原创 CAN总线学习笔记（2）- CAN协议数据帧与遥控帧

2018年03月31日 15:34:11 小兵大将0221 阅读数 8542 更多

依照瑞萨公司的《CAN入门书》的组织思路来学习CAN通信的相关知识，并结合网上相关资料以及学习过程中的领悟整理成笔记。好记性不如烂笔头，加油！

1 CAN 协议中的帧

在了解CAN总线的通信机制之前，首先需要了解CAN协议中五种类型的帧结构：

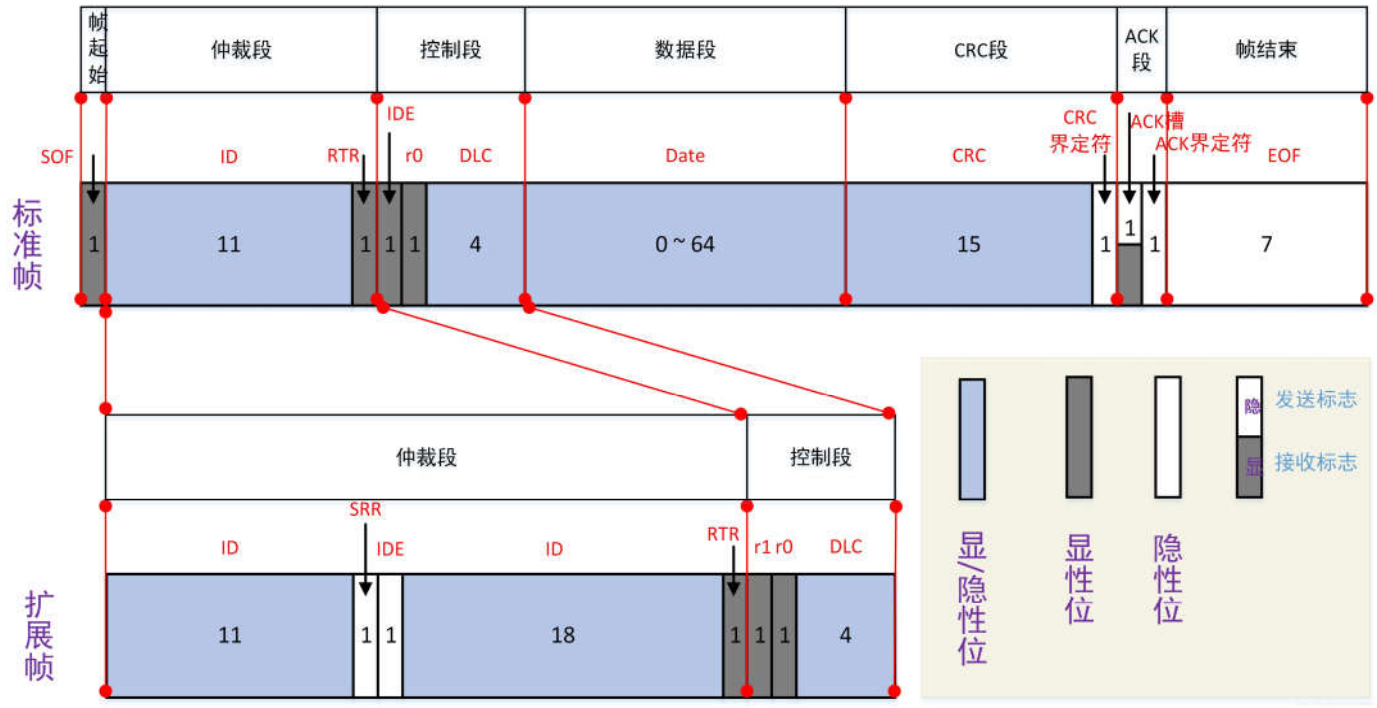
- 数据帧
- 遥控帧
- 错误帧
- 过载帧
- 帧间隔

在讲述五种帧结构的过程中，穿插讲述CAN总线的通信机制。

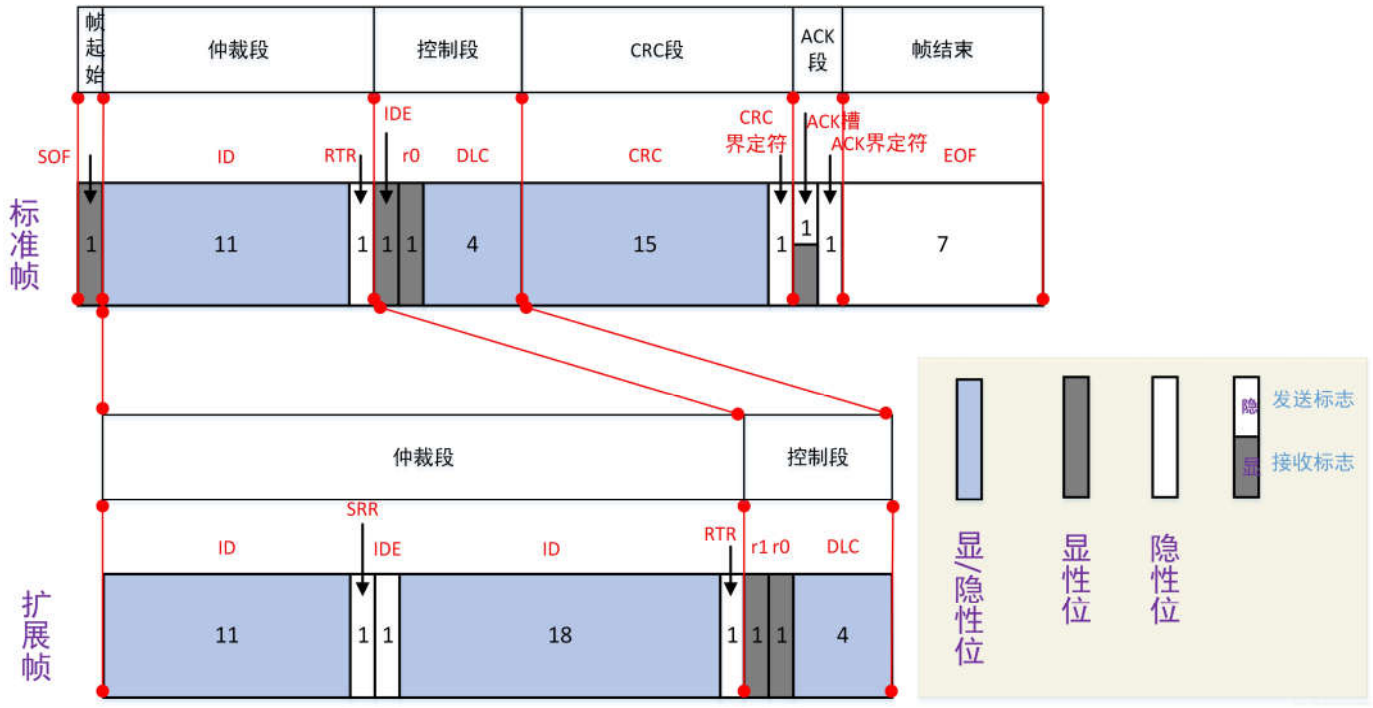
2 数据帧与遥控帧

在CAN协议中，数据帧和遥控帧有着诸多相同之处，所以，在这里，我们将数据帧和遥控帧放在一起来讲。
顾名思义，所谓数据帧，就是包含了我们要传输的**数据**的帧，其作用当然也就是承载发送节点要传递给接收节点的数据。
而遥控帧的作用可以描述为：请求其它节点发出与本遥控帧具有相同ID号的数据帧。
比如：在某一个时刻，节点Node_A向总线发送了一个ID号为ID_2的遥控帧，那么就意味着Node_A请求总线上的其他节点发送ID号为ID_2的数据帧。
节点Node_B能够发出ID号为ID_2的数据帧，那么Node_B就会在收到Node_A发出的遥控帧之后，立刻向总线上发送ID号为ID_2的数据帧。

数据帧 的帧结构如下图所示，包含七个段：**帧起始、仲裁段、控制段、数据段、CRC段、ACK段、帧结束。**



遥控帧 相比于数据帧，从帧结构上来看，只是**少了数据段**，包含六个段：**帧起始、仲裁段、控制段、CRC段、ACK段、帧结束。**



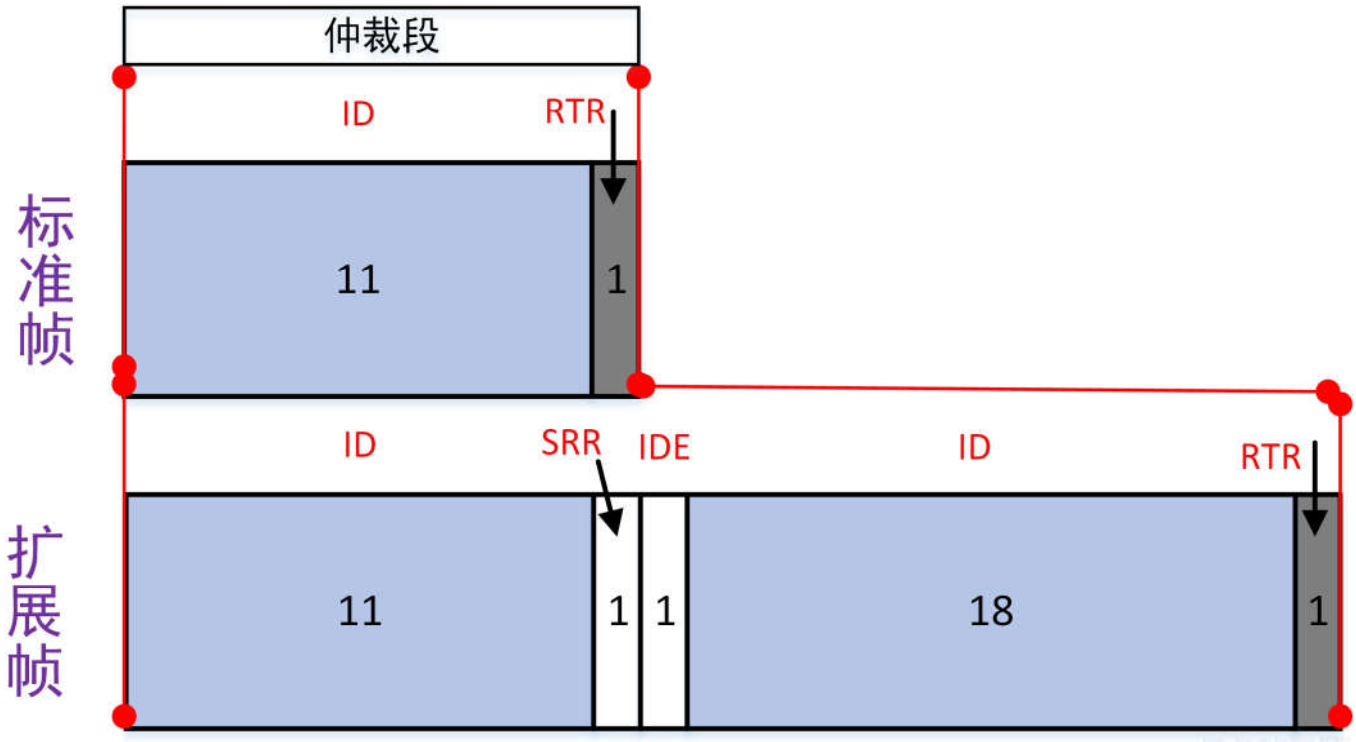
数据帧和遥控帧都分为**标准帧（CAN2.0A）**和**扩展帧（CAN2.0B）**两种结构。
遥控帧相比于数据帧除了**缺少数据段**之外，**遥控帧的RTR位恒为隐性1，数据帧的RTR位恒为显性0。**

2.1 帧起始

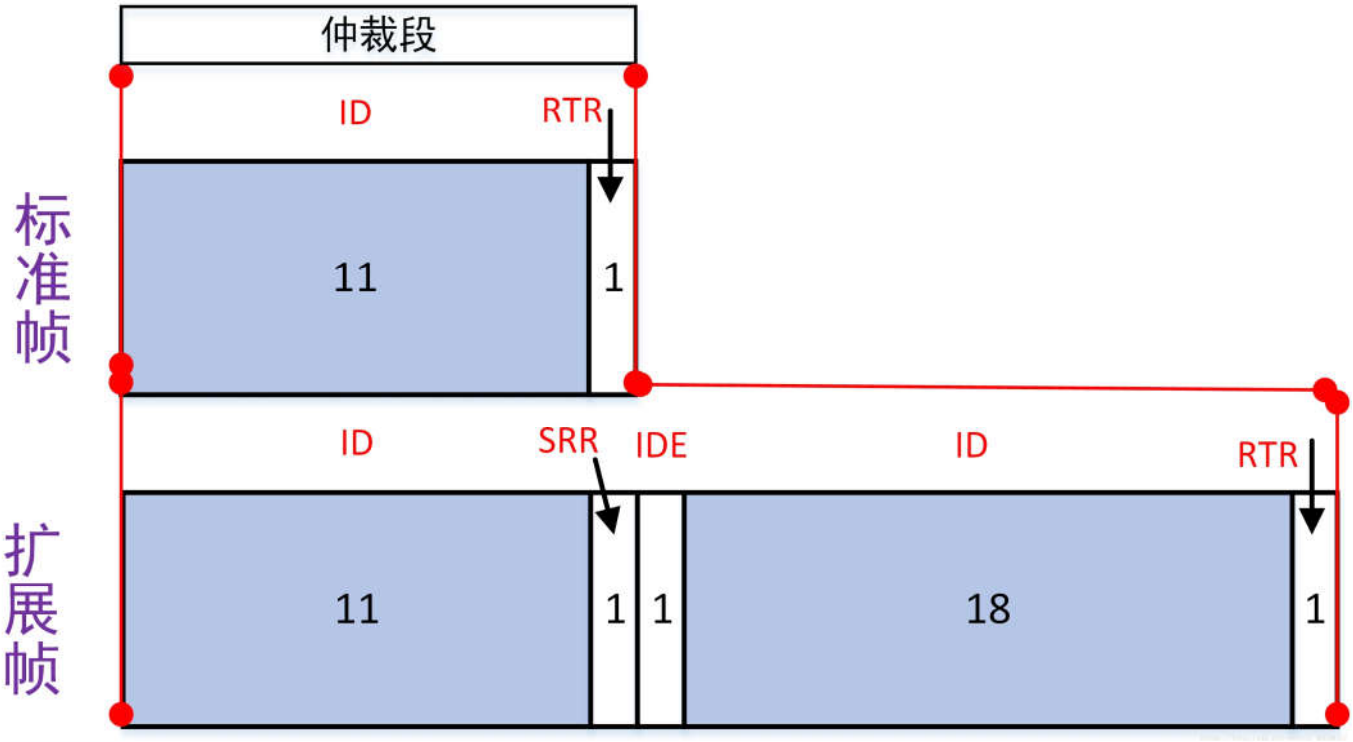
帧的最开始的一位是**帧起始**，也叫SOF（Start Of Frame），**SOF恒为显性位**，即逻辑 0 。
帧起始表示CAN_H 和 CAN_L上有了电位差，也就是说，一旦总线上有了SOF就表示总线上开始有报文了。

2.2 仲裁段

仲裁段是用来判定一帧报文优先级的依据，仲裁段中的ID号也是实现报文过滤机制的基础。仲裁段由以下几个部分组成，
数据帧仲裁段：



遥控帧仲裁段：



可以看到相比于数据帧仲裁段RTR位恒为显性0，遥控帧仲裁段的RTR位恒为隐性1。

2.2.1仲裁过程

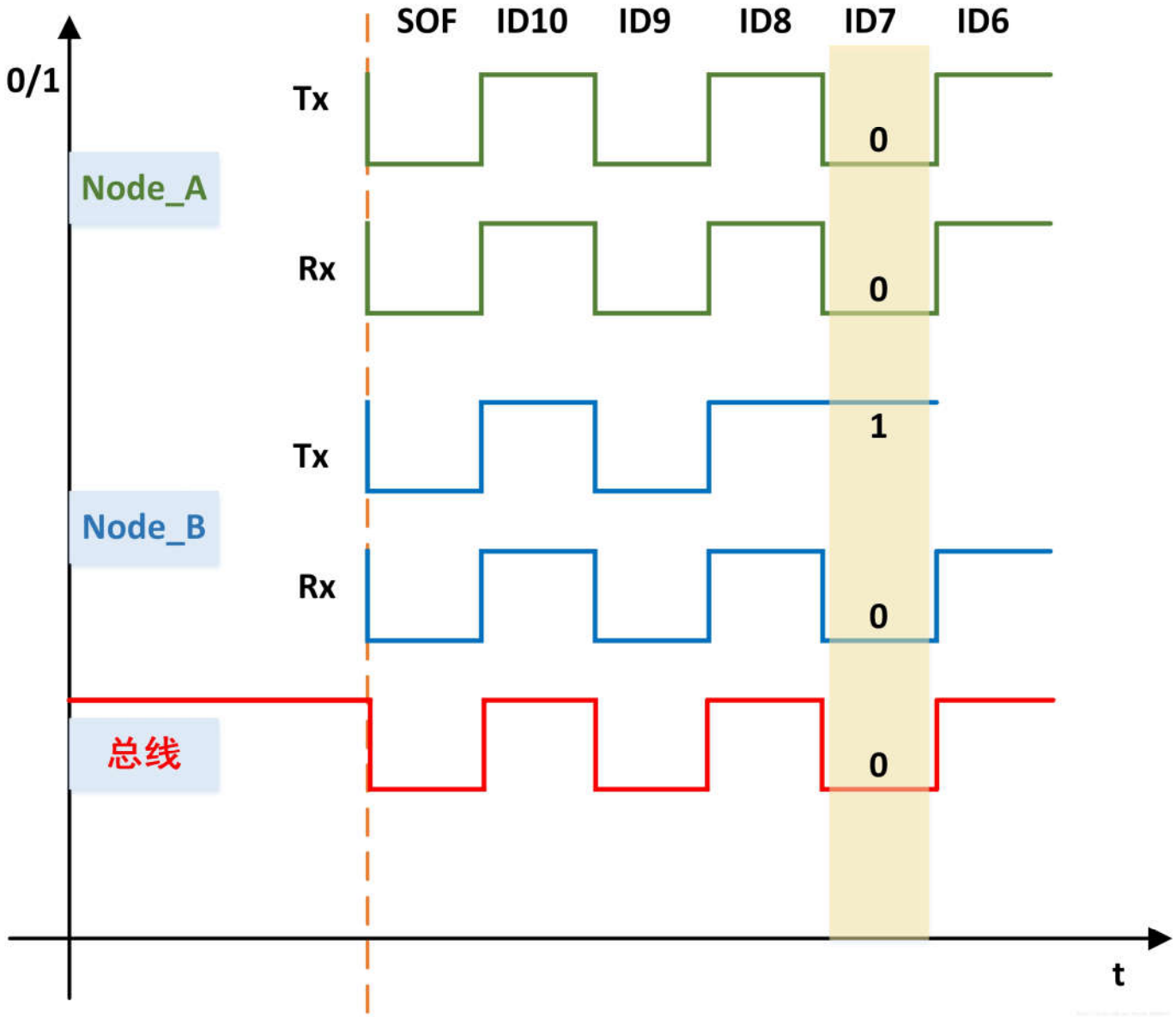
在CAN总线通信中，有一种**回读机制**：指的是节点在向总线上发送报文的过程中，同时也对总线上的二进制位进行“回读”。通过这种机制，节点就可以判断出**本节点发出的二进制位**与**总线上当前的二进制位**是否一致。

还有一种叫做**线与机制**：指的是在总线上，**显性位能够覆盖隐性位。**

举个例子：在某一个时刻，节点Node_A向总线发送了一个显性位0，Node_B向总线发送了一个隐性位1，那么在该时刻，总线上的电平为显性0。

下面将以**标准数据帧**的一个例子来分析CAN总线的**非破坏性逐位仲裁机制。**

一条CAN总线上有Node_A 和 Node_B两个节点，在**总线空闲**时，总线上为**隐性电平**，就在这个时候Node_A 和 Node_B 这两个节点同时向总线上发送数据，如下图：



从图中可以看出，在Node_A 和 Node_B 传输数据前，总线处于空闲状态，为隐性电平1，这也就意味着，此时总线上的任意节点都可以向总线发送数据。
在某一时刻，Node_A 和Node_B两个节点同时向总线上发送数据。按照**线与机制**，总线上的电位为：

节点/ID号	ID10	ID9	ID8	ID7	ID6	...
Node_A	1	0	1	0	1	...
Node_B	1	0	1	1	...	...
总线	1	0	1	0	1	...

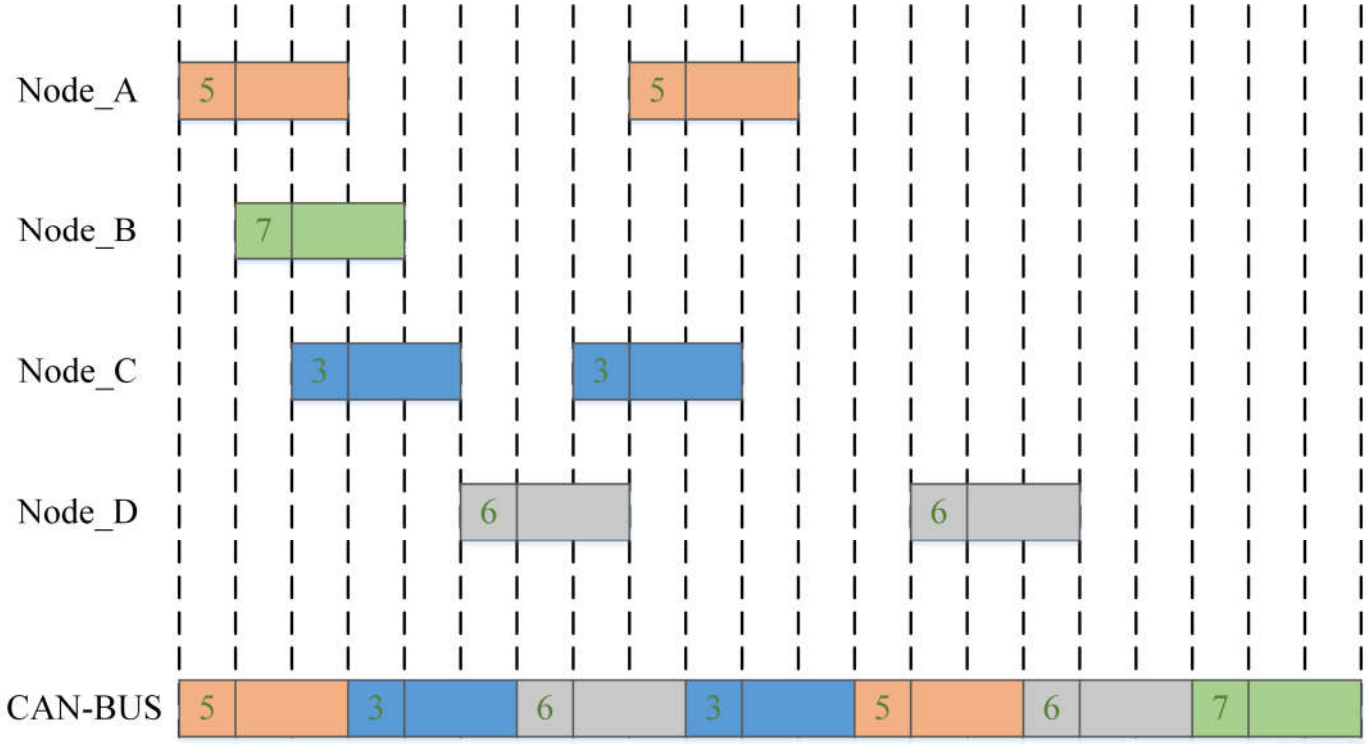
在Node_A和Node_B两个节点向总线发送数据时，他们同时回读总线上的电平。从图中我们可以看到，Node_A 和Node_B的ID10 、ID9、ID8电位相同，因此这两个节点从总线上听到的电位与他们自己发出的电位也相同，这个时候还没有分出胜负。
当Node_B回读总线上的 ID7 这一位时，发现总线上的电平跟它自己发送到总线上的不一样，此时，Node_B知道自己在争夺总线的仲裁中失败了，那么它主动地转换为接收状态，不再发出信息。
于是在此之后，总线上的电平和Node_A发出的电平一致，也就是说，Node_A占据了总线的发送权。
通过上面的分析我们可以看到，在整个仲裁过程中：

- 在Node_A获取总线的发送权之后，Node_A接着发送自己的Msg_A，因此在竞争总线的过程中不会对Msg_A的传输造成延时；
- 在两个节点竞争总线的过程中，不会破坏Msg_A；

正是由于上面的两点，才称之为**非破坏性仲裁机制。**

> Tips: 通过上面仲裁过程的分析，我们可以解释CAN总线通信的三个特点：
> 1）多主控制方式：只要总线空闲，总线上的任意节点都可以向总线上发送数据，直到节点在仲裁中一个个失败，最后只留下一个节点获得总线的发送权。
> 2）非破坏性仲裁机制：仲裁段逐位总裁，依靠**回读机制、线与机制**得以实现。
> 3）半双工通信：所谓半双工通信，指的是节点不能在自己发送报文的时候，同时接收其他节点发送来的报文。这是显然的，一个节点正在发送报文时，已经占据了总线的发送权，其他节点肯定不能向总线上发送报文。

看一个CAN报文发送的实例，CAN总线上有四个节点：Node_A、Node_B、Node_C、Node_D。发送的报文的ID号分别为5、7、3、6.



2.2.2 仲裁段中的RTR，SRR和IDE位

通过上面标准数据帧的仲裁过程分析，我们已经理解了CAN总线的仲裁机制。但同时也注意到仲裁段除了ID号之外，还有其他的位。

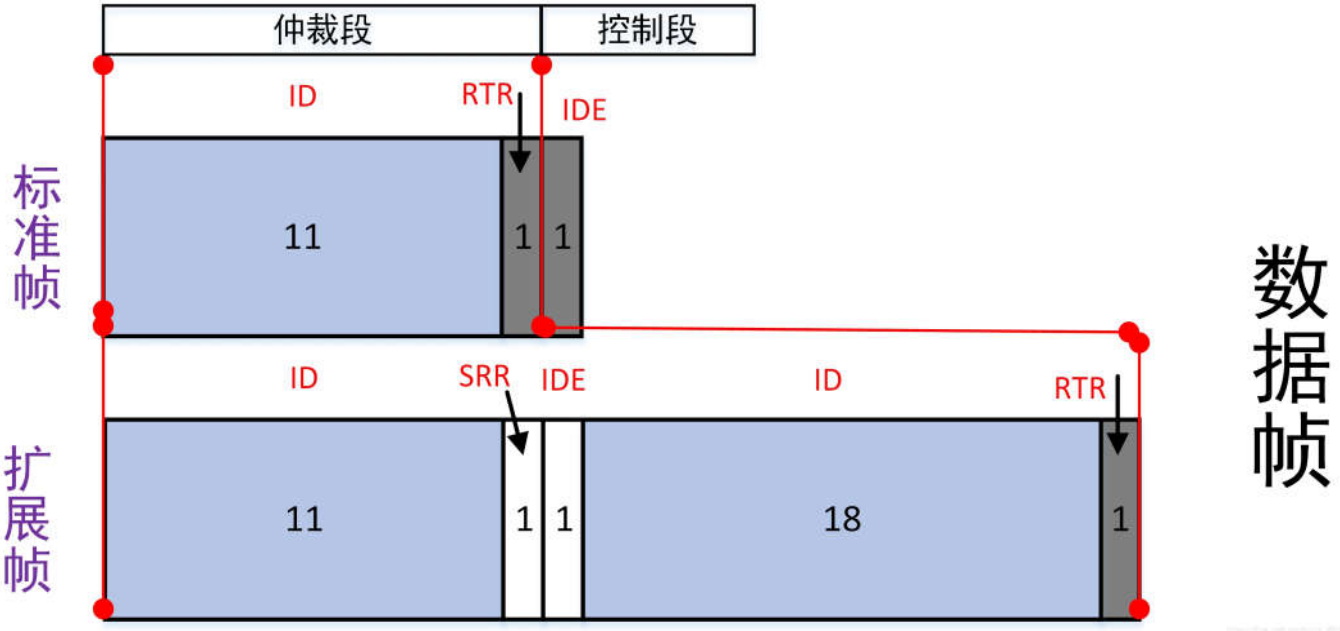
1）RTR位：

Tranmission Request Bit （远程发送请求位）。**在数据帧中，RTR位恒为显性位0，在遥控帧中，恒为隐性1。**

> Tips: 这么做的原因是保证数据帧优先级高于遥控帧。比如：在某一时刻t，节点Node_A发出了ID号为ID_2遥控帧报文来请求总线上的其它节点发出ID号为ID_2的数据帧报文。但是就在同一时刻t，节点Node_B发出了ID号为ID_2的数据帧报文。这个时候怎么办呢，显然依靠ID号不能仲裁出这两帧报文（一个遥控帧，一个数据帧，ID号相同）谁能占据总线的发送权，这种情况下，RTR位就起作用了，由于RTR在数据帧中恒为显性0，在遥控帧中恒为隐性1，所以在ID号相同的情况下，一定是数据帧仲裁获胜。这就解释了 RTR位的作用：在ID号相同的情况下，保证数据帧的优先级高于遥控帧。

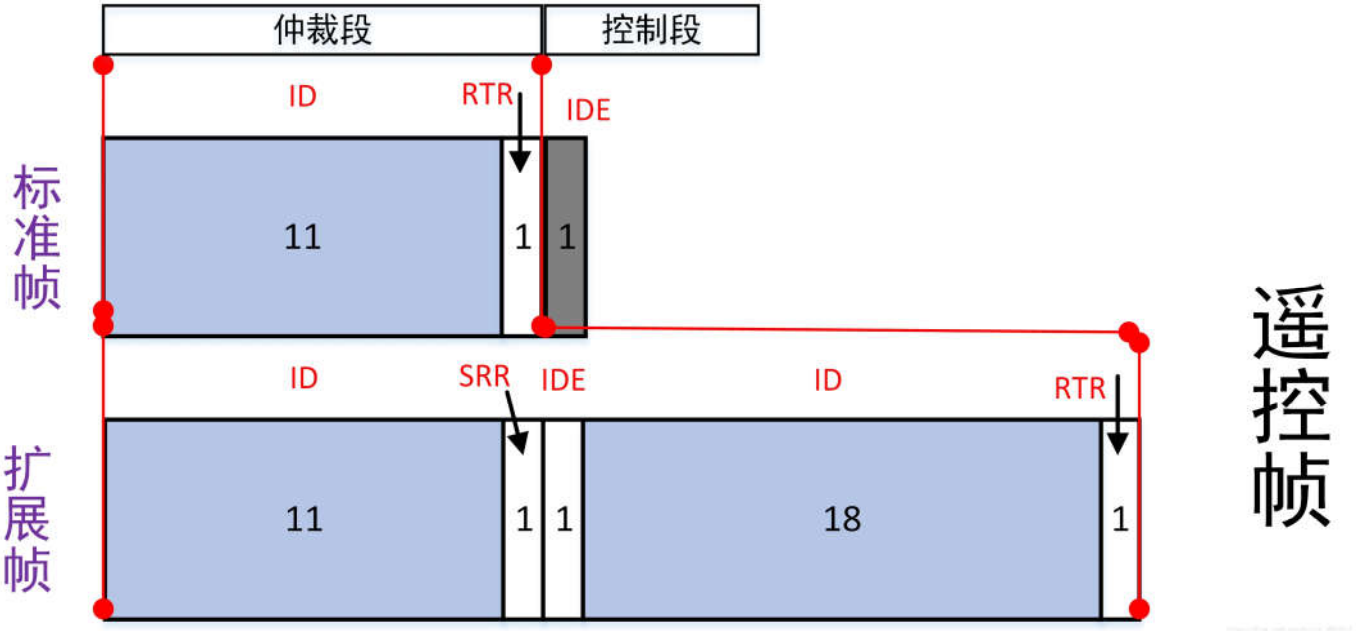
2）SRR位

Substitutes for Remote Requests Bit（替代远程请求位），**在扩展帧（数据帧或遥控帧）中，SRR恒为隐性位1**，并且可以发现，扩展帧的隐性SRR位正好对应标准帧的显性RTR位，这就解释了 **SRR位的作用：在前11位ID号相同的情况下，标准数据帧的优先级高于扩展数据帧**；



3）IDE位

全称：Identifier Extension Bit（标识符扩展位）。**在扩展帧中恒为隐性1，在标准帧中，IDE位于控制段，且恒为显性0。**且扩展帧IDE位和标准帧IDE位位置对应，这就保证了：**在前11位ID号相同的情况下，标准遥控帧的优先级一定高于扩展遥控帧。**



总结：
在ID号前11位相同的情况下：

- RTR：保证数据帧优先级高于遥控帧；
- SRR：保证标准数据帧的优先级高于扩展数据帧。
- IDE：保证标准遥控帧的优先级高于扩展遥控帧。

2.2.3 报文过滤

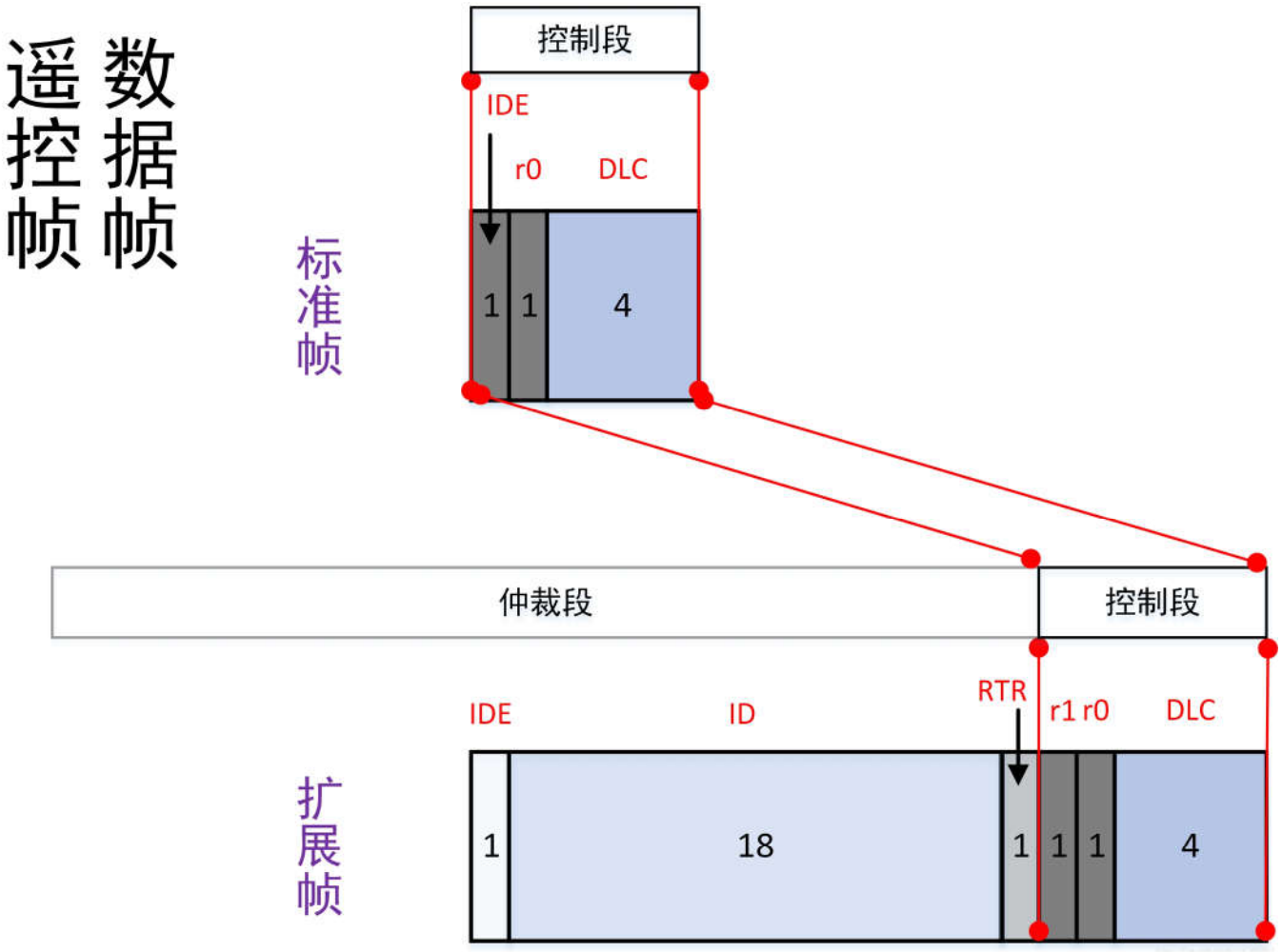
在CAN总线中**没有地址**的概念，CAN总线是通过**报文ID**来实现收发数据的。CAN节点上都会有一个**验收滤波ID表**，其位于CAN节点的验收滤波器中，如果总线上的报文的ID号在某个节点的验收滤波ID表中，那么这一帧报文就能通过该节点验收滤波器的验收，该节点就会接收这一帧报文。

比如：Node_A发送了一帧ID号为ID_1的报文Msg_1，Node_B的验收滤波ID表中恰好有ID_1，于是乎Msg_1就会被Node_B接收。

> Tips: 报文过滤机制体现了CAN通信的两条特点：
> 1）一对一、组播和广播
> 2）系统的柔性：正是因为CAN总线上收发报文是基于报文ID实现的，所以总线上添加节点时不会对总线上已有的节点造成影响。

2.3 控制段

数据帧和遥控帧的控制段结构相同：



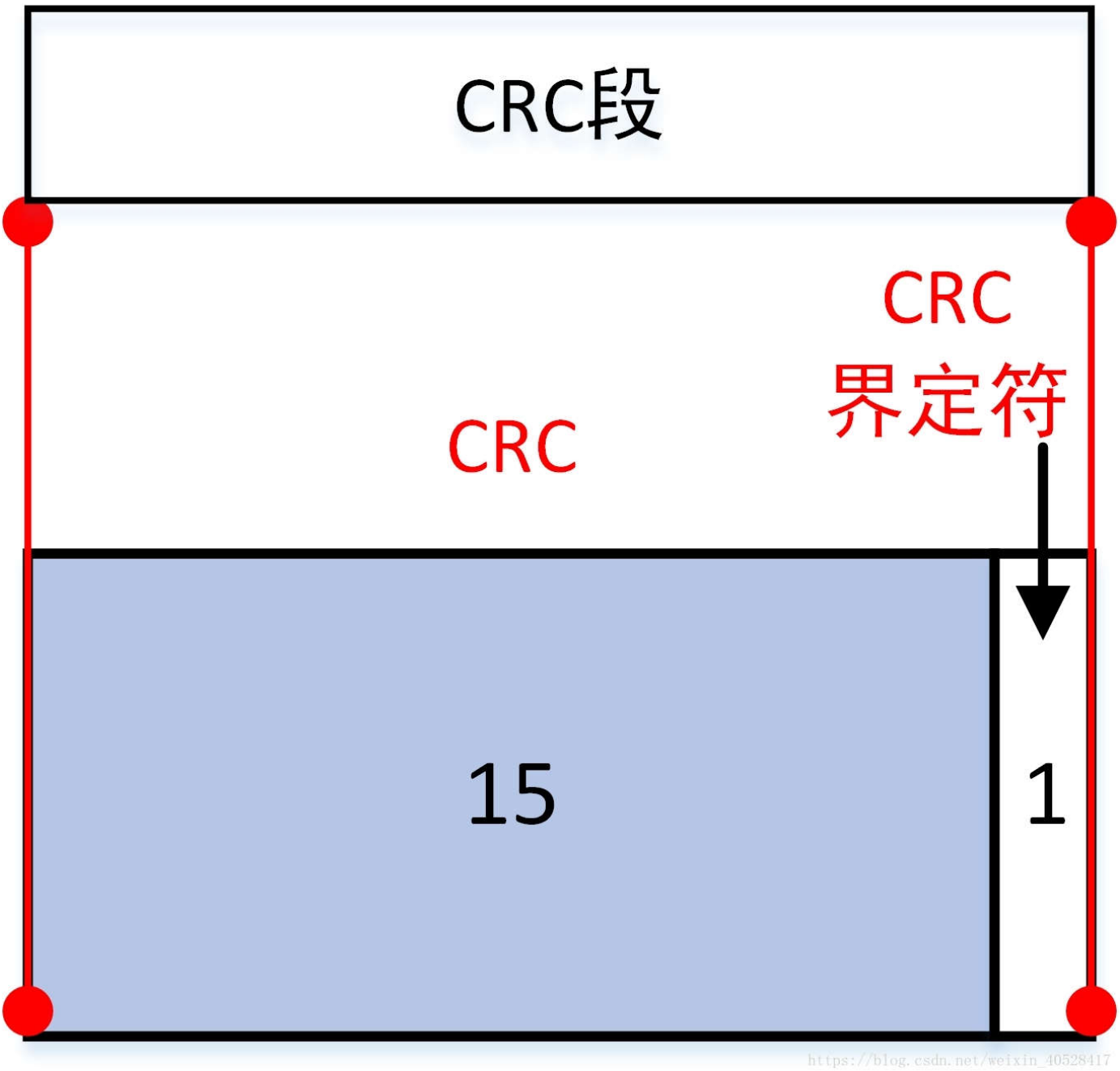
- 标准帧中IDE位对应扩展帧中的IDE位，保证在前11位ID号相同的情况下，标准帧的优先级一定高于扩展帧；
- 然后是保留位r0和r1（扩展帧），保留位r0和r1必须以显性电平发送，但是接受方可以接受显性、隐性及其任意组合的电平；
- 最后是4个字节的**DLC**（DLC3、DLC2、DLC1、DLC0）代表数据长度，指示了**数据段中的字节数。**对于没有数据段的遥控帧，DLC表示该遥控帧对应的数据帧的数据段的字节数。

2.4 数据段

数据段可以包含0~8个字节的数据，从MSB（最高位）开始输出。

2.5 CRC段

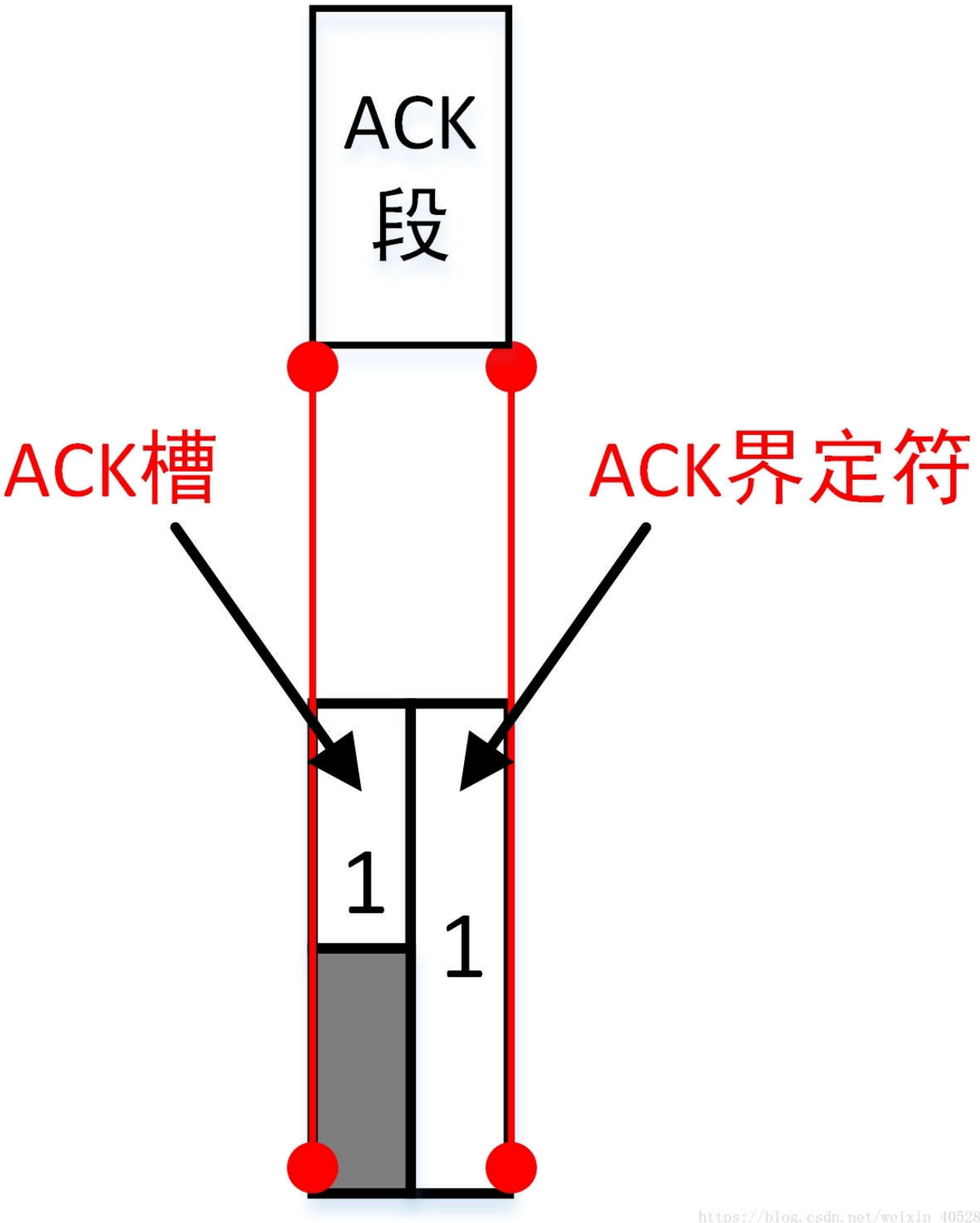
CRC段包含CRC校验序列和CRC界定符。



CRC校验序列是根据多项式生成的CRC值，**其计算范围包括：帧起始、仲裁段、控制段和数据段。**
CRC界定符恒为隐性1。

2.6 ACK段

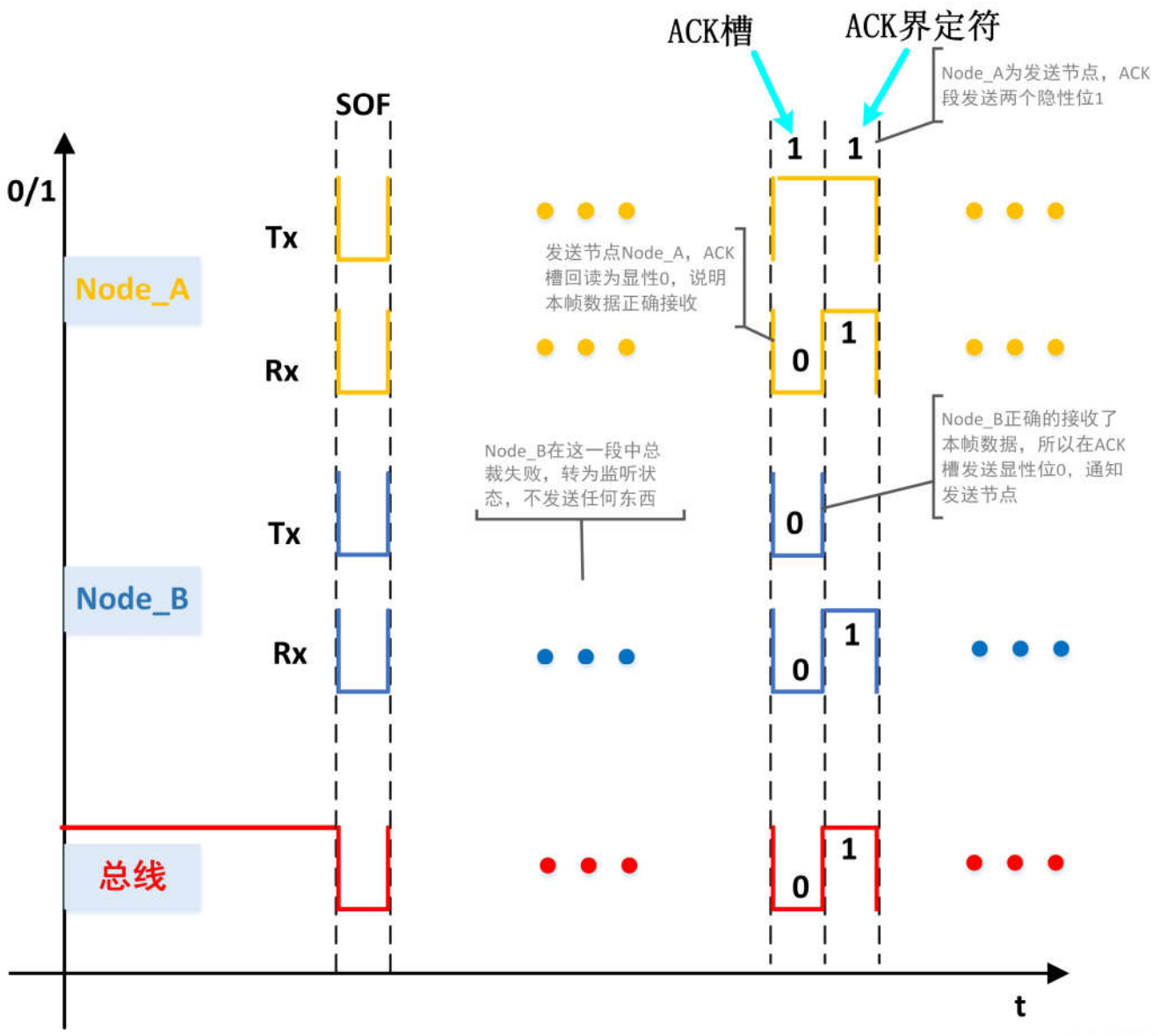
ACK段包含**ACK槽**和**ACK界定符**两个位。



- **发送节点**在ACK段发送**两个隐性位**，即发送方发出的报文中ACK槽为隐性1；
- **接收节点**在接收到正确的报文之后会在ACK槽发送显性位0，通知发送节点正常接收结束。所谓接收到正确的报文指的是接收到的报文没有填充错误、格式错误、CRC错误。

> Tips: 我们以标准数据帧为例来分析**ACK段的工作方式**：如图所示，Node_A为发送节点，Node_B为接收节点。Node_A在ACK段发送两个隐性位1。Node_B正确接收到这一报文后，在ACK段的ACK槽中填充了一个显性位0。注意，这个时候**Node_A回读到的总线上的额电平为显性0**，于是这个时候，Node_A就知道自己发出去的

报文至少有一个节点正确接收了。



2.7 帧结束

帧结束段表示该帧报文的结束，由7个隐性位构成。