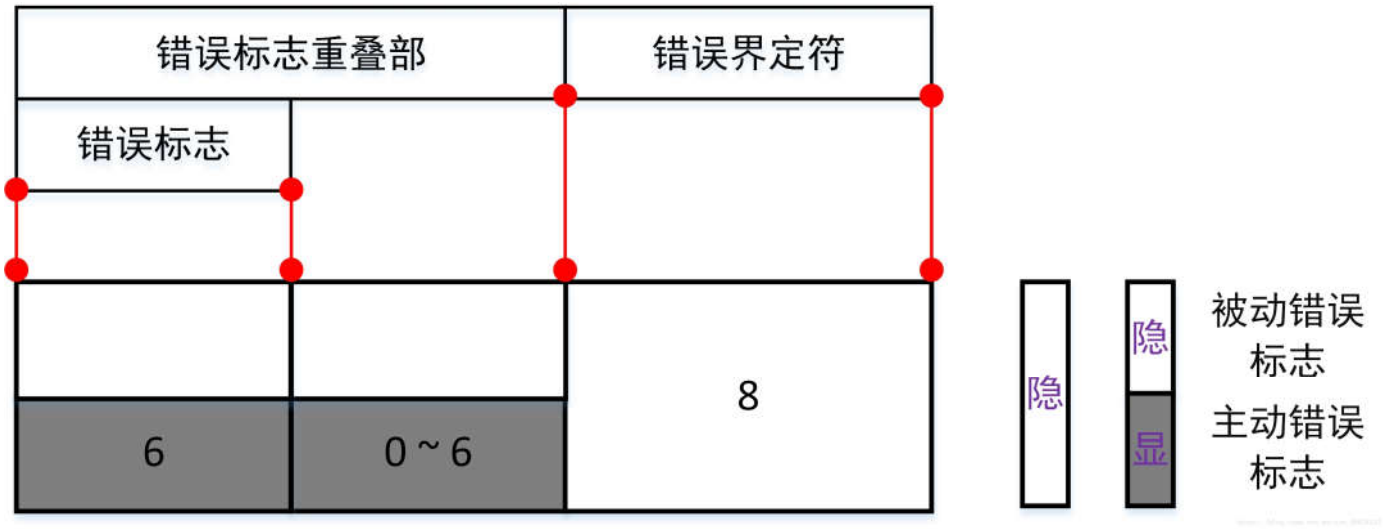
原创 CAN总线学习笔记（3）- CAN协议错误帧

2018年04月09日 10:55:54 小兵大将0221 阅读数 7074 文章标签： CAN总线 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 错误帧的帧结构

在发送和接收报文时，总线上的节点如果检测出了错误，那么该节点就会发送错误帧，通知总线上的节点，自己出错了。
错误帧由**错误标志**和**错误界定符**两个部分组成。



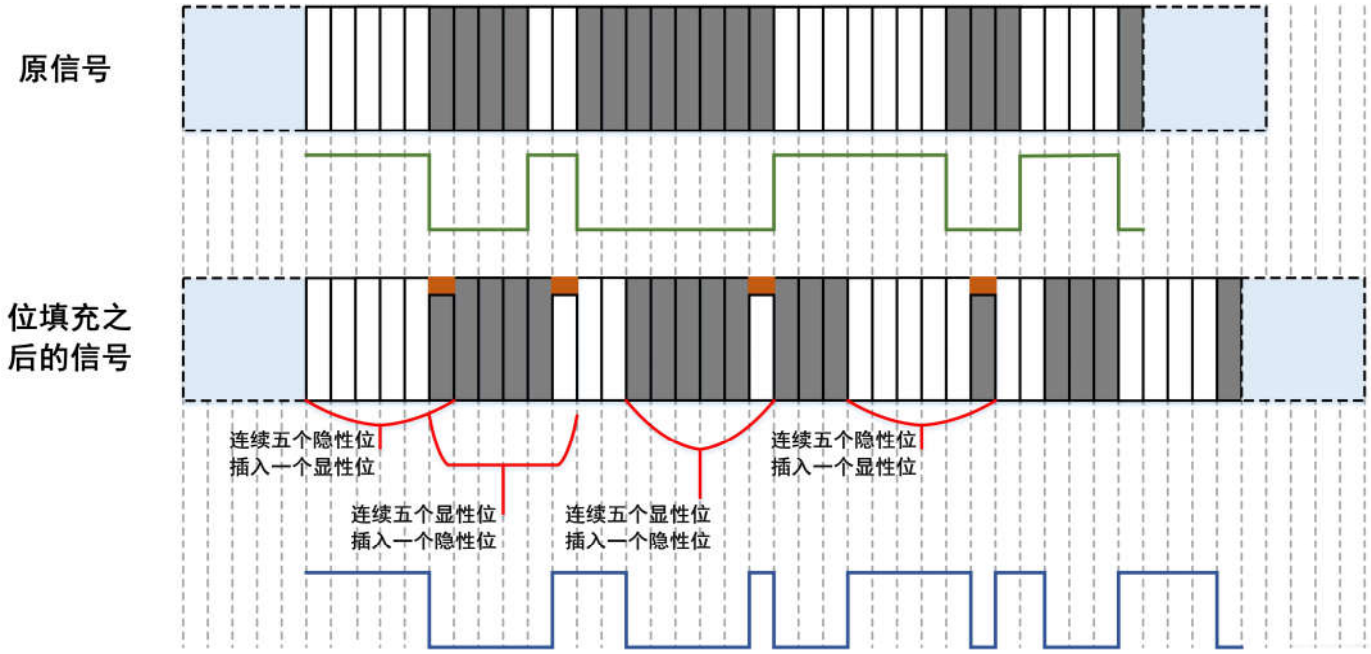
- 主动错误标志：6个连续的显性位；
- 被动错误标志：6个连续的隐性位；
- 错误界定符：8个连续的隐性位。

可以看到在错误标志之后还有0~6位的错误标志重叠，这一段最低有0个位，最多有6个位，关于这一段是怎么形成的，将在下文中解释。

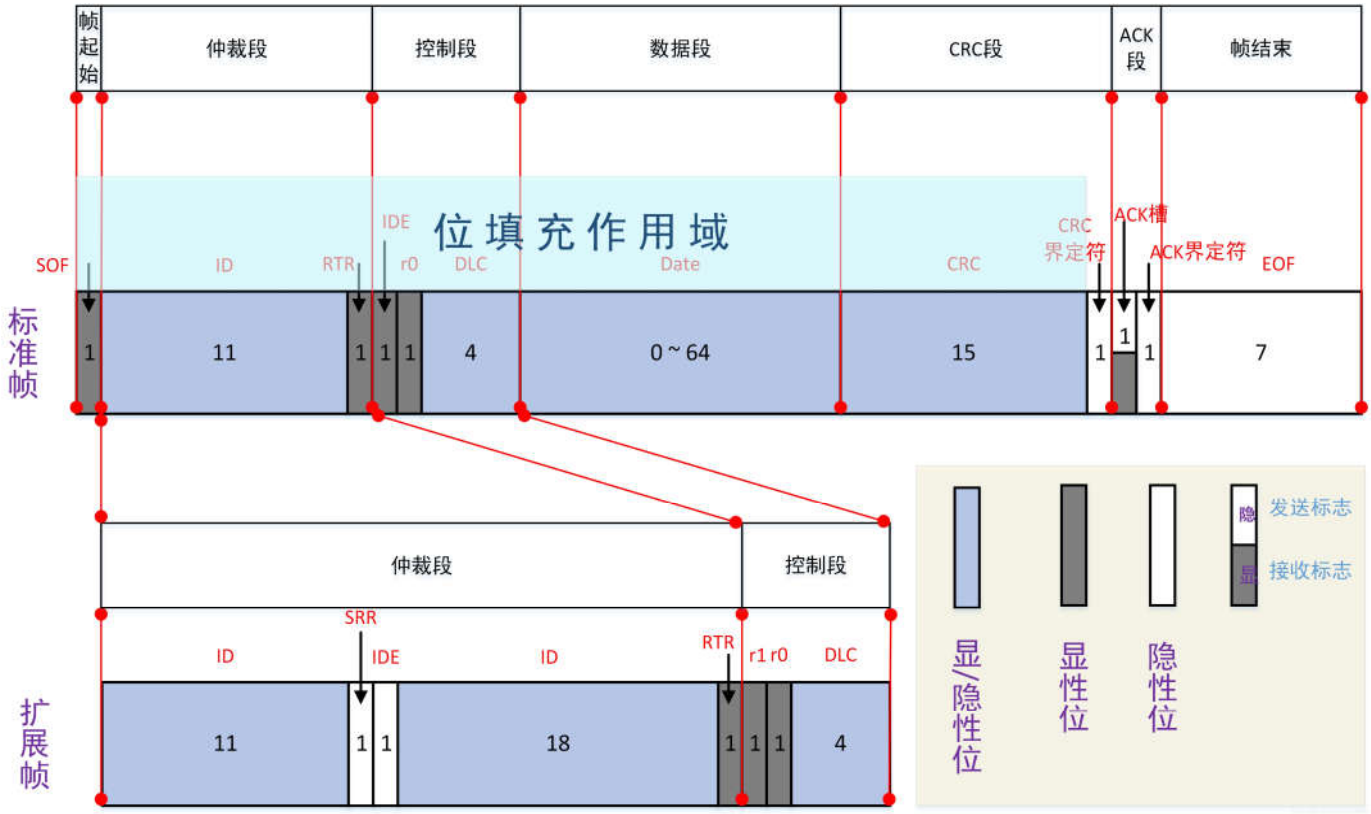
2 错误检测

2.1 位填充原则

在了解CAN总线中的错误检测之前，首先需要了解什么是位填充。
CAN协议中规定，当**相同极性的电平持续五位**时，则**添加一个极性相反的位**。



- 对于**发送节点**而言：
  在发送**数据帧**和**遥控帧**时，对于**SOF~CRC(除去CRC界定符)** 之间的位流，相同极性的电平如果持续5位，那么在下一个位插入一个与之前5位反型的电平；
- 对于**接收节点**而言：
  在接收**数据帧**和**遥控帧**时，对于**SOF~CRC(除去CRC界定符)**之间的位流，相同极性的电平如果持续5位，那么需要删除下一位再接收。



> Tips: 注意：填充位的添加和删除是由发送节点和接收节点完成的，CAN-BUS只负责传输，不会操纵信号。

2.2 错误的种类

在CAN总线通信中，一共有五种错误：

- 位错误
- ACK错误

- 填充错误
- CRC错误
- 格式错误

2.2.1 位错误（Bit Check Error）

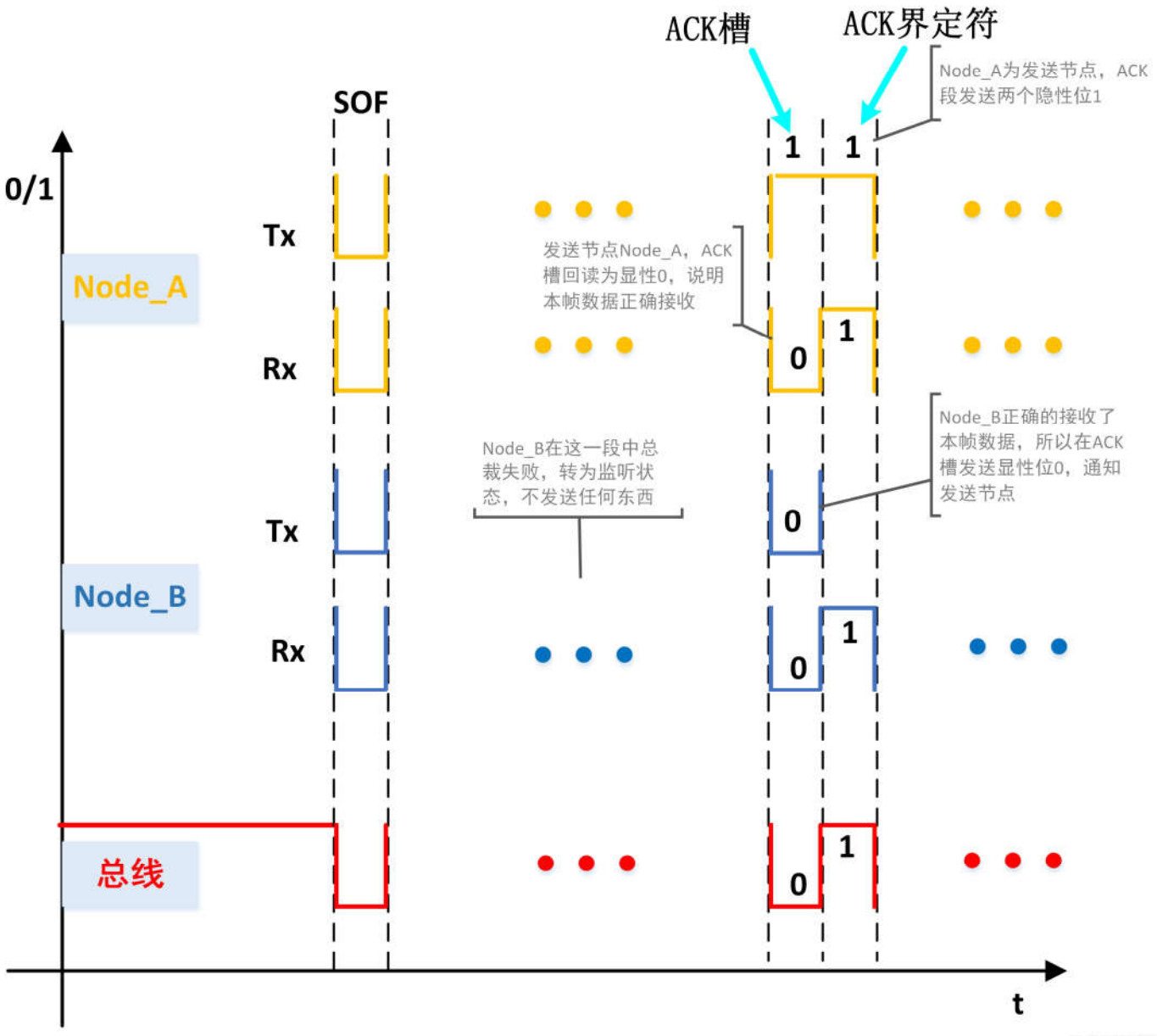
节点将自己发送到总线上的电平与同时从总线上回读到的电平进行比较，如果发现二者不一致，那么这个节点就会检测出一个位错误。

实际上所谓“发出的电平与从总线上回读的电平不一致”，指的就是节点向总线发出隐性位，却从总线上回读到显性位或者节点向总线发出显性位，却从总线上回读到隐性位这两种情况。

> Tips: 有三种例外情况不属于位错误：
>
> - 在**仲裁区**，节点向总线发送隐性位却回读到显性位，不认为是位错误，这种情况表示该节点仲裁失败；
> - 在**ACK槽**，节点向总线发送隐性位却回读到显性位，不认为是位错误，这种情况表示，该节点当前发送的这一帧报文至少被一个其它节点正确接收；
> - **该节点发送被动错误标志**，节点Node_A向总线发送连续六个隐性位（被动错误标志）却回读到显性位，不认为是位错误。因为被动错误标志是六个连续的隐性位，所以在总线上按照线与机制，有可能这六个连续隐性位被其它节点发送的显性电平“吃掉”；

2.2.2 ACK错误（Acknowledgment Error）

按照CAN协议的规定，在一帧报文（数据帧或者遥控帧）发出之后，如果接收节点Node_B成功接收了该帧报文，那么接收节点Node_B就要在该帧报文ACK槽对应的时间段内向总线上发送一个显性位来应答发送节点Node_A。这样发送节点Node_A就会在ACK槽时间段内从总线上回读到一个显性位。因此：
当发送节点Node_A在**ACK槽时间段内没有回读到显性位**，那么发送节点Node_A就会检测到一个ACK应答错误。这表示没有一个节点成功接收该帧报文。

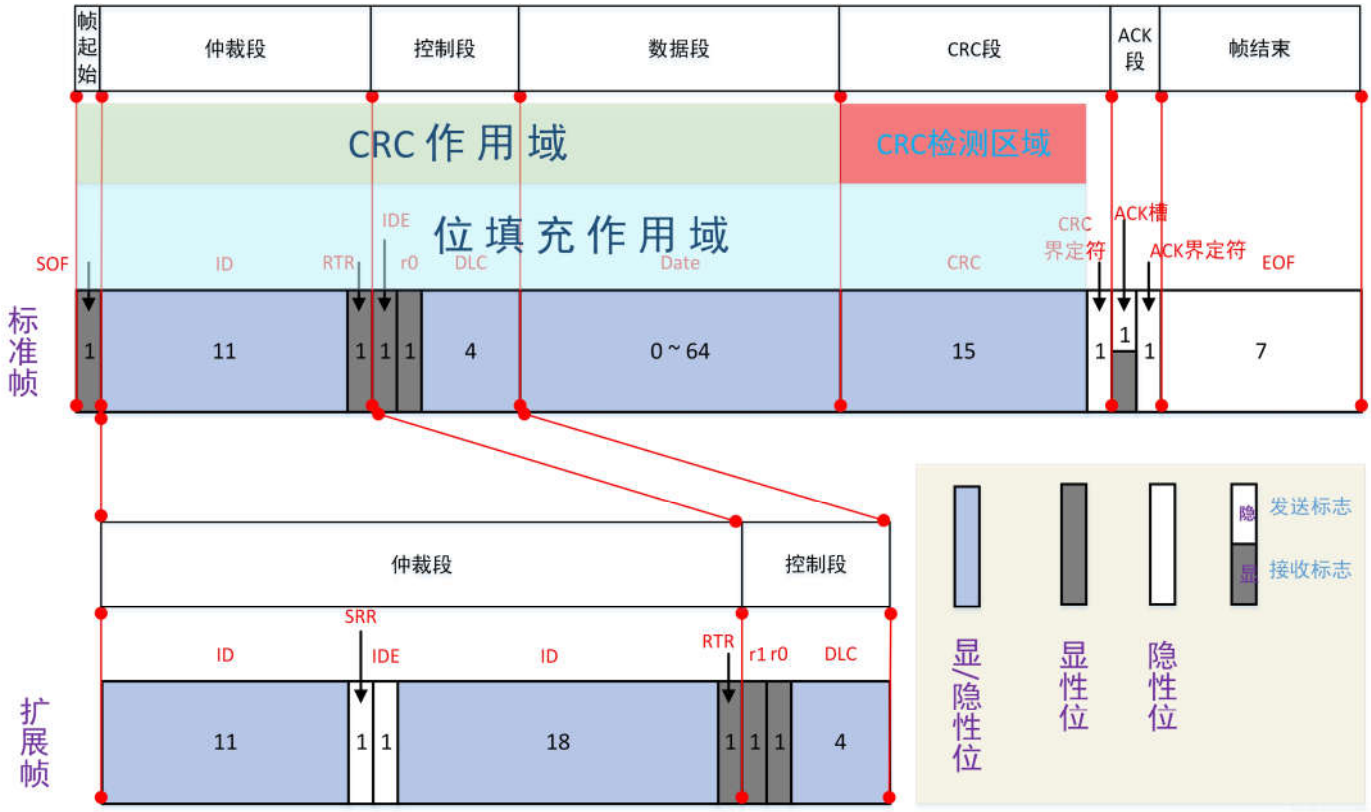


2.2.3 填充错误（Fill Error）

在需要执行**位填充原则**的帧段（数据帧遥控帧的SOF~CRC序列），检测到**连续六个同性位**，则检测到一个填充错误。

2.2.4 CRC错误

发送节点Node_A在发送数据帧或者遥控帧时，会计算出该帧报文的CRC序列。接收节点Node_B在接收报文时也会执行相同的CRC算法，如果接收节点Node_B计算出的CRC序列值与发送节点Node_A发来的CRC序列值不一致，那么接收节点就检测到一个CRC错误。



2.2.5 格式错误

在一帧报文发送时，如果在必须发送预定值的区域内检测到了非法值，那么就检测到一个格式错误。
CAN报文中，有预定值的区域包括：

- 数据帧和遥控帧的**CRC界定符、ACK界定符、EOF**；
- **错误帧界定符**
- **过载帧界定符**

3 错误通知

上一节中，讲到CAN通信中有五种错误，并且介绍了在什么情况下能够检测到这几种错误，在检测到错误之后，检测到错误的节点就要发送错误帧到总线上来通知总线上的其他节点。
错误帧有的带有**主动错误标志**，有的带有**被动错误标志**，而且错误标志重叠部分的字节数也不一样，那么问题就来了：

- 什么情况下发送带有主动错误标志的错误帧；
- 什么情况下发送带有被动错误标志的错误帧；
- 在哪个时间点发送错误帧；
- 错误标志重叠部分是怎样形成的；

3.1 节点错误状态

按照CAN协议的规定，CAN总线上的节点始终处于以下三种状态之一。

- 主动错误状态
- 被动错误状态
- 总关闭状态

当满足一定的条件时，节点可以从一种状态转换为另外一种状态。

> Tips: 需要注意的是：

- 处于主动错误状态，表示该节点具备发出主动错误标志的能力；
- 处于被动错误状态，表示节点具备发出被动错误标志的能力。

1）主动错误状态

- 节点处于主动错误状态**可以正常通信**；
- 处于主动错误状态的节点（可能是接收节点也可能是发送节点）在检测出错误时，**发出主动错误标志。**

2）被动错误状态

- 节点处于被动错误状态**可以正常通信**；
- 处于被动错误状态的节点（可能是接收节点也可能是发送节点）在检测出错误时，**发出被动错误标志。**

> Tips: 注意：这里说处于主动错误状态或被动错误状态的节点仍然可以正常通信，这里的**正常通信**指的是：节点仍然能够从总线上接收报文，也能够竞争总线获胜后向总线上发送报文。但是不代表接收的报文一定正确也不代表一定能正确的发送报文。

3）总线关闭状态

- 节点处于总线关闭状态，那么该节点**不能收发**报文；
- 处于总线关闭状态的节点，只能一直等待，在满足一定条件的时候，再次进入到**主动错误状态。**

3.2 错误状态的转换

现在我们知道：

- 处于主动错误状态的节点在检测到错误时会发送带有主动错误标志的错误帧；
- 处于被动错误状态的节点在检测到错误时会发送带有被动错误标志的错误帧。

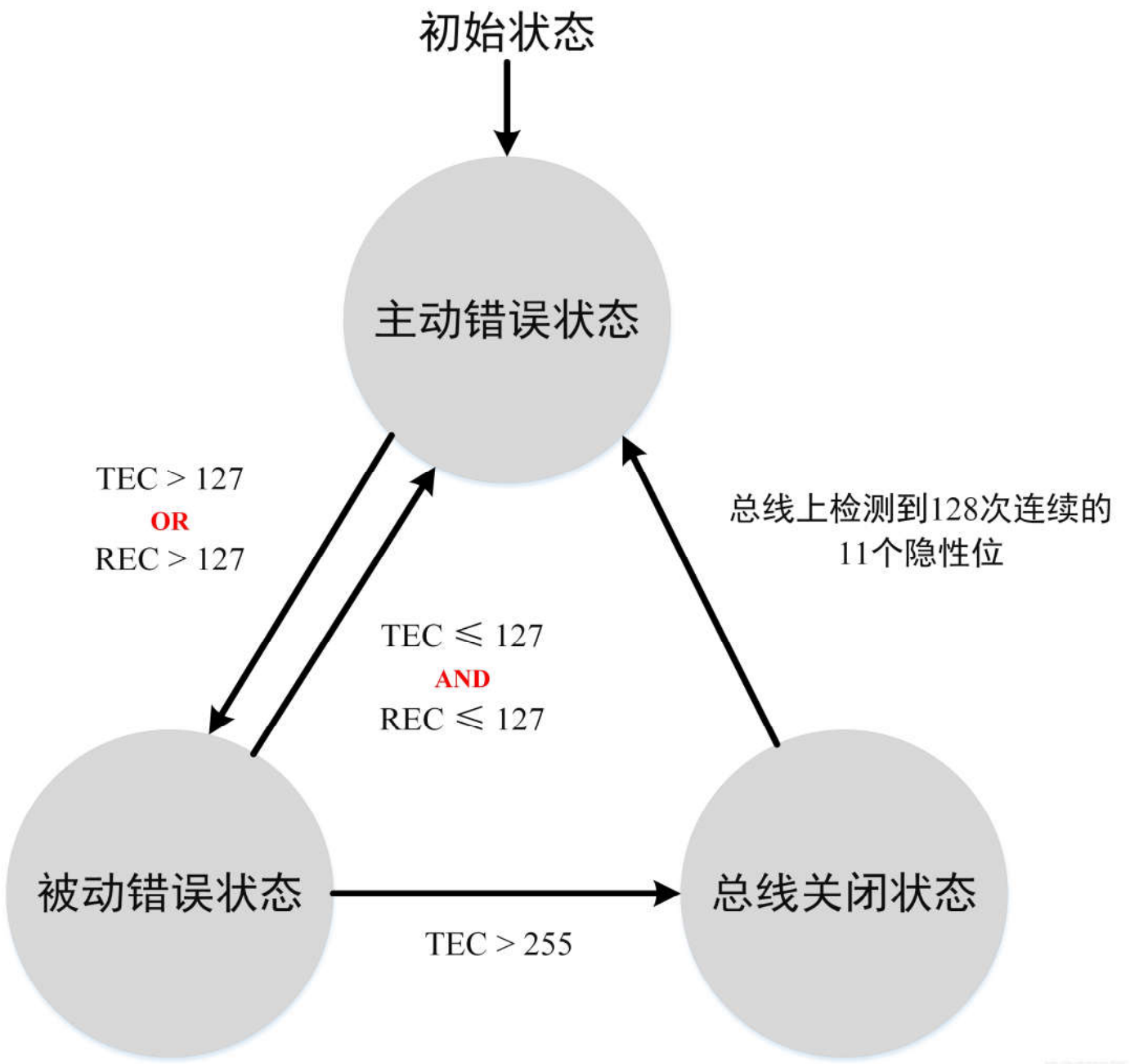
那么一个CAN节点在什么情况下处于主动错误状态，什么情况下处于被动错误状态呢？

根据CAN协议的规定，在CAN节点内，有两个计数器：**发送错误计数器（TEC）和接收错误计数器（REC）**。

> Tips: 需要注意的是：这两个计数器计得不是收发报文的数量，也不是收发错误帧的数量。TEC和RCE计数值的变化，是根据下表的规定来进行的

	接受和发送错误计数值的变动条件	发送错误计数值 (TEC)	接收错误计数值 (REC)
1	接收单元检测出错误时。 例外：接收单元在发送错误标志或过载标志中检测出“位错误”时，接收错误计数值不增加。	—	+1
2	接收单元在发送完错误标志后检测到的第一个位为显性电平时。	—	+8
3	发送单元在输出错误标志时。	+8	—
4	发送单元在发送主动错误标志或过载标志时，检测出位错误。	+8	—
5	接收单元在发送主动错误标志或过载标志时，检测出位错误。	—	+8
6	各单元从主动错误标志、过载标志的最开始检测出连续 14 个位的显性位时。 之后，每检测出连续的 8 个位的显性位时。	发送时 +8	接收时 +8
7	检测出在被动错误标志后追加的连续 8 个位的显性位时。	发送时 +8	接收时 +8
8	发送单元正常发送数据结束时（返回 ACK 且到帧结束也未检测出错误时）。	–1 TEC=0 时±0	—
9	接收单元正常接收数据结束时（到 CRC 未检测出错误且正常返回 ACK 时）。	—	1≤REC≤127 时-1 REC=0 时±0 REC>127 时 设 REC=127
10	处于总线关闭态的单元，检测到 128 次连续 11 个位的隐性位。	TEC=0	REC=0

CAN节点错误状态的转换，就是基于这两个计数器来进行的。



可以看出，节点错误状态的转换就是一个**“量变”**到**“质变”**的过程：

####1) 主动错误状态

最开始TCE和REC都小于127时**，就处于**主动错误状态。**

在这一状态下，节点检测到一个错误就会发送带有**主动错误标志**的**错误帧**，因为主动错误标志是**连续六个显性位**，所以这个时候主动错误标志将会“覆盖”掉总线上其它节点的发送，而之前在CAN总线上传输的报文就被这“六个连续显性位”破坏掉了。

如果发出主动错误帧的节点是发送节点，这个情况下就相当于：刚刚发送的那一帧报文我发错了，现在我破坏掉它（发送主动错误帧），你们不管收到什么都不算数；

如果发出主动错误帧的节点是接收节点，这个情况就相当于：刚刚我收报文的时候发现了错误，不管你们有没有发现这个错误，我现在主动站出来告诉大家这个错误，并把这一帧报文破坏掉（发送主动错误帧），刚才你们收到的东西不管对错都不算数了。

> Tips: 处于主动错误状态，说明这个节点目前是比较可靠的，出现错误的原因可能不是它本身的问题，即刚刚检测到的错误可能不仅仅只有它自己遇到，正是因为这一点，整个总线才相信它报告的错误，允许它破坏掉发送中的报文，也就是将这一次的发送作废。

2）被动错误状态

如果某个节点发送错误帧的次数较多，必将使得**TCE > 127 或者 REC > 127**，那么该节点就处于**被动错误状态。**

在这一状态下，节点Node_A检测到一个错误就会发送带有被动错误标志的错误帧，因为被动错误标志是连续六个隐性位，所以这个时候总线上正在传输的报文位流不会受到该被动错误帧的影响，其它的节点该发送的发送，该接收的接收，没人搭理这个发送被动错误帧的节点Node_A。

如果发出被动错误帧的节点Node_A为报文的发送节点，那么在发送被动错误帧之后，刚刚正在发送的报文被破坏，并且Node_A不能在错误帧之后随着**连续发送**刚刚发送失败的那个报文。随之而来的是**帧间隔**，并且连带着8位隐性位的 **“延迟传送”** 段；这样总线电平就呈现出连续11位隐性位，总线上的其它节点就能判定总线处于空闲状态，就能参与总线竞争。此时如果Node_A能够竞争成功，那么它就能接着发送，如果竞争不能成功，那么就接着等待下一次竞争。这种机制的目的正是为了让其它正常节点（处于主动错误）优先使用总线。

> Tips: 处于**被动错误状态**，说明这个节点目前是不太可靠的，出现错误的原因可能是它本身的问题，即刚刚检测到的错误可能仅仅只有它自己遇到，正是因为这一点，整个总线才不信任它报告的错误，从而只允许它发送六个连续的隐性位，这样它才不会拖累别人。

3）总线关闭状态

如果一个处于被动错误状态的节点，仍然多次发送被动错误帧，那么势必导致**TEC > 255**，这样就处于**总线关闭状态。**

在总线关闭状态下的节点Node_A不能向总线上发送报文，也不能从总线上接收报文，整个节点脱离总线。等到检测到128次11个连续的隐性位时，TEC和REC置0，重新回到主动错误状态。

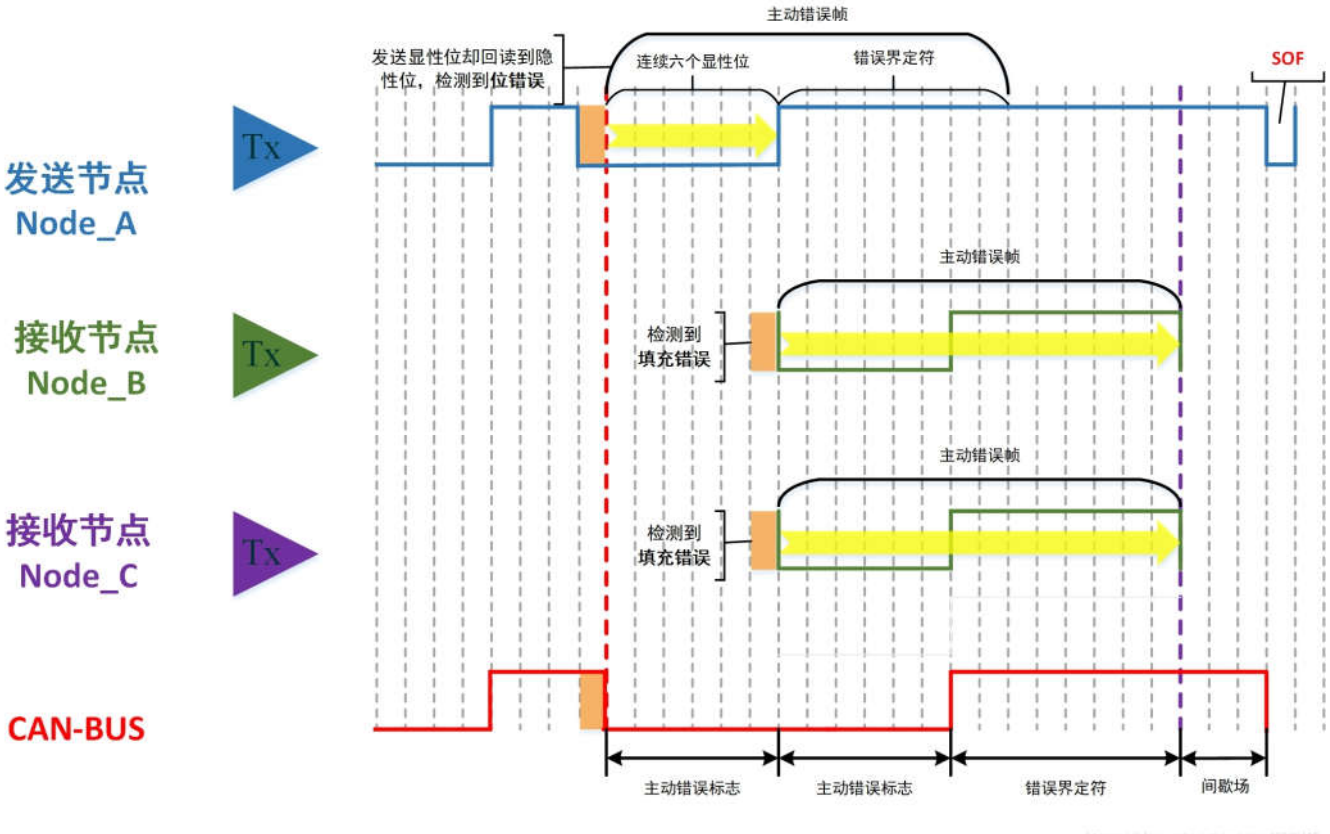
按照我的理解这个所谓“检测到128次11个连续隐性位”其实就是让这个节点隔离一段时间冷静下，因为它一旦处于总线关闭状态，就不会和总线有任何的联系，这个时候只要它计算时间等于达到传送128次11个连续隐性位所用的时间，就可以重新连到总线上。

> Tips: 处于总线关闭状态说明，这个节点目前挂掉了，总线先把它踢开，这样它才不会拖累别人，等到它冷静一段时间之后再回到总线上。

3.3 错误帧的发送

在检测到错误之后，什么时候发送错误帧呢?
按照CAN协议的规定:

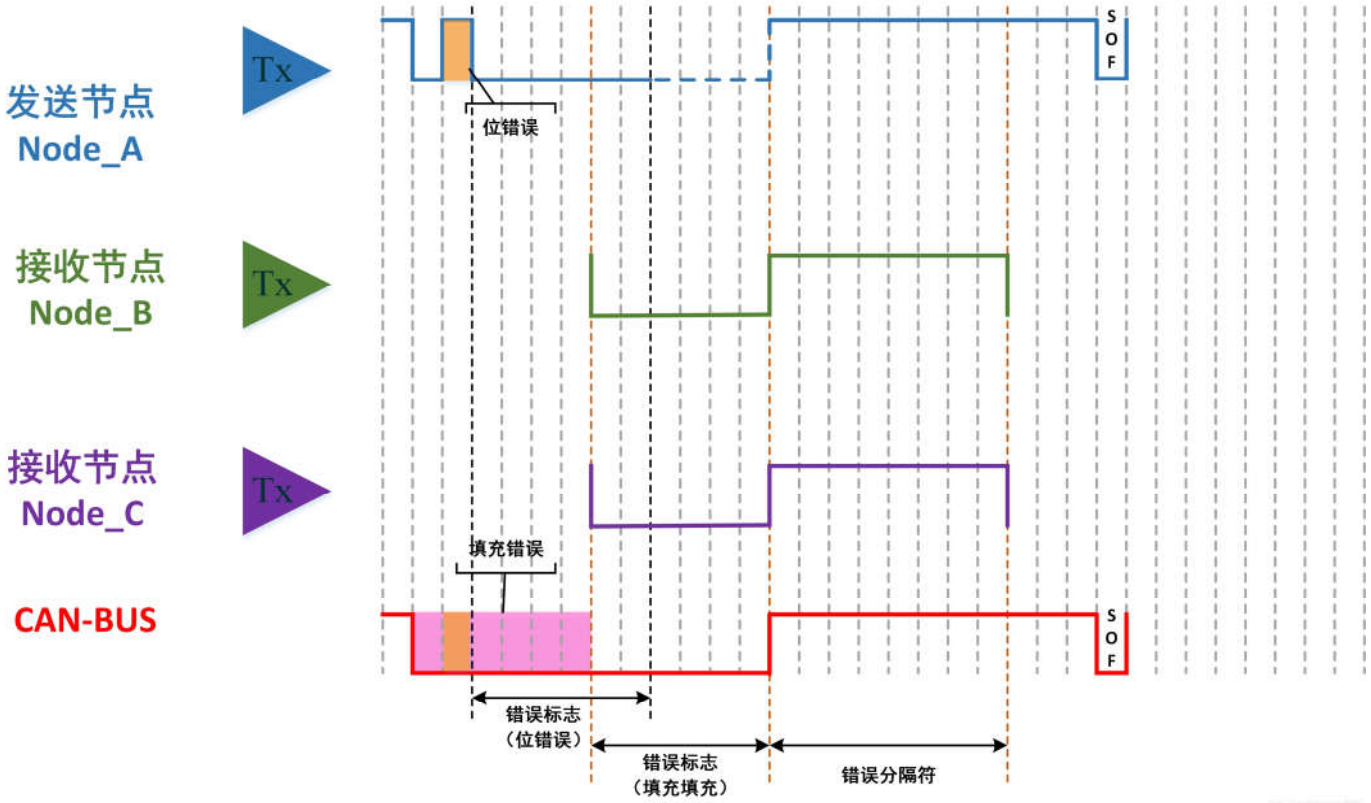
- 位错误、填充错误、格式错误、ACK错误。
  在错误产生的那一位的下一位开始发送错误帧。
- CRC错误
  紧随ACK界定符后的位发送错误帧。
  例子1:



(1) 发送节点Node_A发送一个显性位，但是却从总线上听到一个隐形位，于是Node_A节点就会检测到一个位错误;
(2) Node_A检测到位错误之后，立即在下一位开始发送主动错误帧：6个连续显性位的主动错误标志+8个连续隐性位的错误界定符;
(3) 对应Node_A发出的主动错误标志，总线上电平为6个连续显性位;
(4) 接收节点Node_B和Node_C从总线上听到连续6个显性位，那么就会检测到一个填充错误，于是这两个节点都会发送主动错误帧;
(5) 对应Node_B和Node_C发出的主动错误标志，总线电平又有6个连续显性电平，对应Node_B和Node_C发出的错误界定符，总线电平有8个连续的隐性电平。

（6）在间歇场之后，Node_A节点重新发送刚刚出错的报文。

例子2：



从上图中可以看出错误帧之中，错误标志重叠部分是怎样形成的，这个例子中，位错误的错误标志与填充错误的错误标志重叠两位，剩下的部分还有四位：