



## LIN 2.1规范

LIN(Local Interconnect Network) Specification 2.1

## 概述

协议规范

物理层规范

传输层规范

节点配置和标识规范

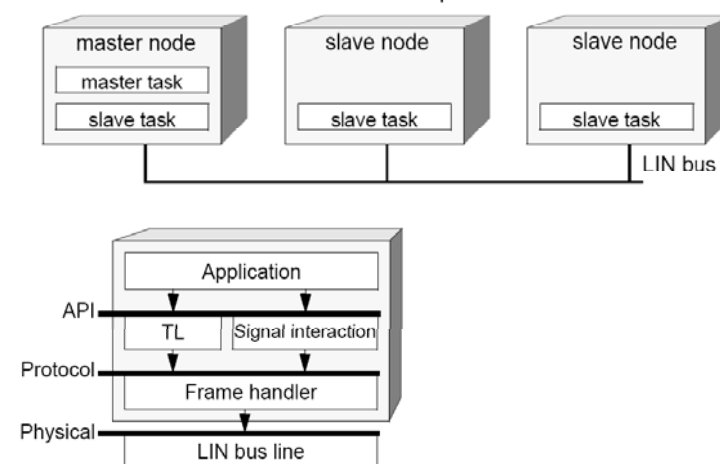
诊断规范

配置语言规范



# 概述

## Overview



# 概述

## 子网(SUB-BUS)的概念

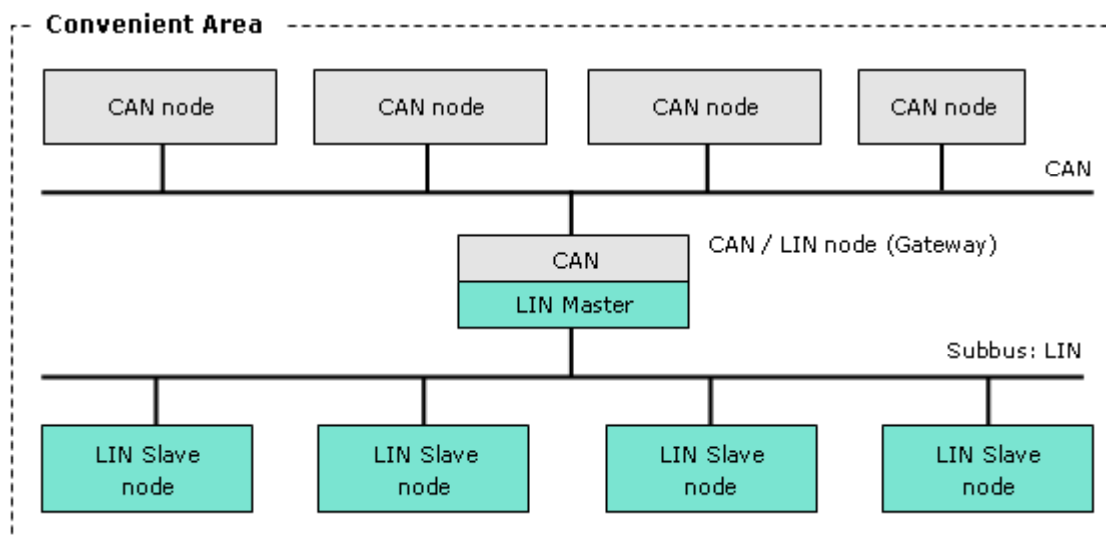
❑ 功能简单，实时性低

❑ 成本低

❑ 传感器/执行器级的总线

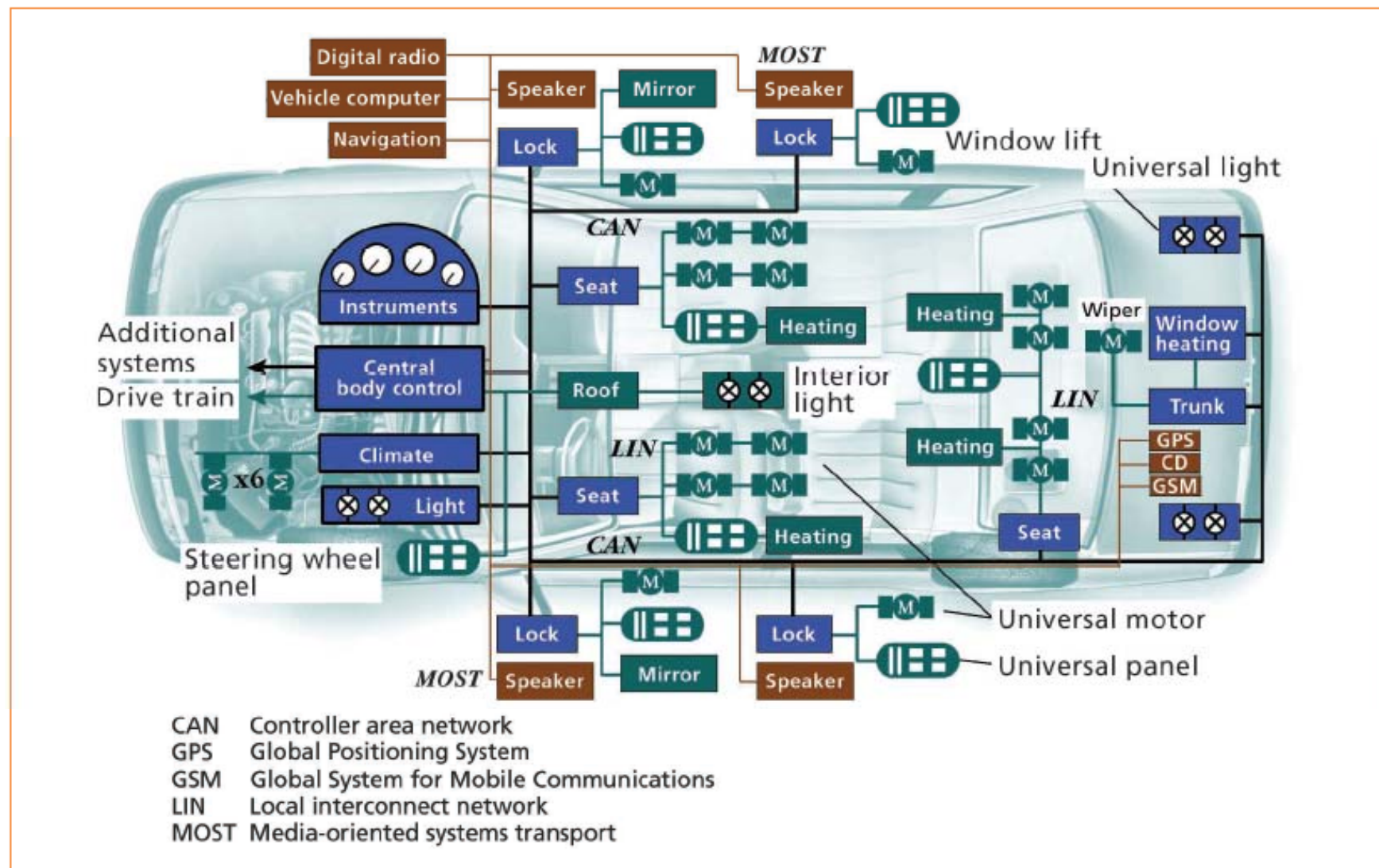
❑ 与主干网(back-bone BUS)之间需要网关

❑ LIN总线是一个SUB-BUS



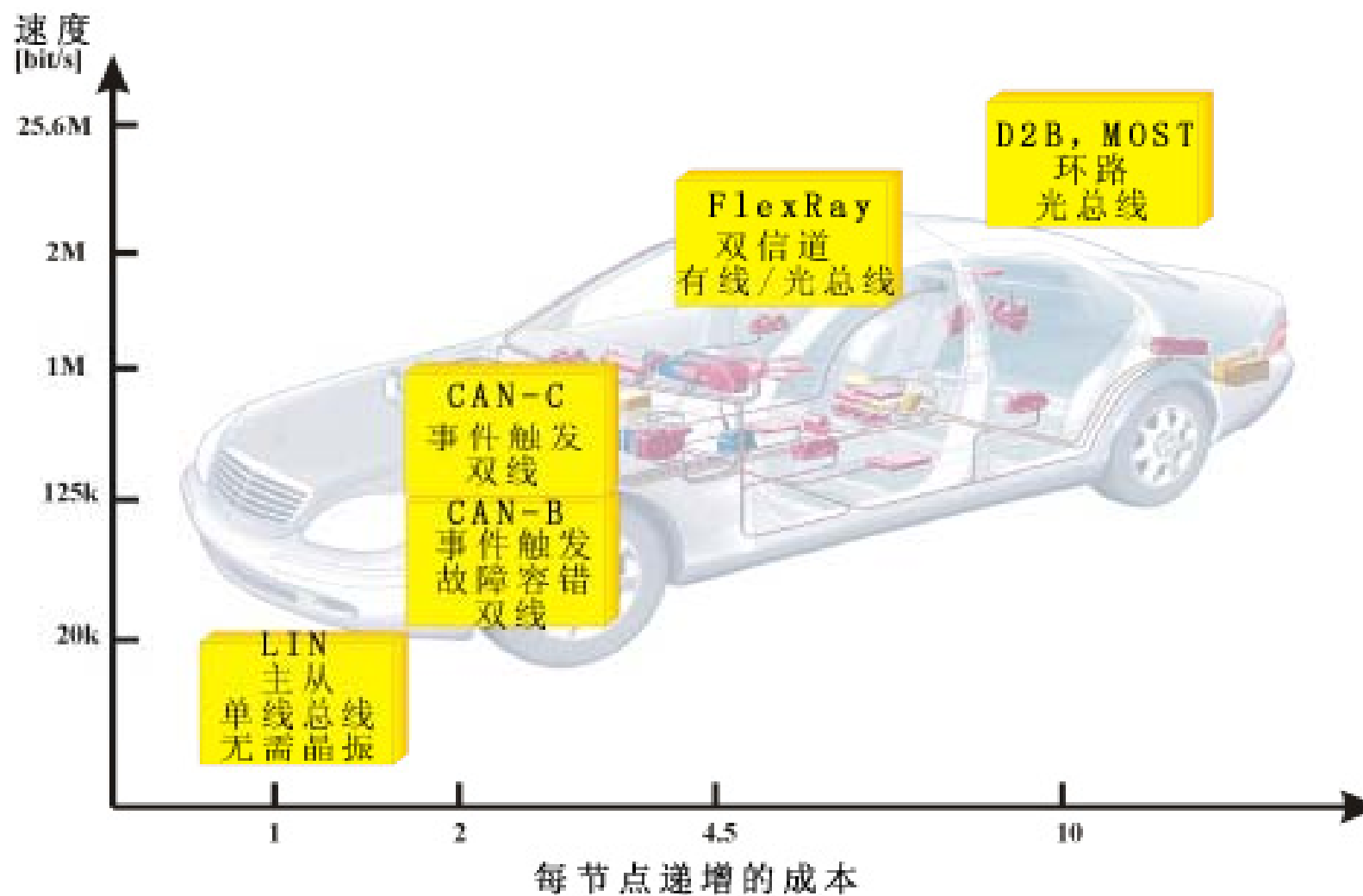
# 概述

## 典型车载网络分布



# 概述

## 车用总线分级



### **LIN(Local Interconnect Network)协会成立于1998年**

成立时的主要成员：

- ❑ 5家整车厂：Audi, BMW, DaimlerChrysler, Volvo, VW
- ❑ 1家半导体制造商：Motorola
- ❑ 1家工具提供商：Mentor Graphics

目前的主要成员：

- ❑ 5家整车厂：Audi, BMW, DaimlerChrysler, Volvo, VW
- ❑ 1家半导体制造商：Freescale
- ❑ 1家工具提供商：Mentor Graphics

主要目的：

- ❑ LIN总线的主要目的在于提供一种低成本的车用总线，从而形成对CAN总线的补充。
- ❑ LIN总线已经广泛地被世界上的大多数汽车公司以及零配件厂商所接受，有望成为公认的A类网络标准。

# 概述

## LIN协议版本(1/2)

### LIN 1.1

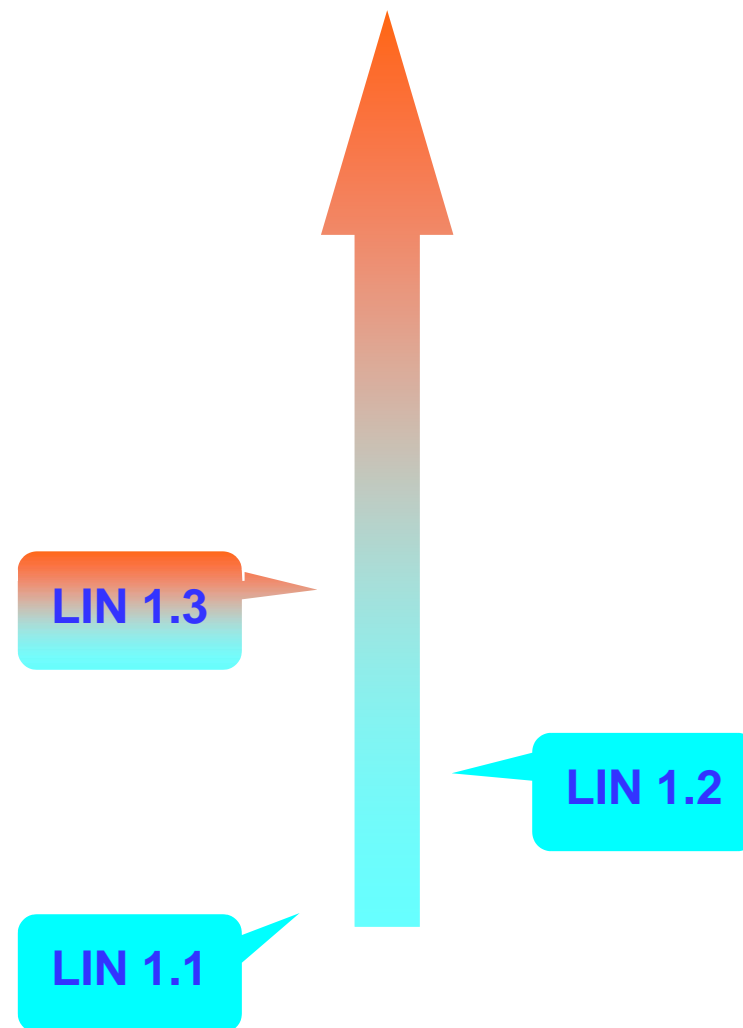
- ❑ 1999年，底特律SAE大会
- ❑ 包括3部分
  - ❑ 协议规范
  - ❑ 配置语言规范
  - ❑ **API**规范

### LIN 1.2

- ❑ 2000年11月

### LIN 1.3

- ❑ 2002年11月
- ❑ 增加了物理层规范



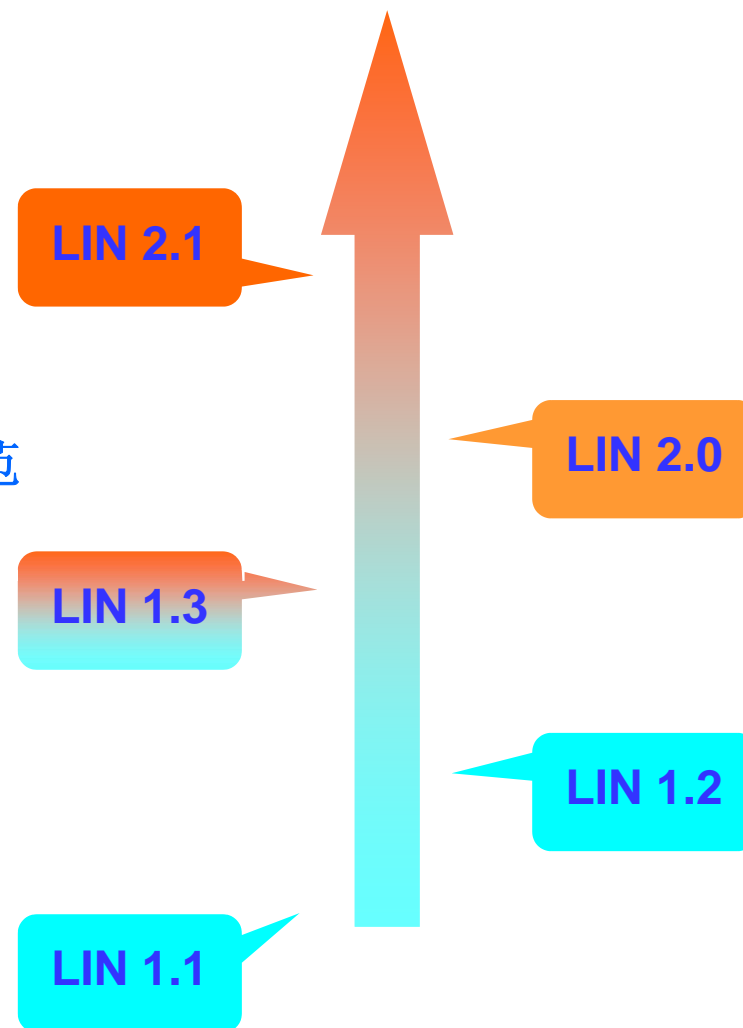


### LIN 2.0

- ❑ 2003年9月
- ❑ 适应当代和未来汽车工业发展趋势
- ❑ 为了实现节点的“即插即用”
- ❑ 增加了**诊断规范**和**节点能力语言规范**

### LIN 2.1

- ❑ 2006年11月
- ❑ 目标是改进LIN 2.0规范的理解力
- ❑ 增加了**传输层规范**和**节点配置规范**
- ❑ 形成了8个子规范



# 概述

## LIN规范组成(1/2)

子规范	内容
物理层规范 (Physical Layer Specification)	<ul style="list-style-type: none"><li>→ 物理层兼容性</li><li>→ 波特率误差</li><li>→ 时间要求</li><li>→ <b>LIN Driver / Receiver</b></li></ul>
协议规范 (Protocol Specification)	<ul style="list-style-type: none"><li>→ 信号管理</li><li>→ 帧传输</li><li>→ 调度表</li><li>→ 任务行为模型</li><li>→ 网络管理</li><li>→ 状态管理</li></ul>
传输层规范 (Transport Layer Specification)	<ul style="list-style-type: none"><li>→ <b>PDU</b>规范</li><li>→ 通信</li><li>→ 错误处理</li><li>→ 规定的请求</li><li>→ 时间约束</li></ul>

# 概述

## LIN规范组成(2/2)

子规范	内容
节点配置和标识规范 (Node Configuration and Identification Specification)	→ <b>LIN</b> 产品标识 → 从节点模型 → <b>PDU</b> 结构 → 节点配置服务 → 标识
诊断规范 (Diagnostic Specification)	→ 诊断级别 → 基于信号的诊断 → 传输协议处理 → 从节点发送句柄 → 从节点诊断时间要求
应用程序接口规范 (API Specification)	→ 主要 <b>API</b> → 节点配置和标识 → 传输层
节点能力语言规范 (Node Capability Language Specification)	→ 文件定义 → 语法
配置语言规范 (Configuration Language Specification)	→ 文件定义 → 语法

### ❑ 串行通信

- ❑ 线间干扰小，节省线束，传输距离长

### ❑ 单线传输

- ❑ 增强的**ISO 9141 (ISO 15765-1)**，总线电压基于 $V_{BAT}$

### ❑ 最高速率20Kbit/s

- ❑ 满足车身上大部分的应用需求

### ❑ 单主多从结构

- ❑ 无需仲裁

### ❑ 基于通用UART/SCI的低成本接口硬件

- ❑ 几乎所有**MCU**有具备**LIN**总线的硬件基础

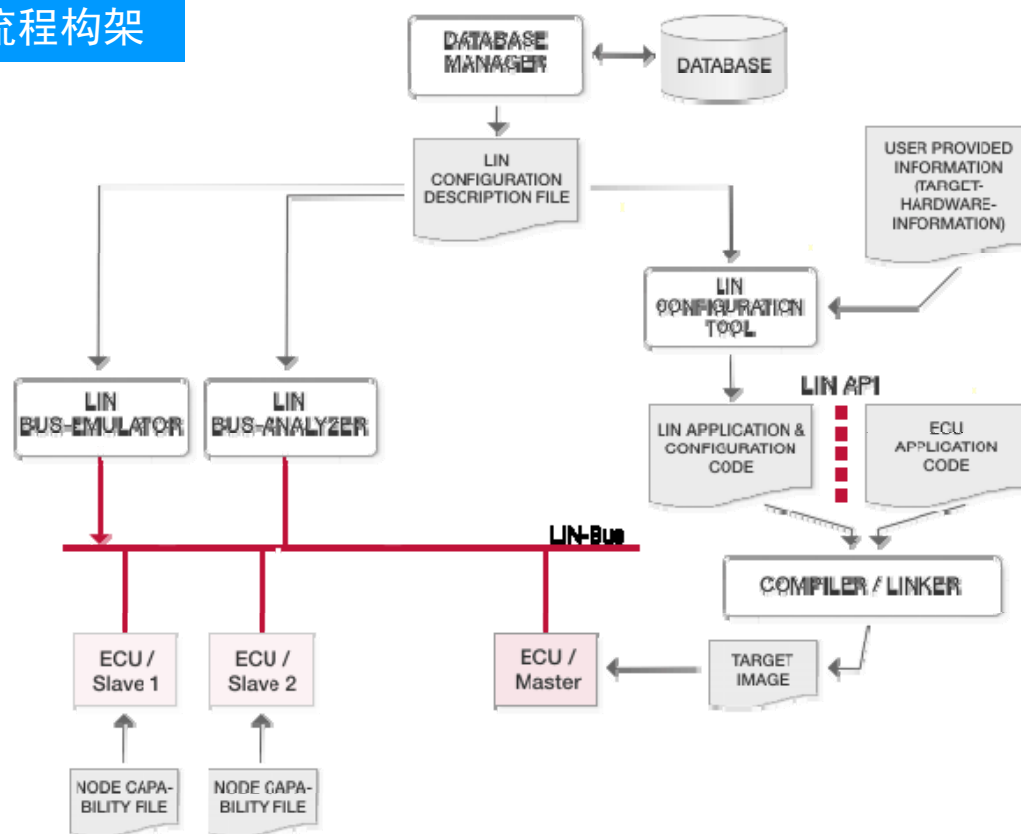
### ❑ 从节点无须晶振或陶瓷震荡器就可以实现同步

- ❑ 大幅度降低成本

- ❑ 可计算的信号传输的延迟时间
  - ❑ 网络通信可预期性
- ❑ 可灵活的增加或减少从节点
  - ❑ 无需改变其他节点的硬件电路
- ❑ 一条总线最多可连接16个节点
  - ❑ 由总线电气特性决定
- ❑ 支持多包报文传输
  - ❑ 基于**ISO15765-2**的传输层规范
- ❑ 支持诊断功能
  - ❑ 支持**ISO14229**的诊断服务

# 概述

## LIN总线开发流程构架



### ❑ LIN Node Capability Language (\*.ncf)

- ❑ 对单个节点的描述

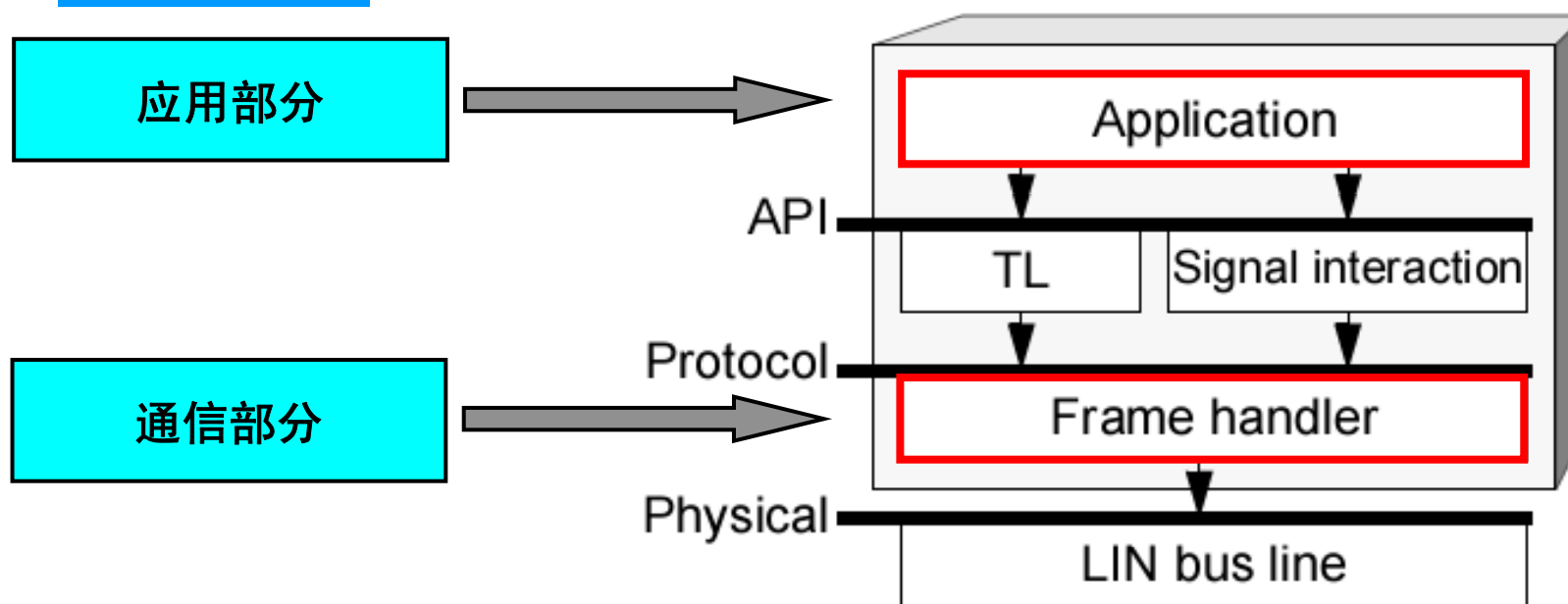
### ❑ LIN Configuration Language (\*.ldf)

- ❑ 在增加/减少节点的情况下保证总线的安全
- ❑ 起到LIN总线的数据库的作用，方便LIN总线的仿真和测试。

### ❑ LIN总线开发流程构架旨在实现开发设计工具之间的无缝衔接，提高开发的效率和可靠性

# 概述

## LIN节点模型

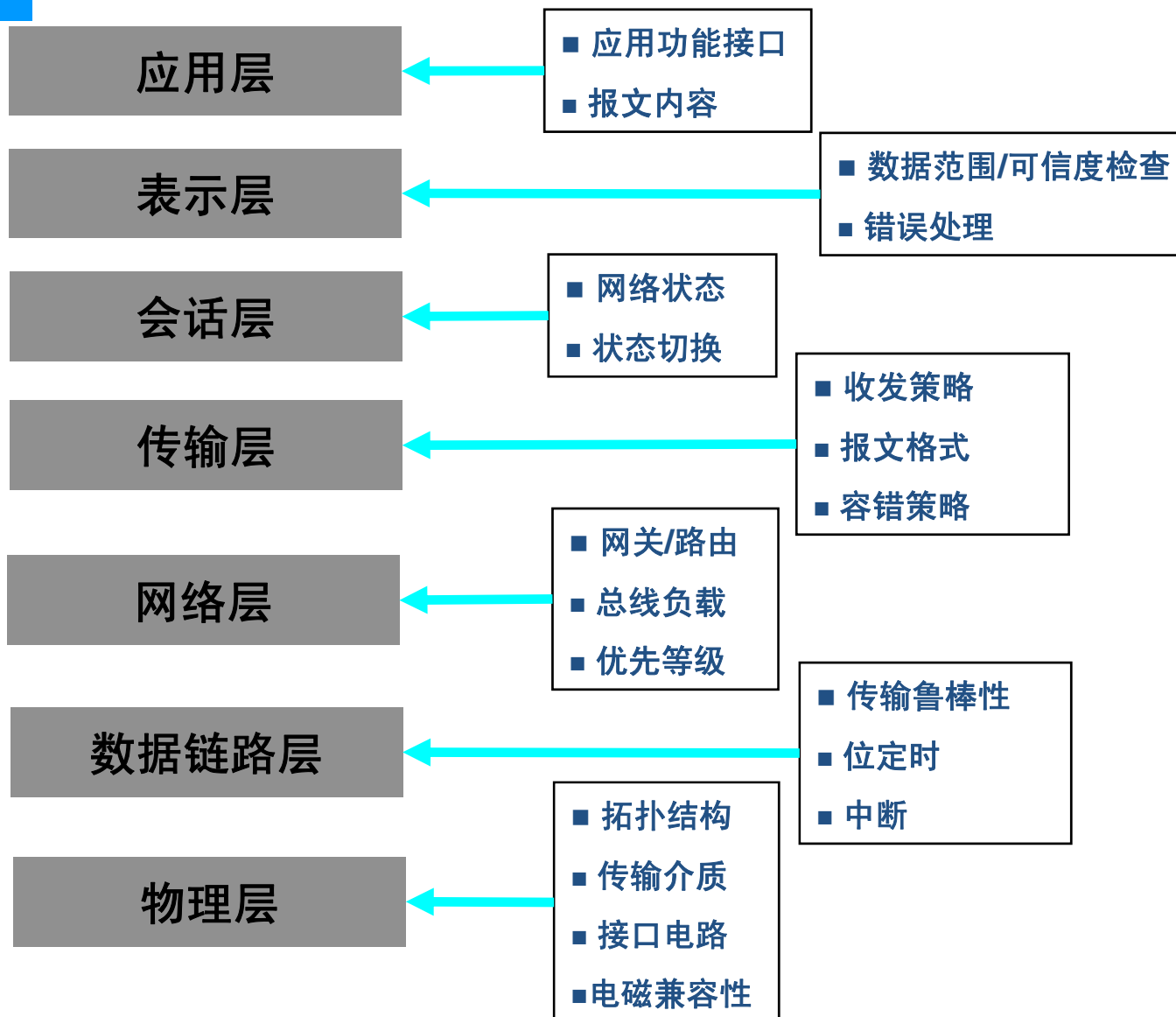


- ❑ 一个节点可以分为应用部分和通信部分
- ❑ 报文不会以帧的结构到达应用部分
- ❑ 在应用部分与通信部分之间加入一个基于信号的交互层
- ❑ 传输层中进行长报文的处理

# 概述

## LIN和OSI模型

### LIN2.1





### ❑ 帧

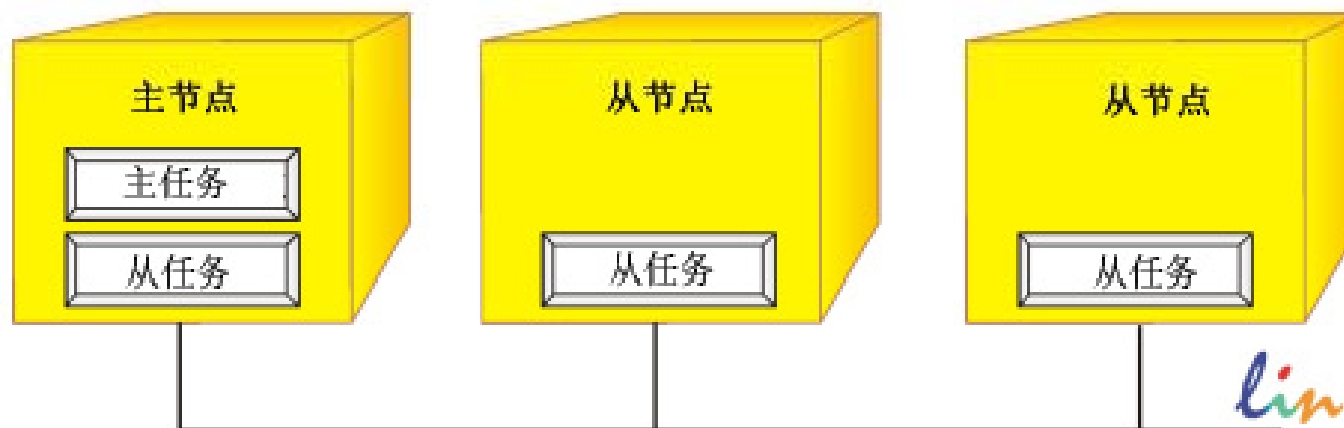
- ❑ 总线上传输的实体，是作为一个完整单元传输的数据。
- ❑ 帧的数据场中包含一个或多个信号
- ❑ 注意：一帧并不代表只能由单一节点发送

### ❑ 信号

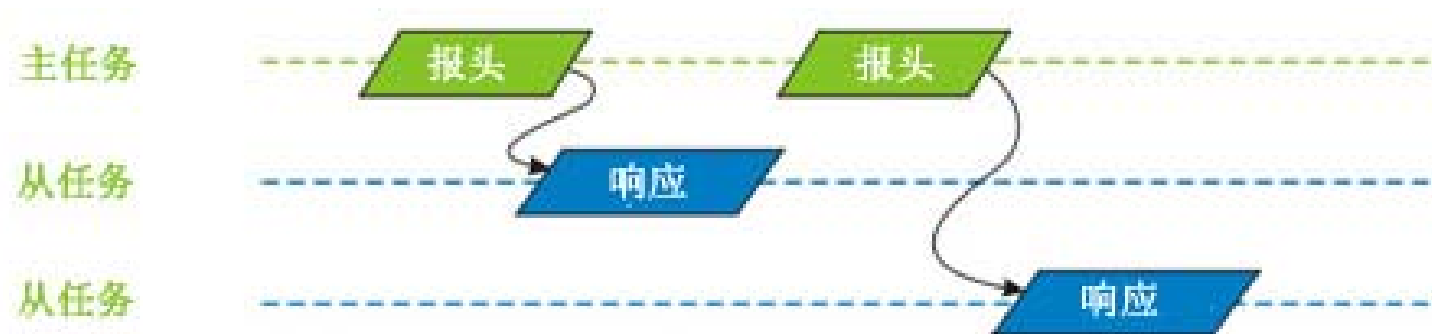
- ❑ 信号一般是反映真实世界的物理量或逻辑量，如发动机转速、电机的状态等
- ❑ 信号由帧来传输，一个帧可以包含一个或多个信号
- ❑ 每个信号由固定的节点产生，但可以由不同的节点接收
- ❑ 信号的大小从一个位到多个字节

# 概述

## LIN网络拓扑结构



- ❑ 一个LIN网络包含一个主任务，多个从任务
- ❑ 主节点同时包含主任务和从任务
- ❑ 从节点只包含从任务
- ❑ 主任务负责决定总线上的报文，从任务负责发送数据



### □通信原理:

- 主任务发送报头，从任务用响应来补充报头形成完整的报文。

### □系统灵活性:

- 可以自由地增减从节点，而不需要改变其它从节点的软件和硬件

### □报文传输:

- 报文的内容由**ID**来定义

### □广播:

- 所有节点都能够接受总线上的帧

### □ LIN总线上可以传输的两类数据

#### □ 信号报文

- 信号位于帧的数据场中，以数字或数组的形式出现
- 同一种信号在相同ID的报文中的位置总是固定的

#### □ 诊断报文

- 诊断报文由两个保留ID来传输，用来传输节点的诊断信息

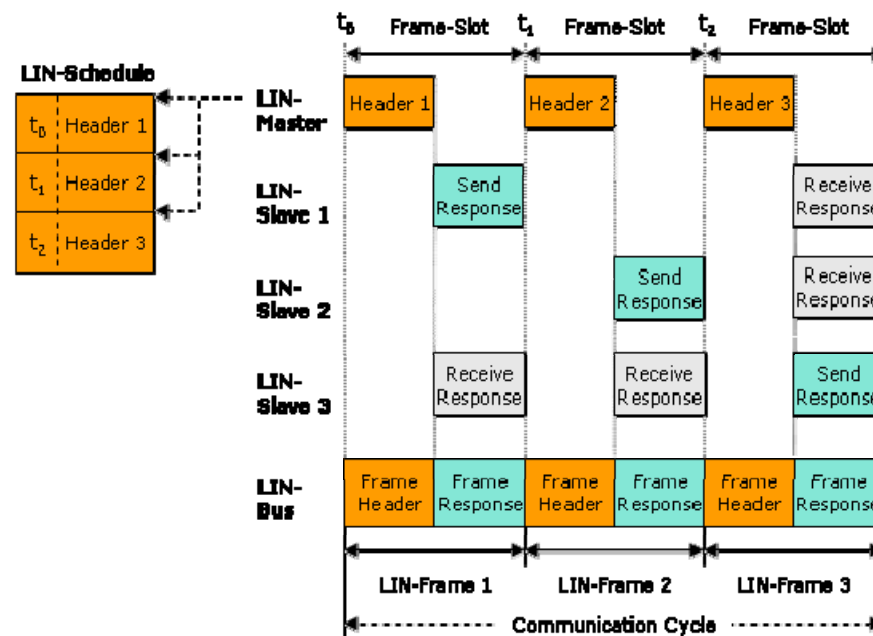
# 概述

## 调度表

### 负责调度网络各报文发送的顺序

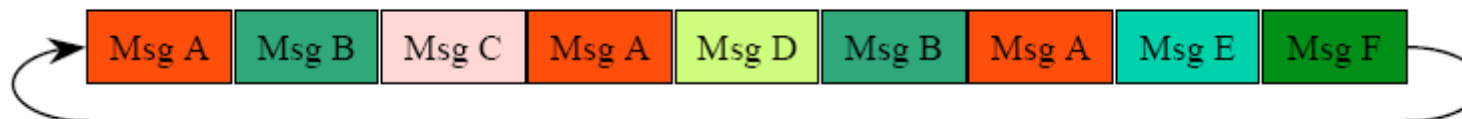
#### 为每帧报文分配发送时隙（slot）

- 发送时隙：报文可以被发送的时间
- 不同报文的发送时隙可能不同

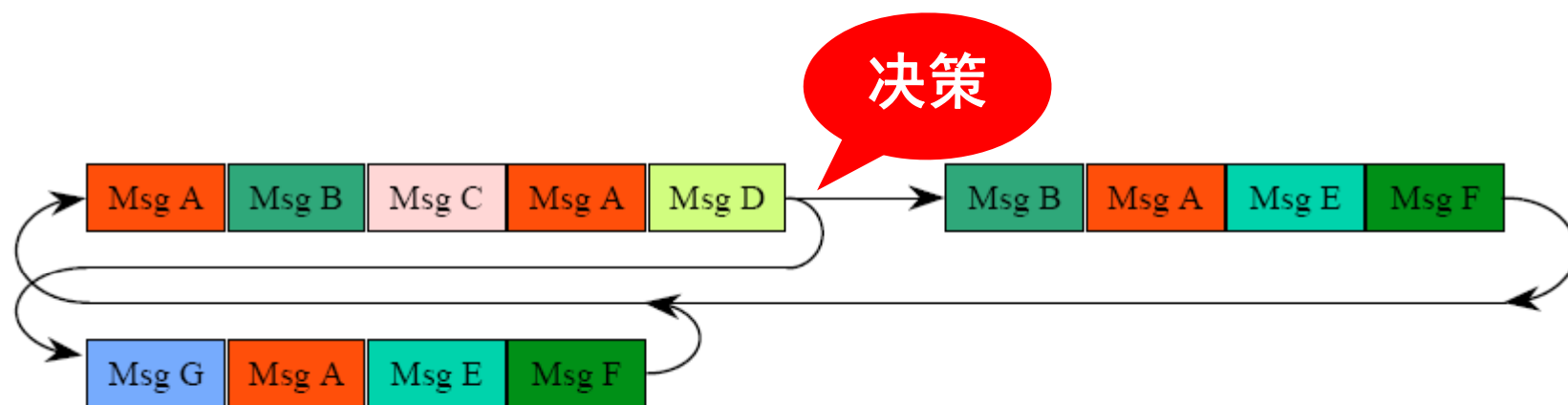


### 调度表在网络系统设计阶段确定

### 调度表使得LIN通信具有可预测性



- ❑ 主任务可以拥有多个调度表，并在不同的调度表之间切换



- ❑ 增加通信的灵活性

概述

**协议规范**

物理层规范

传输层规范

节点配置和标识规范

诊断规范

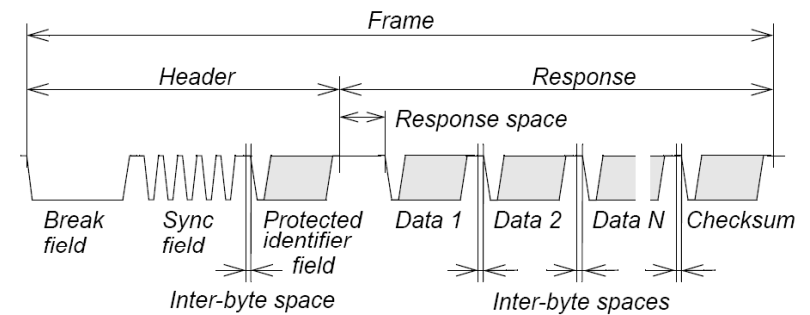
配置语言规范





## 协议规范

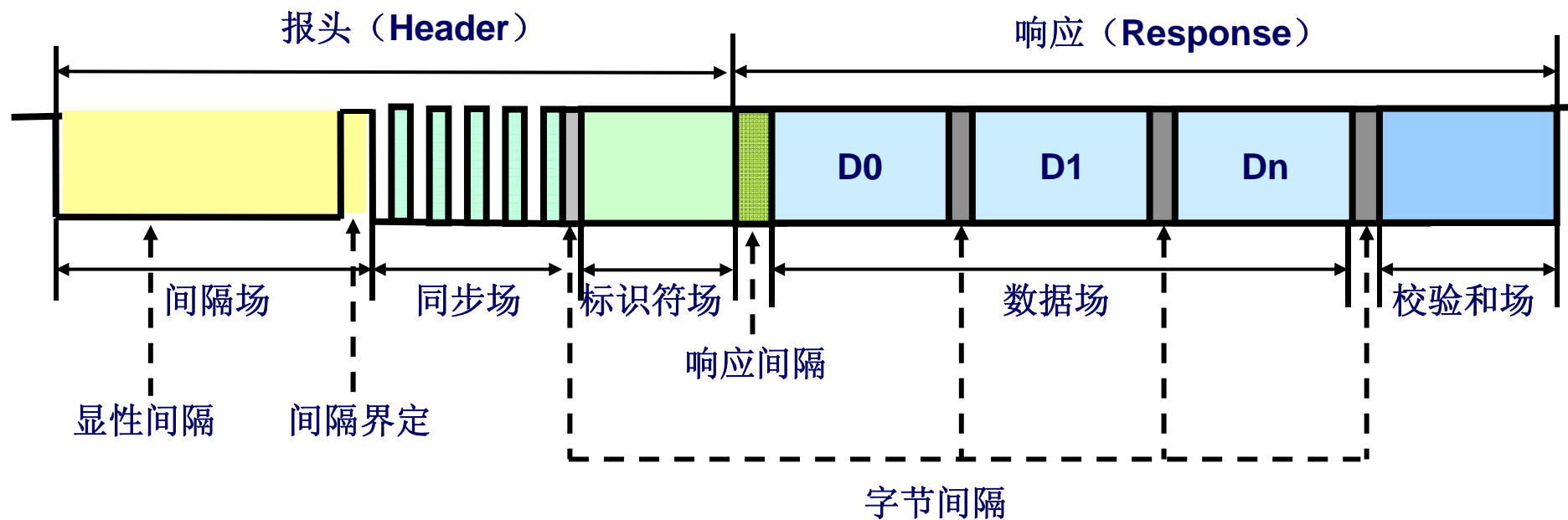
### Protocol Specification





# 协议规范

## 帧的结构(Frame Structure)



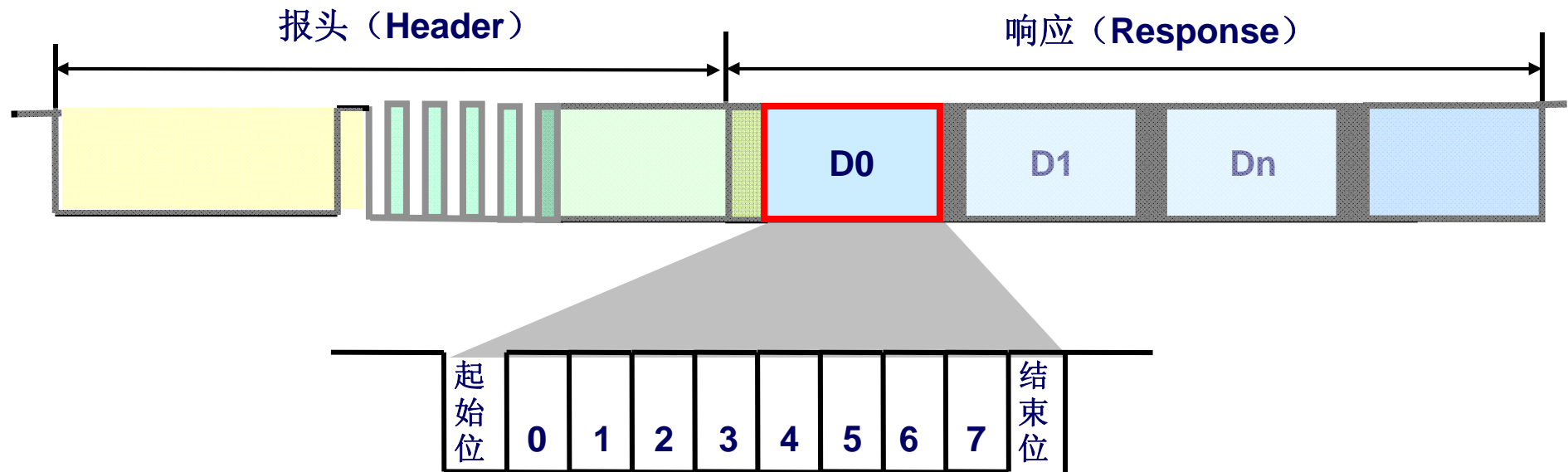
❑ 字节间隔位于每个字节之间，响应间隔位于报头与响应之间

❑ 留给**MCU**足够的处理时间

❑ 帧长度可能增长

# 协议规范

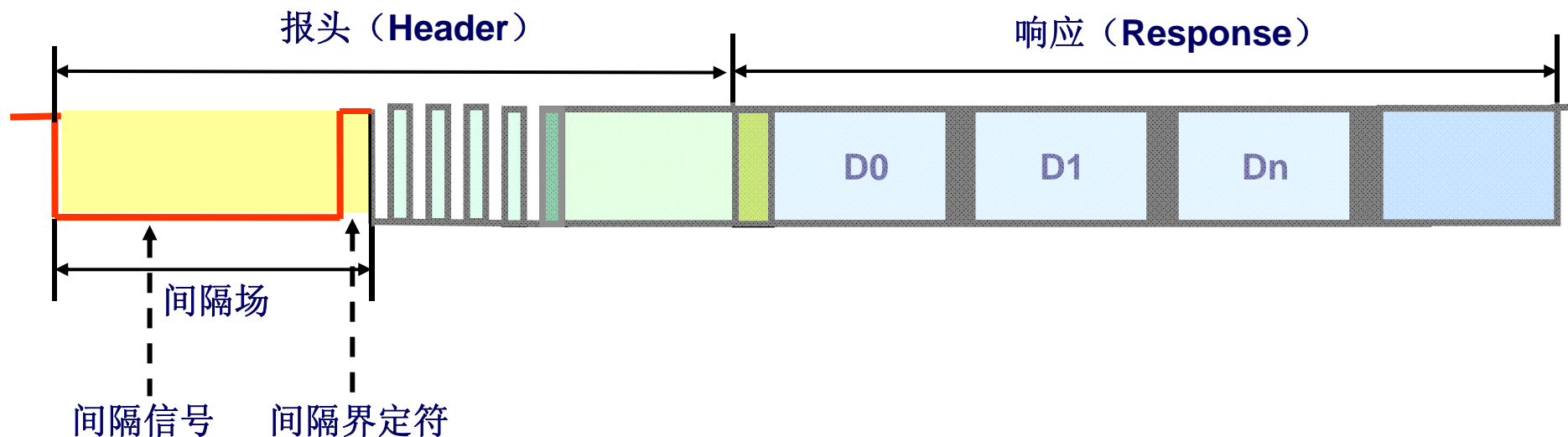
## 字节场(Byte Field)



- ❑ 基于**SCI**的通信格式
- ❑ 发送一个字节需要**10**个位时间 ( $T_{\text{BIT}}$ )

# 协议规范

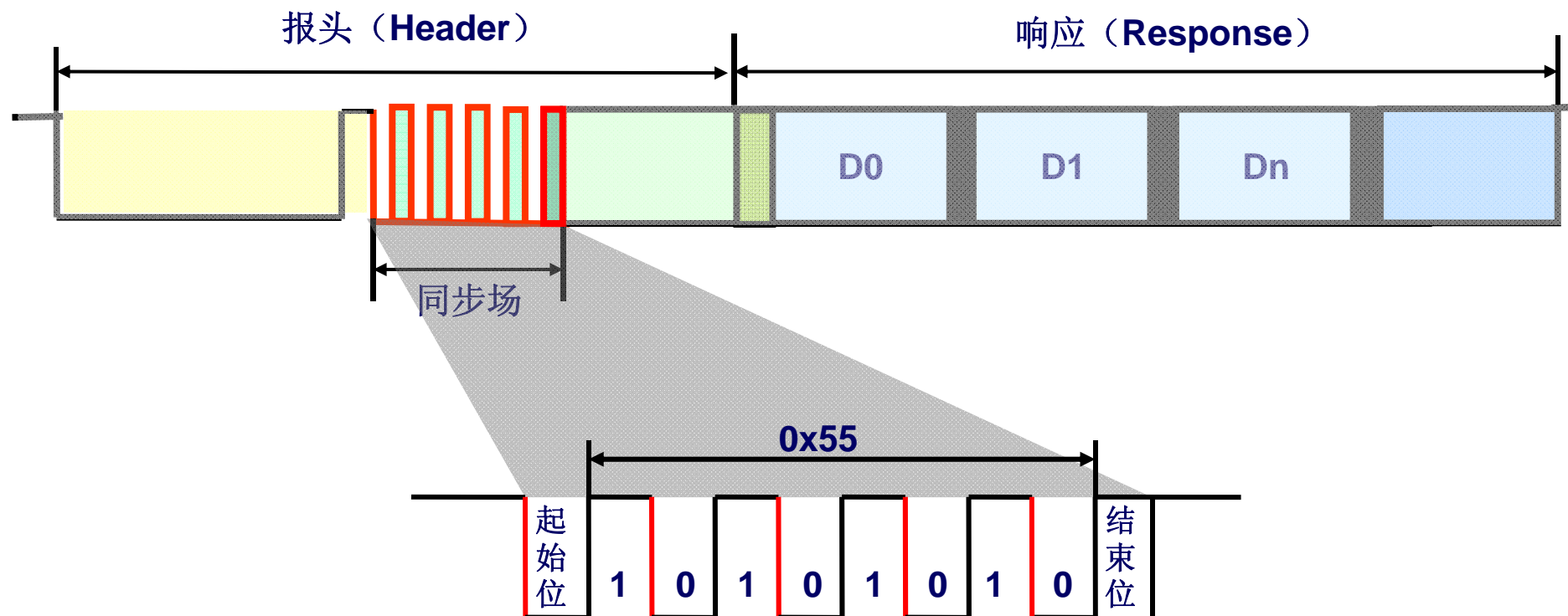
## 间隔场(Break Field)



- ❑ 表示一帧报文的起始，由主节点发出
- ❑ 间隔信号至少由**13**个显性位组成，间隔界定符至少由**1**个隐形位组成
- ❑ 间隔场是唯一一个不符合字节场格式的场
- ❑ 从节点需要检测到至少连续**11**个显性位才认为是间隔信号

# 协议规范

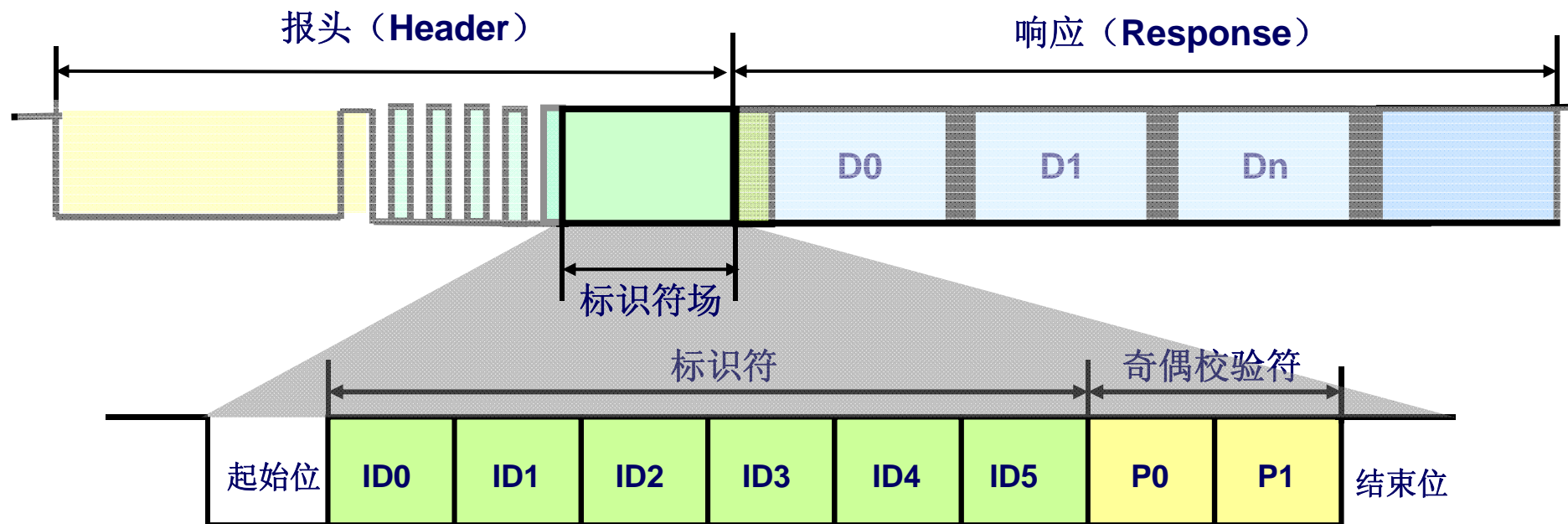
## 同步场(Sync Break Field)



- ❑ 确保所有从节点使用与节点相同的波特率发送和接收数据
- ❑ 一个字节，结构固定：**0X55**

# 协议规范

## 标识符场(Identifier Field)



❑ ID的范围从0到63(0x3f)

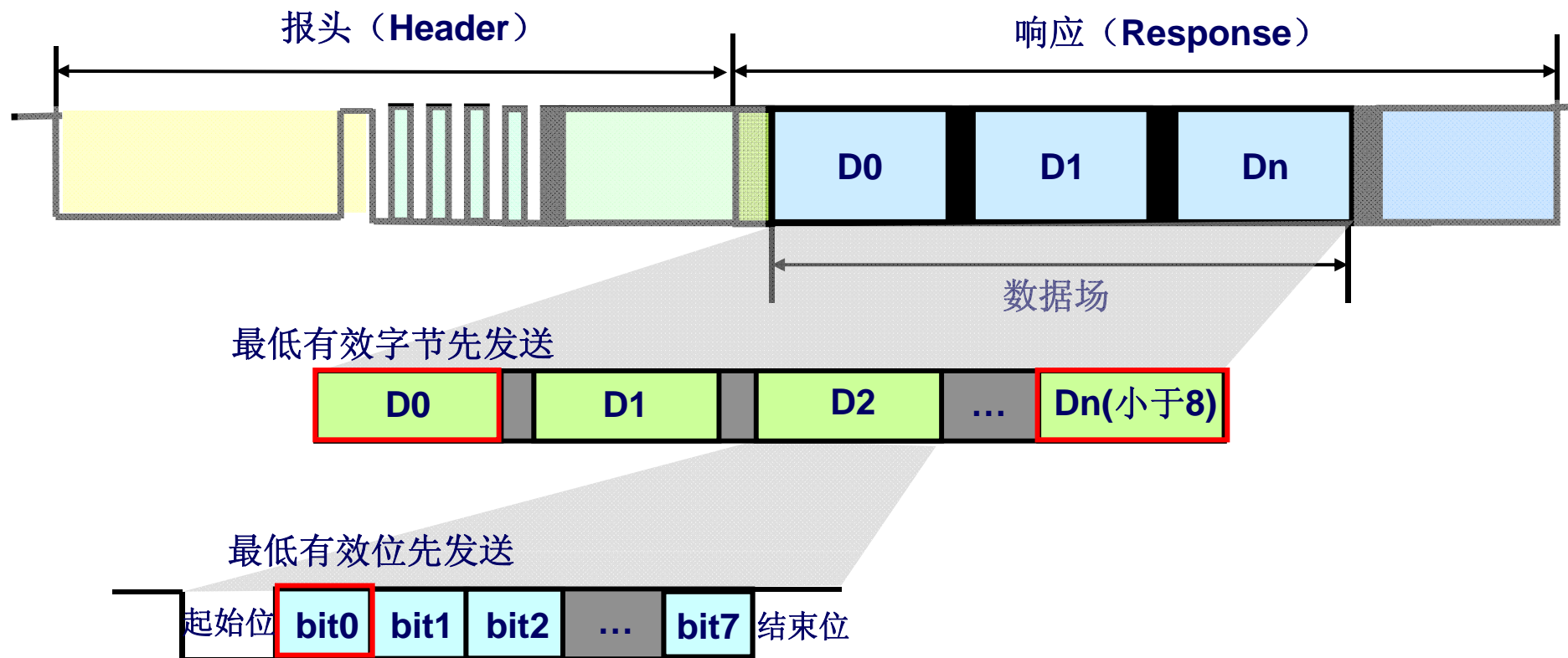
❑ 奇偶校验符(Parity)P0,P1

$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4$$

$$P1 = \overline{ID1 \oplus ID3 \oplus ID4 \oplus ID5}$$

# 协议规范

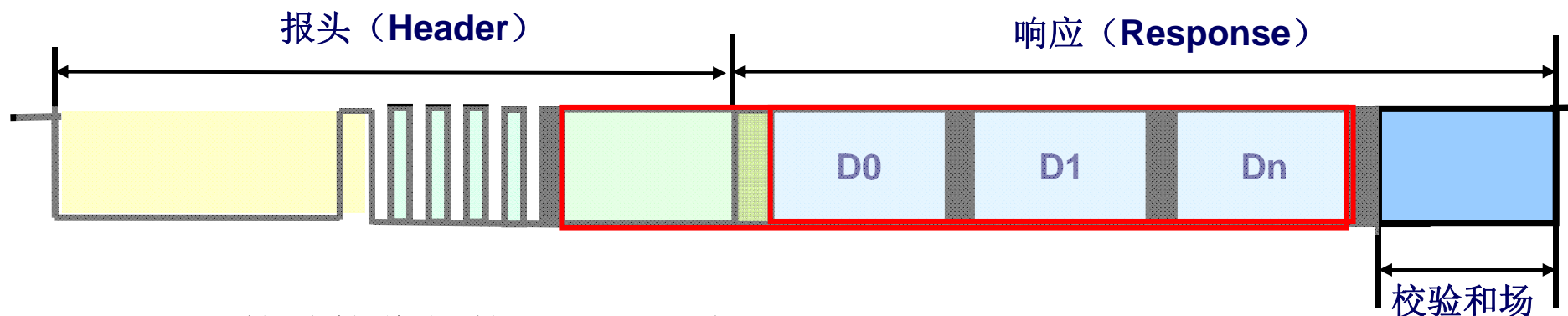
## 数据场(Data Field)



- ❑ 数据场长度**1**到**8**个字节
- ❑ 低字节先发，低位先发
- ❑ 如果某一信号长度超过**1**个字节，采用低位在前的方式发送（小端）

# 协议规范

## 校验和场(Checksum Field)(1/2)



- ❑ 用于校验接收的数据是否正确
- ❑ 经典校验 (**Classic Checksum**)
  - ❑ 仅校验数据场(**1.3**)
- ❑ 增强校验 (**Enhance Checksum**)
  - ❑ 校验标识符场与数据场内容(**2.0**)
- ❑ 标识符为**0x3C**和**0x3D**的帧只能使用经典校验

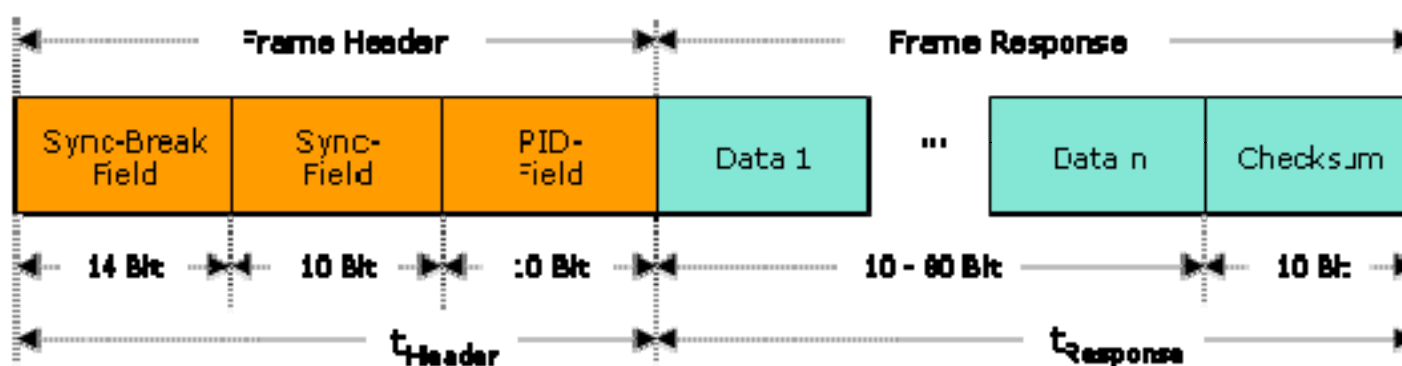
- ❑ 翻转八位和 (inverted eight bit sum)
- ❑ 例: Data = 0x4A, 0x55, 0x93, 0xE5
- ❑ Checksum = 0xE6

	hex
0x4A	0x4A
+0x55 = (Add Carry)	0x9F 0x9F
+0x93 = Add Carry	0x132 0x33
+0xE5 = Add Carry	0x118 0x19
Invert	0xE6
0x19+0xE6 =	0xFF



# 协议规范

## 帧长度(Frame Length)



### 最小帧长度

- ❑  $T_{HEADER\_NOMINAL} = 34 * T_{BIT}$
- ❑  $T_{RESPONSE\_NOMINAL} = 10 * (N_{DATA} + 1) * T_{BIT}$
- ❑  $T_{FRAME\_NOMINAL} = T_{HEADER\_NOMINAL} + T_{RESPONSE\_NOMINAL}$

### 最大帧长度

- ❑  $T_{HEADER\_MAX} = 1.4 * T_{HEADER\_NOMINAL}$
- ❑  $T_{RESPONSE\_MAX} = 1.4 * T_{RESPONSE\_NOMINAL}$
- ❑  $T_{FRAME\_MAX} = T_{HEADER\_MAX} + T_{RESPONSE\_MAX}$

# 协议规范


## 帧类型(Frame Type)



无条件帧



事件触发帧



保留帧



零星帧

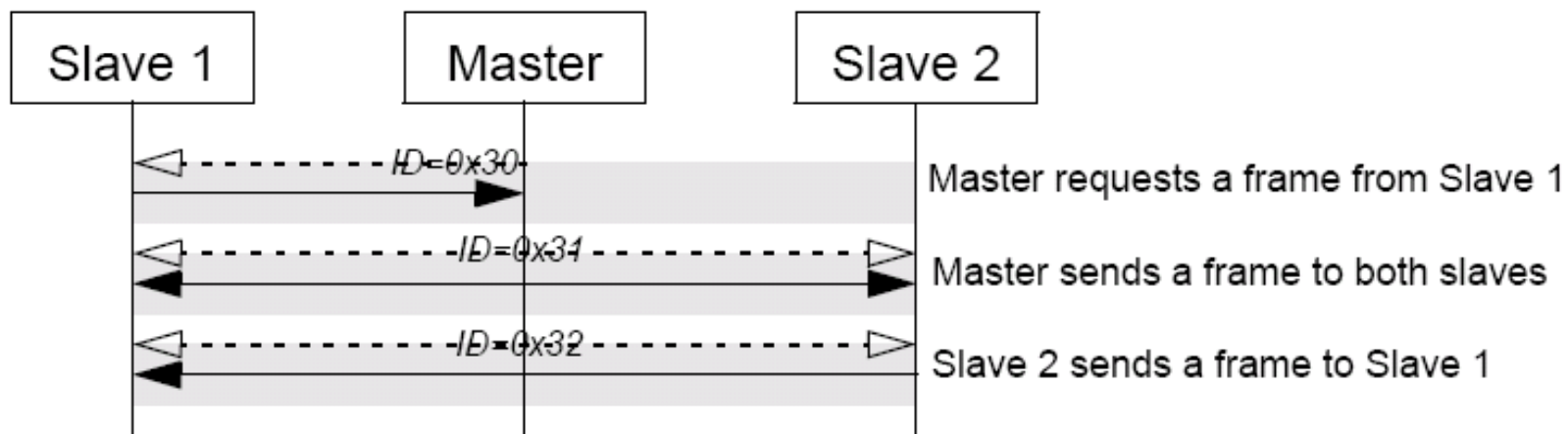


诊断帧



自定义帧

- ❑ 标识符(**ID**)为**0到59(0x3b)**
- ❑ 主任务发出报头，一个任务响应，一个或多个任务接收
- ❑ 三种情况



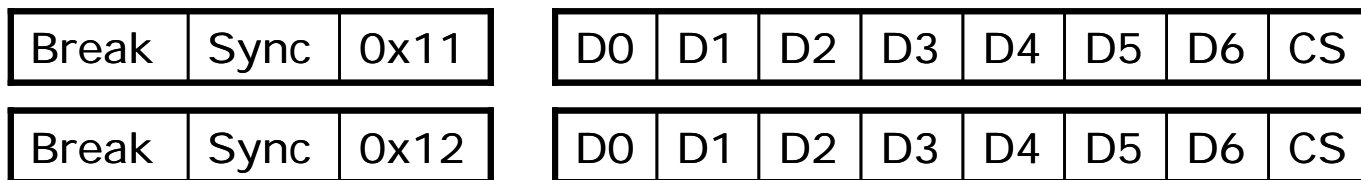
### ❑ 目的:

- ❑ 增强**LIN**总线的响应能力，避免为了很少发生的事件而对从节点进行轮询，从而浪费了大量的带宽

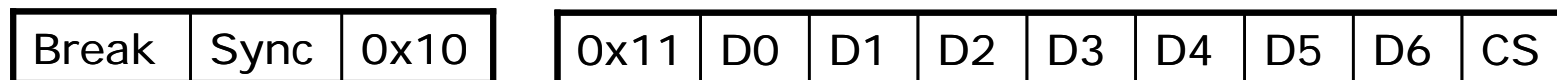
### ❑ 标识符: **0~59(0x3B)**

- ❑ 事件触发帧必须有一个独立的**ID**，该**ID**可以与多个普通帧相关联
- ❑ 在事件触发帧时隙内发送帧头，只有当相关联的无条件帧内有信号被更新时才发送帧响应
- ❑ 帧响应的第一个数据字节等于标识符，即响应最多可以传输**7**个字节的数据
- ❑ 如果没有帧响应，帧头被忽略
- ❑ 帧响应可由多个节点发送，发生冲突时切换到“冲突解决调度表”，之后再切换回到原来的调度表

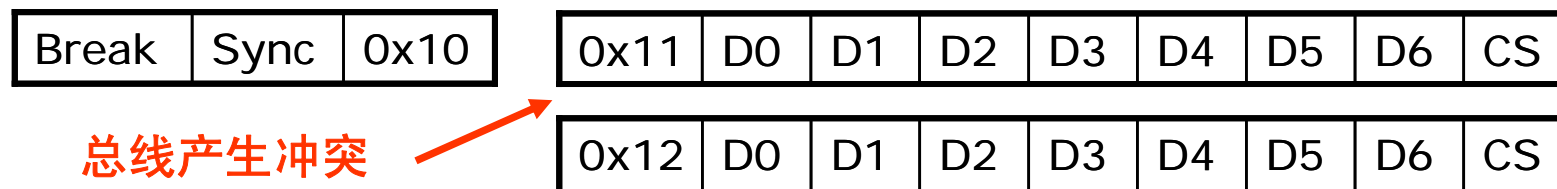
- 假设与事件触发帧0x10相关联的两个普通帧...



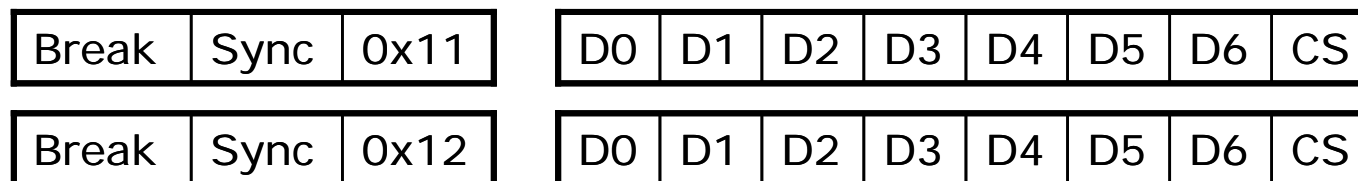
- 如果只有一个节点响应事件触发帧的报头...



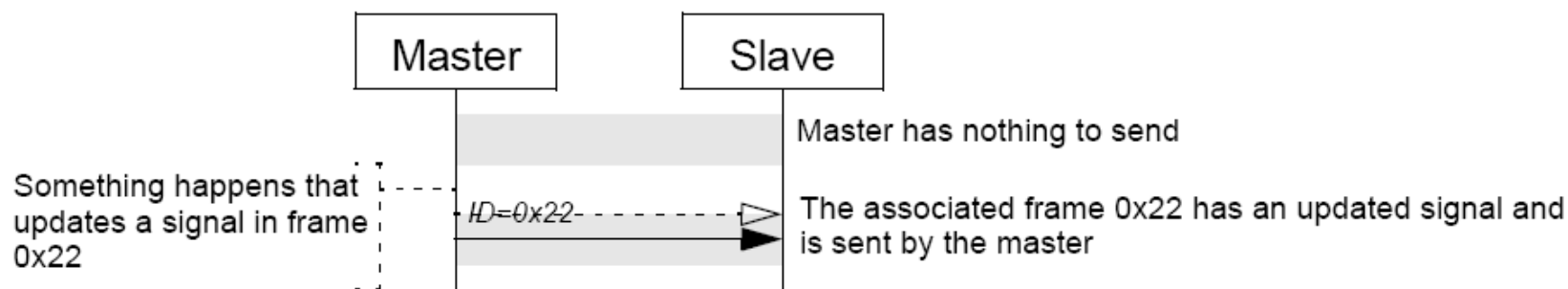
- 如果同时有多个节点响应事件触发帧的报头...



- 切换调度表，进行轮询...



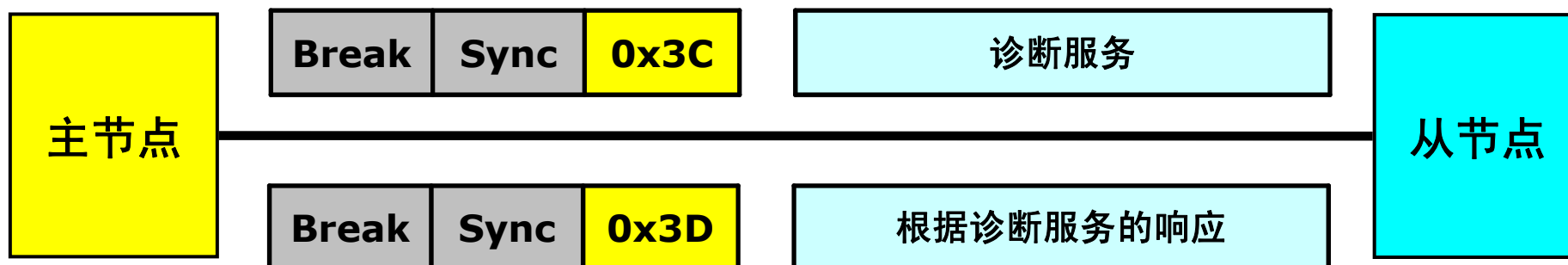
- ❑ 表示共用一个时隙、在需要时才被发送的一组普通帧
- ❑ 目的
  - ❑ 将动态行为和实时性添加到调度表中，同时不影响调度表其余部分的确定性
- ❑ 标识符：**0~59(0x3b)**
- ❑ 仅由主节点发送零星帧内的普通帧
- ❑ 若有普通帧需要发送，则根据帧的优先级裁定发送顺序
- ❑ 若没有普通帧需要发送，则时隙保持空白



# 协议规范

## 诊断帧(Diagnostic Frame)

- ❑ 诊断帧用来传输诊断或配置信息，一般包含**8**个字节数据。
- ❑ 标识符
  - ❑ **60(0x3c)**: 主请求帧
  - ❑ **61(0x3d)**: 从响应帧
- ❑ 诊断响应基于**ISO15765-2**传输层和**ISO14229**应用层
- ❑ 传输方式:



- ❑ 用户自定义帧(**User-defined frame**)

- ❑ 标识符: **62(0x3e)**

- ❑ 可以传输任何信息(由用户自定义)

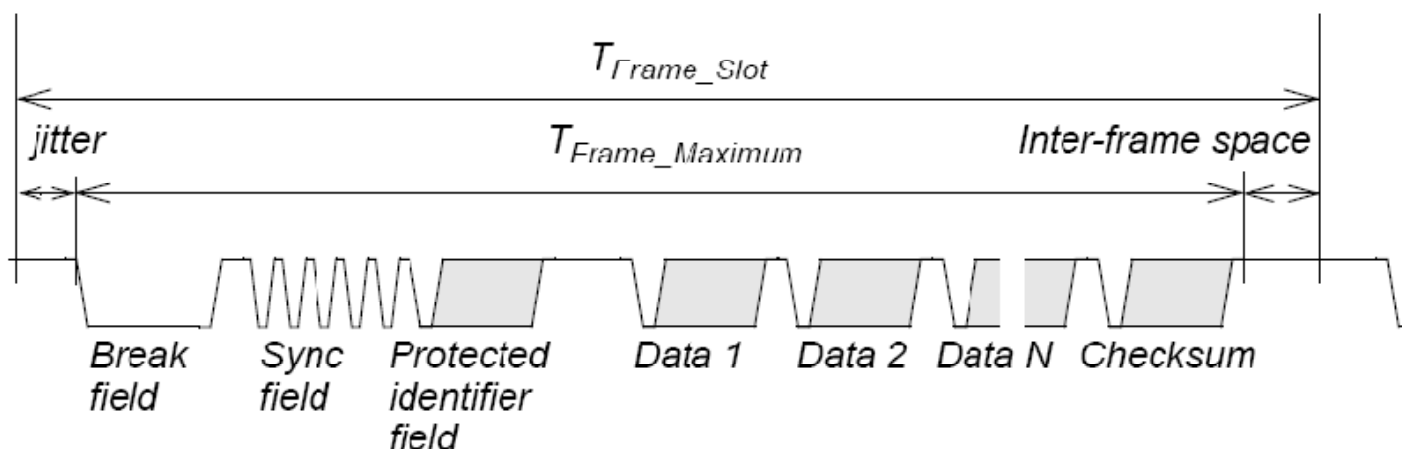
- ❑ 保留帧(**Reserved frame**)

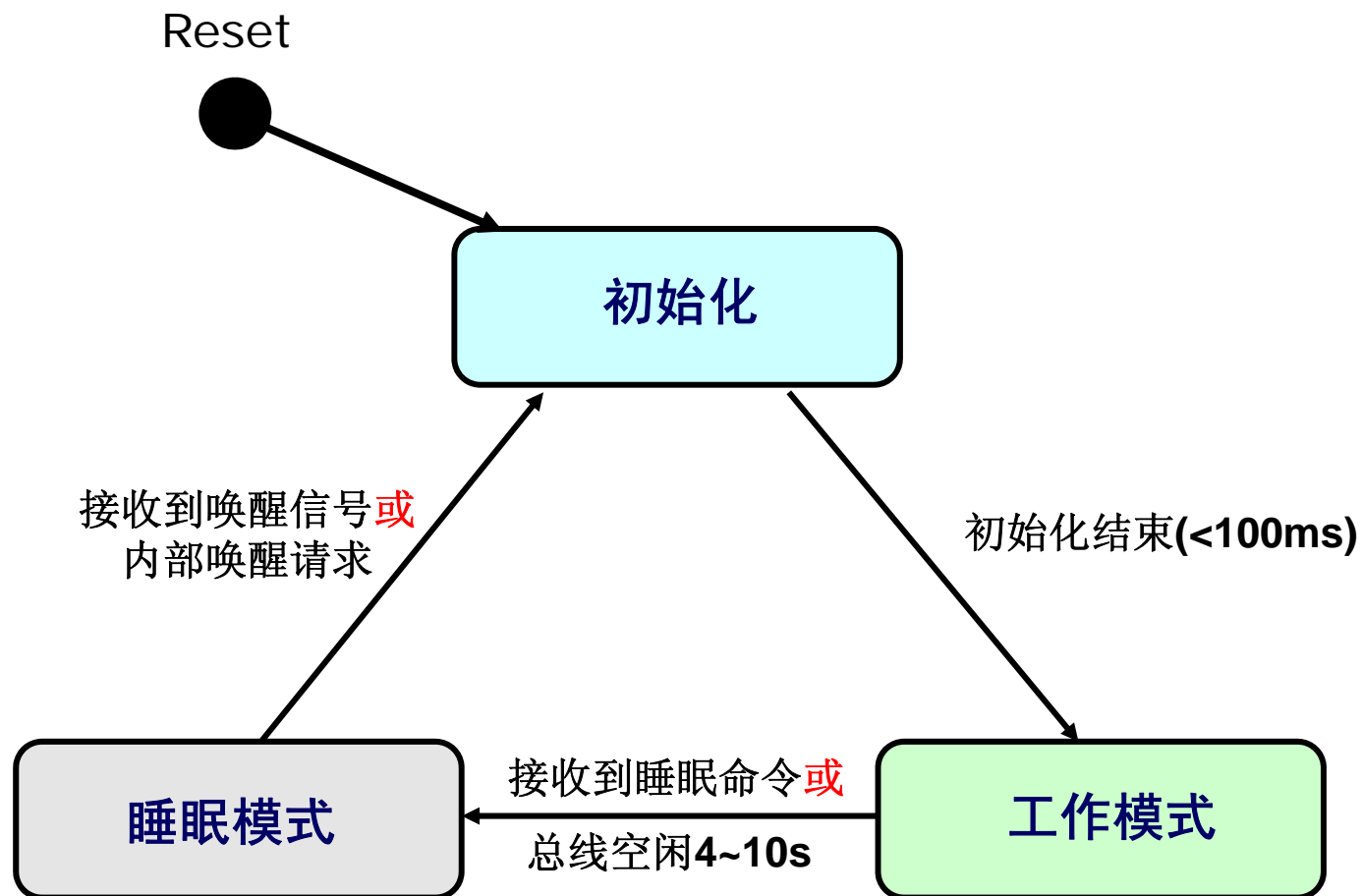
- ❑ 标识符: **63(0x3f)**

- ❑ 在**LIN 2.1**中未对保留帧进行定义



- ❑ LIN总线最小时间单位是时基( $T_{base}$ )。
- ❑ 调度表中用来发送一帧报文的时间称为帧时隙(Frame\_Slot)，帧时隙必须是时基的整数倍，调度表是由帧时隙组成的。
  - ❑  $T_{Frame\_Slot} = T_{base} * n$
- ❑ 偏移(jitter)是指一帧报文实际开始发送的时刻与帧时隙起点的时间差。
  - ❑  $T_{Frame\_Slot} > jitter + T_{Frame\_Maximum}$





- ❑ 主节点可以发送一帧ID为0x3c，第一个字节为零的主请求帧来使处于工作状态的从节点进入睡眠。这帧报文称为睡眠指令。

data1	data2	data3	data4	data5	data6	data7	data8
0	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

- ❑ 从节点在接到睡眠指令之后，也可以选择不进入睡眠状态而继续工作，这根据应用层协议而定。
- ❑ 当总线空闲4到10秒的时候，所有从节点必须进入睡眠状态。  
注：空闲的定义是没有显性位和隐性位之间的转换。

- ❑ 在一个处于睡眠状态的LIN网络中，任何一个节点都可以发送唤醒信号
- ❑ 唤醒信号是一个250us (在20Kbit/s波特率下的0xF0) 到5ms (在1Kbit/s波特率下的0xF0) 的显性电平
- ❑ 每一个从节点都需要做
  - ❑ 检测唤醒信号 (持续150us以上的显性位)
  - ❑ 当检测到唤醒信号之后，在100ms以内完成初始化工作
- ❑ 主节点除了需要完成唤醒以外，还需要检测出发送唤醒信号的节点 (利用信号)
- ❑ 当从节点发出唤醒信号之后150ms，主节点仍未发送报头时，从节点可以再次发送唤醒信号。当连续发送了3次唤醒信号之后如果主节点仍未发送报头，从节点要等待1.5秒以后才可以再次发送唤醒信号

- ❑ 状态管理的目的是来发现工作状态中总线上产生的错误
- ❑ 每个从节点将本地状态发送给主节点，由主节点来统一过滤/综合之后报告是否有节点出现错误
- ❑ 通信错误报告：  
每个从节点用**Response\_Error(1bit)**来表示自身状态，这个位需要通过响应来发送

response_error	解析
0	通信正常
1	间歇性故障
从节点未响应	总线或主节点故障

概述

协议规范

**物理层规范**

传输层规范

节点配置和标识规范

诊断规范

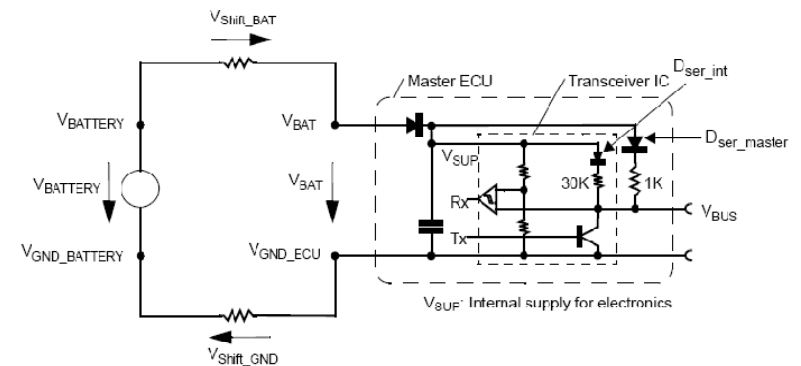
配置语言规范





## 物理层规范

# Physical Layer Specification



# 物理层规范

## 位速率容差(Bit Rate Tolerance)

### 相对于标称位速率

节点类型	符号	$\Delta F / F_{\text{NOM}}$
主节点	$F_{\text{TOL\_RES\_MASTER}}$	$< \pm 0.5\%$
不使用同步场同步的从节点	$F_{\text{TOL\_RES\_SLAVE}}$	$< \pm 1.5\%$
使用同步场同步的从节点（在同步之前）	$F_{\text{TOL\_UNSYNC}}$	$< \pm 14\%$

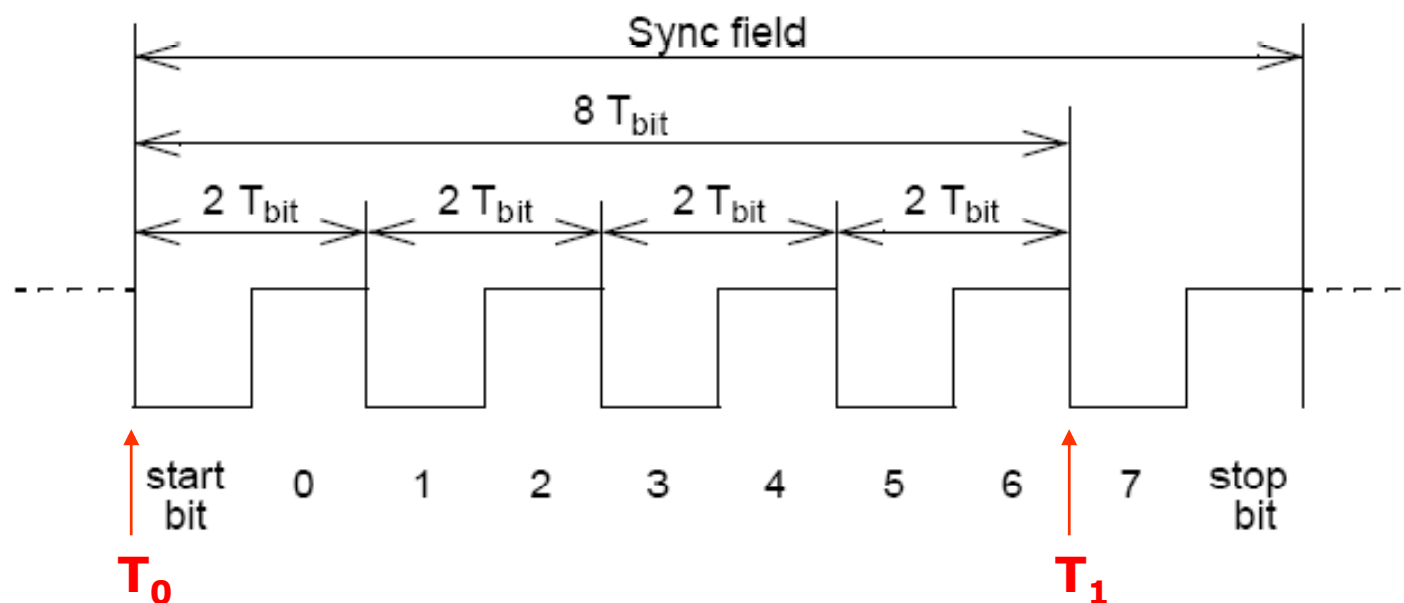
### 相对于主节点位速率

节点类型	符号	$\Delta F / F_{\text{MASTER}}$
同步后从节点	$F_{\text{TOL\_SYNC}}$	$< \pm 2\%$



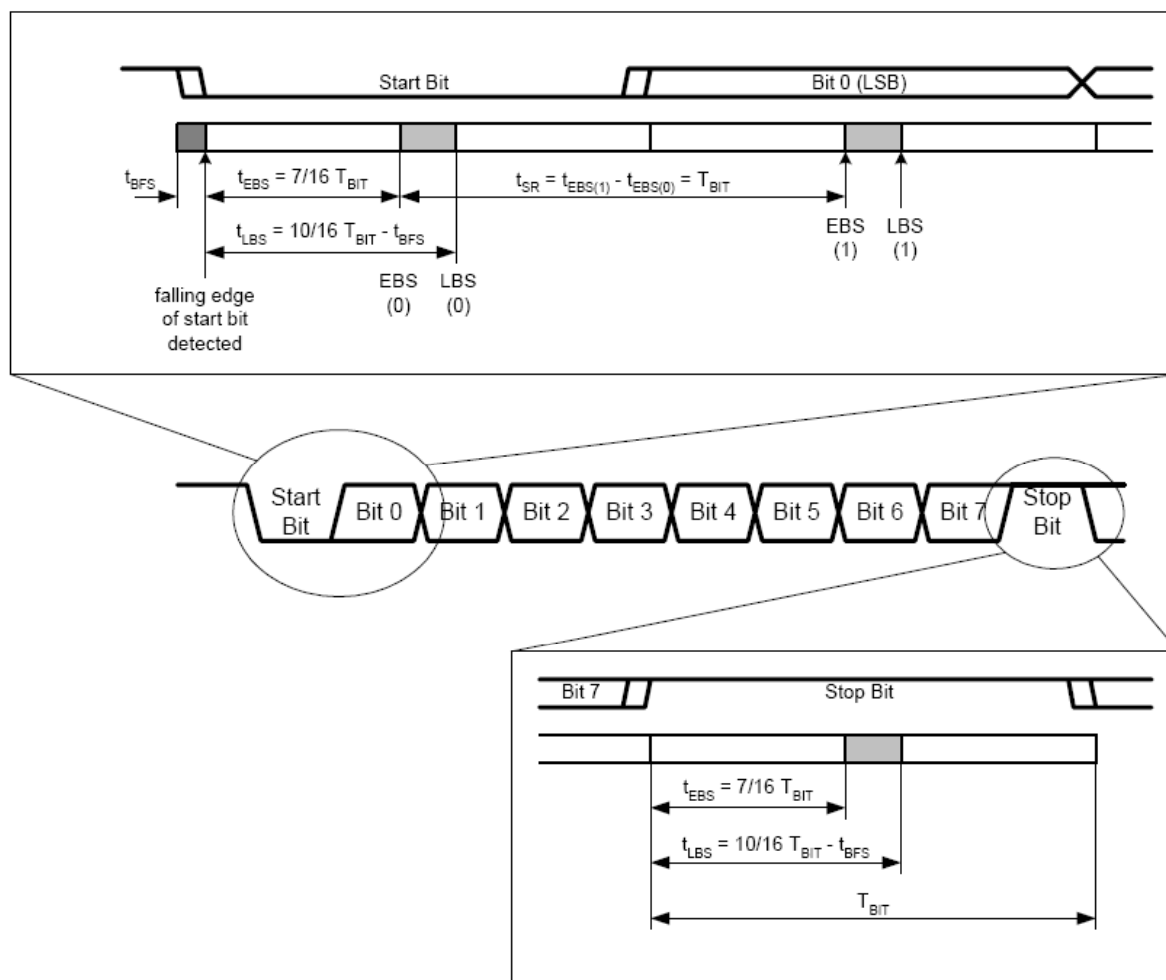
- ❑ 所有从节点的位定时必须以主节点的位定时为参考
- ❑ 同步过程

$$T_{\text{BIT}} = (T_1 - T_0) / 8$$



# 物理层规范

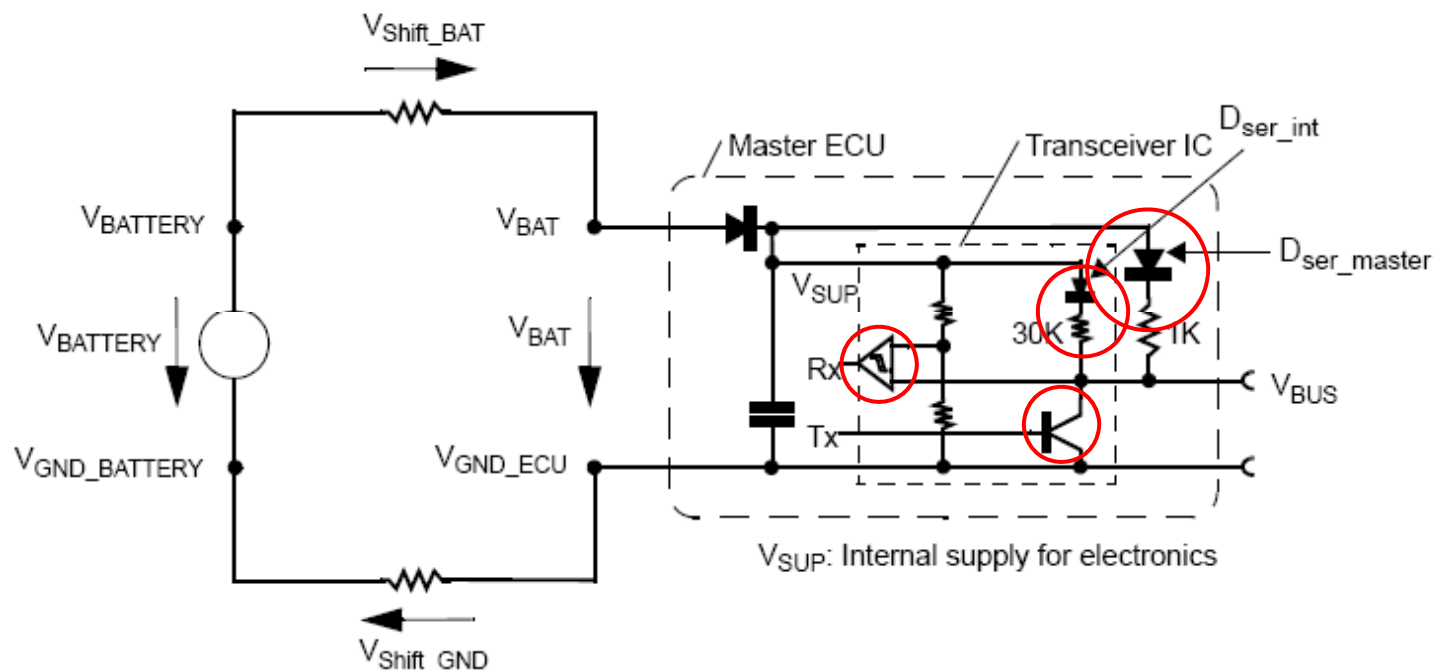
## 位采样定时(Bit Sample Timing)



- EBS = Early bit sample (前采样)
- LBS = Lastest bit sample (后采样)
- $t_{EBS} = 7/16 T_{BIT}$
- $t_{LBS} = 10/16 T_{BIT} - t_{BFS}$
- 采样点在EBS和LBS之间

# 物理层规范

## 总线收发器(Line Driver/Receiver)



- ❑ Tx通过集电极开路电路连到总线，Rx与总线之间有施密特触发器
- ❑ 收发器内部有30K上拉电阻，如果作为主节点使用，必须外接1K上拉电阻
- ❑ 上拉电阻与电源之间有二极管保护，防止在电源掉电的情况下总线电平被拉低

# 物理层规范

## 信号规范(Signal Specification)

❑  $V_{BAT}$ : 8~18V

❑  $V_{SUP} = V_{BAT} - V_{diode}$

❑ 显性位(dominant)

❑ 逻辑0

❑ 发送: 总线电平 < 20%  $V_{SUP}$

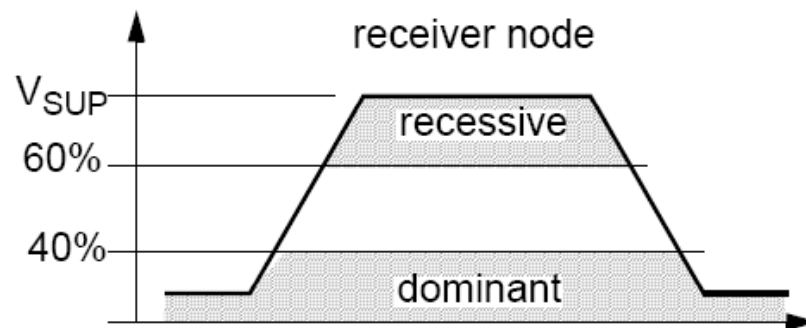
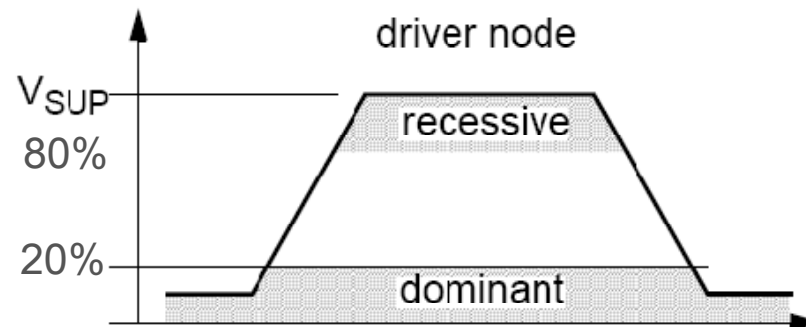
❑ 接收: 总线电平 < 40%  $V_{SUP}$

❑ 隐性位(recessive)

❑ 逻辑1

❑ 发送: 总线电平 > 80%  $V_{SUP}$

❑ 接收: 总线电平 > 60%  $V_{SUP}$



## 物理层规范

### 总线特性(Line Characteristics)(1/2)

		min	typ.	max	unit
total length of bus line	$LEN_{BUS}$			40	m
total capacitance of the bus including slave and master capacitances	$C_{BUS}$	1	4	10	nF
time constant of overall system	$\tau$	1		5	$\mu s$
capacitance of master node	$C_{MASTER}$		220		pF
capacitance of slave node	$C_{SLAVE}$		220	250	pF
line capacitance	$C'_{LINE}$		100	150	pF/m

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

$$R_{BUS} = R_{Master} \parallel R_{Slave1} \parallel R_{Slave2} \parallel \dots \parallel R_{Slave_n}$$

$$\tau = C_{BUS} \cdot R_{BUS}$$

概述

协议规范

物理层规范

**传输层规范**

节点配置和标识规范

诊断规范

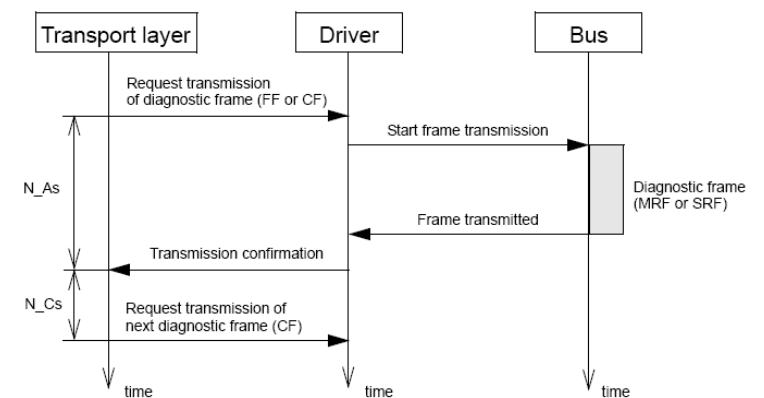
配置语言规范





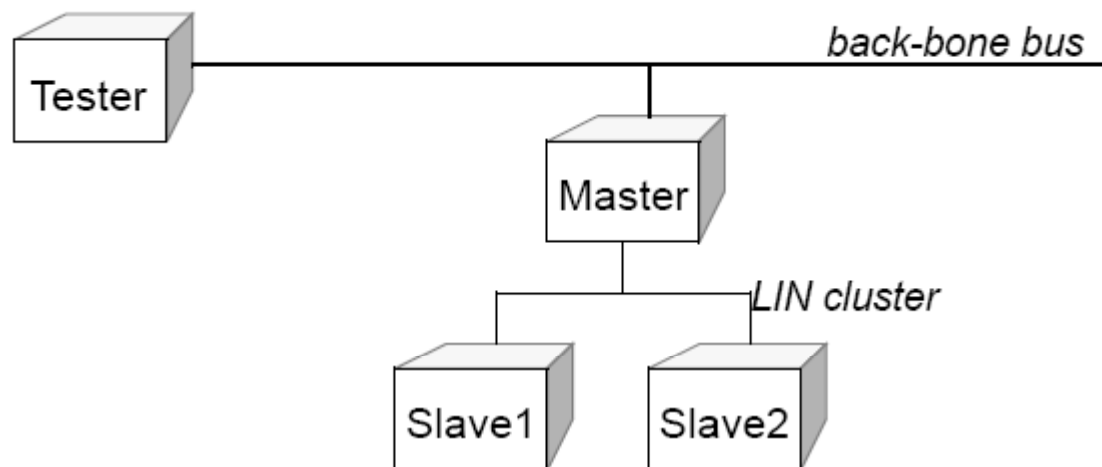
## 传输层规范

### Transport Layer Specification



### ❑ **PDU**——打包数据单元(Packet Data Unit)

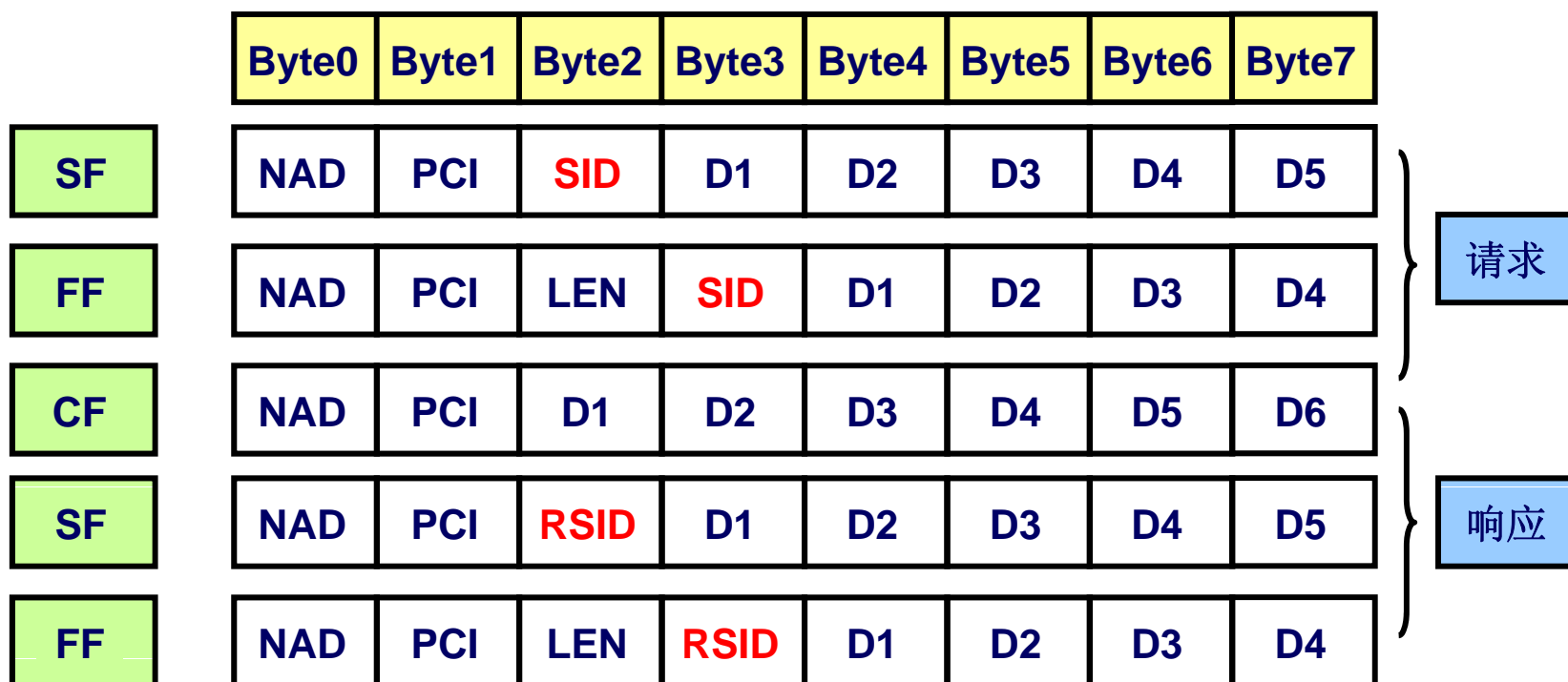
- ❑ 一个PDU可以是一个完整的报文，也可以是一个完整报文的一部分，由多个PDU来组成一个完成的报文。
- ❑ 请求：从服务器端（如测试仪，主节点）发出的报文
- ❑ 响应：从客户端（如主节点，从节点）发出的报文





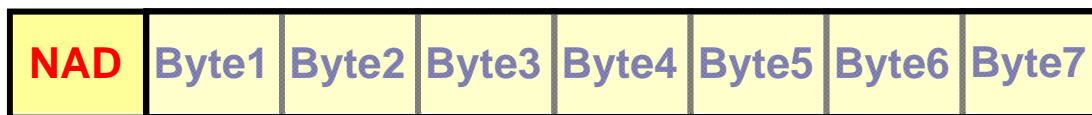
### □ PDU类型

- 单帧（**SF**）：只用一帧传输的诊断报文
- 第一帧（**FF**）：多包报文传输时，最先被发送的第一包数据帧
- 连续帧（**CF**）：多包报文传输时，在第一帧发送后，紧接着发送的数据帧



### ❑ **NAD** ——从节点地址(Node Address)

- ❑ 睡眠指令: **0**
- ❑ 物理地址: **0x01~0x7D**
  - ❑ 访问一个特定节点
- ❑ 功能地址: **0x7E** (在LIN中建议不使用)
  - ❑ 访问所有从节点
  - ❑ 从节点不允许响应
- ❑ 广播地址: **0x7F**
  - ❑ 访问所有从节点
  - ❑ 从节点将**0x7F**解析为本地物理地址
  - ❑ 从节点不响应(推荐)
- ❑ 用户自定义: **0x80~0xFF**



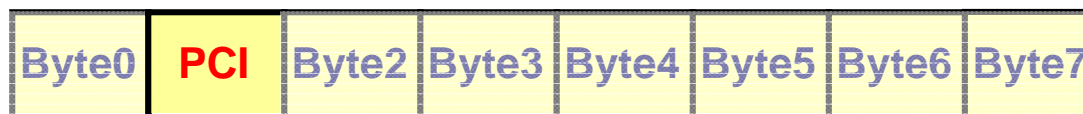
### ❑ **PCI** ——协议控制信息(Protocol Control Information)

- ❑ **PCI**包含**PDU**的类型、编号和报文的有效数据长度等信息

	PCI类型信息				附加信息			
	B7	B6	B5	B4	B3	B2	B1	B0
SF	0	0	0	0	有效数据长度			
FF	0	0	0	1	有效数据长度高4位			
CF	0	0	1	0	包编号			

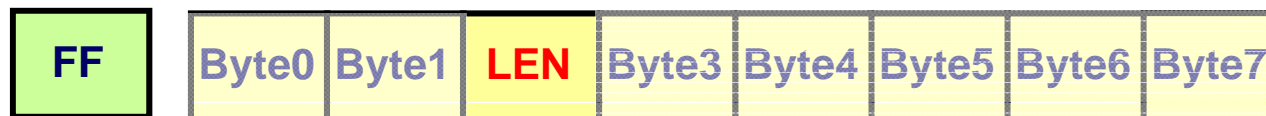
### ❑ 包编号:

- ❑ 在多个**PDU**组成的报文中，在每个连续帧**CF**中必须加入包编号，第一个**CF**的包编号是**1**，第二个**CF**的包编号是**2.....**以此类推，如果包编号超过**15**则重新从**0**开始计数



### ❑ LEN —— 帧长度(Length)

- ❑ 只存在于多PDU报文的第一帧中
- ❑ 表示报文的有效数据长度的低8位，有效数据长度的高4位在PCI的低4位中
- ❑ 由于要发送SID和RSID，所以有效数据长度 = 数据总长度 + 1



#### ❑ **SID(Service Identifier)**

- ❑ 服务标识符

#### ❑ **RSID(Response Service Identifier)**

- ❑ 对应于**SID**的响应服务标识符
- ❑ **RSID = SID + 0x40**

概述

协议规范

物理层规范

传输层规范

**节点配置和标识规范**

诊断规范

应用程序层规范

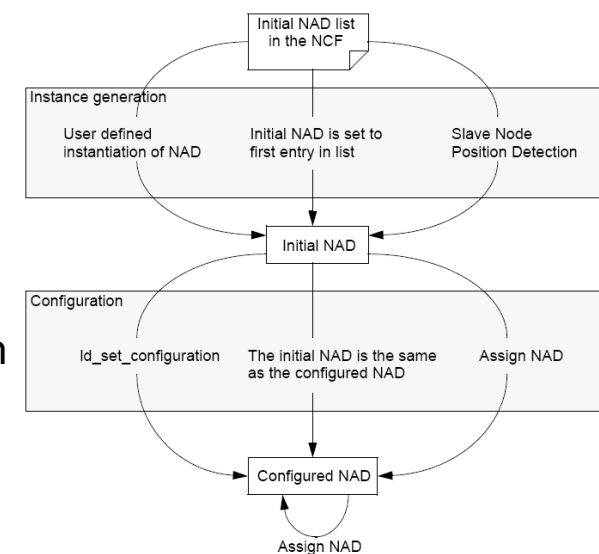
节点能力语言规范

配置语言规范



## 节点配置和标识规范

## Node configuration and Identification Specification



### ❑ 标识了节点的供应商和功能

#### ❑ 类似于条形码

D1	D2	D3	D4	D5
供应商ID 低字节	供应商ID 高字节	功能ID 低字节	功能ID 高字节	变量ID

- ❑ 供应商ID是一个16位的数字，最高位是0，由LIN协会规定
- ❑ 功能ID是一个16位的数字，由各供应商规定
- ❑ 变量ID是一个8位的数字，用于区分同网络中相同功能的节点
- ❑ PID保存在各节点的ROM中，不进行更改

### ❑ 序列号(Serial Number)

D1	D2	D3	D4
低字节	...	...	高字节



- ❑ **NAD**的通配符**0x7F**用来访问所有节点的地址
- ❑ 供应商**ID**和功能**ID**可以代表所有未定义的供应商与功能

属性	通配符
<b>NAD</b>	<b>0x7F</b>
供应商 <b>ID</b>	<b>0x7FFF</b>
功能 <b>ID</b>	<b>0xFFFF</b>

# 节点配置和标识规范

## 与配置相关的服务(Service Associated with Configuration)

- ❑ 服务标识符 (**SID**)
  - ❑ **0xB0~0xB7**
- ❑ 节点配置只能用单帧通信

服务	服务
0~0xAF	Reserved
0xB0	Assign NAD
0xB1	Assign frame identifier
0xB2	Read by Identifier
0xB3	Conditional Change NAD
0xB4	Data Dump
0xB5	Assign NAD via SNPD
0xB6	Save Configuration
0xB7	Assign frame identifier range
0xB8~0xFF	Reserved

概述

协议规范

物理层规范

传输层规范

节点配置和标识规范

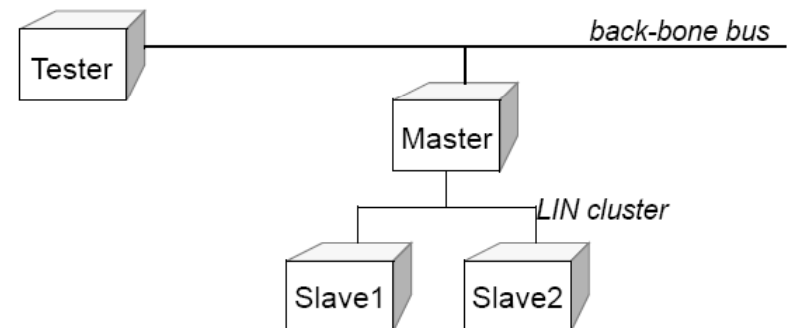
**诊断规范**

配置语言规范



# 诊断规范

## Diagnostic Specification

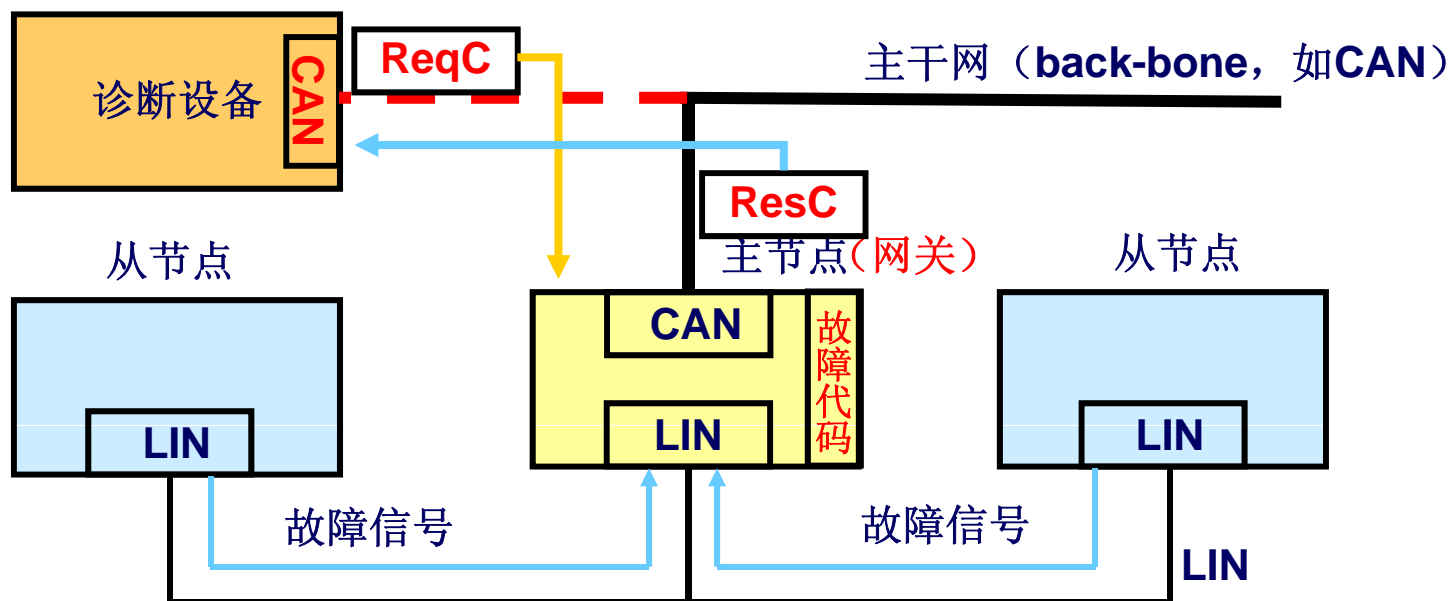


# 诊断规范

## 两种诊断方式(Diagnostic Methods)(1/2)

### ❑ 第一种方式:

- ❑ 从节点将故障信号发送到主节点（基于信号）
- ❑ 主节点将各故障信号处理、存储为故障代码(DTC)
- ❑ 诊断设备请求故障代码
- ❑ 主节点发送故障代码

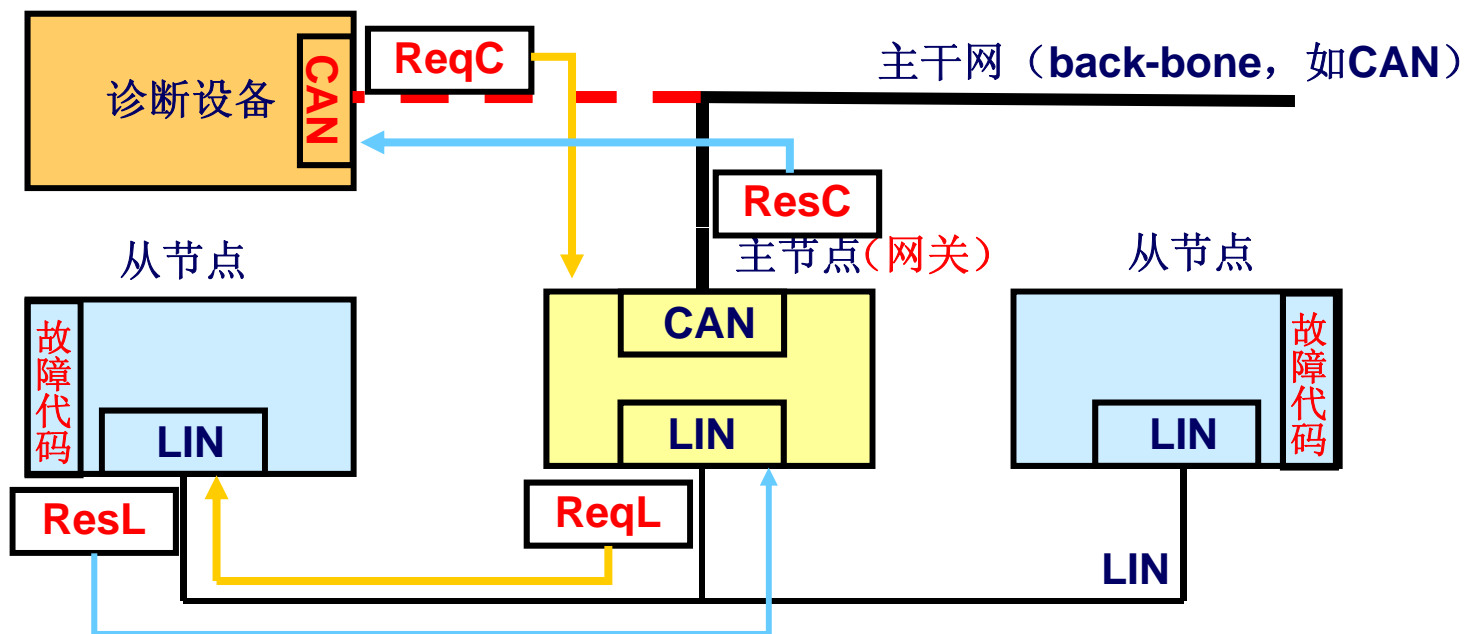


# 诊断规范

## 两种诊断方式(Diagnostic Methods)(2/2)

### □ 第二种方式:

- 诊断设备连接到主干网
- 主节点作为网关
- 从节点存储故障代码(DTC)
- 主节点转发请求与响应
- 适用于所有诊断服务



### ❑ 从节点根据自身的特点和需要的诊断功能分为**3**个级别

#### ❑ 诊断级别 **1**

- ❑ 从节点主要由简单元件组成(如传感器等)，故障都由主节点来读取、分析处理和保存

##### ❑ 传输层

- ❑ 只有在进行节点配置的时候才用到诊断帧，单帧(**SF**)就能够满足需求

##### ❑ 诊断服务

- ❑ 只支持节点配置服务

### ❑ 诊断级别 2

- ❑ 和诊断级别**1**的节点类似，但是支持节点标识(产品**id**和序列号)，故障同样由主节点来读取、分析处理和保存

#### ❑ 传输层

- ❑ 支持传输层全部功能

#### ❑ 诊断服务

- ❑ 可以支持节点标识服务，读取参数服务，写参数服务，但不硬性规定
- ❑ 如果节点同时支持级别**1**和级别**2**，不要求必须遵守级别**2**



### ❑ 诊断级别 3

- ❑ 诊断级别**3**的节点是具有较强应用功能的节点，通常在本地处理各种工作(如具有本地传感器，本地执行器)。除了基本的节点配置服务以外，还支持**ISO 14229-1**的其他诊断服务
- ❑ 诊断级别**3**的节点具有内部故障存储设备，有些还支持在线刷写，需要完整的**boot-loader**工具和相应的诊断服务去实现

#### ❑ 节点地址

- ❑ 每个节点都需要有唯一的**NAD**

#### ❑ 传输层

- ❑ 支持传输层全部功能，必须支持多帧传输

#### ❑ 诊断服务

- ❑ 诊断级别**3**的节点支持所有诊断级别**2**所支持的服务，除此之外，还可以支持其他的**UDS**服务
- ❑ 只有诊断级别**3**的节点才能支持**FLASH**在线刷写以及相关服务

# 诊断规范

## 诊断级别总结(Summary of Diagnostic Class)(1/2)

诊断级别	1	2	3	UDS服务号 [HEX]
传输协议层需求				
仅单帧传输	+			
完整传输协议(包括多帧)		+	+	
需要的配置服务				
Assign frame identifier range	+	+	+	0xB7
Read by identifier	+	+	+	0xB2
Assign NAD	可选	可选	+	0xB0
Conditional change NAD	可选	可选	可选	0xB3
Positive response on supported configuration services	+	+	+	service+ 0x40

# 诊断规范

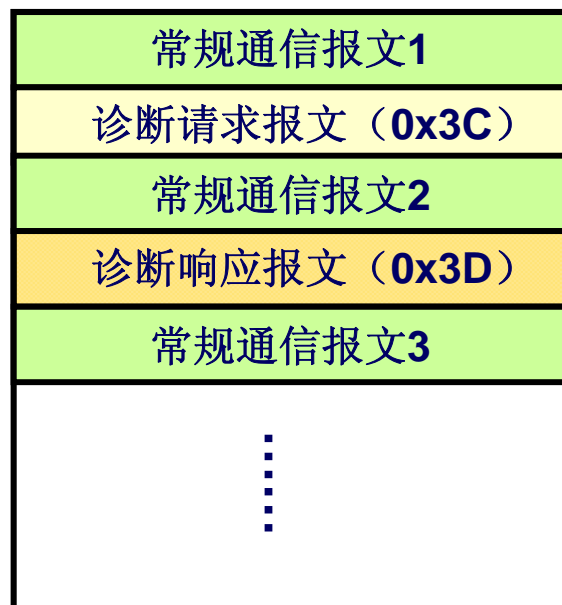
## 诊断级别总结(Summary of Diagnostic Class)(2/2)

诊断级别	1	2	3	UDS服务号 [HEX]
需要的UDS服务				
Read data by identifier				0x22
hardware and software version		+	+	0x22
hardware part number (OEM specific)		+	+	0x22
diagnostic version		+	+	0x22
Read by identifier (parameters)		+	+	0x22
Write by identifier (parameters)		如果支持	如果支持	0x2E
Read by identifier (sensor and actuator data)			+	0x22
I/O control by identifier			+	0x2F
Read and clear DTC (fault memory)			+	0x19,0x14
Routine control			如果支持	0x31
Other diagnostic services			如果支持	...
Flash刷写服务				
Flash programming services			可选	/

### ❑ 交叉诊断模式

❑ 默认模式

❑ 不终止常规通信



# 诊断规范

## 纯诊断模式(Diagnostics Only Mode)

### ❑ 纯诊断模式

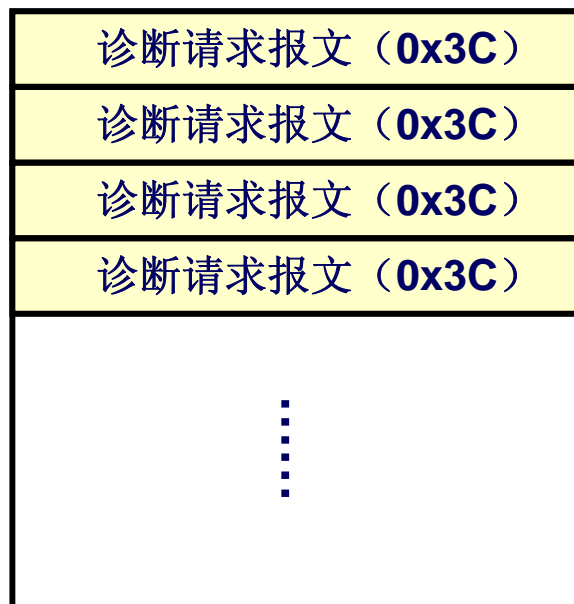
❑ 可选模式

❑ 终止常规通信

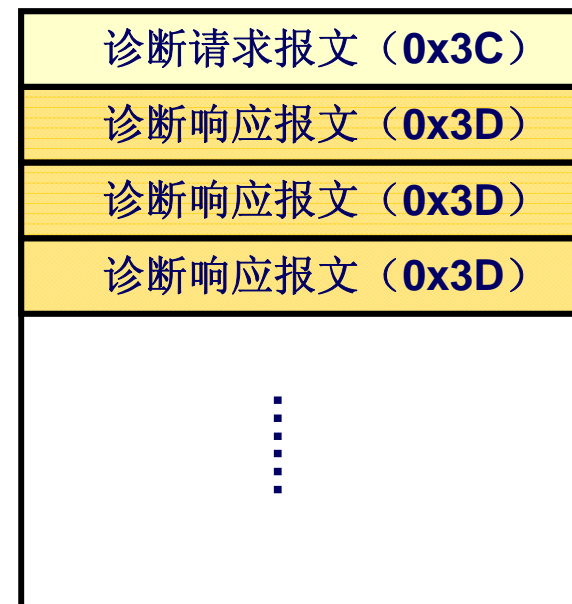
#### 单帧传输



#### 多帧传输 (数据来自主节点)



#### 多帧传输 (数据来自从节点)



概述

协议规范

物理层规范

传输层规范

节点配置和标识规范

诊断规范

**配置语言规范**



## 配置语言规范

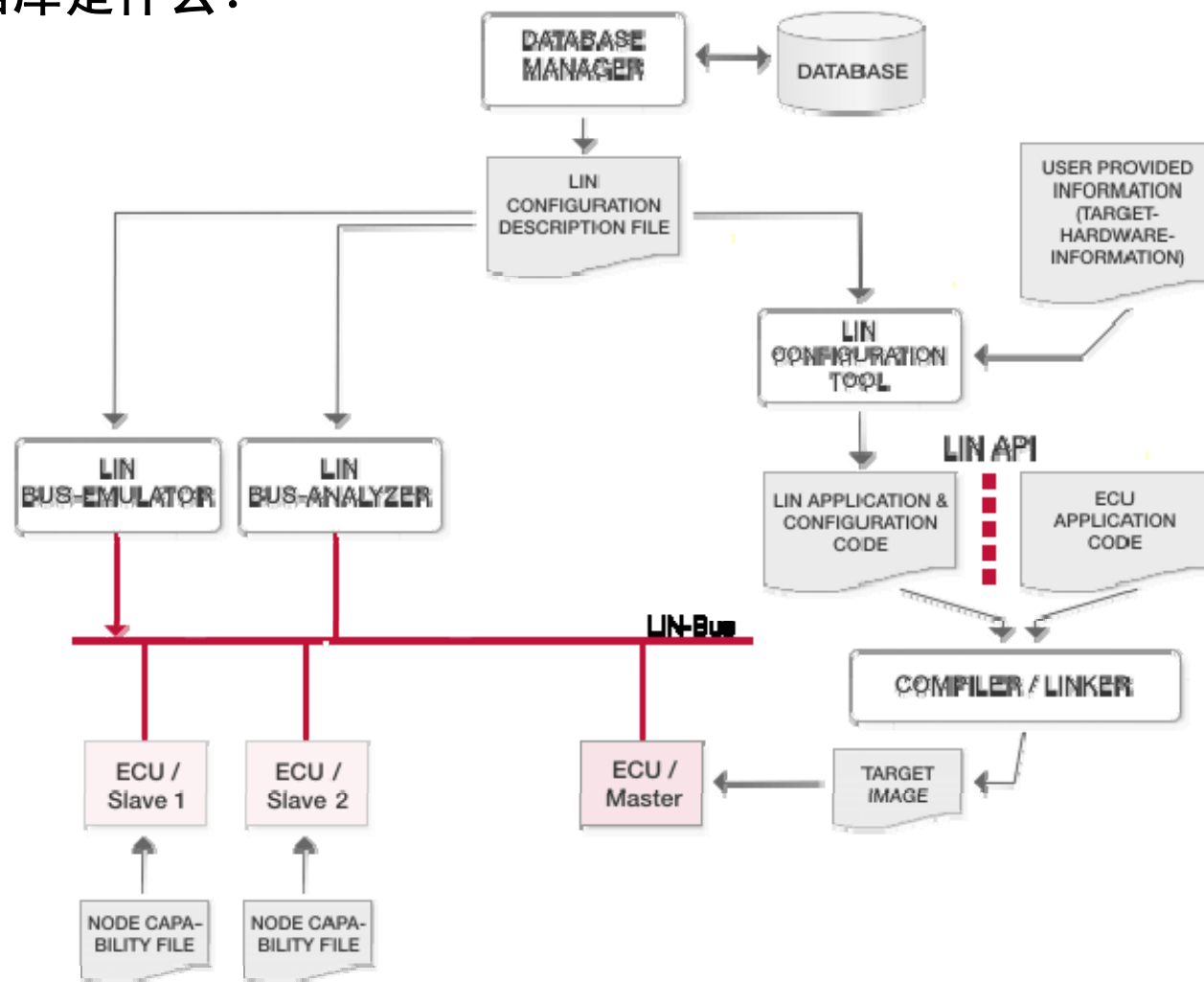
LIN Description File



# 配置语言规范

## LDF在LIN开发流程中(LDF in LIN Workflow)

### ❑ 数据库是什么？

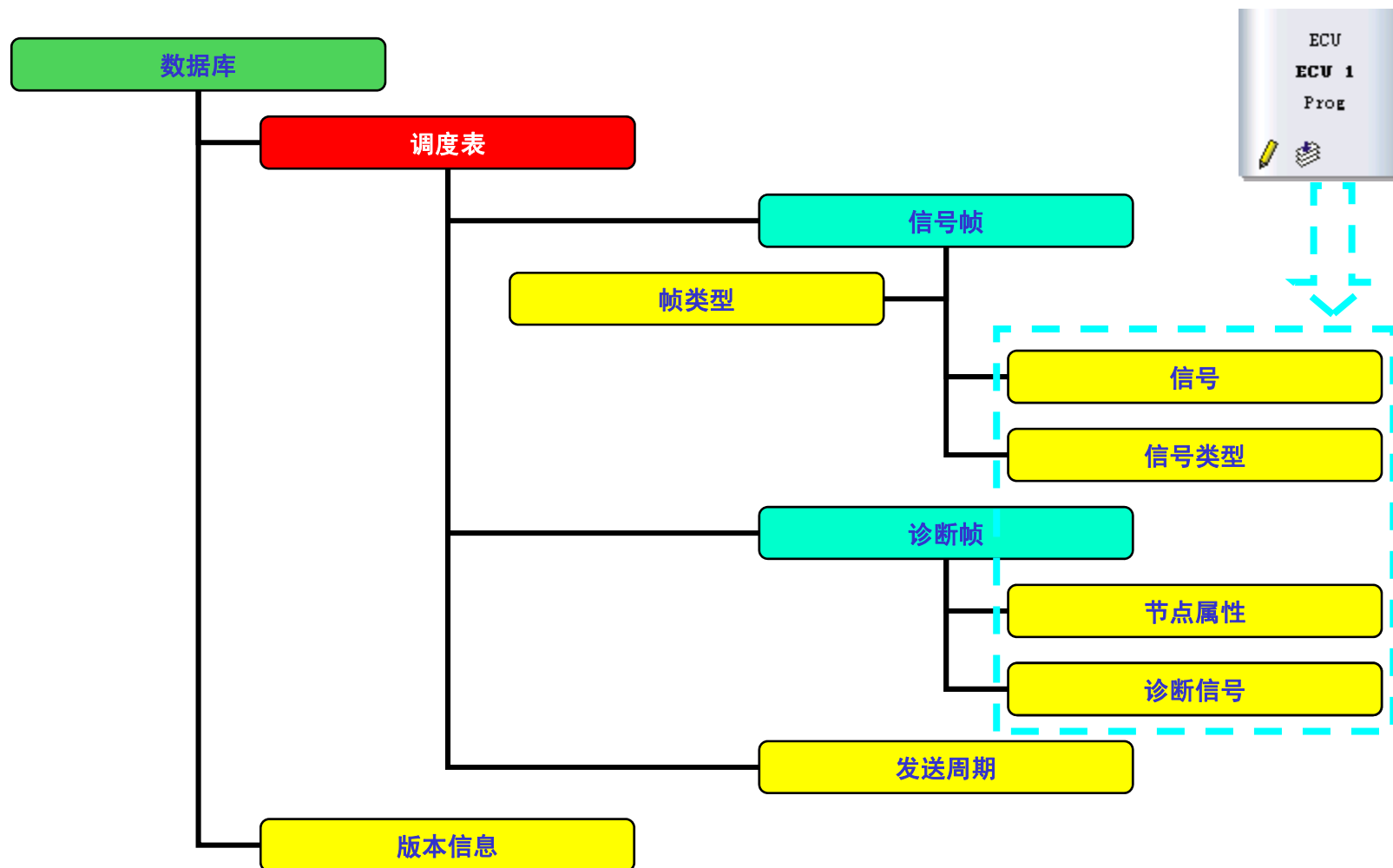




# 配置语言规范

## LIN数据库构成(Composition of LIN database)

### ❑ 需要在数据库中定义些什么？



### ❑ **LIN**描述文件定义

#### ❑ 全局定义

#### ❑ 节点定义

##### ❑ 参与节点定义

##### ❑ 节点属性定义

##### ❑ 节点构成定义

#### ❑ 信号定义

#### ❑ 帧定义

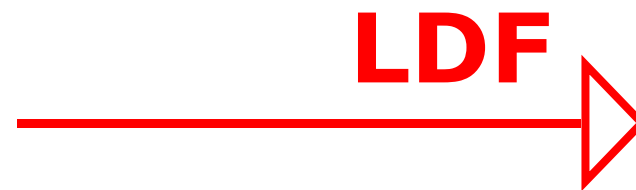
#### ❑ 调度表定义

#### ❑ 附加信息

##### ❑ 信号编码类型

##### ❑ 信号表示类型

# 创建自己的数据库



# 配置语言规范

## 我的第一个数据库(My first database)

```
LIN_description_file;  
LIN_protocol_version = "2.0";  
LIN_language_version = "2.0";  
LIN_speed = 9.6 kbps;
```

```
Nodes{  
  Master: Mymaster, 10 ms, 0 ms;  
  Slaves: Myslave;  
}
```

```
Signals{  
  My_first_signal: 1, 0, Mymaster, Myslave;  
}
```

```
Frames{  
  Myframe: 1, Mymaster, 1{  
    My_first_signal, 0;  
  }  
}
```

```
Schedule_tables{  
  Myscheduletable{  
    Myframe delay 20 ms;  
  }  
}
```

- ❑ LIN 协议版本号
- ❑ LIN 配置语言版本号
- ❑ LIN 总线速率

### 节点定义

- ❑ 主节点, 时基, 抖动
- ❑ 从节点

### 信号定义

- ❑ 信号: 长度, 初始值, 发送节点, 接受节点

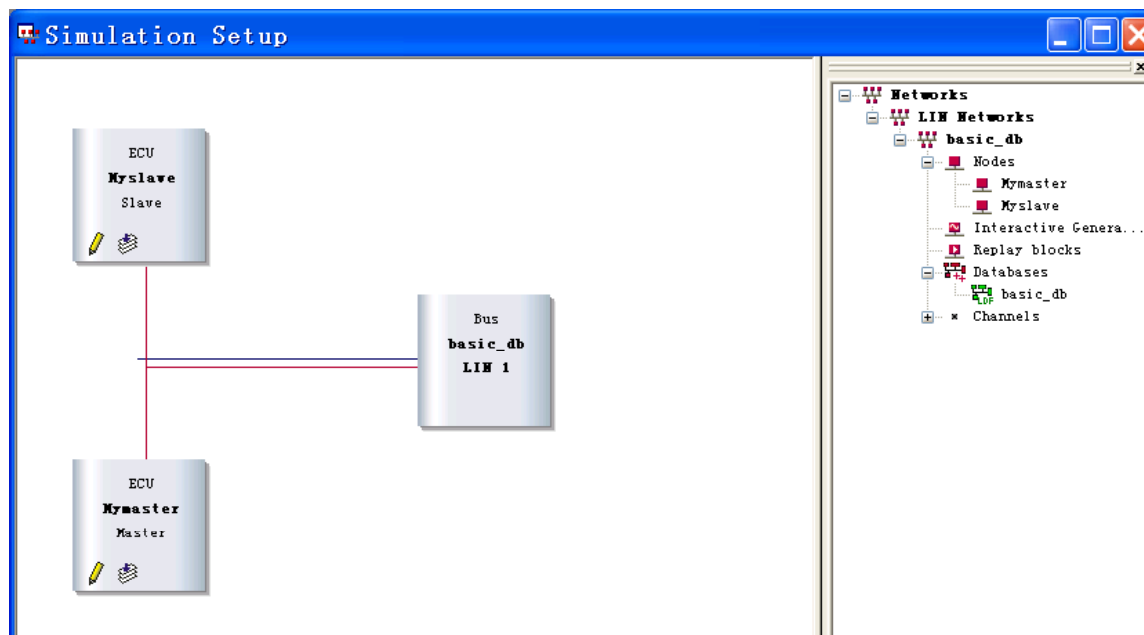
- ❑ 帧: ID, 发送节点, 长度
- ❑ 信号列表, 起始位

### 调度表定义

- ❑ 调度表
- ❑ 帧列表, 时隙长度

# 配置语言规范

## 验证(Verify)



□ 在CANoe中验证

**Great, I See it!**

Trace

Time	Start of Frame	C...	Dir	Event Type	Frame Name	Id	Data	Publisher
43.7...	43.780000	L...	Tx	LIN Frame (Unconditional)	Myframe	01	FE	Mymaster (simulated)

My\_first\_signal 0 [ 0]

# 配置语言规范

## 进阶(Advance)

- ❑ 尝试着给数据库增加新元素
- ❑ 第一步，建立一个更复杂的系统

### Nodes{

Master: Mymaster, 10 ms, 0 ms;

Slaves: Myslave1, Myslave2;

}

### Signals{

My\_signal1: 1, 0, Mymaster, Myslave1;

My\_signal2: 1, 0, Mymaster, Myslave2;

My\_signal3: 1, 0, Mymaster, Myslave1, Myslave2;

My\_signal4: 2, 0, Myslave1, Mymaster;

My\_signal5: 2, 0, Myslave2, Mymaster;

}

现在我们的数据库有了：

- ❑ 1个主节点，2个从节点
- ❑ 5个信号
- ❑ 4个帧
- ❑ 1个4时隙的调度表

### Frames{

Myframe1: 11, Mymaster, 2{

My\_signal1, 0;

My\_signal2, 1;

}

Myframe2: 12, Mymaster, 1{

My\_signal3, 0;

}

Myframe3: 13, Myslave1, 1{

My\_signal4, 3;

}

Myframe4: 14, Myslave2, 1{

My\_signal5, 2;

}

}

### Schedule\_tables{

Myscheduletable{

Myframe1 delay 20 ms;

Myframe2 delay 20 ms;

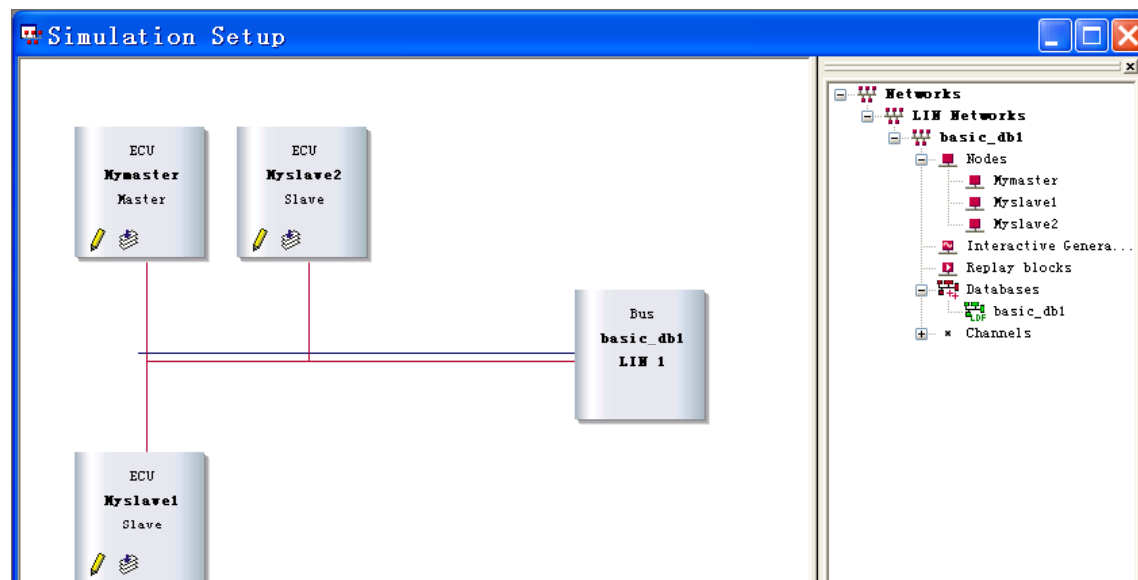
Myframe3 delay 20 ms;

Myframe4 delay 20 ms;

}

}

# 配置语言规范 验证(Verify)



在CANoe中验证

**It becomes better!**

Trace window showing a list of LIN frames and their associated signals. The table has columns: Time, Start of Frame, C..., Dir, Event Type, Frame Name, Id, Data, and Publisher.

Time	Start of Frame	C...	Dir	Event Type	Frame Name	Id	Data	Publisher
23.2...	23.200000	L...	Tx	LIN Frame (Unconditional)	Myframe1	0B	FC FF	Mymaster ...
				My_signal1		0	[ 0]	
				My_signal2		0	[ 0]	
23.2...	23.220000	L...	Tx	LIN Frame (Unconditional)	Myframe2	0C	FE	Mymaster ...
				My_signal3		0	[ 0]	
23.2...	23.240000	L...	Tx	LIN Frame (Unconditional)	Myframe3	0D	E7	Myslave1 ...
				My_signal4		0	[ 0]	
23.2...	23.260000	L...	Tx	LIN Frame (Unconditional)	Myframe4	0E	F3	Myslave2 ...
				My_signal5		0	[ 0]	

# 配置语言规范

## 事件触发帧和零星帧(ETF & Sporadic frame)

- ❑ 为数据库来点调味料!
- ❑ 增加事件触发帧和零星帧

```
Event_triggered_frames {  
    ETF_Myframes: ETFCollisionResolving, 58,  
    Myframe3, Myframe4 ;  
} //将Myframe3和Myframe4关联成事件触发帧  
ETF_Myframes  
  
Schedule_tables {  
    NormalTable {  
        ETF_Myframes delay 20 ms ;  
    } //在调度表中只需放入ETF_Myframes即可!  
  
    ETFCollisionResolving {  
        Myframe3 delay 20 ms ;  
        Myframe4 delay 20 ms ;  
    } //万一发生冲突了, 就执行这个解决ETF冲突调度表!!  
}
```

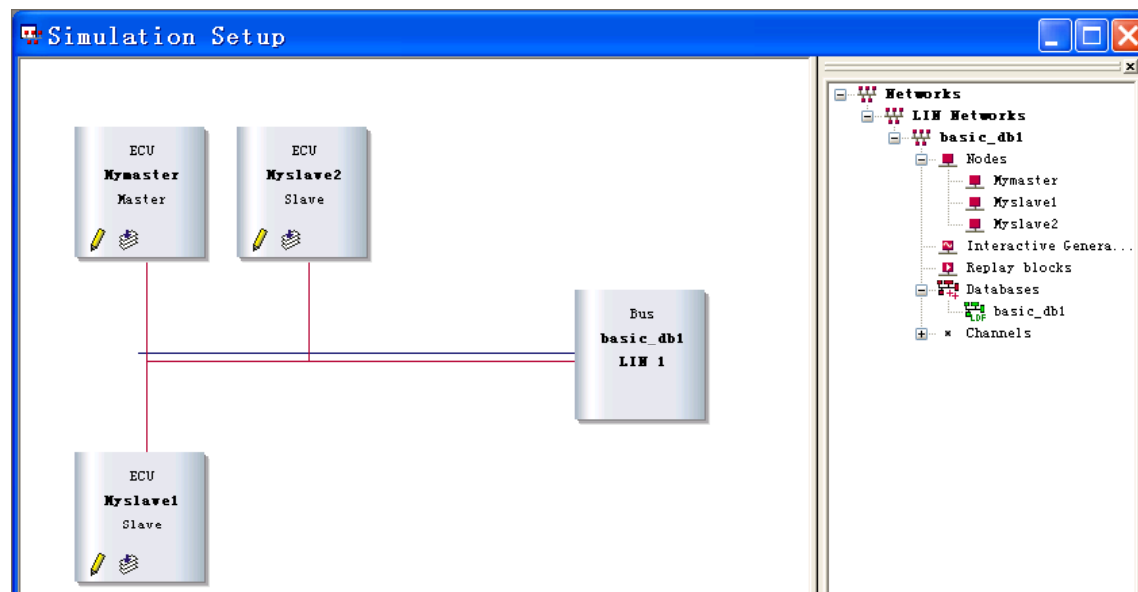
```
Sporadic_frames {  
    MySporadicFrame: Myframe1, Myframe2 ;  
} //将Myframe1和Myframe2关联成零星帧, 注意! 它们必须  
//都由Master发送  
  
Schedule_tables {  
    NormalTable {  
        MySporadicFrame delay 20 ms ;  
    } //在调度表中放入零星帧即可!
```

数据库中有形有色!



# 配置语言规范

## 验证(Verify)



### 思考

#### 在Trace窗口中

- 为什么出现错误?
- 为什么没有零星帧?

Time	Start of Frame	C...	Dir	Event Type	Frame Name	Id	Data	Publisher
47.2...	47.260000	L...		TransmError (ETF no response)	ETF_Myframes	3A		

□ 上帝说：“要有诊断”

于是就有了诊断！

□ 给数据库添加诊断功能

**Diagnostic\_signals {**

```
MasterReqB0: 8, 0 ;
MasterReqB1: 8, 0 ;
MasterReqB2: 8, 0 ;
MasterReqB3: 8, 0 ;
MasterReqB4: 8, 0 ;
MasterReqB5: 8, 0 ;
MasterReqB6: 8, 0 ;
MasterReqB7: 8, 0 ;
SlaveRespB0: 8, 0 ;
SlaveRespB1: 8, 0 ;
SlaveRespB2: 8, 0 ;
SlaveRespB3: 8, 0 ;
SlaveRespB4: 8, 0 ;
SlaveRespB5: 8, 0 ;
SlaveRespB6: 8, 0 ;
SlaveRespB7: 8, 0 ;
```

**}**

诊断信号的定义

**Diagnostic\_frames {**

```
MasterReq: 0x3c {
    MasterReqB0, 0 ;
    MasterReqB1, 8 ;
    MasterReqB2, 16 ;
    MasterReqB3, 24 ;
    MasterReqB4, 32 ;
    MasterReqB5, 40 ;
    MasterReqB6, 48 ;
    MasterReqB7, 56 ;
```

**}**

```
SlaveResp: 0x3d {
    SlaveRespB0, 0 ;
    SlaveRespB1, 8 ;
    SlaveRespB2, 16 ;
    SlaveRespB3, 24 ;
    SlaveRespB4, 32 ;
    SlaveRespB5, 40 ;
    SlaveRespB6, 48 ;
    SlaveRespB7, 56 ;
```

**}**

**}**

诊断帧的定义

### ❑ 每个节点都有独有的节点属性

**Node\_attributes {**

**Myslave1{**

**LIN\_protocol = "2.0" ;**

**configured\_NAD = 0x2 ;**

**initial\_NAD = 0xa ;**

**product\_id = 0x1e, 0x1, 0 ;**

**response\_error = Myslave1Error ;**

**P2\_min = 100 ms ;**

**ST\_min = 20 ms ;**

**N\_As\_timeout = 1000 ms ;**

**N\_Cr\_timeout = 1000 ms ;**

**configurable\_frames {**

**Myframe3 = 0x01;**

**ETF\_Myframes = 0x02;**

**}**

**}**

**}**

这些是它的条形码

可以通过assign frame id服务来配置它的帧ID

### ❑ 把诊断帧放进调度表里去

#### ❑ 第一种实现方法:

```
Schedule_tables{  
    MyDiag1{  
        MasterReq delay 20 ms;  
        SlaveResp delay 20 ms;  
    }  
}
```

留下诊断调度表，让应用层去调用诊断API函数

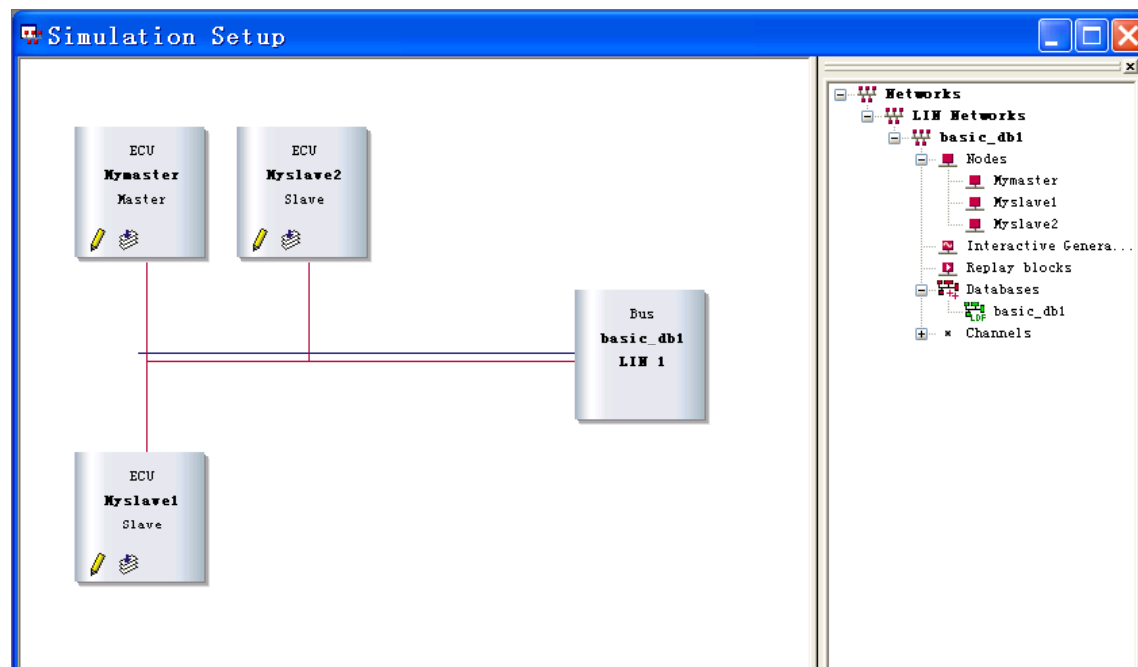
#### ❑ 第二种实现方法:

```
Schedule_tables{  
    MyDiag1{  
        AssignNAD { 0xa, 0x4, 0x1e, 0x1 } delay 20 ms ;  
        SlaveResp delay 20 ms;  
        AssignFrameId { Myslave1, Myframe3 } delay 20 ms ;  
        SlaveResp delay 20 ms;  
    }  
}
```

在数据库中直接定义具体诊断服务，应用层不调用API函数

# 配置语言规范

## 验证(Verify)



- ❑ 总线上出现了诊断帧!
- ❑ 可以随意配置节点了

Time	Start ...	Dir	Event Type	Frame Name	Id	Data	Publisher
56.8...	56.82...	L...	Tx	LIN Frame (Configuration Response)	SlaveResp ("Positive response")	3D 0A 01 F0 FF FF FF FF FF	Myslave1...
56.7...	56.76...	L...	Tx	LIN Frame (Configuration Request)	MasterReq ("AssignNAD")	3C 0B 06 B0 1E 00 01 00 03	Mymaster...
56.7...	56.78...	L...	Tx	LIN Frame (Configuration Response)	SlaveResp ("Positive response")	3D 0B 01 F0 FF FF FF FF FF	Myslave2...
56.8...	56.80...	L...	Tx	LIN Frame (Configuration Request)	MasterReq ("AssignNAD")	3C 0A 06 B0 1E 00 01 00 02	Mymaster...

# 配置语言规范

## 附：常用的节点配置函数

函数名	参数
AssignNAD {initial_NAD, conf_NAD,supplier ID, function ID}	{初始NAD, 配置NAD, 供应商ID, 功能ID}
AssignFrameId {message_ID, PID}	{报文ID, PID}
FreeFormat { D0,D1,D2,D3,D4,D5,D6,D7}	{ MasterReq的数据场的8个字节 }

# 配置语言规范

## 信号编码类型和信号表示

- ❑ 看不懂这信号是什么意思？
- ❑ 信号编码类型和信号表示

```
Signal_encoding_types {  
    Mysignal_Type {  
        logical_value, 0, "no ";  
        logical_value, 1, "yes";  
    }  
} //信号编码类型，相当于c语言中的typedef
```

```
Signal_representation {  
    Mysignal_Type : My_signal1, My_signal2;  
}  
//用刚才定义好的信号编码类型去对信号进行分类
```

让信号更生动！

# 配置语言规范

## 验证(Verify)

Simulation Setup

Networks

- LIN Networks
  - basic\_db1
    - Nodes
      - Mymaster
      - Myslave1
      - Myslave2
    - Interactive Genera...
    - Replay blocks
    - Databases
      - basic\_db1
    - Channels

Trace

Time	Start ...	C...	Dir	Event Type	Frame Name	Id	Data	Publisher
16.8...	16.80...	L...	Tx	LIN Frame (Unconditional)	Myframe1	0B	FC FF	Mymaster...
				My_signal1	[ 0]			
				My_signal2	[ 0]			
16.8...	16.82...	L...	Tx	LIN Frame (Unconditional)	Myframe2	0C	FE	Mymaster...
16.8...	16.84...	L...	Tx	LIN Frame (Unconditional)	Myframe3	0D	E7	Myslave1...
16.7...	16.78...	L...	Tx	LIN Frame (Unconditional)	Myframe4	0E	F3	Myslave2...

□ 请注意信号边上的“NO”