CANOpen 系列教程 04

CAN 总线波特率、位时序、帧类型及格式说明

作者：strongerHuang

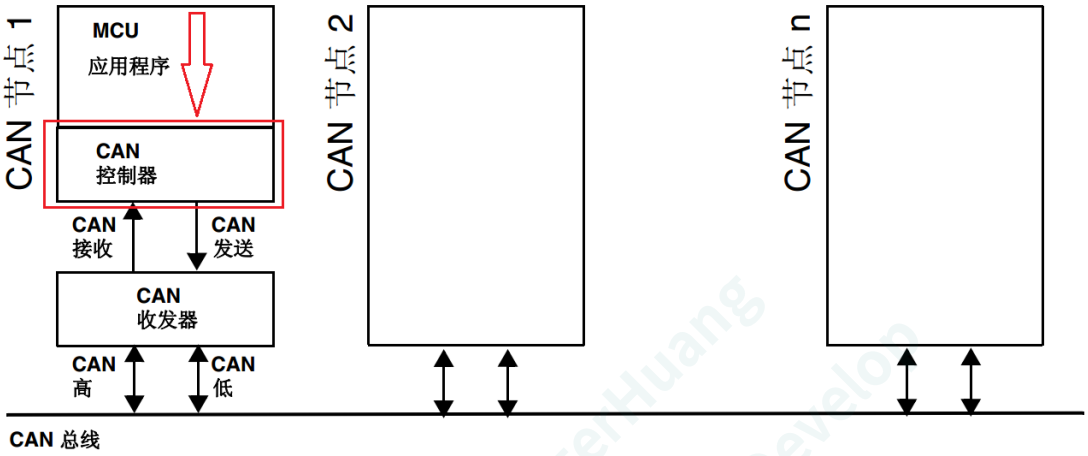
申明：该文档仅供个人学习使用

归类	CANOpen 系列教程
标签	CAN、 CANOpen、 CanFestival
网站	http://www.strongerhuang.com

目 录

一、写在前面

上一篇文章讲述了 **CAN 收发器**的重要作用，也提及了一下 CAN 总线的优势主要在于 **CAN 控制器**。CAN 控制器在 CAN 网络中所处的位置如下图：



本文讲述的 **CAN 控制器**实现的几个重要功能：**CAN 总线波特率、位时序、帧类型**。

本文章收录于【CANOpen 系列教程】，在我的博客分类“CANOpen 系列教程”也能查找到。

为了方便大家平时公交、地铁、外出办事也能用手机随时随地查看该教程，该系列教程也同步更新于微信公众号【EmbeddedDevelop】，关注微信公众号回复【CANOpen 系列教程】即可查看。

二、CAN 总线波特率

CAN 总线属于**异步通信**，因此就有通信**波特率**，而这个波特率发生器就位于 CAN 控制器内部。我们不需要了解它是如何产生的，但需要了解它的含义。这章节针对初学者讲述以下两点内容。

2.1 异步通信

在串行通信中，主要分**异步通信和同步通信**。

同步通信：通信设备之间通过**同步信号**（CLK 时钟）来实现数据传输的通信叫同步通信。如 I2C、SPI 这类通信中都具有一个时钟信号，其实在 STM32 中

USART 也具有同步功能，只是我们大多数人都只用了它的异步功能。

异步通信：简单来说，就是通信设备之间通过约定一样的时间来收发数据。而这个时间就会决定本节说的波特率。

2.2 波特率

很多工程师一直都没彻底搞明白什么是波特率，我这里还是结合 UART 波特率来简述一下其含义。

在电子通信领域，波特（Baud）即调制速率，指的是有效数据信号调制载波的速率，即单位时间内载波调制状态变化的次数。它是对符号传输速率的一种度量，**1 波特即指每秒传输 1 个符号**。

UART 每秒钟传送 240 个字符，而每个字符格式包含 10 位（1 个起始位，1 个停止位，8 个数据位），这时的波特率为 **240Bd**，比特率为 **10 位*240 个/秒=2400bps**。

从上面的描述可以总结：
比特率：即单位时间内传送的**二进制位数**；

波特率：即单位时间内传输的**符号个数**；

只有在每个符号只代表一个比特信息的情况下，波特率与比特率才在**数值上相等**，但是它们的意义并不相同。

三、位时序

上一章节讲述了波特率，而**决定波特率大小的就是本节说的位时序**。在 CAN 标准中**一个位可分为 4 段**：

- **同步段（SS）**
- **传播时间段（PTS）**
- **相位缓冲段 1（PBS1）**
- **相位缓冲段 2（PBS2）**

这些段又由可称为 Time Quantum（简称 Tq）的最小时间单位构成。

1 位分为 4 个段，每个段又由若干个 Tq 构成，这称为**位时序**。

而在 STM32 参考手册中，将位时序分为三段，但它将它传播段和位段 1 合并在一起了，如下图：

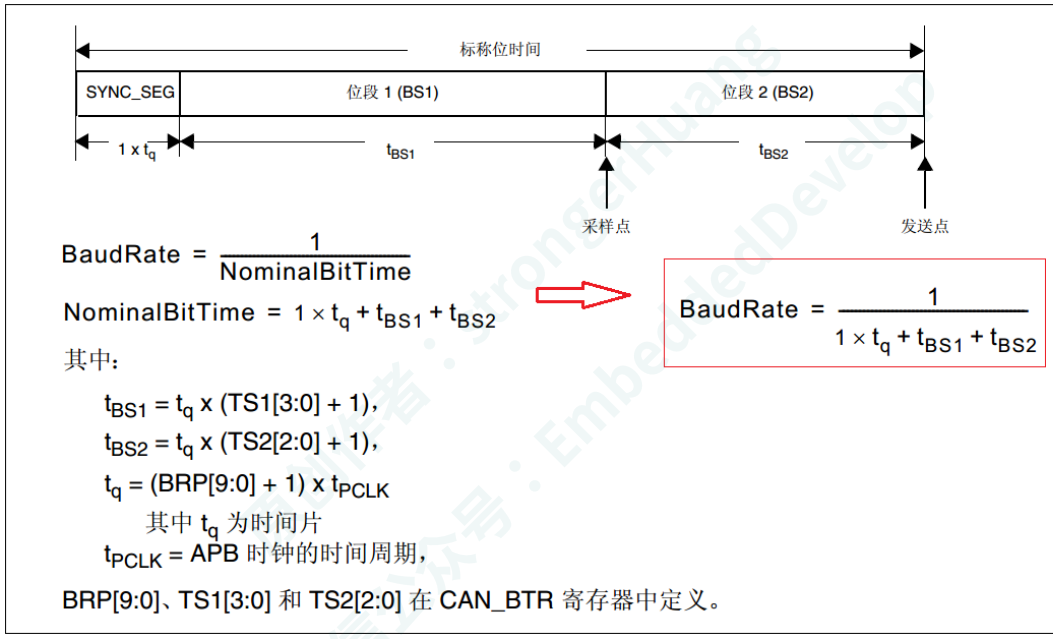
位时序

位时序逻辑将监视串行总线，执行采样并调整采样点，在调整采样点时，需要在起始位边沿进行同步并后续的边沿进行再同步。

通过将标称位时间划分为以下三段，即可解释其工作过程：

- **同步段 (SYNC_SEG)**：位变化应该在此时间段内发生。它只有一个时间片的固定长度 (1 x t_{CAN})。
- **位段 1 (BS1)**：定义采样点的位置。它包括 CAN 标准的 PROP_SEG 和 PHASE_SEG1。其持续长度可以在 1 到 16 个时间片之间调整，但也可以自动加长，以补偿不同网络节点的频率差异所导致的正相位漂移。
- **位段 2 (BS2)**：定义发送点的位置。它代表 CAN 标准的 PHASE_SEG2。其持续长度可以在 1 到 8 个时间片之间调整，但也可以自动缩短，以补偿负相位漂移。

1 位由多少个 Tq 构成、每个段又由多少个 Tq 构成等，**可任意设定位时序**。通过设定位时序，决定传输的波特率：



这几个参数会在以后编程中进行配置，从而**决定通信的波特率**。

关于同步，还有硬件同步、再同步等操作。但初学者可以不必过多理解，掌握上面基础内容就行了。更多关于位时序的内容可以参看 ISO 11898 标准。

四、帧类型及格式说明

CAN 总线是通过以下 5 种类型的帧进行通信：

数据帧：用于发送单元向接收单元传送数据的帧。

遥控帧：用于接收单元向具有相同 ID 的发送单元请求数据的帧。

错误帧：用于当检测出错误时向其它单元通知错误的帧。

过载帧：用于接收单元通知其尚未做好接收准备的帧。
帧间隔：用于将数据帧及遥控帧与前面的帧分离开来的帧。

数据帧和遥控帧有标准格式和扩展格式两种格式。标准格式有 11 个位的标识符 ID，扩展格式有 29 个位的 ID。

4.1 数据帧

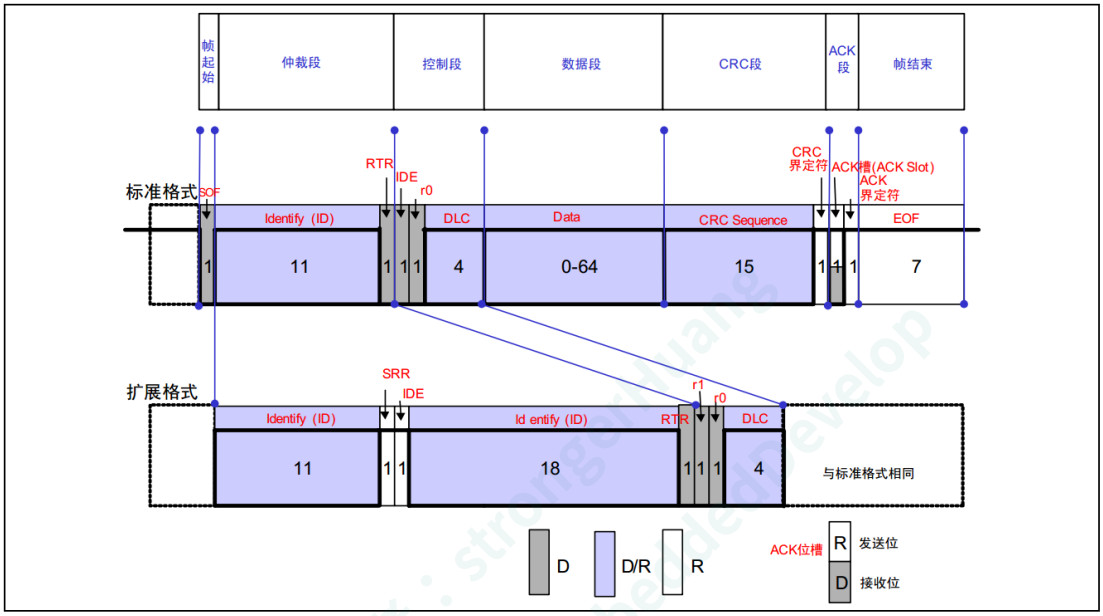


图 16. 数据帧的构成

如上图，数据帧由 7 个段构成：

(1) 帧起始
表示数据帧开始的段。

(2) 仲裁段
表示该帧优先级的段。

(3) 控制段
表示数据的字节数及保留位的段。

(4) 数据段
数据的内容，可发送 0～8 个字节的数据。

(5) CRC 段
检查帧的传输错误的段。

(6) ACK 段
表示确认正常接收的段。

(7) 帧结束
表示数据帧结束的段。

理解数据帧的含义，请从认真理解它的定义：**用于发送单元向接收单元传送数据的帧**。

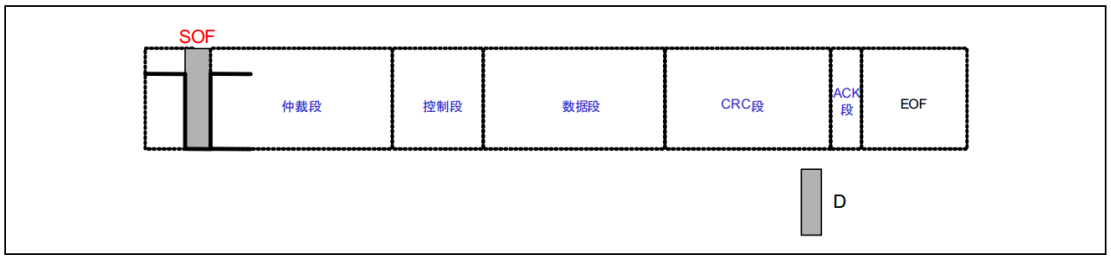
一般的 CAN 总线通信，总线上通信绝大部分时候都是数据帧。像在 CANOpen

协议中，用的最多的 PDO 过程数据对象就是通过数据帧进行的通信。

初学者可以先理解数据帧，然后其他就容易理解了。下面再来讲述一下**数据帧 7 段的详情**。

4.1.1 帧起始

标准和扩展格式相同。表示帧开始的段，1 个位的**显性位**（如下图）：



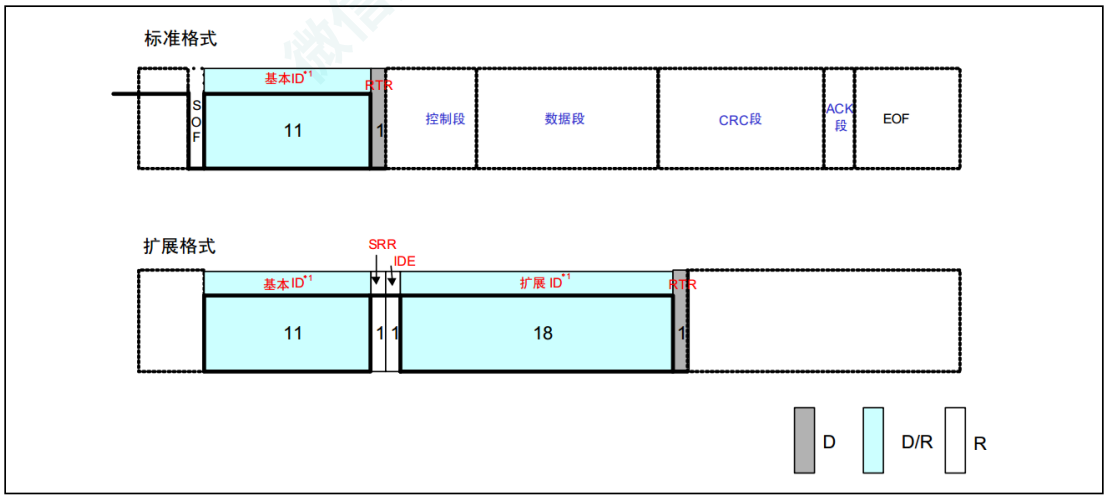
关于显性和隐性电平，请参看我上一篇文章差分信号章节。

总线上的电平有显性电平和隐性电平两种。
总线上执行逻辑上的线“与”时，显性电平的逻辑值为“0”，隐性电平为“1”。

“显性”具有“优先”的意味，只要有一个单元输出显性电平，总线上即为显性电平。并且，“隐性”具有“包容”的意味，**只有所有的单元都输出隐性电平，总线上才为隐性电平**。（显性电平比隐性电平更强）

4.1.2 仲裁段

标准格式和扩展格式在此的构成有所不同。仲裁段表示该帧优先级的段，扩展格式多了 18 位 ID（如下图）：

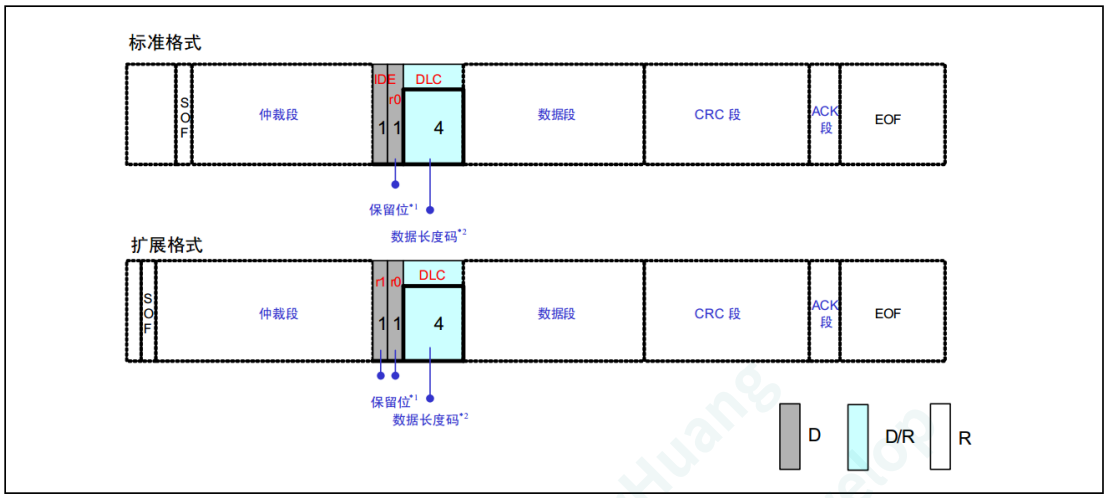


RTR = 0 代表数据帧，RTR = 1 代表远程帧。

为什么叫仲裁段，就是通过 ID 来判断总线上哪一个节点具有优先发送的权利。ID 越小（0 代表显性），优先级越高。

4.1.3 控制段

标准和扩展格式的构成有所不同。控制段由 6 个位构成（如下图）：



它们除了都有 **4 位表示数据段长度代码（DLC）外**，标准帧有 IDE（数值为 0）位和 r0 保留位，扩展帧有 r0 和 r1 保留位。

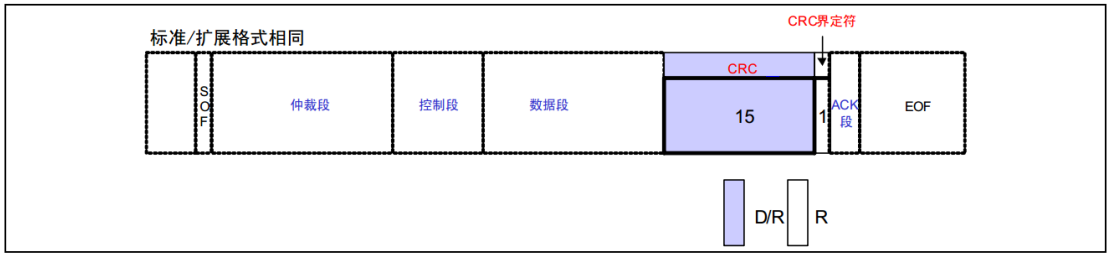
保留位必须全部以显性电平发送。但接收方可以接收显性、隐性及其任意组合的电平。

4.1.4 数据段

标准和扩展格式相同。数据段表示传输数据的内容，从 MSB（最高位）开始输出，可发送 0～8 个字节的数据，长度由前面控制段决定。

4.1.5 CRC 段

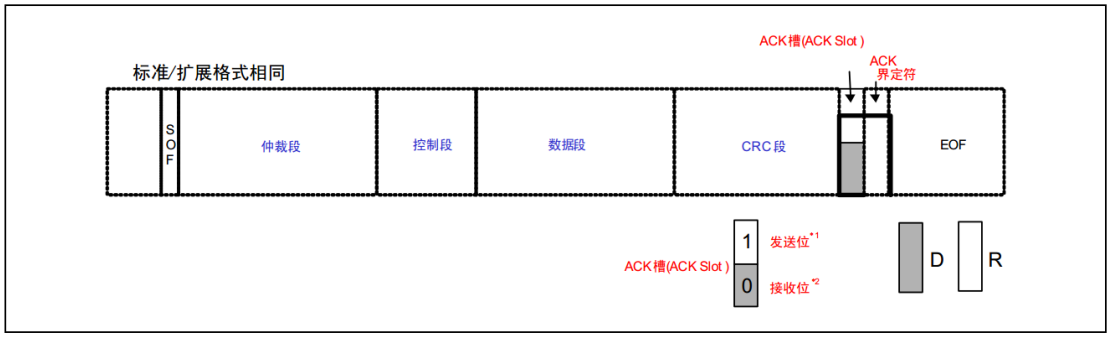
标准和扩展格式相同。CRC 段是检查帧传输错误的帧，由 15 个位的 CRC 顺序和 1 个位的 CRC 界定符（用于分隔的位）构成。



相比 485 这类通信，CAN 控制器就已经把 CRC 校验做了，不需要你的程序再次去计算，从而节约了处理器资源。

4.1.6 ACK 段

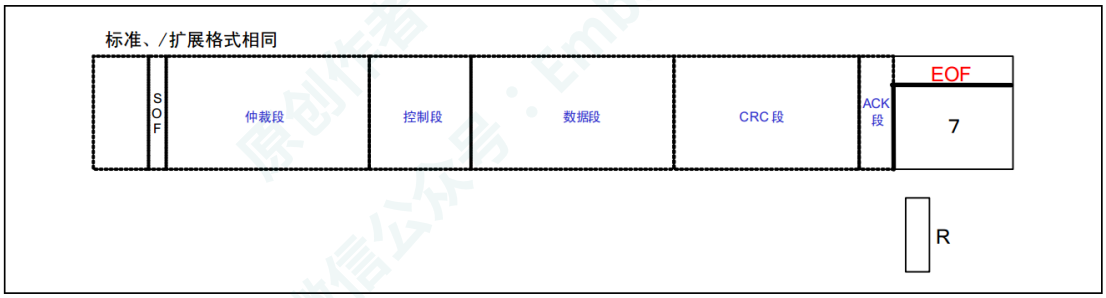
标准和扩展格式相同。ACK 段用来确认是否正常接收。由 ACK 槽(ACK Slot)和 ACK 界定符 2 个位构成。



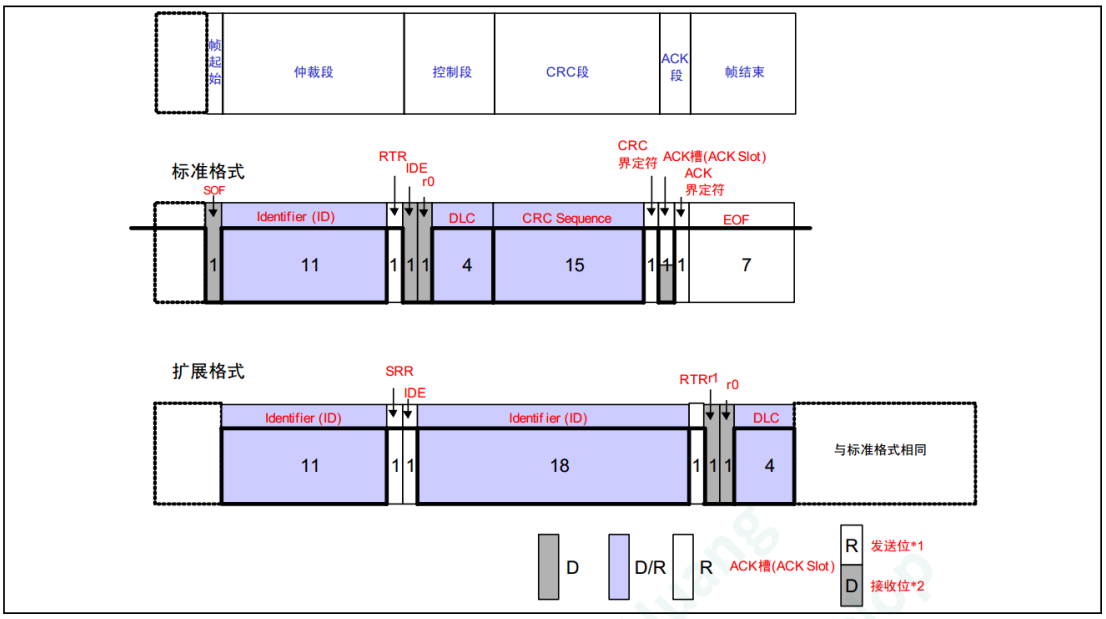
A.发送单元在 ACK 段发送 2 个位的隐性位。
B.接收到正确消息的单元在 ACK 槽(ACK Slot)发送显性位，通知发送单元正常接收结束。这称作“发送 ACK”或者“返回 ACK”。

4.1.7 帧结束

标准和扩展格式相同。帧结束是表示该该帧的结束的段。由 7 个位的隐性位构成。



4.2 遥控帧



和数据帧相比，遥控帧是接收单元向发送单元**请求**发送数据所用的帧。所以，遥控帧没有数据段。因此，遥控帧由如下 6 个段组成：

(1) 帧起始（SOF）

表示帧开始的段。

(2) 仲裁段

表示该帧优先级的段。可请求具有相同 ID 的数据帧。

(3) 控制段

表示数据的字节数及保留位的段。

(4) CRC 段

检查帧的传输错误的段。

(5) ACK 段

表示确认正常接收的段。

(6) 帧结束

表示遥控帧结束的段。

这 6 个段和上面数据帧的内容基本一样，这里就不一一讲述了。讲一下遥控帧和数据帧的区别：

- **遥控帧的 RTR 位为隐性位，没有数据段。**
- **没有数据段的数据帧和遥控帧可通过 RTR 位区别开来。**

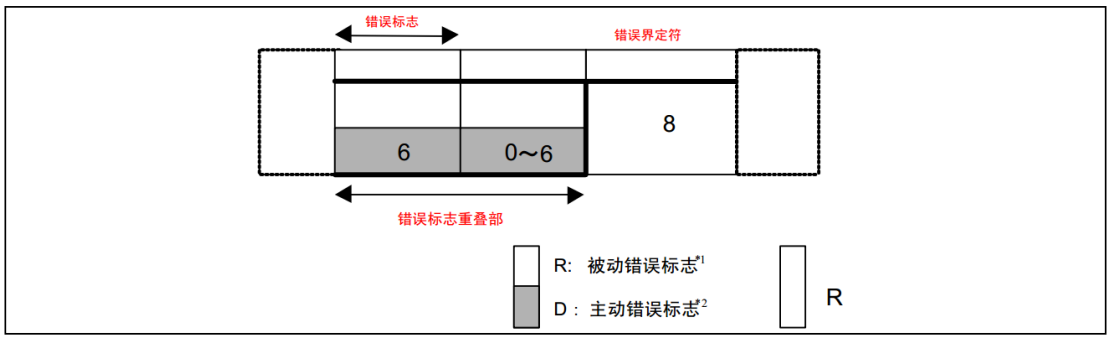
问题一：遥控帧没有数据段，数据长度码该如何表示？

遥控帧的数据长度码以所请求数据帧的数据长度码表示。

问题二：没有数据段的数据帧有何用途？

例如，可用于各单元的定期连接确认/应答、或仲裁段本身带有实质性信息的情况下。

4.3 错误帧



用于在接收和发送消息时检测出错误通知错误的帧。错误帧由错误标志和错误界定符构成。

(1) 错误标志

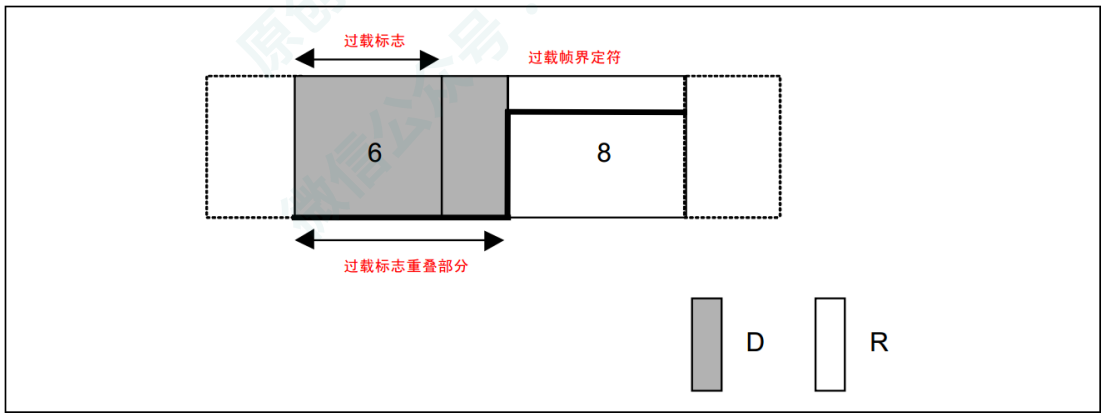
错误标志包括主动错误标志和被动错误标志两种。

- 主动错误标志： 6 个位的显性位。
- 被动错误标志： 6 个位的隐性位。

(2) 错误界定符

错误界定符由 8 个位的隐性位构成。

4.4 过载帧



过载帧是用于接收单元通知其尚未完成接收准备的帧。过载帧由过载标志和过载界定符构成。

(1) 过载标志

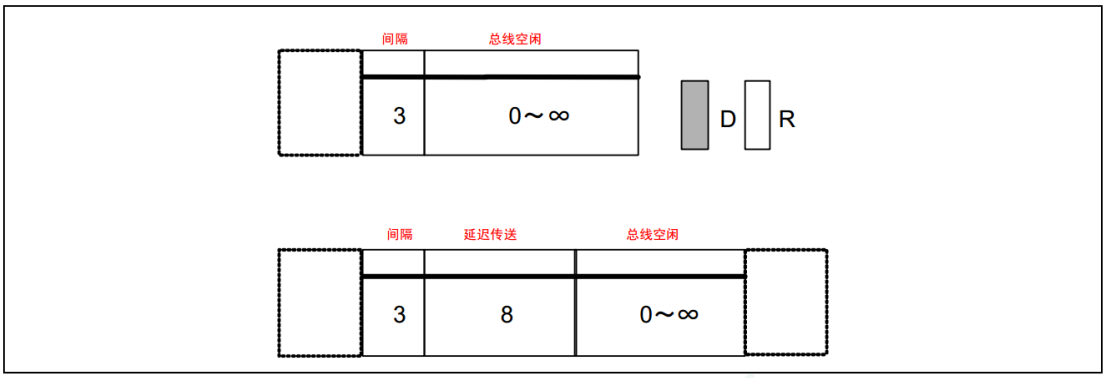
6 个位的显性位。

过载标志的构成与主动错误标志的构成相同。

(2) 过载界定符

8 个位的隐性位。
过载界定符的构成与错误界定符的构成相同。

4.5 帧间隔



帧间隔是用于分隔数据帧和遥控帧的帧。数据帧和遥控帧可通过插入帧间隔将本帧与前面的任何帧（数据帧、遥控帧、错误帧、过载帧）分开。

过载帧和错误帧前不能插入帧间隔。

(1) 间隔
3 个位的隐性位。

(2) 总线空闲
隐性电平，无长度限制（0 亦可）。
本状态下，可视为总线空闲，要发送的单元可开始访问总线。

(3) 延迟传送（发送暂时停止）
8 个位的隐性位。
只在处于被动错误状态的单元刚发送一个消息后的帧间隔中包含的段。

五、说明

1.该文档部分文字来自网络，仅供个人学习使用，版权所有，禁止商用。

2. 本文由我一个人编辑并整理，难免存在一些错误。

3.本教程收录于微信公众号「嵌入式专栏」，关注微信公众号回复【CANOpen 系列教程】即可查看全系列教程。

六、最后

我的博客：http://www.strongerhuang.com
我的 GitHub：https://github.com/EmbeddedDevelop

我的微信公众号（ID：strongerHuang）还在分享 STM8、STM32、Keil、IAR、FreeRTOS、UCOS、RT-Thread、CANOpen、Modbus...等更多精彩内容，如果想查看更多内容，可以关注我的微信公众号。