引

```
Call_Write_Node_Directory ::= RECORD
{
  clear_directory      ENUM8
  {
    REPLACE           (0),           ——不清除,仅替代项
    CLEARFIRST        (1)           ——插入前清除索引
  },
  nr_nodes            UNSIGNED8,      ——列表中节点数
  nodes_list          ARRAY [nr_nodes] OF
  {
    node_address      UNSIGNED8,      ——8 位节点地址
    reserved1         WORD8 (=0),     ——保留
    device_address    UNSIGNED16      ——节点设备地址
  }
}
```

8.4.5.10.3 应答写节点索引

应答写节点索引格式见图 269。

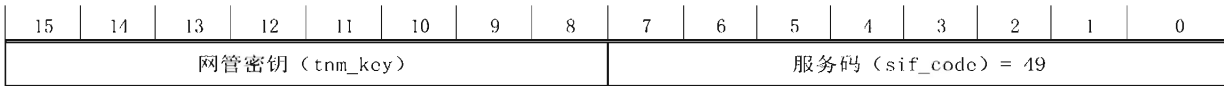


图 269 应答写节点索引

```
Reply_Write_Node_Directory ::= RECORD
{ }           ——无参数
```

8.4.6 域服务

8.4.6.1 读存储器

8.4.6.1.1 描述

本服务在一次操作中读多个存储区,每个区由若干个相同的连续项组成,每个项长度为八位位组的整数倍,每个八位位组有一个地址。

地址对齐排列应规定为:

- 若项长度为 1,则存储区可从奇地址或偶地址开始;
- 若项长度为 2,则存储区应从偶地址开始;
- 若项长度为 4,则存储区应从能被 4 整除的地址开始。

8.4.6.1.2 呼叫读存储器

呼叫读存储器格式见图 270。

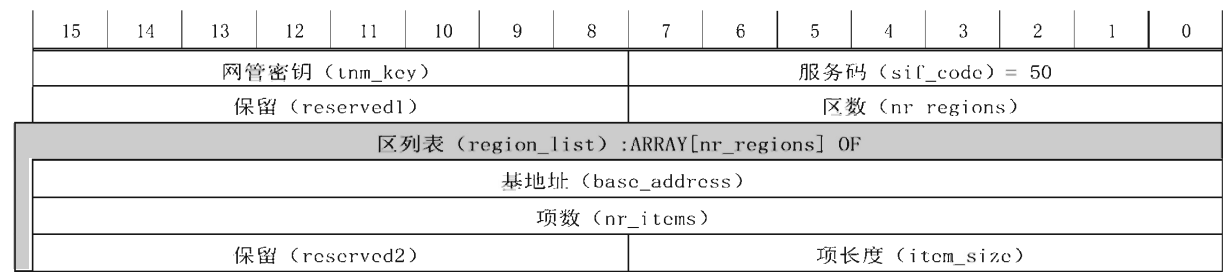


图 270 呼叫读存储器

```
Call_Read_Memory ::= RECORD
{
    reserved1      WORD8 (=0),
    nr_regions     UNSIGNED8,      ——待读取的区数
    region_list    ARRAY [nr_regions] OF
    {
        base_address  UNSIGNED32,  ——区基地址
        nr_items     UNSIGNED16,   ——区长度,以项长度计
        reserved2    WORD8 (=0),   ——保留
        item_size     UNSIGNED8     ——以八位位组计的每项长度,允许值为 1、2、4
    }
}
```

8.4.6.1.3 应答读存储器

应答读存储器格式见图 271。

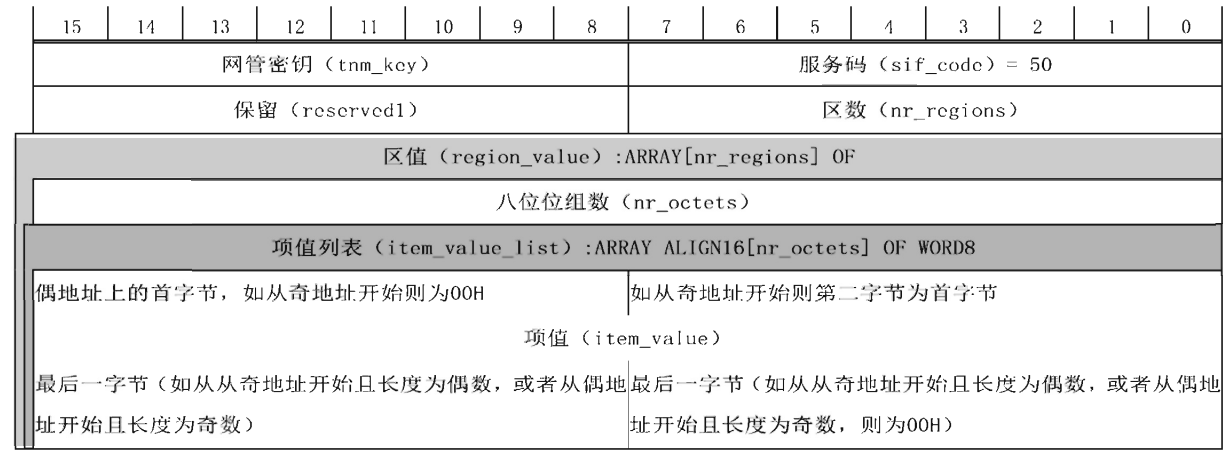


图 271 应答读存储器

```
Reply_Read_Memory ::= RECORD
{
    reserved1      WORD8 (=0),      ——保留
    nr_regions     UNSIGNED8,      ——实际读取的区数
    region_values   ARRAY [nr_regions] OF
    {
        ——区值数组,每项包含:
```

nr_octets	UNSIGNED16,	—— 传送字段中八位位组数, 包含填充
item_value_list	ARRAY ALIGN16 [nr_octets] OF	
{		—— 长度为 nr_octets 的数组, 每项包含:
item_value	WORD8	—— 以连续顺序发送的项值。若区开始于奇地址, 则第一个八位位组应是填充八位位组
}		
}		

注：位于存储器偶地址的八位位组总是以消息中偶数八位位组偏置发送。若存储区开始于奇地址，则第一个发送的元素是无意义的。相反，若存储区开始于偶地址且八位位组数为奇数，则最后一个八位位组是无意义的。若基地址是奇数且八位位组数为偶数，则第一个和最后一个八位位组是无意义的。字段“nr_octets”含有有效发送的八位位组数，仅当开始于偶地址且八位位组数为偶数时该数才等于区长度。

8.4.6.2 写存储器

8.4.6.2.1 描述

本服务在一次操作中写多个存储区,每个区由若干个给定长度的相同项组成。

8.4.6.2.2 呼叫写存储器

呼叫写存储器格式见图 272。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 51							
保留 (reserved1)								区数 (nr_regions)							
区列表 (region_list):ARRAY[nr_regions] OF															
基地址 (base_address)															
项数 (nr_items)															
保留 (reserved2)								项长度 (item_size)							
区值列表 (region_value_list):ARRAY[nr_regions] OF															
项值列表 (item_value_list):ARRAY[nr_items] OF															
nr_octets为偶数时为第一个字节, 否则无意义								nr_octets为奇数时为第一个字节							
偶八位位组								奇八位位组							
最后一个字节 (nr_octets为奇数时)								最后一个字节 (若nr_octets为奇数则为00H)							

图 272 呼叫写存储器

```

Call_Write_Memory ::= RECORD
{
    reserved1          WORD8 (=0),           —— 保留
    nr_regions          UNSIGNED8,           —— 待写区数
    region_list         ARRAY [nr_regions] OF
    {
        base_address    UNSIGNED32,          —— 区基地址(可能是奇数)
        nr_items         UNSIGNED16,         —— 区长度,以项长度整数倍计
        reserved2        WORD8 (=0),         —— 保留
        item_size         UNSIGNED8          —— 以八位位组计的每项长度,允许值为 1、2、4
    },

```



```
region_value_list    ARRAY [nr_regions] OF
{
    ———区值数组,每项包含:
    item_value_list    ARRAY ALIGN16 [nr_items] OF
    {
        ———nr_items 项项值数组,每项包含:
        ONE_OF [item_size]
        ———连续顺序发送
        {
            1:          WORD8,          ———偶地址的八位位组首先发送
            2:          WORD16,         ———逐字写入
            4:          WORD32          ———逐双字写入
        }
    }
}
```

注：位于存储器偶地址的八位位组总是以消息中偶数八位位组偏置发送。若存储区开始于奇地址,则第一个发送的元素是无意义的。相反,若存储区开始于偶地址且八位位组数为奇数,则最后一个八位位组是无意义的。若基地址是奇数且八位位组数为偶数,则第一个和最后一个八位位组是无意义的。字段“nr_octets”含有有效发送的八位位组数,仅当开始于偶地址且八位位组数为偶数时该数才等于区长度。

8.4.6.2.3 应答写存储器

应答写存储器格式见图 273。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 51							

图 273 应答写存储器

Reply_Write_Memory ::= RECORD
{ } ———无参数

8.4.6.3 下载设置

8.4.6.3.1 描述

下载设置准备对不同段中的一个域进行下载操作。
若在两段连续加载时间间隔大于 16 s,则代理者应复位该站。

8.4.6.3.2 呼叫写下载设置

呼叫写下载设置格式见图 274。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 53							
保留 (reserved1)								下载命令 (download_command)							
保留 (reserved2)								下载超时 (download_time_out)							
保留 (reserved3)								域数 (nr_domains)							
域列表 (domain_list) :ARRAY[nr_domains] OF															
基地址 (base_address)															
域长度 (domain_size)															

图 274 呼叫写下载设置

```
Call_Write_Download_Setup ::= RECORD
{
    reserved1                WORD8 (=0),
    download_command         ENUM8,
    {
        DNLD_PREPARE        (0),          —— 强迫站启动下载程序。忽略其他参数
        DNLD_CHECK_ONLY     (1),          —— 检查下载参数是否合法(根据块、基址、长度和分区),但不影响站
        DNLD_START_ERASE    (2),          —— 若参数有效,则无效化域,擦除存储器为下载准备
        DNLD_START_NOERASE  (3),          —— 若参数有效,则无效化域,为下载准备存储器
        DNLD_TERMINATE_BOOT (4),          —— 终止下载并重新启动
        DNLD_TERMINATE_NOBOOT (5),        —— 终止下载并停止超时计数器。等待进一步服务呼叫
        DNLD_VERIFY         (6)          —— 为该域呼叫校验过程
    },
    reserved2                WORD8 (=0),
    download_time_out         UNSIGNED8,   —— 以秒计的两段装载间允许时间间隔(最大 16s)
    reserved3                WORD8 (=0),
    nr_domains               UNSIGNED8,    —— 设置的域数
    domain_list              ARRAY [nr_domains] OF
    {
        base_address        WORD32,       —— 代理者地址空间中域基地址
        domain_size         WORD32        —— 以八位位组计的域长度
    }
}
```

8.4.6.3.3 应答写下载设置

应答写下载设置格式见图 275。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 53							
最大段长度 (max_segment_size)															
保留 (reserved1)								域数 (nr_domains)							
设置结果列表 (setup_result_list) :ARRAY[nr_domains] OF															
第一个设置结果								第二个设置结果							
最后一个设置结果 (若nr_domains为奇)								最后一个设置结果 (若nr_domains为偶), 否则为00H							

图 275 应答写下载设置

```
Reply_Write_Download_Setup ::= RECORD
{
    max_segment_size         UNSIGNED32,   —— 下载/上传缓冲区最大长度
    reserved1               WORD8 (=0),
    nr_domains               UNSIGNED8,    —— 呼叫中 nr_domains 的拷贝
    setup_result_list        ARRAY [nr_domains] OF
    setup_result             ENUM8
```

```
{
  DOMAIN_OK                (0),           ——域成功设置
  DOMAIN_BAD_BASE_ADDR     (1),           ——无效的域基地址
  DOMAIN_BAD_SIZE          (2),           ——无效的域长度
  DOMAIN_ERASE_ERR         (3),           ——域不可擦除
  DOMAIN_WRITE_ERR         (4),           ——域不可写
  DOMAI_BAD_CHECKSUM       (5),           ——不正确的校验和
}
```

8.4.6.4 段下载

8.4.6.4.1 描述

本服务向由写下载设置(Write_Download_Setup)打开的域发送指定长度的段。

8.4.6.4.2 呼叫写下载段

呼叫写下载段格式见图 276。

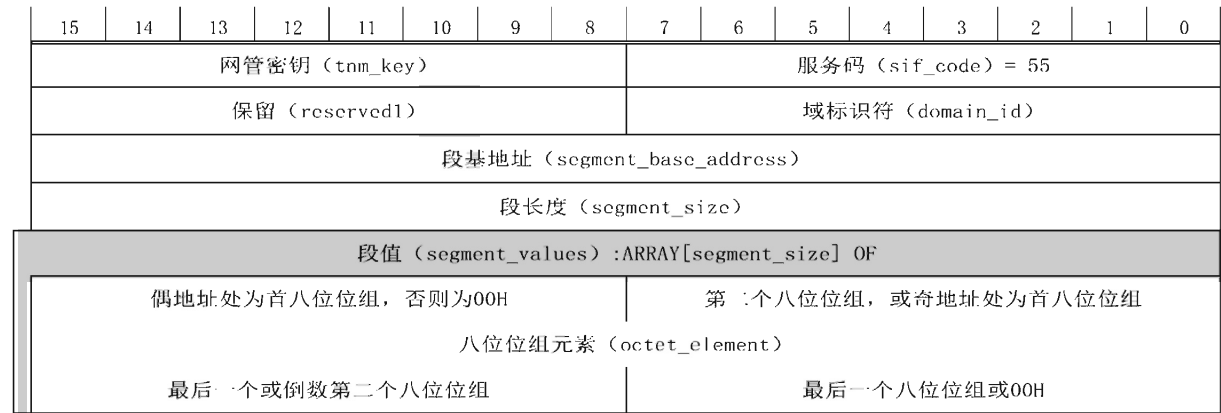


图 276 呼叫写下载段

```
Call_Write_Download_Segment ::= RECORD
{
  reserved1          WORD8 (=0),      ——填充
  domain_id          UNSIGNED8,       ——标识域(在前一“呼叫写下载设置”域列表中域的数组索引)
  segment_base_address  UNSIGNED32,   ——段基地址(可能是奇数)
  segment_size       UNSIGNED32,     ——用八位位组计的段长度
  segment_values     ARRAY [segment_size] OF
  {
    octet_element     WORD8           ——八位位组列表
  }
}
```

8.4.6.4.3 应答写下载段

应答写下载段格式见图 277。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 55							

图 277 应答写下载段

Reply_Write_Download_Segment ::= RECORD
{ } ——— 无参数

8.4.7 任务服务

8.4.7.1 读任务状态

8.4.7.1.1 描述

本服务读取装载在站中的任务名和任务状态。

8.4.7.1.2 呼叫读任务状态

呼叫读任务状态格式见图 278。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 60							

图 278 呼叫读任务状态

Call_Read_Task_Status ::= RECORD
{ } ——— 无参数

8.4.7.1.3 应答读任务状态

应答读任务状态格式见图 279。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 60							
保留 (reserved1)								任务数 (nr_tasks)							
任务列表 (tasks_list) :ARRAY[nr_tasks] OF															
任务名 (task_name) :STRING16															
								CHARACTER8或00H							
优先级 (priority)								状态 (status)							
CPU负荷 (cpu_load)															
堆栈边界 (stack_margin)															
任务描述 (task_comment) :STRING26															
								CHARACTER8或00H							

图 279 应答读任务状态

Reply_Read_Tasks_Status ::= RECORD
{
 reserved1 WORD8 (=0), —— 填充
 nr_tasks UNSIGNED8, —— 已返回的任务状态描述数
 tasks_list ARRAY [nr_tasks] OF

{		
task_name	STRING16,	——以字符串表示的任务名或任务号
priority	UNSIGNED8,	——任务优先级(0 = 最高优先级)
status	ENUM8	——任务状态
{		
READY	(0),	
SUSPENDED	(1),	
PENDING	(2),	
RUNNING	(3),	
FAULTY	(4)	
},		
cpu_load	UNSIGNED16,	——以%表示的该任务导致的 CPU 负荷(0~100%),其他值表示不支持 CPU 负荷测量
stack_margin	UNSIGNED16,	——该任务的堆栈边界(“FFFFH”=不支持该服务)
task_comment	STRING26	
}		
}		

8.4.7.2 写任务控制

8.4.7.2.1 描述

停止或启动所有任务。
当请求停止任务时,站状态中的“dnr”(设备未就绪)位和所有 MVB 链路层的设备状态字中的“dnr”位(设备未准备好)应置位;在启动成功后该位应清除置位。

8.4.7.2.2 呼叫写任务控制

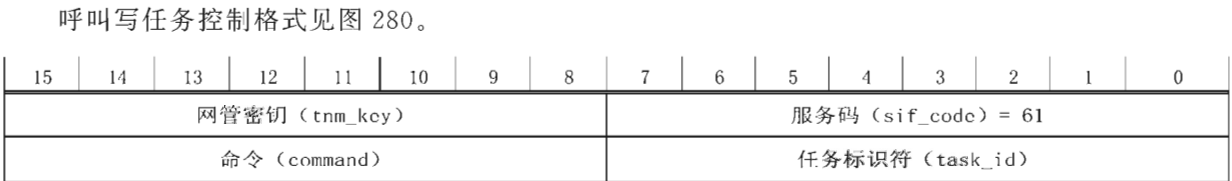


图 280 呼叫写任务控制

Call_Write_Tasks_Control ::= RECORD		
{		
command	ENUM8	
{		
STOP_TASK	(0),	——停止所有任务
START_TASK	(1)	——启动所有任务
},		
task_id	UNSIGNED8	——标识任务(“应答读任务状态”任务列表中任务的数组索引)以停止或启动任务,“FF”H:启动/停止所有任务
}		

8.4.7.2.3 应答写任务控制

应答写任务控制格式见图 281。

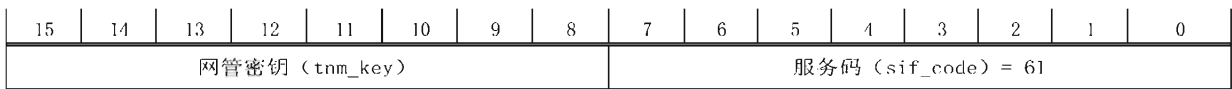


图 281 应答写任务控制

Reply_Write_Tasks_Control ::= RECORD
{ } ——— 无参数

8.4.8 时钟服务

8.4.8.1 读时钟

8.4.8.1.1 描述

读取选中站的时钟值。

8.4.8.1.2 呼叫读时钟

呼叫读时钟格式见图 282。

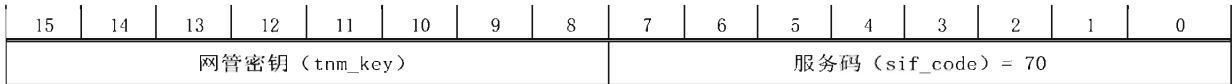


图 282 呼叫读时钟

Call_Read_Clock ::= RECORD
{ } ——— 无参数

8.4.8.1.3 应答读时钟

应答读时钟格式见图 283。

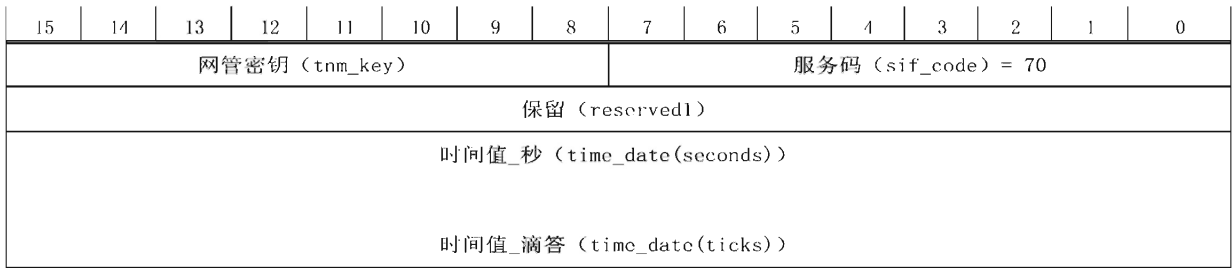


图 283 应答读时钟

Reply_Read_Clock ::= RECORD
{
 reserved WORD16 (=0), ——— 用于对齐
 time_date TIMEDATE48 ——— 以秒或滴答计的时间值
}

8.4.8.2 写时钟

8.4.8.2.1 描述

设置选中站的时钟值,格式见图 284。

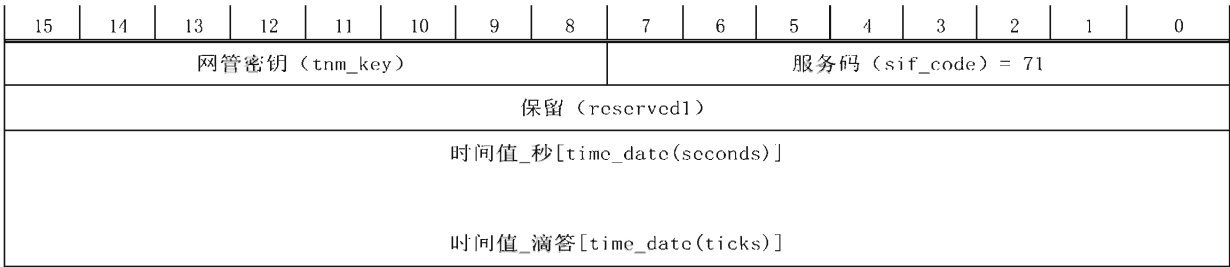


图 284 设置选中站的时钟值

8.4.8.2.2 呼叫写时钟

```
Call_Write_Clock ::= RECORD
{
    reserved1      WORD16 (=0),      ——用于对齐
    time_date      TIMEDATE48        ——以秒或滴答计的时间值
}
```

8.4.8.2.3 应答写时钟

应答写时钟格式见图 285。

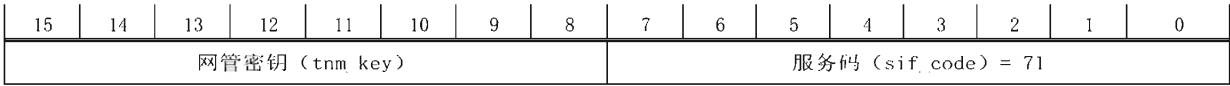


图 285 应答写时钟

```
Reply_Write_Clock ::= RECORD
{ }      ——无参数
```

8.4.9 记录服务

8.4.9.1 读记录描述

本服务读取记录中最后一个记录项。项的含义依赖于应用。索引号的处理在对象描述中解释。

8.4.9.2 呼叫读记录

呼叫读记录格式见图 286。

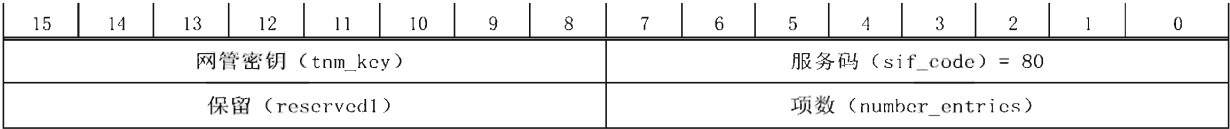


图 286 应答读记录

```
Call_Read_Journal ::= RECORD
{
    reserved1      WORD8 (=0),      ——保留
    number_entries UNSIGNED8        ——最多 255 项
}
```

8.4.9.3 应答读记录

应答读记录格式见图 287。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 80							
保留 (reserved1)								项数 (nr_entries)							
事件列表 (event_list) :ARRAY[number_entries] OF															
时间戳 (time_stamp) :TIMEDATE48															
文件名 (file_name) :STRING16															
(CHARACTER8)								00H							
行数 (line_number)															
保留 (reserved2)								事件类型 (event_type)							
事件描述 (event_description) :STRING78															
(CHARACTER8)								00H							

图 287 应答读记录

```
Reply_Read_Journal ::= RECORD
{
    reserved1          WORD8 (=0),      —— 保留
    number_entries     UNSIGNED8,       —— 已返回的项数
    event_list         ARRAY [number_entries] OF
    {
        time_stamp     TIMEDATE48,     —— 事件发生时的时间戳
        file_name       STRING16,       —— ANSI C 中由 __FILE__ 提供 (以 NULL 结尾的字符串)
        line_number     UNSIGNED16,     —— ANSI C 中由 __LINE__ 提供
        reserved2       WORD8 (=0),
        event_type      ENUM8          —— 事件类型
        {
            INFO        (0),
            WARNING      (1),
            ERROR        (2)
        },
        event_description STRING78      —— 事件描述 (以 NULL 结尾的字符串)
    }
}
```

8.4.10 读设备服务 (Read_Equipment)

8.4.10.1 描述

本服务读取指向存放所支持设备完整描述的存储域的指针。该数据结构的格式不属于本部分范围。

8.4.10.2 呼叫读设备

呼叫读设备格式见图 288。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 82							
保留 (reserved1)								保留 (reserved2)							

图 288 呼叫读设备

```
Call_Read_Equipment ::= RECORD
{
    reserved1      WORD8 (=0),      ——保留
    reserved2      WORD8 (=0)      ——保留
}
```

8.4.10.3 应答读设备

应答读设备格式见图 289。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
网管密钥 (tnm_key)								服务码 (sif_code) = 82							
保留 (reserved1)								项数 (number_entries)							
设备列表 (equipment_list) :ARRAY[number_entries] OF															
设备名 (equipment_name) :STRING32															
(CHARACTER8)								00H							
设备基地址 (equipment_root)															
设备描述符长度 (equipment_size)															

图 289 应答读设备

```
Reply_Read_Equipment ::= RECORD
{
    reserved1      WORD8 (=0),
    number_entries  UNSIGNED8,      ——已返回的项数
    equipment_list  ARRAY [number_entries] OF
    {
        equipment_name  STRING32,      ——标识设备
        equipment_root   UNSIGNED32,    ——域基地址
        equipment_size   UNSIGNED32     ——设备描述符长度
    }
}
```

8.5 接口过程

8.5.1 概述

接口过程分成一个管理者接口和一个代理者接口。

8.5.2 管理者接口

管理者接口的所有服务由两个通用过程提供,见表 168、表 169。

- 服务请求过程 mm_service_req;
- 服务确认过程 mm_service_conf。

管理者接口过程以 mm_xxx 为前缀。

表 168 服务请求过程

描述	呼叫远程服务	
语法	MM_RESULT mm_service_req (UNSIGNED8 station_id, const AM_ADDRESS * agent_adr, struct MM_CALL * mm_call);	
输入	station_id	本站站标识符
	agent_adr	代理者网络地址
	mm_call	mm_call 等同于管理呼叫消息正文格式(该结构格式取决于服务码)
结果	呼叫结果是在 8.4.1.5 中定义的错误码 MM_RESULT	

表 169 服务确认过程

描述	本服务确认过程 mm_service_conf 返回对管理者服务呼叫的结果。该过程可是一个轮询过程或指示过程	
语法	MM_RESULT mm_service_conf (UNSIGNED8 station_id, const AM_ADDRESS * agent_adr, struct MM_CALL * mm_reply);	
输入	station_id	本站站标识符
	agent_adr	代理者网络地址
	mm_reply	在 MM_RESULT 为 OK 时,此结构返回应答消息正文,否则无定义(该结构格式取决于服务码)
结果	呼叫结果是在 8.4.1.5 中定义的错误码 MM_RESULT	

8.5.3 代理者接口

8.5.3.1 描述

代理者接口过程不定义代理者和网络之间的接口,但定义代理者和站其他过程之间的接口。
代理者接口使用户可访问代理者,以轮询两种情况:
——修改(当前允许修改站吗?);
——停止(允许停止站吗?)。
代理者可能需要访问实时内核,以协调用户任务的执行。
代理者接口过程以 ma_xxx 为前缀。

8.5.3.2 代理者控制过程

8.5.3.2.1 过程 ma_ask_permission

过程 ma_ask_permission 见表 170。

表 170 过程 ma_ask_permission

描述	允许用户轮询有什么管理请求。 使用该功能的应用以 ma_permit 响应		
语法	MA_PERMISSION ma_ask_permission (UNSIGNED8 task_id);		
输入	task_id	调用的任务(由“应答读任务状态”任务列表中任务的数组索引标识)	
输出	—		
返回	MA_PERMISSION	0:MA_CHANGE_REQU, 1:MA_CHANGE_NOREQU, 2:MA_STOP_REQU, 3:MA_STOP_NOREQU	修改请求 不修改请求 停止请求 不停止请求

8.5.3.2.2 过程 ma_give_permission

过程 ma_give_permission 见表 171。

表 171 过程 ma_give_permission

描述	应用程序使用此过程响应修改请求,指示是否允许修改		
语法	MA_PERMISSION ma_give_permission (ENUM decision);		
输入	decision	0:MA_CHANGE_ALLOWED, 1:MA_CHANGE_DENIED, 2:MA_STOP_ALLOWED, 3:MA_STOP_DENIED	允许修改 不准许修改 任务可停止 任务不可停止
输出	—		

8.5.3.3 用户服务订阅

8.5.3.3.1 类型 MA_SERVICE_CALL

类型 MA_SERVICE_CALL 见表 172。

表 172 类型 MA_SERVICE_CALL

描述	用于给定服务呼叫的被调用过程类型声明,该服务呼叫返回应答消息所需参数	
语法	typedef void (* MA_SERVICE_CALL) (AM_ADDRESS * manager_address, void * call_msg_adr, UNSIGNED32 call_msg_size, void * * reply_msg_adr, UNSIGNED32 * reply_msg_size, MM_RESULT * agent_status);	
输入	manager_address	指向呼叫管理者全网络地址的指针
	call_msg_adr	指向待处理呼叫消息服务起始处的指针(tnm_key 字段)
	call_msg_size	以八位位组计的呼叫消息长度
	reply_msg_adr	指向返回的应答消息服务起始处的指针(tnm_key 字段)
	reply_msg_size	以八位位组计的应答消息长度
	agent_status	作为代理者状态将传送给管理者的结果

8.5.3.3.2 类型 MA_SERVICE_CLOSE

类型 MA_SERVICE_CLOSE 见表 173。

表 173 类型 MA_SERVICE_CLOSE

描述	用于关闭服务的被调用过程类型声明	
语法	typedef void (* MA_SERVICE_CLOSE) (void);	
输入	未定义	用户定义

8.5.3.3.3 过程 ma_subscribe

过程 ma_subscribe 见表 174。

表 174 过程 ma_subscribe

描述	当接收到用户定义服务呼叫时指示调用哪一个用户过程。先前分配的服务码无警示覆盖	
语法	MM_RESULT ma_subscribe (ENUM16 command, ENUM8 sif_code, MA_SERVICE_CALL service_call, MA_SERVICE_CLOSE service_close, void * service_desc);	

表 174 (续)

输入	command	0:订阅;1:解除订阅
	sif_code	用户服务码(≥128)
	service_call	MA_SERVICE_CALL 类型的过程变量,调用时执行服务
	service_close	当应答消息被完全发送(例如释放缓冲区)时代理者调用的过程
	service_desc	服务描述符,以“00”H 字符结尾的可显示字符串
返回	MM_RESULT	

8.5.3.4 重启过程订阅

8.5.3.4.1 类型 MA_STATION_RESTART

类型 MA_STATION_RESTART 见表 175。

表 175 类型 MA_STATION_RESTART

描述	用于在超时或重启命令后重新启动站所调用的过程类型声明。此过程可能无返回值
语法	<pre>typedef void (* MA_STATION_RESTART) ();</pre>

8.5.3.4.2 过程 ma_subscribe_restart

过程 ma_subscribe_restart 见表 176。

表 176 过程 ma_subscribe_restart

描述	当站复位或保留超时发生时指示调用哪个用户过程	
语法	<pre>MM_RESULT ma_subscribe_restart (MA_STATION_RESTART station_restart);</pre>	
输入	station_restart	当保留超时发生或接收到复位命令时,代理者将调用的过程
返回	MM_RESULT	

附 录 A
(资料性附录)

本部分与 IEC 61375-2-1:2012 相比的结构变化情况

本部分与 IEC 61375-2-1:2012 相比在结构上有较多调整,具体章条编号对照情况见表 A.1。

表 A.1 本部分与 IEC 61375-2-1:2012 的章条编号对照情况

本部分章条编号	对应的 IEC 61375-2-1:2012 章条编号
—	3.1.3、3.1.9、3.1.13、3.1.18、3.1.36、3.1.48、3.1.49、3.1.51、3.1.65、 3.1.83、3.1.84、3.1.85、3.1.87、3.1.96、3.1.98、3.1.128、3.1.129、 3.1.132、3.1.143、3.1.149、3.1.167、3.1.192、3.1.199
3.4.6	3.4.6
3.4.6.1	—
3.4.6.2~3.4.6.5	3.4.6.1~3.4.6.4
4	4
4.1	—
4.2~4.8	4.1~4.7
4.2.1~4.2.5	4.1.1~4.1.5
4.3.1~4.3.6	4.2.1~4.2.6
4.3.3.1	—
4.3.3.2~4.3.3.6	4.2.3.1~4.2.3.5
4.3.4.1~4.3.4.9	4.2.4.1~4.2.4.9
4.3.5.1	—
4.3.5.2、4.3.5.3	4.2.5.1、4.2.5.2
4.4.1	—
4.4.2~4.4.5	4.3.1~4.3.4
4.5.1.1	—
4.5.1.2~4.5.1.4	4.4.1.1~4.4.1.3
4.5.2	4.4.2
4.5.2.1	—
4.5.2.2、4.5.2.3	4.4.2.1、4.4.2.2
4.5.3	—
4.6.1	—
4.6.2~4.6.6	4.5.1~4.5.5
4.6.3.1~4.6.3.3	4.5.2.1~4.5.2.3
4.6.6.1	—
4.6.6.2、4.6.6.3	4.5.5.1、4.5.5.2
4.7、4.7.1	4.6、4.6.1
4.7.2~4.7.4	4.6.2~4.6.4

表 A.1 (续)

本部分章条编号	对应的 IEC 61375-2-1:2012 章条编号
4.7.2.1~4.7.2.6	4.6.2.1~4.6.2.6
4.7.3.1~4.7.3.6	4.6.3.1~4.6.3.6
4.8.1	4.7.1
4.8.1.1~4.8.1.5	4.7.1.1~4.7.1.5
4.8.1.5、4.8.1.5.2	4.7.1.5
4.8.1.5.2~4.8.1.5.5	4.7.1.5.1~4.7.1.5.4
4.8.2	4.7.2
4.8.2.1	—
4.8.2.2~4.8.2.7	4.7.2.1~4.7.2.6
4.8.2.5.1、4.8.2.5.2	4.7.2.4.1、4.7.2.4.2
4.8.3	4.7.3
5.4.1.1	—
5.4.1.2	5.4.1.1
5.4.1.2.1、5.4.1.2.2	5.4.1.1.1、5.4.1.1.2
5.5.1.4	5.5.1.3.1
5.5.1.5	5.5.1.3.2
5.5.2.1	—
5.5.2.2~5.5.2.9	5.5.2.1~5.5.2.8
5.5.4.7.1	—
5.5.4.7.2~5.5.4.7.4	5.5.4.7.1~5.5.4.7.3
5.6.4.2、5.6.4.2.1	5.6.4.2
5.6.4.2.2、5.6.4.2.3	5.6.4.2.1、5.6.4.2.2
6.2.3.3.5.1	—
6.2.3.3.5.2~6.2.3.3.5.7	6.2.3.3.5.1~6.2.3.3.5.6
6.3.2.1	—
6.3.2.2~6.3.2.7	6.3.2.1~6.3.2.6
6.3.3.1	—
6.3.3.2~6.3.3.5	6.3.3.1~6.3.3.4
6.3.4.5.1	—
6.3.4.5.2、6.3.4.5.3	6.3.4.5.1、6.3.4.5.2
6.3.4.6.1	—
6.3.4.6.2~6.3.4.6.13	6.3.4.6.1~6.3.4.6.12
6.3.6.7.1	—
6.3.6.7.2~6.3.6.7.10	6.3.6.7.1~6.3.6.6.9
6.3.7.1.1	—
6.3.7.1.2~6.3.7.1.7	6.3.7.1.1~6.3.7.1.6

表 A.1 (续)

本部分章条编号	对应的 IEC 61375-2-1:2012 章条编号
6.3.7.6.1	—
6.3.7.6.2~6.3.7.6.8	6.3.7.6.1~6.3.7.6.7
6.3.10.6.1	—
6.3.10.6.2~6.3.10.5.5	6.3.10.6.1~6.3.10.6.4
6.3.10.7.1	—
6.3.10.7.2~6.3.10.7.6	6.3.10.7.1~6.3.10.7.5
6.3.10.8.1	—
6.3.10.8.2~6.3.10.8.6	6.3.10.8.1~6.3.10.8.5
6.3.10.9.1	—
6.3.10.9.2~6.3.10.9.4	6.3.10.9.1~6.3.10.9.3
6.3.10.10.1	—
6.3.10.10.2~6.3.10.10.8	6.3.10.10.1~6.3.10.10.8
6.4.3.3.3.1	—
6.4.3.3.3.2、6.4.3.3.3.3	6.4.3.3.3.1、6.4.3.3.3.2
6.4.3.4.3.1	—
6.4.3.4.3.2~6.4.3.4.3.4	6.4.3.4.3.1~6.4.3.4.3.3
6.4.3.5.3.1	—
6.4.3.5.3.2、6.4.3.5.3.3	6.4.3.5.3.1、6.4.3.5.3.2
6.4.3.7.3.1	—
6.4.3.7.3.2	6.4.3.7.3.1
6.4.3.12.3.1	—
6.4.3.12.3.2、6.4.3.12.3.3	6.4.3.12.3.1、6.4.3.12.3.2
6.4.4.3.3.1	—
6.4.4.3.3.2~6.4.4.3.3.5	6.4.4.3.3.1~6.4.4.3.3.4
6.4.6.1	—
6.4.6.2~6.4.6.5	6.4.6.1~6.4.6.4
7.1.1.1	—
7.1.1.2、7.1.1.3	7.1.1.1、7.1.1.2
7.2.1	—
7.2.2~7.2.8	7.2.1~7.2.7
8.4.9.1	8.4.9.1、8.4.9.1.1
8.4.9.2、8.4.9.3	8.4.9.1.2、8.4.9.1.3
8.5.1	—
8.5.2、8.5.3	8.5.1、8.5.2
8.5.3.1~8.5.3.4	8.5.2.1~8.5.2.4
8.5.3.3.1~8.5.3.3.3	8.5.2.3.1~8.5.2.3.3
8.5.3.4.1、8.5.3.4.2	8.5.2.4.1、8.2.3.4.2

参 考 文 献

- [1] GB/T 9387(所有部分) 信息技术 开放系统互连 基本参考模型
 - [2] GB/T 15127 信息技术 系统间远程通信和信息交换 双扭线多点互连
 - [3] GB/T 15629.3 信息技术 系统间远程通信和信息交换 局域网和城域网 特定要求 第3部分:带碰撞检测的载波侦听多址访问(CSMA/CD)的访问方法和物理层规范
 - [4] GB/T 18657.1 远动设备及系统 第5部分:传输规约 第1篇:传输帧格式
 - [5] UIC 557 Diagnostics on passenger rolling stock
-