原创 CAN总线学习笔记（4）- CAN协议过载帧和帧间隔

2018年04月09日 19:28:38 小兵大将0221 阅读数 4670 更多

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/weixin_40528417/article/details/79871311

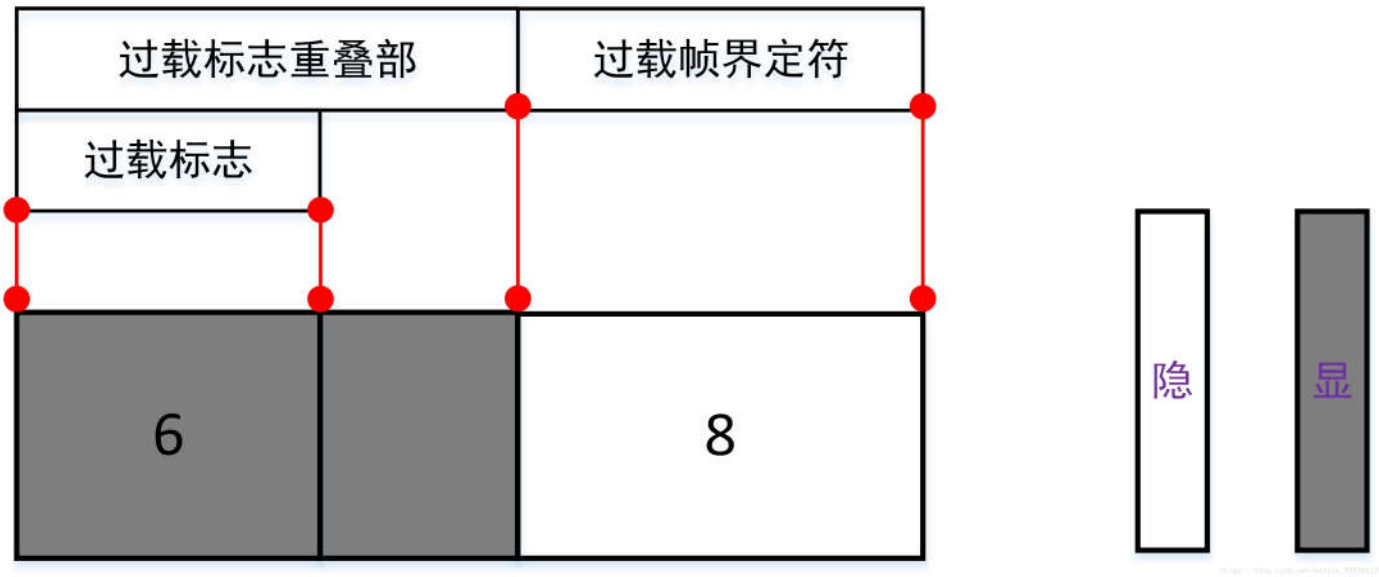
依照瑞萨公司的《CAN入门书》的组织思路来学习CAN通信的相关知识，并结合网上相关资料以及学习过程中的领悟整理成笔记。好记性不如烂笔头，加油！

1 过载帧

过载帧是**接收节点**向总线上其它节点报告自身**接收能力达到极限**的帧。
上面这句话可以这样理解：接收节点Node_A接收报文的能力达到极限了，于是Node_A就会发出过载帧来告诉总线上的其它节点（包括发送节点），我接收节点Node_A已经没有能力处理你们发来的报文了。

过载帧包括：**过载标志**和**过载界定符**两个部分

- **过载标志**：连续6个显性位；
- **过载界定符**：连续8个隐性位。
- 与错误帧类似，过载帧中有**过载帧重叠部分**，且形成过载重叠标志的原因与形成错误帧中的错误重叠标志的原因是相同的



对于过载帧的帧结构我们可以这样理解：接收节点Node_A达到接收极限时，就会发出过载帧到总线上，显然，过载标志的6个连续显性位会屏蔽掉总线上其它节点的发送，也就是说这个时候Node_A通过发送过载帧的方式来破坏其它节点的发送，这样在Node_A发送过载帧期间，其它节点就不能成功发送报文，于是就相当于把其它节点的发送推迟了，也就是说Node_A在其发送过载帧的这段时间得以“休息”。

有三种情况会引起过载帧：

- 接收节点自身原因。接收节点由于某种原因需要延迟接收下一个数据帧或者遥控帧。
- 在帧间隔的间歇段的第一位和第二位检测到一个显性位（正常的间歇段都是隐性位）
  帧间隔的间隔段本应是三个连续的隐性位，如果接收节点Node_A在间隔段检测到显性位，那么就意味着此时有报文发向接收节点Node_A，但这个时候是不应该有报文发来的，于是Node_A发送过载帧。
- CAN节点在**错误界定符**或**过载界定符**的第八位(最后一位)听到一个**显性位0**，节点会发送一个过载帧，且错误计数器不会增加。
  接收节点Node_A在错误界定符和过载界定符的最后一位听到显性位，也意味着有报文发向Node_A，但这个时候是不应该有报文发来的，于是Node_A发送过载帧。

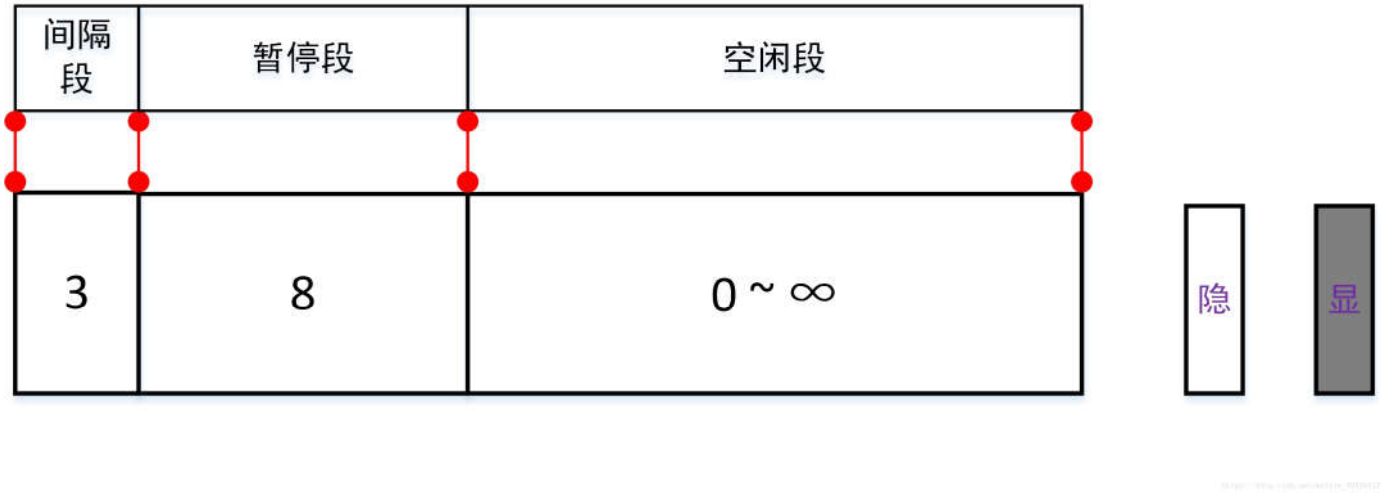
2 帧间隔

帧间隔是用来隔离数据帧（或者遥控帧）的，也就是说，数据帧（或者遥控帧）通过插入**帧间隔**可以将本帧与先行帧（数据帧、遥控帧、错误帧、过载帧）分隔开来。

> Tips: 过载帧和错误帧的前面不能插入帧间隔。

帧间隔有两种不同的形式：
主动错误状态的帧间隔：

被动错误状态的帧间隔：



间隔段：连续三个隐性位；间隔段期间，所有节点不允许发送数据这或遥控帧，只要在这期间监听到显性位，接收节点就会发送过载帧。
空闲段：连续隐性位，个数不一定，0个或者多个都可以。总线空闲的时间是任意长的，只要总线空闲，节点就可以竞争总线。
暂停段：只有处于被动错误状态的节点在发送帧间隔的时候，才会在帧间隔中插入8个连续隐性位的暂停段。

暂停段，又叫做延迟传送段，为什么节点处于被动状态时会有这样一段呢。原因如下：首先，考虑主动错误状态的节点Node_A，发送主动错误标志之后，随之就要重新发送刚刚发送失败的报文，但是为了间隔开与前面刚刚发送的错误帧，总线在错误帧之后就会插入３个隐形位的帧间隔，在这３个隐形位期间，其它的节点不足以判定总线空闲（需要连续11个隐性位才能判定），所以Node_A仍然占据着总线的控制权，于是在帧间隔之后，Node_A能够接着发送报文。现在Node_A转入到被动错误状态了，说明它已经不是很可靠了，这个时候如果没有延迟传送段，在Node_A发出被动错误标志之后，它仍然能够在3位的帧间隔之后立即重新发送报文，这是不符合我们对被动错误状态的处理要求的当然也是不符合CAN协议的，于是乎对于发送出被动错误标志的节点，总线在帧间隔中加入了８个连续隐性位的延迟传送段，这样的3+8=11个连续隐性位。就能让Node_A在这个帧间隔期间失去对总线的控制权，从而优先保证其它正常(处于主动错误状态)节点能够使用总线，而不必等着一个已经不可靠的Node_A占据总线。