## CAN总线协议

### 1 介绍

#### 1.1 概述

CAN是一种串行通信协议，能够支持分布式实时控制，具有非常高的安全性。从高速网络到低成本的多路布线中都有应用。为了实现设计的透明性和实现的灵活性，CAN被细分为CAN对象层、CAN-传输层和物理层。对象层和传输层由ISO/OSI模型定义的数据链路层的所有服务和功能组成。

对象层主要找出要传递哪些信息，决定传输层接收到的哪些消息实际要使用并为应用层相关硬件提供接口。

在定义对象处理方面有很大的自由。传输层的作用范围主要是传输协议，即控制帧、执行仲裁、错误检查、错误信令和故障约束。在传输层中，决定总线是否可以开始新的传输，或者接收是否刚刚开始。此外，比特定时的一些一般特征也被视为传输层的一部分。

物理层是不同节点之间有传输比特的电气特性代表。在一个网络中，物理层对所有节点来说当然都是一样的。然而，在选择物理层时可能有很大的自由。

#### 1.2 CAN协议特性

* 消息的优先级；
* 保证延迟时间；
* 配置灵活；
* 具有时间同步的多播接收；
* 系统范围的数据一致；
* 支持多master；
* 错误检测和错误信号；
* 总线空闲时自动重传消息；

### 2 CAN总线分层

#### 2.1 CAN协议分层

* 物理层：物理层定义了信号实际传输的方式；包含信号电平以及传输介质；
* 传输层：CAN协议的核心；将接收到的消息呈现给对象层并接受要从对象层传输的消息；负责同步、消息帧、仲裁、确认、错误检测和信令以及故障约束；
* 对象层：包含消息过滤以及状态和消息处理；

#### 2.2 CAN特性概述

* 消息：总线上的信息格式固定，但长度有限的消息发送；当总线空闲时，任何连接的单元都可以开始发送新消息；
* 比特率：CAN总线的速度在不同的系统中可能会有所不同。然而，在给定的系统中，比特率是统一和固定的；
* 多master系统：当总线空闲时，任何单元都可以开始发送消息。具有最高优先级的单元获得总线访问权；
* 安全：为了实现数据传输的最大安全性，在每个CAN节点中都实施了强大的错误检测、信令和自检；

#### 2.3 相关术语

* 信息路由：在CAN系统中，CAN节点不使用任何关于系统配置的信息。这样可以将节点添加到CAN网络中，而不需要更改任何节点和应用层的软硬件；
* 消息路由：消息的内容被称为标识符。标识过滤不会指示消息的目的地，而是描述数据的含义。网络中的所有节点都可以通过消息来决定过滤数据是否由接收并进行处理；
* 数据一致性：在CAN网络中，可以保证消息同时被所有节点或不被节点接受。系统的数据一致性是通过组播和错误处理来实现的；
* 优先级：标识定义了总线访问期间的静态消息优先级；
* 远程数据请求：节点通过发送远程帧可以请求另一个节点发送相应的数据帧。数据帧和相应的远程帧由相同的标识符命名；
* 仲裁：当总线空闲时，任何单元都可以开始发送消息。如果有2个或更多的单元同时开始传输消息，则使用标识通过按位仲裁来解决总线访问冲突。仲裁机制保证既不损失信息，也不损失时间。如果具有相同标识符的数据帧和远端帧同时初始化，则数据帧优先于远端帧。在仲裁期间，每个发送器将传输的比特的水平与总线上监控的水平进行比较。如果这些水平是相等的，单位可以继续发送；
* 确认：所有接收方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；

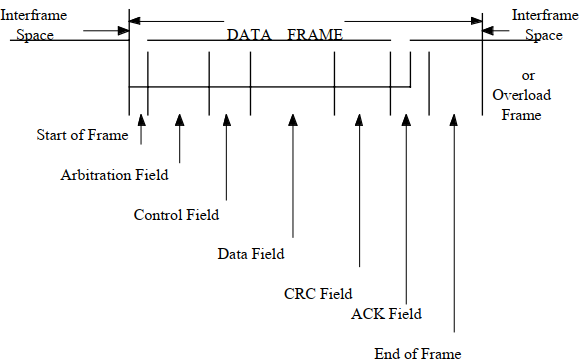
### 3 消息传输

#### 3.1 消息格式

消息传输由四种不同的帧类型表示和控制：

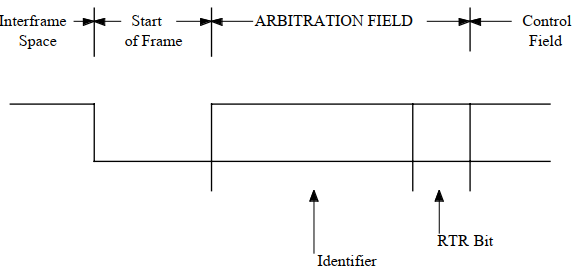
* 数据帧：将数据从发送端传送到接收端；
* 远程帧：由总线单元传输以请求传输具有相同标识符的数据帧；
* 错误帧：在检测总线错误时，任何单元都要发送一个错误帧；
* 过载帧：用于在前面和后面的数据或远程帧之间提供额外的延迟。数据帧和远程帧通过帧间隙与前面的帧分开；

#### 3.2 数据帧



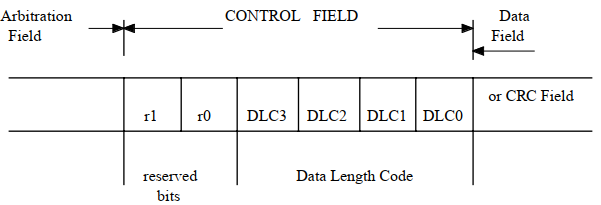
数据帧由帧起始域、仲裁域、控制域、数据域、CRC域，ACK域和帧结束域组成。数据域字段的长度可以是0。

起始域标志着数据帧和远程帧的开始。由一个明显电平位组成。节点只允许在总线空闲时开始传输。所有节点必须与进行传输的节点的起始域进行同步。



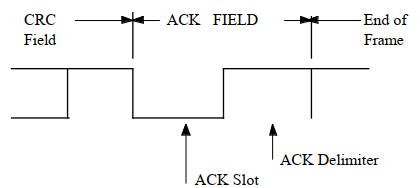
仲裁域仲由标识符和RTR位组成。标识符长度为11位。最低有效位是第0位。高7位不能都是低电平的。RTR位为远程传输请求位。在数据帧中，RTR位必须为高电平。在远程帧内，RTR位必须是低电平。

控制域由6位组成。包括数据长度和为将来扩展保留的两位。保留的比特必为高电平。接收器接受所有组合。



数据域由要在数据帧内传输的数据组成。可以包含0到8个字节，每个字节包含8位，首先传输MSB。

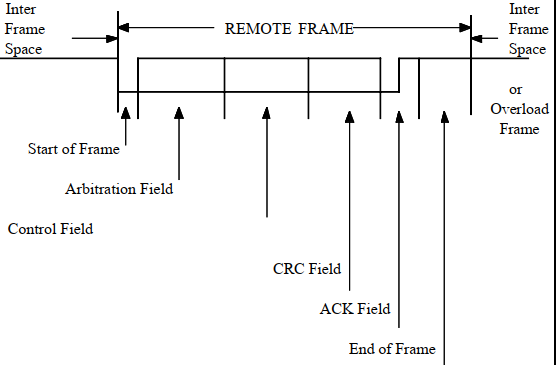
ACK域有两位，包含ACK位和ACK分隔符。在ACK字段中，发送方发送两个低电平比特。正确接收到有效消息的接收方通过在ACK SLOT中发送一个高电平位将此消息报告给发送方。



每个数据帧和远程帧具有一个由7个低电平位组成的标志序列分隔。

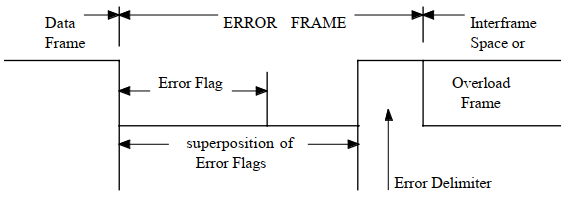
#### 3.3 远程帧

作为某些数据的接收方，可以通过发送远程帧来启动源节点对相应数据的传输。一个远程帧由帧起始域、仲裁域、控制域、CRC域、ACK域和帧结束域组成。与数据帧相反，远程帧的RTR位是低电平且没有数据域字段。



#### 3.4 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节点的错误标志组成。第二个字段是错误分隔符。



错误标志分为主动错误标志和被动错误标志。错误活跃标志由六个连续的高电平位组成。被动错误标志由六个连续的低电平位组成，除非被来自其他节点的高电平位覆盖。

错误活跃节点检测到错误情况时，通过发送错误活跃标志发出信号。

错误分隔符由八个低电平位组成。发送错误标志后，每个节点发送低电平位并监视总线，直到检测到低电平位之后。

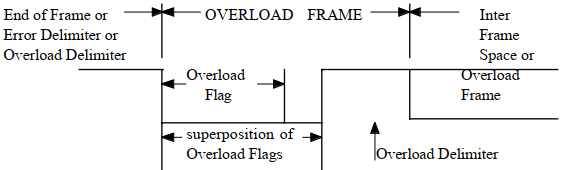
#### 3.5 过载帧

过载帧包含两个位字段过载标志和过载分隔符。

有两种过载情况：

1. 接收方的内部要求下一个数据帧或远程帧延迟；
2. 在间断期间检测到高电平；

由于过载条件1导致的过载帧的开始只允许在预期的间歇的第一个比特时间开始，而过载由于过载条件2的帧在检测到高电平位后开始一个位。



最多可以产生两个过载帧来延迟下一个数据或远程帧。过载标志由六个高电平位组成。

重载分隔符由八个低电平位组成。重载分隔符与错误分隔符的形式相同。在传输过载标志后，该节点监视总线，直到检测高电平转到低电平。每个总线节点已经完成了发送过载标志，然后所有节点开始重合传输另外七个低电平。

#### 3.6 消息验证

消息被视为有效的时间点对于消息的发送方和接收方是不同的。

发送方：如果在帧结束域之前没有错误，则消息对发送方是有效的。如果消息损坏，将根据优先级自动进行重传。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传。

接收方：如果在帧结束域的最后一个比特之前没有错误，消息对接收方是有效的。

#### 3.7 编码

帧起始域、仲裁域、控制域、数据字段和CRC序列采用位填充法编码。每当发送器在要传输的比特流中检测到五个连续的相同值的比特时，就自动在实际传输的比特流中插入一个填充比特。

数据帧或远程帧的剩余位域是固定的形式，不是填充的。错误帧和过载帧也是固定形式，不采用位填充的方式编码。

消息中的位流是根据非归零(NRZ)方法编码。这意味着，在传输时间内，生成的比特电平是正向电平或反向电平。

”

### 4 错误处理

#### 4.1 错误检测

* 位错误：在总线上发送比特的单元也监视总线。当监控的比特值与发送的比特值不同时，必须在该比特时间检测到比特错误；
* 填充错误：在用位填充方法编码的消息字段中，必须在连续第6个相等位的位时间检测到STUFF错误；
* CRC错误：CRC序列由发送方CRC计算的结果组成。接收端以与发送端相同的方式计算CRC。如果计算结果与在CRC序列中接收到的结果不相同，则必须检测到CRC错误；
* 格式错误：当固定格式位字段包含一个或多个非法位时，必须检测到格式错误；
* ACK错误：当发送器在ACK SLOT中没有监视到高电平位时，必须检测到ACK错误；

#### 4.2 错误信号

检测到错误情况的节点通过发送错误标志来发出信号。对于一个错误主动节点是主动错误标志，错误被动节点是被动错误标志错误标志。每当节点检测到位错误，STUFF错误，格式错误或确认错误时发送一个错误位。

FLAG在下一个位在相应的站点启动。无论何时检测到CRC错误，错误标志的传输从ACK分隔符后面的位开始，除非另一个错误条件的错误标志已经开始。

#### 4.3 错误约束

对于一个单元的错误约束而言，可能处于错误活动、错误不活动和总线关闭状态。

一个错误活跃单元通常可以参与总线通信并发送一个错误活跃标志位(检测到错误)。

错误被动单元不能发送错误活跃标志位。参与总线通信，但是当检测到错误时，只发送被动错误标志。同样，在传输之后，错误被动单元将在启动下一步传输之前等待。

总线关闭单元不允许对总线产生任何影响。

### 5 位时序要求

#### 5.1 相关术语

* 标准比特率：标准比特率是一个理想的发送方在没有重新同步的情况下每秒传输的比特数；
* 同步段：这部分位时间用于同步总线上的各个节点。在这个区段内预计会有一个周期；
* 传播时间段：这部分比特时间用于补偿网络中的物理延迟时间，为信号在总线上的传播时间、输入比较器延迟和输出驱动器延迟之和的两倍；
* 相位缓冲段1和相位缓冲段2：这些相位缓冲段用于补偿时钟边缘相位误差。这些段可以通过重新同步来延长或缩短；