第 16 章　CAN 总线原理和应用

控制器局域网络(controller area network,CAN)是一种功能丰富的车用总线标准。该总线允许在没有主机的情况下,实现网络上单片机和设备之间的相互通信。它是基于消息传递协议,最初是在车辆上复用通信电缆,以降低铜导线的使用量,后来也用于其他行业。

本章首先详细介绍了 CAN 总线协议规范,在此基础上详细介绍了 STC32G 系列单片机中所集成的 CAN 模块功能,最后通过设计实例说明基于 CAN 总线的通信实现方法。

16.1　CAN 规范基础

本节介绍 CAN 规范,内容包括发展历史、总线架构、总线节点、消息传输、消息过滤、消息验证、位流编码、错误管理、故障界定和位时序要求。

16.1.1　发展历史

博世公司(Robert Bosch GmbH)于 1983 年开发了控制器局域网(controller area network,CAN)。该协议于 1986 年在美国密歇根州底特律市举办的国际汽车工程师学会(society of automotive engineers,SAE)会议上正式发表。直到 1987 年,Intel(英特尔)公司发布了第一个 CAN 控制器芯片 82526,随后 Philips(飞利浦)公司发布了 CAN 控制器芯片 82C200。1991 年,在梅赛德斯-奔驰 W140 上搭载了基于 CAN 的连线系统。

博世公司发布了关于 CAN 规范的几个版本,最新的 CAN 2.0 于 1991 年发布。该规范分成两个部分。其中,A 部分适用于使用 11 位标识符的标准格式;B 部分适用于使用 29 位标识符的扩展格式。通常,将使用 11 位标识符的 CAN 设备称为 CAN 2.0A,将使用 29 位标识符的 CAN 设备称为 CAN 2.0B。

在 1990 年早些时候,博世公司的 CAN 2.0 提交给国际标准化组织。在经历多次讨论后,在一些法国汽车厂商的要求下,增加了汽车局域网络(vehicle area network,VAN)的内容,并于 1993 年颁布了 CAN 的国际标准 ISO11898。除了 CAN 协议外,它也规定了最高到 1Mbps 波特率时的物理层。后来,CAN 标准重新编译分成两部分。其中,ISO11898-1 涵盖了数据链路层;ISO11898-2 涵盖了高速 CAN 总线的物理层。

后来,晚些时候公布了 ISO11898-3 涵盖了低速 CAN 总线的物理层和 CAN 总线容错规范,其传输速率为 40~125kb/s。它使用线性主线、星型主线或者连接到一个线性主线上的多星结构,如图 16.1 所示。每个节点都有终端电阻作为全局终端电阻的一部分。全局终端电阻不小于 100Ω。

注:在 CAN2.0 规范中并不包含 ISO11898-2 和 ISO11898-3,需要单独从国际标准化组织(International Organization for Standardization,ISO)购买。

后来,博世公司仍然在积极地扩展 CAN 标准。2012 年,博世公布 CAN FD 1.0(也称为可变数据速率的 CAN)。该规范使用不同的架构,允许在仲裁之后,切换到更快的比特率,传输不同的数据长度。CAN FD 兼容现有的 CAN 2.0 网络,因此新的 CAN FD 设备能够与现有的 CAN 设备共存于同一网络。

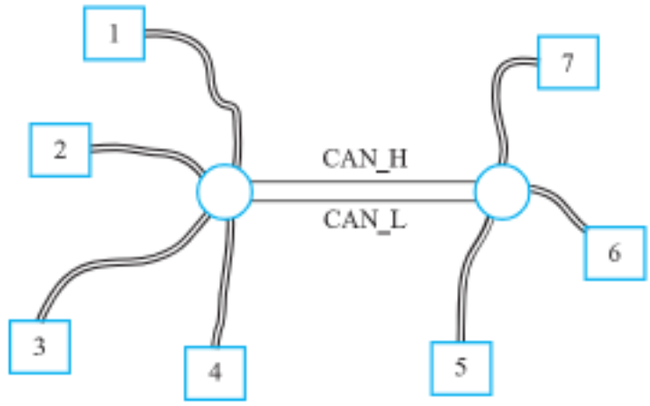


图 16.1　低速 CAN 拓扑结构

16.1.2　总线架构

CAN 是一个用于连接电子控制单元(electronic control unit,ECU)的多主设备串行总线标准。ECU 也称为节点。CAN 网络上至少需要两个节点才能通信。节点的复杂度可以只是简单的 I/O 设备,也可以是运行许多软件的嵌入式计算机。节点也可以是网关,允许普通的计算机通过 USB 或以太网端口与 CAN 网络上的设备通信。

通过两根平行的总线,将所有节点连接在一起,两条导线组成一条双绞线,并且连接了 120Ω 的特性阻抗。ISO 11898-2,也称为高速 CAN(CAN 上的位速率最高为 1Mb/s,CAN-FD 上为 5Mb/s)。它在总线的两端均连接 120Ω 的电阻,如图 16.2 所示。

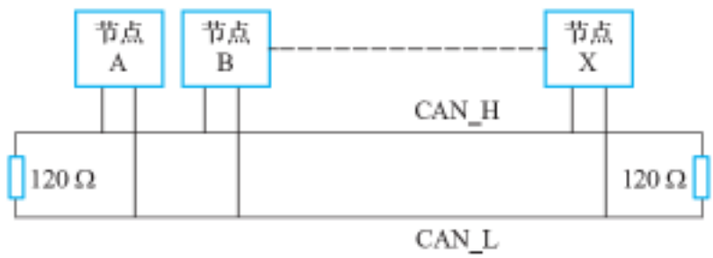


图 16.2　带有端接电阻的 CAN 拓扑结构

CAN 使用了差分“线与”信号,如图 16.3 所示。两个信号 CAN 高(CANH)和 CAN 低(CANL)要么被驱动到 CANH>CANL 的“显性”状态(将 CANH 线驱动到 3.5V,将 CANL 线驱动到 1.5V),要么没有被无源电阻驱动并拉到 CANH≤CANL 的“隐性”状态(即 CANL 线上的电压和 CANH 上的电压为 2V,因此 CANH 和 CANL 线上的差分电压为 0V 的状态)。数据位‘0’编码为显性状态,数据位‘1’编码为隐性状态,支持“线与”规约,该规约为总线上 ID 号较低的节点提供了优先级。

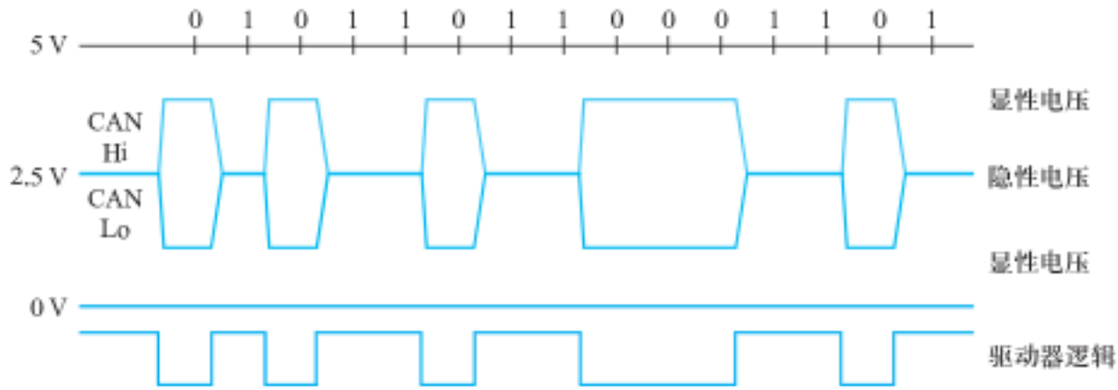


图 16.3　CAN 信号的电平标准

低速/容错 CAN 信号在传输显性信号‘0’时，驱动 CANH 端到 5V，将 CANL 端降到 0V。在传输隐性信号‘1’时，并不驱动 CAN 总线的任何一端。在电源电压为 5V 时，显性信号差分电压需要大于 2.3V，隐形信号的差分电压要小于 0.6V，如图 16.4 所示。

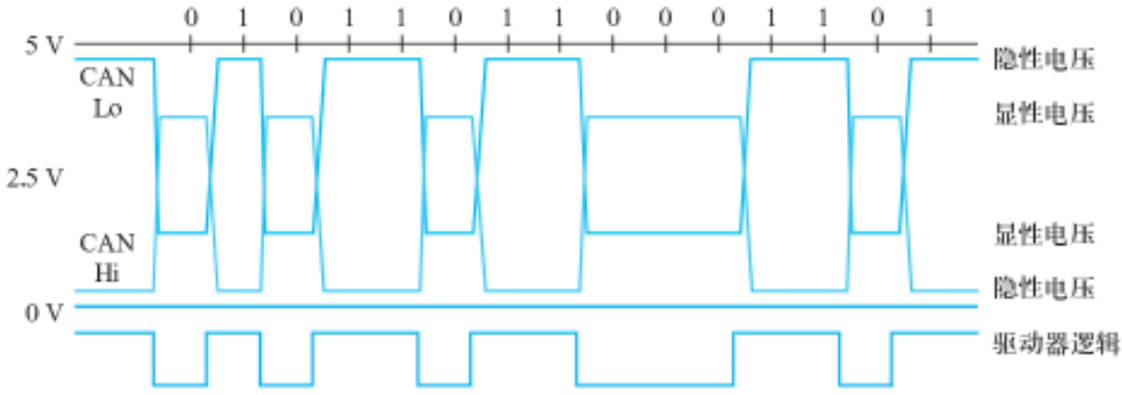


图 16.4　低速/容错 CAN 信号的电平标准

16.1.3　总线节点

CAN 总线中的每个节点包括微控制器（或微处理器/主处理器）、CAN 控制器和收发器，如图 16.5 所示。其中：

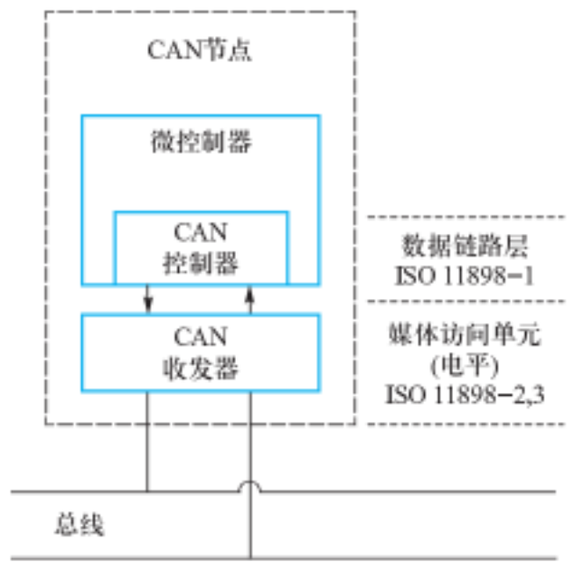


图 16.5　CAN 总线节点

（1）中央处理器、微处理器或主处理器。处理主设备确定收到信息的含义以及想要传输的信息。传感器、驱动器和控制设备可以与主处理器连接在一起。

（2）CAN 控制器。通常集成在单片机中，作为单片机中的一个外设模块。当接收消息时，CAN 控制器将从总线上接收到的串行字节保存到整个消息可用，之后主处理器就可以获取这个消息（通常由于 CAN 控制器触发一个中断）。当发送消息时，主处理器要发送的消息传递到 CAN 控制器，之后当总线空闲时将串行信息传递到总线。

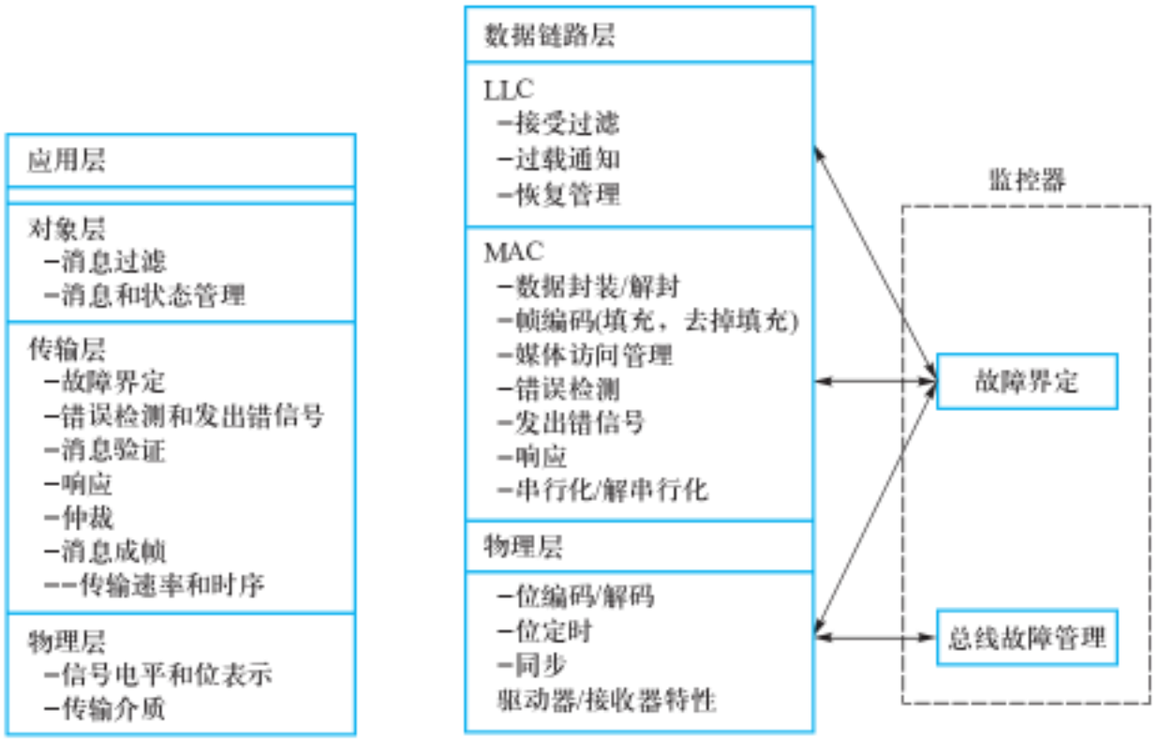
（3）收发器。由 ISO11898-2/3 媒体访问单元（Medium Access Unit，MAU）定义。当接收时，把数据流从 CAN 总线层转换成 CAN 控制器可以使用的标准。CAN 控制通常配有保护电路；当发送时，把来自 CAN 控制器的数据流转换到 CAN 总线层。

CAN 总线中的每个节点都能够发送和接收消息，但不能同时进行。一个消息帧主要包括：标识符（ID），它表示信息的优先级，最多 8 个数据字节。循环冗余校验（Cyclic Redundancy Check，CRC）、应答（ACK）和其他帧部分也是消息的一部分。改进的 CAN FD 将每个帧扩展到最多 64 字节。消息采用非归零格式（non-return zero，NRZ）串行传输到总线并可以被所有节点接收。

对于 CAN 2.0A 规范，CAN 总线节点的分层结构如图 16.6(a)所示；对于 CAN2.0B 规范，根据 OSI 参考模型的 CAN 分层结构如图 16.6(b)所示。

下面对 CAN 2.0A 分层结构中的一些术语进行说明。

（1）系统灵活性。节点可以添加到 CAN 网络中，而无需对任何节点和应用层的软件或硬



(a) CAN 2.0A分层结构　　(b) CAN 2.0B分层结构

图 16.6　CAN 2.0 节点分层结构

件进行任何更改。

(2) 消息路由。消息的内容由一个 IDENTIFIER(标识符)命名。IDENTIFIER 不指示消息的目的地,但是描述了数据的含义,这样网络上的所有节点可以通过消息过滤来确定是否对数据采取行为。

(3) 多播。作为消息过滤(message filtering)概念的结果,任何数量的节点都可以接收并同时对同一消息采取行动。

(4) 数据一致性。在 CAN 网路中,可以保证一条消息同时被所有节点或没有节点接受。因此,系统的数据一致性时通过多播的概念和错误处理来实现的。

(5) 位速率。在不同的系统中,CAN 的速度是不同的。然而,在一个给定的系统中位速率是一致的并且是固定的。

(6) 优先级。标识符(IDENTIFIER)定义在总线访问期间的一个静态消息优先级。

(7) 远程数据请求。通过发送一个远程帧(REMOTE FRAME),一个需要数据的节点可以请求另一个节点发送相应的数据帧(DATA FRAME)。数据帧和相应的远程帧由相同的标识符命名。

(8) 多主设备。当总线空闲时,任何单元都可以开始传输消息。具有较高优先级消息的单元将获得总线访问权。

(9) 仲裁。每当总线空闲时,任何单元都可以开始发送消息。如果两个或更多单元同时开始发送消息,则通过使用标识符的按位仲裁解决。仲裁机制保证不会丢失信息和时间。如果同时启动具有相同标识符的数据帧和远程帧,则数据帧优先于远程帧。在仲裁期间,每个发送器都会将发送的位的电平与总线上监视的电平进行比较。如果电平相等,则单元可以继续发送,当发送“隐性“电平并监视到”显性“电平时,该单元失去仲裁,必须退出而不会发送更多的位。

(10) 安全性。为了实现数据传输的最大安全性,每个 CAN 结点都实现了强大的错误检测、发送错误指示和自检。

① 错误检测。对于检测错误,使用了下面的措施,包括监视(发送器将要发送位的电平与总线上检测的位电平进行比较)、循环冗余校验、位填充和消息帧检查。

② 错误检测的性能。错误检测的机制有以下属性,包括:

a. 检测到所有的全局错误。

b. 检测到在发送器的所有本地错误。

c. 检测到消息中最多 5 个随机分布的错误。

d. 检测到消息中长度小于 15 的突发错误。

e. 检测到消息中任何奇数的错误。

(11) 发出错误指示和恢复时间。任何检测到错误的节点都会将损坏的消息进行标记。将丢掉该消息,并自动重传。如果没有进一步的错误,从检测到错误到下一条消息开始的恢复时间最多为 29 位时间。

(12) 故障界定。CAN 节点能够区分短暂的干扰和永久性故障。将关闭有缺陷的节点。

(13) 连接。CAN 串行通信链路是可连接多个单元的总线。这个数字没有理论上的限制。实际上,单元的总数将受到延迟时间和/或总线线路上的电气负载的限制。

(14) 单个通道。总线由一个承载位的通道组成。从这个数据可以得到重新同步的信息。该通道的实现方式在 2.0 规范中并不固定。例如,单线(加地)、两根差分线、光纤等。

(15) 总线值。总线可以具有两个互补值之一,即“显性”或“隐性”。在同时传输“显性”和“隐性”位期间,生成的总线值将是“显性”。例如,在总线的“线与”实现中,“显性”电平将由逻辑‘0‘表示,“隐性”电平将由逻辑‘1’表示。

(16) 响应。所有接收器都会检查接收到的消息的一致性,并将确认一致的消息并标记不一致的消息。

(17) 休眠模式/唤醒。为了减低系统的功耗,可以将 CAN 设备设置为休眠模式,而无需任何内部的活动且断开总线驱动器。通过唤醒来完成休眠模式,重新启动内部的活动。尽管传输层将等待系统振荡器稳定,然后等待直到它自己与总线活动同步(通过检查连续 11 个‘隐性’位),在将总线驱动器再次设置为“在总线(on-bus)之前,

为了唤醒系统中的其他节点,可以使用具有专用的、最低可能的标识符(IDENTIFIER)(rrr rrrd rrrr;r=”隐性(recessive)“,d=显性(dominant)“)。

在图 16.6(b)中,LLC 的全称为 logic link control(逻辑链路控制),MAC 全称为 medium access control(MAC)。显然将数据链路层(data link layer,DLL)分成了 LLC 子层和 MAC 子层。

16.1.4 消息传输

消息传输由四种不同的帧类型表现和控制:

(1) 数据帧(data frame)将数据从发送器传送到接收器。

(2) 远程帧(remote frame)由总线单元发出,请求发送具有相同标识符(identifier)的数据帧(data frame)。

(3) 错误帧(error frame)。任何单元检测到一个总线错误时都会发送一个错误帧。

(4) 过载帧(overload frame)。过载帧用于在前面或后面的数据或远程帧之间提供额外的延迟。

数据帧和远程帧通过帧间空间与前面的帧分开。

1. 数据帧

数据帧由七个不同的字段构成，包括帧的起始（start of frame，SOF）、仲裁字段（arbitration filed）、控制字段（control filed）、数据字段（data field）、CRC 字段（CRC field）、ACK 字段（ACK field）、帧的结束（end of frame，EOF）。数据字段的长度可以为零，如图 16.7 所示。

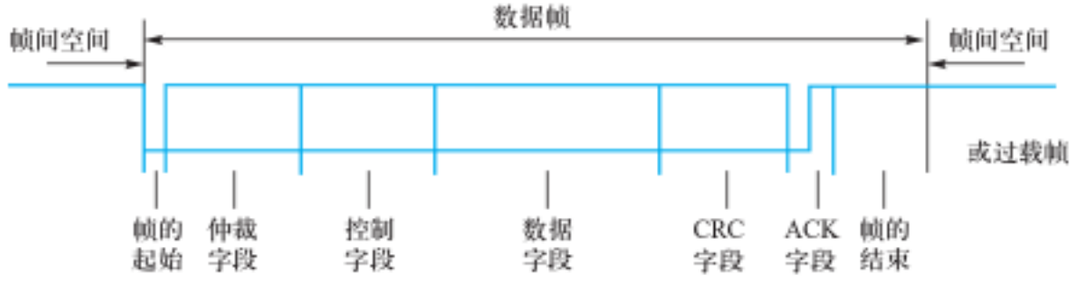


图 16.7　数据帧的格式

对于 CAN 2.0A 来说，使用标准格式；对于 CAN 2.0B 来说，使用扩展格式。

（1）帧的起始（start of frame，SOF）。标记数据帧和远程帧的开始。它由一个“显性”位构成。当总线处于空闲时，只允许一个站开始发送。所有的站必须同步到由首先开始传输的站的帧开始引起的前沿。

（2）仲裁字段。

对于标准格式来说，该字段由标识符（identifier）和远程传输请求（remote transmission request，RTR）位构成，如图 16.8 所示。其中：

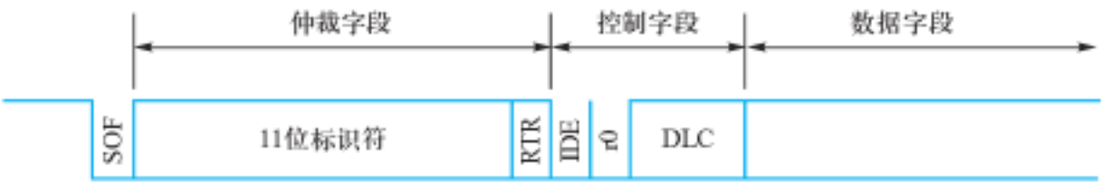


图 16.8　数据帧的标准格式

① 标识符的长度为 11 位。这些位按照从 ID10 到 ID0 的顺序发送。其中 ID0 位为最低有效位（least significant bit，LSB）。最高的 7 个有效位（most significant bit，MSB）（ID10～IDE4）不能都是“隐性”位。

② 在数据帧中，RTR 位必须是‘显性“位。在远程帧中，RTR 位必须是‘隐性’位。

对于扩展格式来说，该字段由标识符（identifier）、代替远程请求（substitute remote request，SRR）位、标识符扩展（Identifier Extension，IDE）位和远程传输请求（remote transmission request，RTR）位构成，如图 16.9 所示。其中：

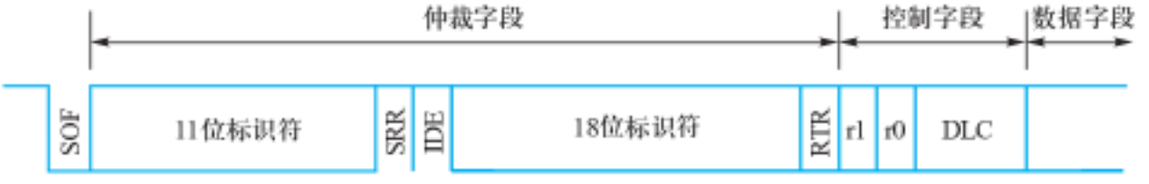


图 16.9　数据帧的扩展格式

① 标识符的长度为 29 位。格式由两个子段构成，包括 11 位的基本 ID 和 18 位的扩展 ID。其中，基本 ID 由 11 位构成，按照从 ID28 到 ID18 的顺序发送。它与标准标识符的格式相同。基本 ID 定义了扩展的帧的基本优先级；扩展 ID 由 18 位构成，按照从 ID17 到 ID0 的顺序

发送。

② SRR：是一个“隐性”位。它在标准帧中 RTR 位位置的扩展帧中传输，因此替代标准帧中的 RTR 位。

③ IDE 位。在扩展帧中属于仲裁字段，在标准格式中属于控制字段。在标准格式中，发送 IDE 位为“显性”；在扩展格式中，发送 IDE 位为“隐性”。

（3）控制字段。

对于标准格式来说，控制字段由 6 位组成，包括数据长度编码和用于将来扩展的 2 位，如图 16.10 所示。

图 16.10　数据帧标准格式中控制字段的内容

注：为了区分标准格式和扩展格式，以前的 CAN 规范版本 1.0~1.2 中的保留位 r1 现在表示为 IDE 位。

图 16.10 中的保留位必须‘显性’发送。接收器接受所有组合中的‘显性’和‘隐性’位。数据字段中的字节个数由数据长度编码确定，数据长度编码为 4 位宽，编码含义如表 16.1 所示。

表 16.1　数据长度编码含义

数据字节的个数	数据长度代码			
	DLC3	DLC2	DLC1	DLC0
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d	d	d

注：表中的 d 表示 dominant（显性），r 表示 recessive（隐性）。

从表中可知，允许数据字节数的范围为 0~8 之间，不能使用其他值，也就是不能超过 0~8 的范围。

（4）数据字段。数组字段由要与数据帧一起传输的数据组成。它可以包含 0 到 8 个字节，每个字节包含 8 位，按照从 MSB 到 LSB 的顺序传输。

(5) CRC 字段。CRC 字段包括 CRC 序列,后面跟着 CRC 界定符,如图 16.11 所示。

图 16.11 CRC 字段

① CRC 序列。

帧校验序列源自循环冗余码,最适合位数小于 127 位的帧(BCH 码)。

为了执行 CRC 计算,将要除的多项式定义为多项式,其系数由无填充的比特流给出,该比特流由帧的起始、仲裁字段、控制字段和数据字段(如果出现)组成,且对于 15 个最低的系数为 0。该多项式是用生成多项式除(系数以 2 为模计算):

$$X^{15}+X^{14}+X^{10}+X^{8}+X^{7}+X^{4}+X^{3}+1$$

这个多项式除法的余数就是发送到总线上的 CRC 序列。为了实现这个功能,使用 15 位的移位寄存器 CRC_RG(14:0)。如果用 NXTBIT 指示比特流(从 SOF 开始到数据字段结束的无填充位序列)的下一位,CRC 序列的计算如下:

```
    CRC_RG=0;                          //初始化移位寄存器
REPEAT
    CRCNXT=NXTBIT EXOR CRC_RG(14);
    CRC_RG(14:1)=CRC_RG(13:0);         //寄存器左移一位
    CRC_RG(0)=0;
    IF CRCNXT THEN
        CRC_RG(14:0)=CRC_RG(14:0) EXOR (4599hex);
    ENDIF
```

UNTIL(CRC 序列开始或者出现错误条件)

在发送/接收数据字段的最后一位后,CRC_RG 包含 CRC 序列。

② CRC 定界符。CRC 定界符跟在 CRC 序列后,该定界符由一个‘隐形‘位构成。

(6) ACK 字段(响应字段)

ACK 字段为两位长度,包含 ACK 时隙(ACK slot)和 ACK 定界符(ACK delimiter),如图 16.12 所示。在 ACK 字段中,发送站发送两个“隐性”位。当接收器正确接收一个有效消息时,在 ACK 槽隙期间通过发送一个”显性“位向发送器报告该信息。

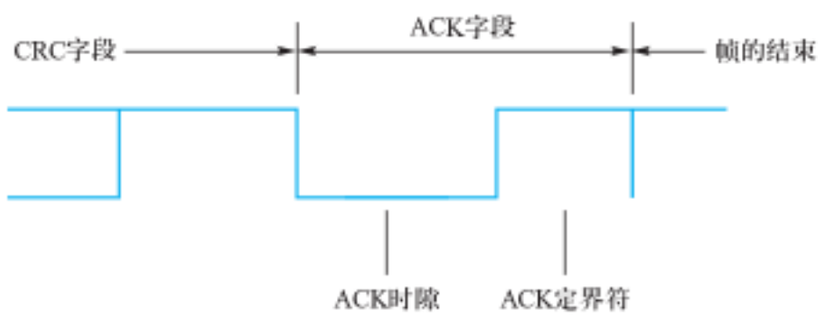


图 16.12 ACK 字段

① ACK 时隙。所有接收到匹配 CRC 序列的站都在 ACK 时隙内通过在发送器的‘隐性’位上加上一个‘显性’位来报告这一点。

② ACK 定界符。ACK 定界符是 ACK 字段的第二位，必须是一个‘隐性’位。因此，ACK 时隙被两个‘隐性’位包围（CRC 定界符、ACK 定界符）。

（7）帧结束。每个数据帧和远程帧由一个标志序列分割，该标志序列由 7 个‘隐性’位组成。

2. 远程帧

作为默写数据的接收者的站可以通过其源节点通过发送远程帧来启动相应数据的传输。远程帧以标准格式和扩展格式的形式存在。在这两种格式中，它由六个不同的字段构成，包括帧的起始、仲裁字段、控制字段、CRC 字段、ACK 字段和帧结束，如图 16.13 所示。

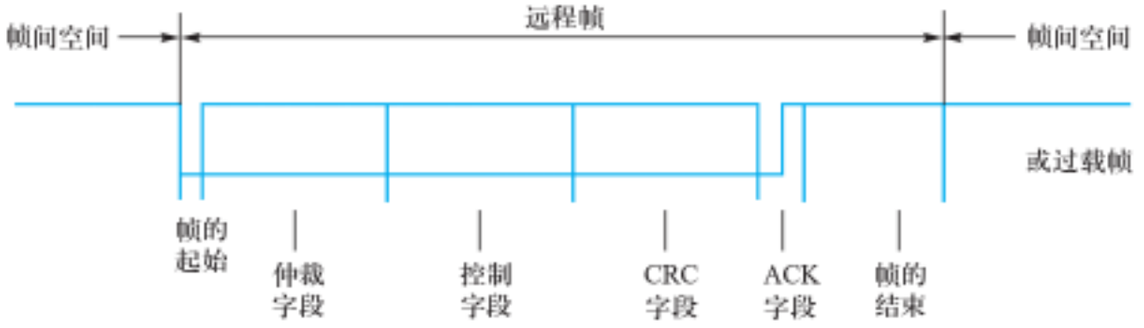


图 16.13　远程帧的格式

与数据帧相比较，远程帧的 RTR 位是‘隐性’的。在远程帧中，没有数据字段，独立于数据长度编码的值，该值范围 0~8。该值为对应数据帧的数据长度编码。

3. 错误帧

错误帧由两个不同的字段组成。第一个字段由来自不同站的错误标志的叠加给出。接下来的第二个字段是错误定界符，如图 16.14 所示。

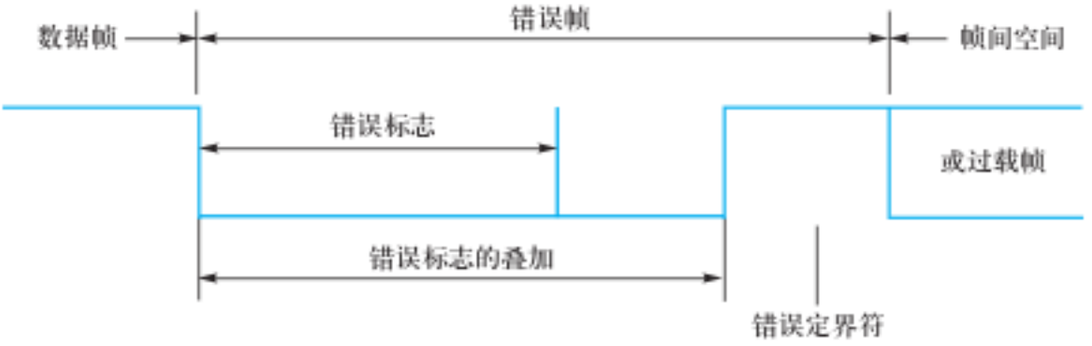


图 16.14　错误帧的构成

为了正确终止错误帧，‘错误被动“节点可能需要总线’总线空闲‘至少三个比特位时间（如果在’错误被动‘接收器存在本地错误）。因此总线不应被加载到 100%。

（1）错误标志。一个错误标志有两种形式，包括一个主动的错误标志（active error flag）和一个被动错误标志（passive error flag）。主动错误标志由六个连续的‘显性‘位构成。被动错误表示由六个连续的’隐性‘位构成，除非它被来自其他节点的’显性‘位覆盖。

检测到错误条件的“错误主动”的站通过发送主动错误标志来指示错误。错误标志的形式与从帧开始到 CRC 定界符的位填充规则冲突，或者破坏了应答字段或帧结束字段的固有形

式。结果,所有其他的站检测到错误条件并与此同时开始发送错误标志。因此,可以在总线上监视的“显性”位序列导致将各个单独站发送的不同错误标志叠加在一起。这个序列的总长度在 6 位到 12 位之间变化。

检测到错误条件的“错误被动“的站尝试通过发送被动错误标志来指示错误。”错误被动“站等待 6 个相同具有相同极性的连续位,从被动错误标志的开始处开始。当检测到这 6 个相同的位时,完成被动错误标志。

(2) 错误定界符。

错误定界符由 8 个‘隐性‘位构成。在发送错误标志后,每个站都会发送’隐性‘位并监视总线,直到它检测到’隐性‘位。之后,它开始传输另外 7 个’隐性‘位。

4. 过载帧

过载帧包含两个字段,即过载标志(overload flag)和过载定界符,如图 16.15 所示。这里有两种过载条件,它们都将导致发送过载标志:

图 16.15 过载帧的构成

(1) 接收器的内部条件,需要延迟下一个数据帧或远程帧。

(2) 在间歇(Intermission)的第 1 位和第 2 位检测到一个‘显性’位。

(3) 如果 CAN 节点在错误定界符或者过载定界符的第 8 位采样一个‘显性’位时,它将开始传输一个过载帧(而不是一个错误帧)。错误计数器不会递增。

由于过载条件 1 导致的过载帧的开始只允许在间歇的第一位时间开始,而由于过载条件 2 和条件 3 导致的过载帧在检测到‘显性’位后开始一位。

最多可以产生两个过载帧,用于延迟下一个数据帧或者远程帧。

(1) 过载标志。过载标志由 6 个‘显性’位构成。整体形式对应于主动错误标志的形式。过载标志的形式破坏了间歇(intermission)字段的固定形式。因此,所有其他站也检测到一个过载条件,并且开始传输过载标志。如果在间歇地第 3 位检测到一个‘显性’位,则将该位理解成帧的开始。

(2) 过载定界符。过载定界符由 8 个‘隐性’位构成。

过载定界符与错误定界符有相同的格式。当发送过载标志后,站监视总线,直到它检测到从一个‘显性’位跳变到到一个‘隐性‘位为止。在这个时间点,每个总线站都完成了其过载标志的发送,并且所有站开始传输另外 7 个’隐性‘位。

5. 帧间空间

数据帧和远程帧通过称为帧间空间(interframe space)的位字段与之前的帧(无论它们是什么类型(数据帧、远程帧、错误帧、过载帧))分开。相反,过载帧和错误帧之间没有帧间空间,并且多个过载帧没有被帧间空间分隔。

帧间空间包含位字段间歇(intermission)和总线空闲(bus idle),并且,对于‘错误被动‘站,它们已经是前一个消息的发送者,暂停传输。

对于不是‘错误被动‘或者前一个消息接收者的站,帧间空间如图 16.16 所示。对于‘错误被动‘站,它们已经是前一个消息的发送者,帧间空间如图 16.17 所示。

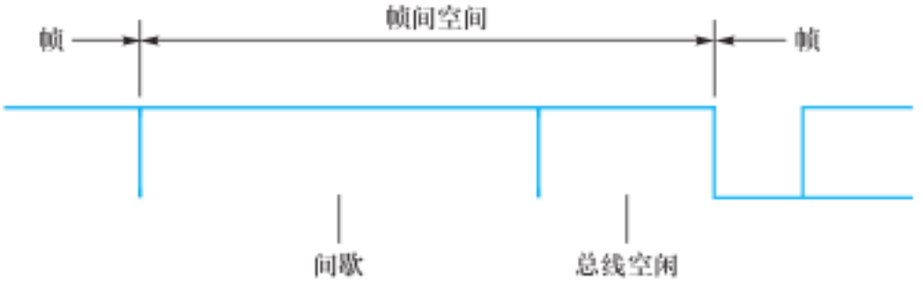


图 16.16　帧间空间构成(1)

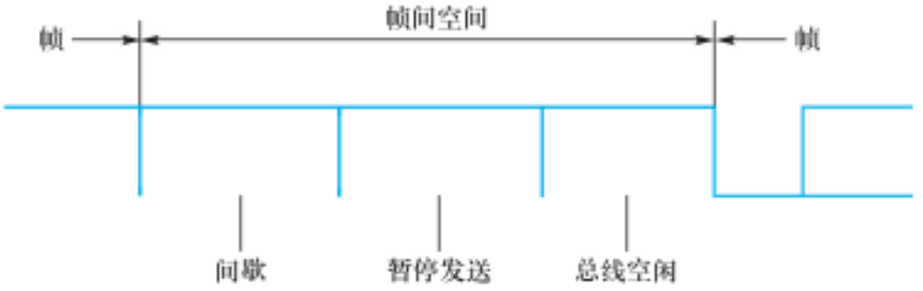


图 16.17　帧间空间构成(2)

(1) 间歇。间歇由 3 个‘隐性‘位构成。注意,如果 CAN 节点有一条消息等待发送,并且它在间歇的第 3 位采样一个’显性‘位,它将其理解为帧开始位,并且在下一位开始传输其消息标识符的第一位,而没有首先发送 SOF 位,且不会变成接收器。

(2) 总线空闲。总线空闲的周期可以是任意长度。将总线识别为空闲的,并且任何需要传输消息的站都可以访问总线。在传输另一个消息期间等待传输的消息在间歇后的第一位开始。

在总线上检测到一个‘显性‘位将理解为帧开始(SOF)。

(3) 暂停发送。在一个“错误被动“站已经发送一条消息后,它跟在间歇后发送 8 个”隐性“位,然后再开始发送进一步的消息或识别总线空闲。如果同时开始传输另一个站引起的传输,则该站将成为该消息的接收者。

16.1.5　消息过滤

消息过滤基于整个标识符。(可选)允许将任何标识符位设置为‘无关’,以用于消息过滤的屏蔽寄存器,可用于选择一组标识符映射到附加的接收缓冲区。

如果实现了屏蔽寄存器,则屏蔽寄存器的每一位都必须是可编程的,即可以使能/禁止它们以进行消息过滤。屏蔽寄存器的长度可以包括整个标识符或仅仅是其中一部分。

16.1.6　消息验证

将消息看作有效的时间点对于消息的发送器和接收器是不同的。

(1) 对于发送器来说,如果在帧的结束之前没有错误,则消息是有效的。如果消息已经破坏,将根据优先级自动重新传输。为了能够与其他消息竞争总线访问,必须在总线空闲时立即开始重新发送。

(2) 对于接收器,如果在帧结束的最后一位之前没有错误,则消息是有效的。将帧结束的最后一位值看作‘无关’,一个“显性“值不会导致格式错误。

16.1.7 位流编码

帧的起始、仲裁字段、控制字段、数据字段和 CRC 序列采用比特填充的方法进行编码。每当发送器在要发送的比特流中检测到 5 个具有相同的值的连续比特时,它会自动在实际发送的比特流中插入一个互补的比特。

数据帧或远程帧的剩余位字段(CRC 定界符,ACK 字段和帧的结束)是固定格式,没有位填充。错误帧和过载帧也是固定格式,没有采用位填充的方法。

根据非归零(non return to zero,NRZ)方法,对消息中的位流进行编码。这意味着,在总的位时间期间内,生成的位电平是‘显性’或‘隐性’的。

16.1.8 错误管理

本节介绍错误管理。

1. 错误检测

此处有 5 个不同的错误类型(不是互相排斥的)。

(1) 位错误

在总线上发送位的单元也监视总线。当监视的比特位的值与发送位的值不同时,必须在该位时间检测位错误。一个例外是在仲裁字段的填充位流期间或在确认时隙期间发送‘隐性’位。然后,当监视到一个‘显性’位时,不会发生位错误。发送一个被动错误标志并且检测到‘显性’位的发送器并不会将此解释为位错误。

(2) 填充错误

在消息字段中出现 6 个连续相同的位电平,而它们应该用位填充编码时,检测到填充错误。

(3) CRC 错误

CRC 序列由发送器的 CRC 计算结果组成。接收器以与发送器相同的方式计算 CRC。如果计算结果与 CRC 序列中接收的结果不同,则检测到 CRC 错误。

(4) 格式错误

当固定格式的位字段包含一个或多个非法位时,则检测到格式错误。(注意,对于接收器,在帧结束的最后一位期间的“显性“位不看做格式错误)。

(5) 响应错误

当在 ACK 时隙期间没有监视到‘显性’位时,发送器检测到一个响应错误。

2. 错误指示

检测到错误条件的站通过发送错误标志来发出信号。对于“错误主动“节点,它是一个主动错误标志;对于”错误被动“节点,它是一个被动错误标志。当任何站检测到位错误、填充错误、格式错误或确认错误时,在相应站的下一位开始传输错误标志。

每当检测到 CRC 错误时,都会从 ACK 定界符之后的位开始传输错误标志,除非已经启动了另一个条件的错误标志。

16.1.9 故障界定

关于故障界定，一个单元可能的状态为以下三种之一，即“错误主动“、“错误被动”和“总线关闭”。

一个“错误主动”的单元可以正常地参与总线通信并且当检测到错误时发送一个主动错误标志。

一个“错误被动”单元不得发送一个主动错误标志。它参与总线通信，但是当检测到错误时，只能发送一个被动错误标志。并且，在发送以后，一个“错误被动”单元将在初始化下一个发送之前处于等待状态。

一个“总线关闭”单元不允许对总线有任何影响。（比如，关闭输出驱动器）

对于故障界定，在每个总线单元中实现两种计数器，即发送错误计数和接收错误计数。这些计数将根据以下的规则进行修改（注意在给定的消息传输期间，可能要用到规则不知一个）：

（1）当接收器检测到一个错误时，接收计数错误将递增 1。下面情况除外，即在发送主动错误标志或过载标志期间检测到位错误。

（2）当发送一个错误标志后，接收器检测到第一个位问‘显性’时，接收错误计数将递增 8。

（3）当发送器发送一个错误标志时，发送错误计数增加 8。

例外情况 1：

如果发送器是‘错误被动’，并且由于没有检测到一个“显性”确认而检测到一个响应错误，并且在发送其被动错误标志时没有检测到一个‘显性’位。

例外情况 2：

如果发送器发送一个错误标志，因为在仲裁期间发生了填充错误，并且应该是“隐性”的，并且已经作为“隐性”发送但被监控为“显性”。

在例外情况 1 和 2 中，发送错误计数不变。

（4）发送器正在发送一个主动错误标志或一个过载标志，此时如果检测到位错误时，发送错误计数增加 8。

（5）接收器正在发送一个主动错误标志或一个过载标志，此时如果检测到位错误时，接收错误计数增加 8。

（6）任何节点在发送一个主动错误标志、被动错误标志或过载标志后，容忍最多 7 个连续的“显性”位。在检测到第 14 个连续的“显性”位之后（在主动错误表示或过载标志的情况下）或在一个被动错误标志后检测到第 8 个连续的“显性”位后，以及在每个额外的 8 个连续“显性”位序列之后，每个发送器将其发送错误计数增加 8，每个接收器将其错误计数增加 8。

（7）在成功传输消息后（得到 ACK 且没有错误，直到完成帧结束），发送错误计数递减 1，除非它已经是 0。

（8）当成功接收消息（接收没有错误，一直到 ACK 时隙，且成功发送 ACK 位），接收错误计数递减 1（如果它在 1～127 的范围内）。如果接收错误计数已经是 0，它呆在 0，如果它大于 127，则它将设置为 119～127 范围内的一个值。

（9）当发送错误计数等于或超过 128 时，或者接收错误计数等于或超过 128 时，一个节点是“错误被动”。让节点变为“错误被动”的错误条件会导致节点发送主动错误标志。

（10）当发送错误计数大于等于 256 时，节点为“总线关闭”。

(11) 当发送错误计数和接收错误计数都小于或等于 127 时,“错误被动”将重新变为“错误主动”。

(12) 允许“总线关闭”的节点变为“错误激活”(不再是“总线关闭”),在总线上检测到 11 个连续的“隐性”位出现 128 次后,其错误计数器都设置为 0。

注:(1) 大于约 96 的错误计数值表明总线受到严重干扰。提供测试这种情况的方法可能是有利的。

(2) 启动/唤醒。如果在启动期间只有一个节点在线,且如果节点发送一些消息,则它将得到非响应,检测一个错误并且重复消息。由于这个原因,它将变成“错误被动”而不是“总线关闭”。

16.1.10 位时序要求

本节介绍 CAN 总线的时序要求。

1. 标称位速率

标称比特率是在没有理想发送器重新同步的情况下每秒传输的比特数。

2. 标称位时间

$$\text{标称位时间} = 1/\text{标称位速率}$$

可以将标称位时间划分为单独的非重叠时间段,如图 16.18 所示。这些段包括:

图 16.18 标称位时间的划分

(1) 同步段(synchronization segment,SYNC_SEG)

这部分位时间用于同步总线上的各个节点。在该段内期望有一个边沿。

(2) 传播时间段(propagation time segment,PROP_SEG)

这部分位时间用于补偿网络内的物理延迟时间。它是信号在总线上传播时间、输入比较器延迟和输出驱动器延迟之和的两倍。

(3) 相位缓冲段 1(PHASE BUFFER SEGMENT1,PHASE_SEG1)和相位缓冲段 2(PHASE BUFFER SEGMENT2,PHASE_SEG2)

PHASE_SEG1 和 PHASE_SEG2 用于补偿边沿的相位误差。这些段可以通过重新同步来延迟或缩短。

图中,采样点是读取总线电平并将其解释为相应位的值的时间点。它的位置在 PHASE_SEG1 的末尾。此外,信息处理时间是从采样点开始的时间段,为计算后续比特级而保留。

时间份/时间量子(time quantum)是从振荡器周期导出的固定时间单位。有一个可编程的预分频器,整数值,范围是 1~32。从最小时间份开始,时间份可以是最小时间份的 m 倍,其中 m 为预分配器的值。图 16.18 中的时间段和时间份的关系为:

（1）SYNC_SEG 是一个时间份长度。

（2）PROP_SEG 是 1~8 个时间份长度。

（3）PHASE_SEG 是 1~8 个时间份长度。

（4）PHASE_SEG 是 PHASE_SEG1 和信息处理时间的最大值。

（5）信息处理时间小于等于 2 个时间份长度。

在一个位时间内，总的时间份的个数至少在范围 8~25 内。

通常情况下，控制单元不为本地 CPU 及通信设备使用不同的振荡器。因此，CAN 设备的振荡器频率往往是本第 CPU 的频率，并且由控制单元的要求决定。为了获得所需的比特率，位时间的可编程性是必要的。在设计中，在没有本地 CPU 而使用 CAN 实现的情况下，无法对位时间进行编程。另一方面，这些设备允许选择外部振荡器，以便将设备调整到适当的比特率，此时可编程性显得并不重要。

然而，采样点的位置应该为所有节点共同选择。因此，串行链路输入/输出设备（serial link input & output，SLIO）设备的位时序必须与以下位时间定义兼容，如图 16.19 所示。

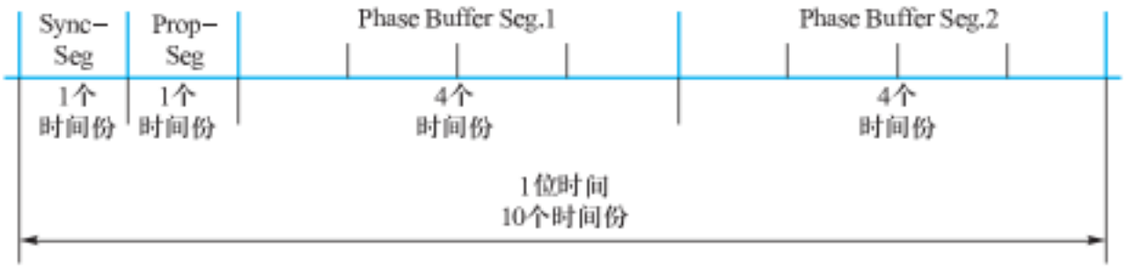


图 16.19　位时间定义

（1）硬同步

硬同步后，内部位时间通过 SYNC_SEG 重新启动。因此，硬同步迫使导致硬同步的边沿位于重新启动位时间的同步段内。

（2）重新同步跳转宽度

重新同步的结果是，PHASE_SEG1 可能变长或者 PHASE_SEG2 可能缩短。相位缓冲段变长或缩短的数量有一个上限，该上限由重新同步跳转宽度给定。通过编程，将重新同步跳转宽度设置在 1~min(4，PHASE_SEG1)的范围内。

时钟信息可以从一位值跳变到另一个位值中得到。只有固定最大数量的连续位具有相同值的特性提供了在帧期间将总线单元重新同步到位流的可能性。用于重新同步的两个跳变之间的最大长度为 29 位时间。

（3）一个边沿的相位误差

一个边沿的相位误差由相对于 SYNC_SEG 的边沿位置给出，以时间份测量。相位误差的符号定义如下：

① e=0，如果边沿在 SYNC_SEG 内。

② e>0，如果边沿在采样点之前。

③ e<0，如果边沿处于前一个位的采样点之后。

（4）重新同步

当引起重新同步边沿的相位误差的幅度小于或等于重新同步跳转宽度的设定值时，重新同步和硬同步的效果相同。当相位误差的幅度大于重新同步跳转宽度时：

① 如果相位误差为正，则将 PHASE_SEG1 延长到等于重新同步跳转长度。

② 如果相位误差为负，则将 PHASE_SEG2 缩短到等于重新同步跳转长度。

(5) 重新同步规则

硬同步和重新同步是同步的两种形式。它们遵守下面的规则：

① 在一个位时间内只允许一个同步。

② 仅当采样点之前探测到的值与紧跟沿之后的总线值不同的时候，边沿才用于同步。

③ 在总线空闲期间，只要有一个“隐性”位到“显性”位的跳变，就会执行硬件同步。

④ 满足规则 1 和规则 2 得所有其他‘隐性’到‘显性“边沿将用于重新同步，但是传输显性位得节点将不会执行重新同步，因为”隐性“到”显性“得边沿与如果仅使用”隐性“到”显性“边沿就行重新同步，则为正的相位误差。