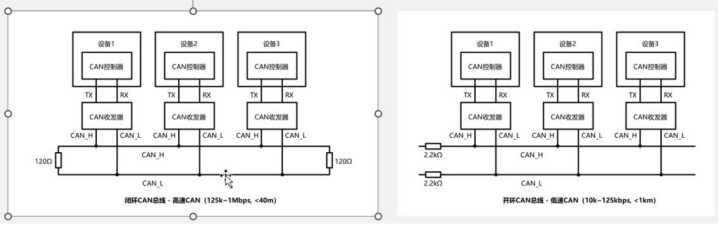
# CAN总线

- CAN总线（Controller Area Network Bus）控制器局域网总线
- CAN总线是由BOSCH公司开发的一种简洁易用、传输速度快、易扩展、可靠性高的串行通信总线，广泛应用于汽车、嵌入式、工业控制等领域
- CAN总线特征：
  - 两根通信线（CAN_H、CAN_L），线路少，无需共地
  - 差分信号通信，抗干扰能力强
  - 高速CAN（ISO11898）：125k~1Mbps, <40m
  - 低速CAN（ISO11519）：10k~125kbps, <1km
  - 异步，无需时钟线，通信速率由设备各自约定
  - 半双工，可挂载多设备，多设备同时发送数据时通过仲裁判断先后顺序
  - 11位/29位报文ID，用于区分消息功能，同时决定优先级
  - 可配置1~8字节的有效载荷
  - 可实现广播式和请求式两种传输方式
  - 应答、CRC校验、位填充、位同步、错误处理等特性

## 1 CAN硬件电路

江协科技 jiangxiekeji.com

- 每个设备通过CAN收发器挂载在CAN总线网络上
- CAN控制器引出的TX和RX与CAN收发器相连，CAN收发器引出的CAN_H和CAN_L分别与总线的CAN_H和CAN_L相连
- 高速CAN使用闭环网络，CAN_H和CAN_L两端添加120Ω的终端电阻
- 低速CAN使用开环网络，CAN_H和CAN_L其中一端添加2.2kΩ的终端电阻



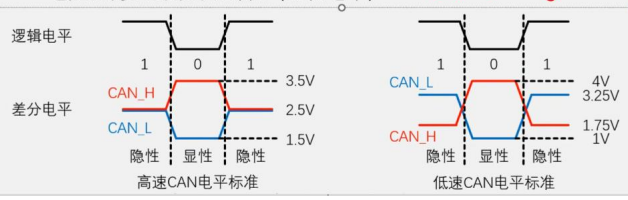

can总线（高速）想发1，不动。想发0，拉开总线

## 1 CAN电平标准

江协科技

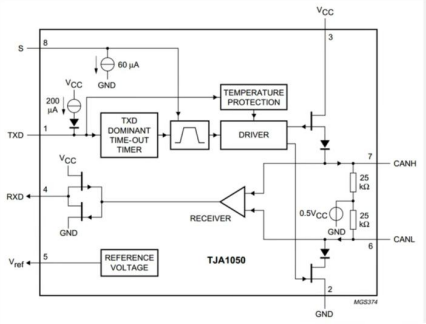
- CAN总线采用差分信号，即两线电压差（$V_{CAN_H}$-$V_{CAN_L}$）传输数据位
- 高速CAN规定：
  - 电压差为0V时表示逻辑1（隐性电平）
  - 电压差为2V时表示逻辑0（显性电平）
- 低速CAN规定：
  - 电压差为-1.5V时表示逻辑1（隐性电平）
  - 电压差为3V时表示逻辑0（显性电平）

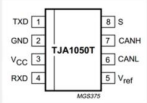
长距离传输有压降




## 1 CAN收发器 – TJA1050（高速CAN）

江协科技




PINNING

| SYMBOL | PIN | DESCRIPTION |
|---|---|---|
| TXD | 1 | transmit data input; reads in data from the CAN controller to the bus line drivers |
| GND | 2 | ground |
| $V_{CC}$ | 3 | supply voltage |
| RXD | 4 | receive data output; reads out data from the bus lines to the CAN controller |
| $V_{ref}$ | 5 | reference voltage output |
| CANL | 6 | LOW-level CAN bus line |
| CANH | 7 | HIGH-level CAN bus line |
| S | 8 | select input for high speed mode/silent mode |

## 2 CAN总线帧格式

- CAN协议规定了以下5种类型的帧：

| 帧类型 | 用途 |
| --- | --- |
| 数据帧 | 发送设备主动发送数据（广播式） |
| 遥控帧 | 接收设备主动请求数据（请求式） |
| 错误帧 | 某个设备检测出错误时向其他设备通知错误 |
| 过载帧 | 接收设备通知其尚未做好接收准备 |
| 帧间隔 | 用于将数据帧及遥控帧与前面的帧分离开 |

RO. 保留位

RTR：远程请求标志位，判断数据帧与遥控帧

IDE：标准格式还是扩展格式

## 2 数据帧

江协科技 jiangxiekeji.com

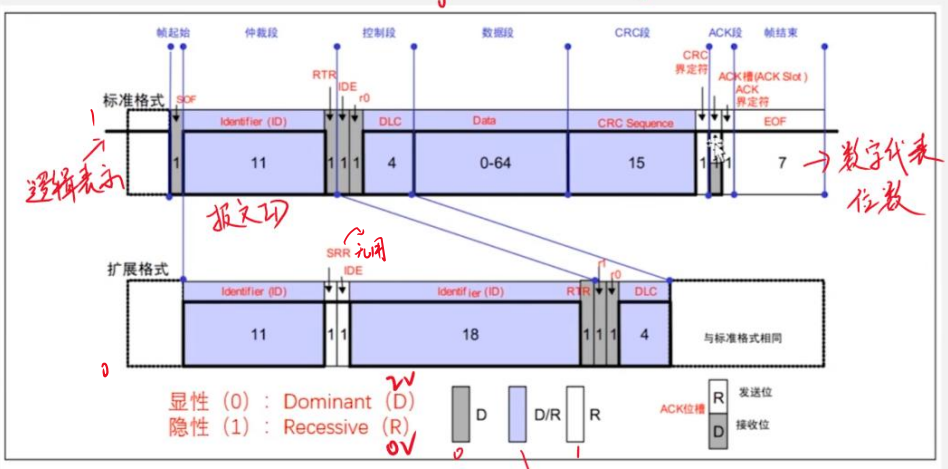



图 11. 数据帧的构成

根据数据自行决定0/1

DLC：表示Data长度，发1字节→0001，发8个字节→1000

报文ID：不用说明从机地址，所有设备都能接收，若是操控设备，报文ID是从机地址就行

CRC：校验位

ACK：发送后将ACK置为隐性1，无此时ACK槽发到为0

说明有设备接收到此请求，

反之同理

说明在ACK之前，只有发送方有权操作总线，ACK时接收可以操作，CRC界定符是给其操控时间

注：一条时序波形是由接收与发送共同构成，发送方每发一位，接收方都会ACK一下

EOT：停止位

## 2 遥控帧

江协科技 jiangxiekeji.com

- 遥控帧无数据段，RTR为隐性电平1，其他部分与数据帧相同

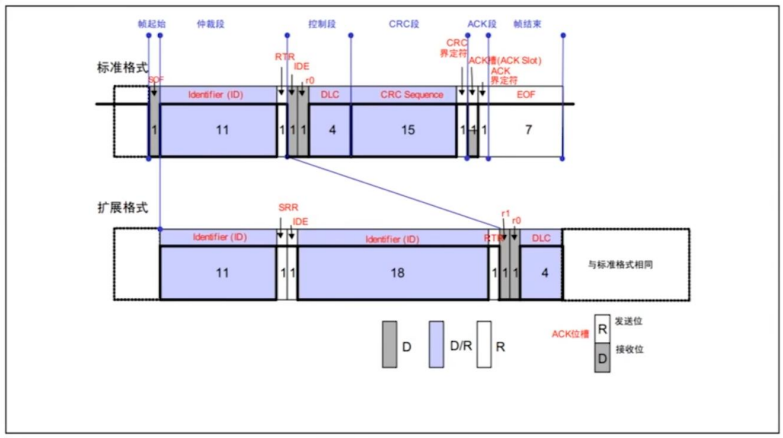



图 12. 遥控帧的构成

遥控帧通过广播设备接收到遥控信号也通过广播发送其所请求的数据。

遥控帧 id 为所请求的数据，响应请求的一方通过相同 ID 的数据帧反馈数据

## 2 错误帧

江协科技 jiangxiekeji.com

- 总线上所有设备都会监督总线的数据，一旦发现“位错误”或“填充错误”或“CRC错误”或“格式错误”或“应答错误”，这些设备便会发出错误帧来破坏数据，同时终止当前的发送设备

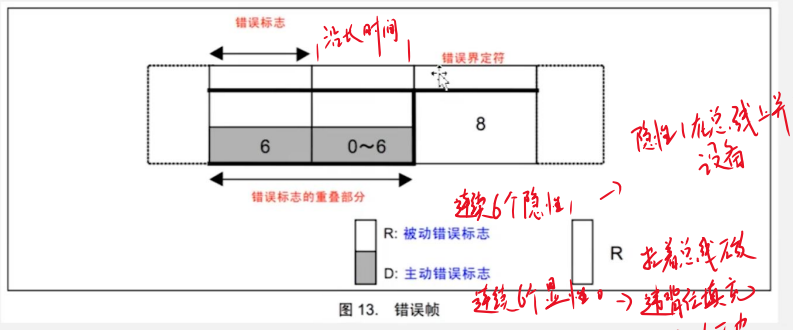



图 13. 错误帧

## 2 过载帧

江协科技 jiangxiekeji.com

- 当接收方收到大量数据而无法处理时，其可以发出过载帧，延缓发送方的数据发送，以平衡总线负载，避免数据丢失

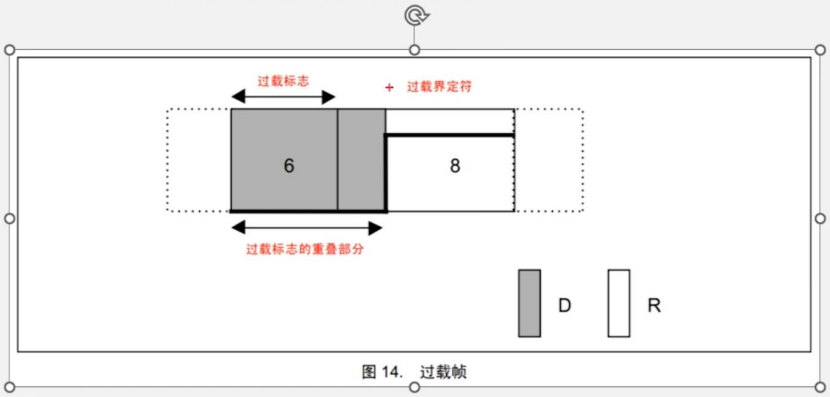



图 14. 过载帧

## 2 帧间隔

江协科技 jiangxiekeji.com

- 将数据帧和远程帧与前面的帧分离开

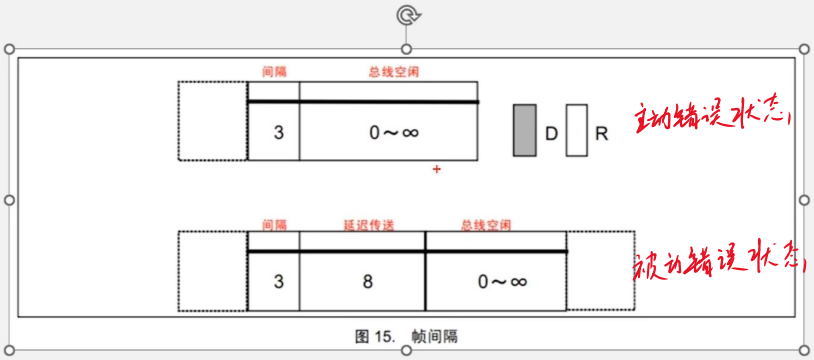



图 15. 帧间隔

## 2 位填充

江协科技 jiangxiekeji.com

- 位填充规则：发送方每发送5个相同电平后，自动追加一个相反电平的填充位，接收方检测到填充位时，会自动移除填充位，恢复原始数据
- 例如：

| | | | |
|---|---|---|---|
| 即将发送： | 100000110 | 10000011110 | 0111111111110 |
| 实际发送： | 100000**1**110 | 10000**1**1111**0**0 | 0111**1**10111**1**10 |
| 实际接收： | 100000**1**110 | 10000**1**1111**0**0 | 0111**1**10111**1**10 |
| 移除填充后： | 100000110 | 10000011110 | 0111111111110 |

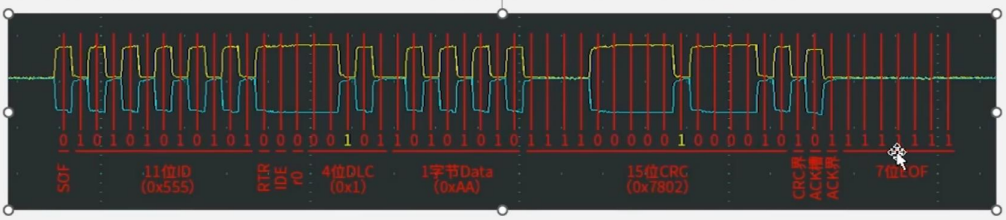
- 位填充作用：
  - 增加波形的定时信息，利于接收方执行“再同步”，防止波形长时间无变化，导致接收方不能精确掌握数据采样时机
  - 将正常数据流与“错误帧”和“过载帧”区分开，标志“错误帧”和“过载帧”的特异性
  - 保持CAN总线在发送正常数据流时的活跃状态，防止被误认为总线空闲

- **问题：** 如果原始数据中出现长串的连续 0 或连续 1（例如发送一个很大的数值），信号线会长期处于恒定电平。
- **后果：** 接收方的采样时钟会产生累积偏移（Drift），导致判断错误，分不清到底是发送了 5 个位还是 6 个位。
- **位填充的解决方法：** 协议规定，当信号线出现**连续 5 个相同电平**时，控制器会自动强制插入一个**相反电平**的位。这个额外的“跳变”强制接收方进行时钟重同步。

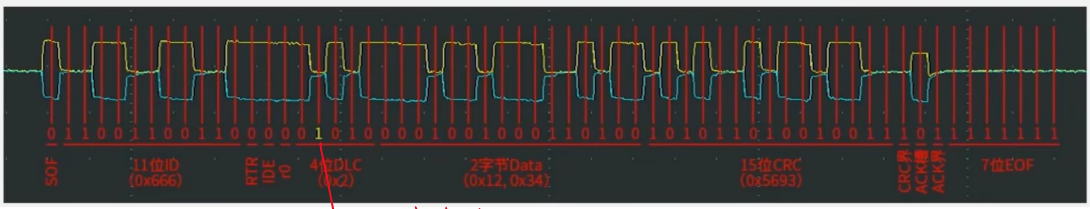
## 2 波形实例

江协科技 jiangxiekeji.com

- 标准数据帧，报文ID为0x555，数据长度1字节，数据内容为0xAA



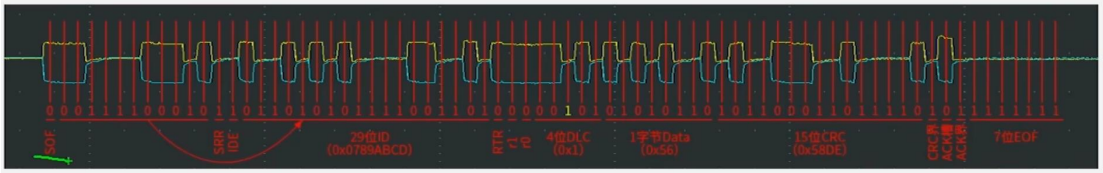

- 标准数据帧，报文ID为0x666，数据长度2字节，数据内容为0x12, 0x34



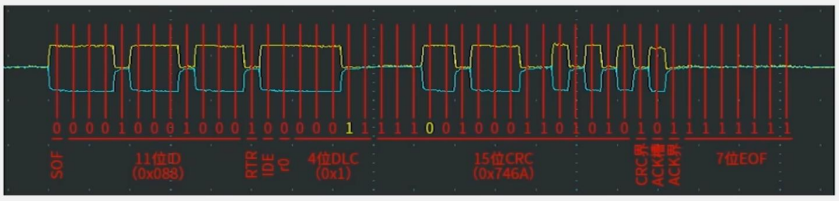
## 2 波形实例

江协科技 jiangxiekeji.com

- 扩展数据帧，报文ID为0x0789ABCD，数据长度1字节，数据内容为0x56




- 标准遥控帧，报文ID为0x088，数据长度1字节，无数据内容




**接收方:**

## 3 接收方数据采样

江协科技 jiangxiekeji.com

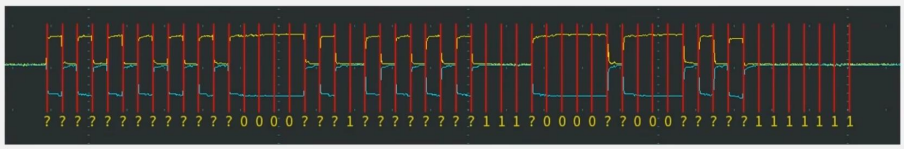
- CAN总线没有时钟线，总线上的所有设备通过约定波特率的方式确定每一个数据位的时长
- 发送方以约定的位时长每隔固定时间输出一个数据位
- 接收方以约定的位时长每隔固定时间采样总线的电平，输入一个数据位
- 理想状态下，接收方能依次采样到发送方发出的每个数据位，且采样点位于数据位中心附近



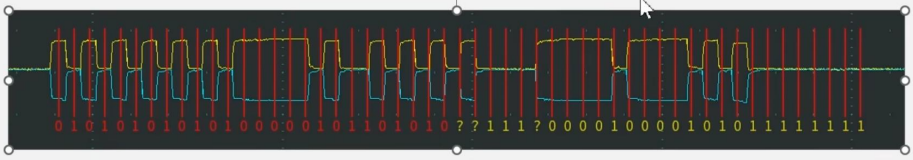
## 3 接收方数据采样遇到的问题

江协科技 jiangxiekeji.com

- 接收方以约定的位时长进行采样，但是采样点没有对齐数据位中心附近



- 接收方刚开始采样正确，但是时钟有误差，随着误差积累，采样点逐渐偏离



## 3 位时序

每一帧都能拆成 SS PTS PBS1 PBS2

Tq = Time Quantum

江协科技 jiangxiekeji.com

- 为了灵活调整每个采样点的位置，使采样点对齐数据位中心附近，CAN总线对每一个数据位的时长进行了更细的划分，分为同步段（SS）、传播时间段（PTS）、相位缓冲段1（PBS1）和相位缓冲段2（PBS2），每个段又由若干个最小时间单位（Tq）构成

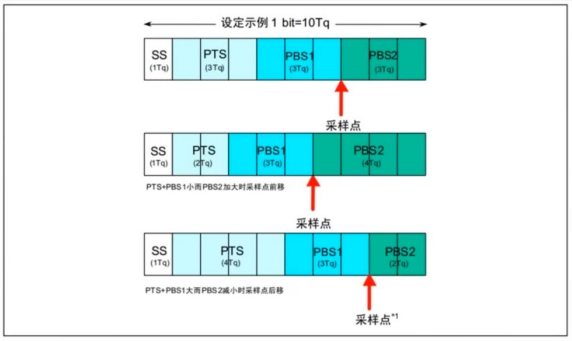



图 32. 1 个位的构成

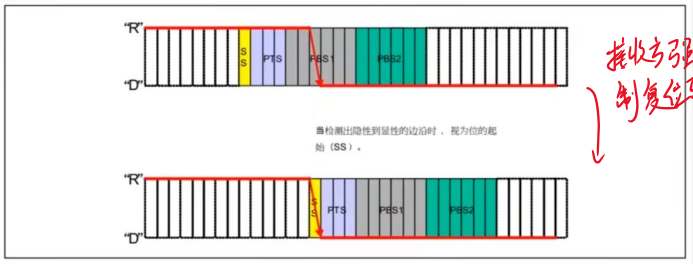
- SS = 1Tq
- PTS = 1~8Tq
- PBS1 = 1~8Tq
- PBS2 = 2~8Tq

采样点一般位于 PBS1 结束语 PBS2 开始交界处，一般汽车就是 87.5%

## 3 硬同步

江协科技 jiangxiekeji.com

- 每个设备都有一个位时序计时周期，当某个设备（发送方）率先发送报文，其他所有设备（接收方）收到SOF的下降沿时，接收方会将自己的位时序计时周期拨到SS段的位置，与发送方的位时序计时周期保持同步
- 硬同步只在帧的第一个下降沿（SOF下降沿）有效
- 经过硬同步后，若发送方和接收方的时钟没有误差，则后续所有数据位的采样点必然都会对齐数据位中心附近




接收方强制复位至SS

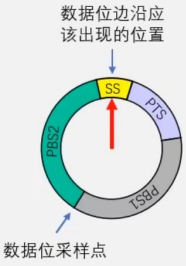



图 33. 硬件同步

发送方一般的开始的跳变沿就是在 ss 处开始的，接收方在 ss 检测到 sof 下降沿，立刻强制复位到 SS 同步段，对表完成。

## 3 再同步

- 若发送方或接收方的时钟有误差，随着误差积累，数据位边沿逐渐偏离SS段，则此时接收方根据再同步补偿宽度值（SJW）通过加长PBS1段，或缩短PBS2段，以调整同步
- 再同步可以发生在第一个下降沿之后的每个数据位跳变边沿

接收方快了
SJW加长PBS1处

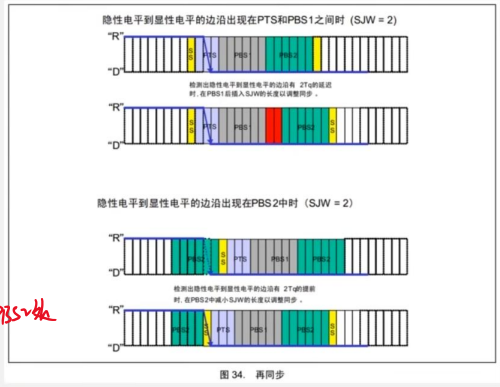


图 34. 再同步

接收方慢了
SJW减去前方相位缓冲段

➢ SJW=1~4Tq

这也解释了为什么要 5 个相同的信号要反向一次，因为长时间没有跳变，无法执行再同步，长时间会出错。

sjw 也不宜太高，容易把噪声当成信号，硬件会认为发生了巨大的相位偏移，并试图瞬间“跳”过去对齐。这会导致采样点发生剧烈抖动，原本在位中心的采样点可能直接跳到了位的边缘，导致误读数据。也很容易破坏“非破坏性仲裁”

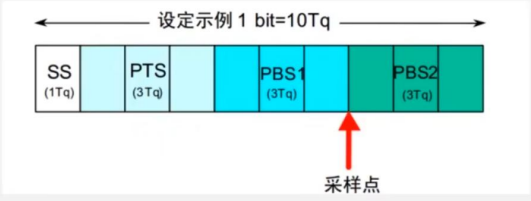
## 3 波特率计算

- 波特率 = 1 / 一个数据位的时长 = 1 / ($T_{SS}$ + $T_{PTS}$ + $T_{PBS1}$ + $T_{PBS2}$)
- 例如：

  SS = 1Tq，PTS = 3Tq，PBS1 = 3Tq，PBS2 = 3Tq

  Tq = 0.5us

  波特率 = 1 / (0.5us + 1.5us + 1.5us + 1.5us) = 200kbps



仲裁：

## 4 资源分配规则1 - 先占先得

江协科技 jiangxiekeji.com

- 若当前已经有设备正在操作总线发送数据帧/遥控帧，则其他任何设备不能再同时发送数据帧/遥控帧（可以发送错误帧/过载帧破坏当前数据）
- 任何设备检测到连续11个隐性电平，即认为总线空闲，只有在总线空闲时，设备才能发送数据帧/遥控帧
- 一旦有设备正在发送数据帧/遥控帧，总线就会变为活跃状态，必然不会出现连续11个隐性电平，其他设备自然也不会破坏当前发送
- 若总线活跃状态其他设备有发送需求，则需要等待总线变为空闲，才能执行发送需求

## 4 资源分配规则2 - 非破坏性仲裁

江协科技 jiangxiekeji.com

- 若多个设备的发送需求同时到来或因等待而同时到来，则CAN总线协议会根据ID号（仲裁段）进行非破坏性仲裁，ID号小的（优先级高）取到总线控制权，ID号大的（优先级低）仲裁失利后将转入接收状态，等待下一次总线空闲时再尝试发送
- 实现非破坏性仲裁需要两个要求：
  - 线与特性：总线上任何一个设备发送显性电平0时，总线就会呈现显性电平0状态，只有当所有设备都发送隐性电平1时，总线才呈现隐性电平1状态，即：0 & X & X = 0，1 & 1 & 1 = 1
  - 回读机制：每个设备发出一个数据位后，都会读回总线当前的电平状态，以确认自己发出的电平是否被真实地发送出去了，根据线与特性，发出0读回必然是0，发出1读回不一定是1

## 4 非破坏性仲裁过程

江协科技 jiangxiekeji.com

- 数据位从前到后依次比较，出现差异且数据位为1的设备仲裁失利

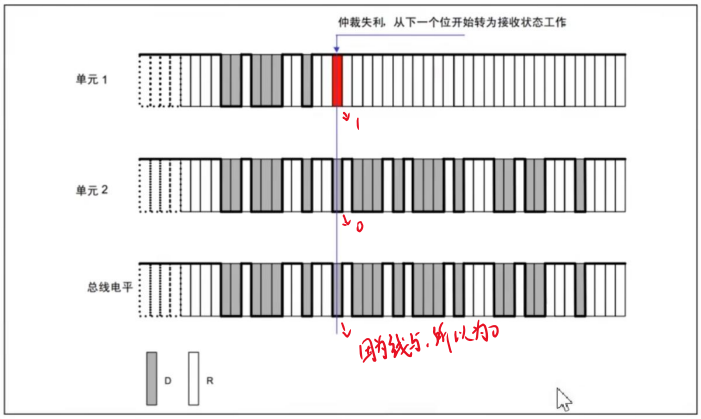



图 28. 仲裁过程

id 号越小 1 出现的也就越晚，所以优先级更高

## 4 数据帧和遥控帧的优先级

江协科技 jiangxiekeji.com

- 数据帧和遥控帧ID号一样时，数据帧的优先级高于遥控帧

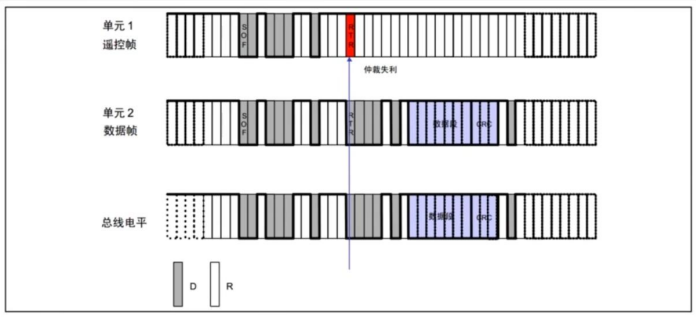



图 29. 数据帧和遥控帧的仲裁过程

## 4 标准格式和扩展格式的优先级

江协科技 jiangxiekeji.com

- 标准格式11位ID号和扩展格式29位ID号的高11位一样时，标准格式的优先级高于扩展格式（SRR必须始终为1，以保证此要求）

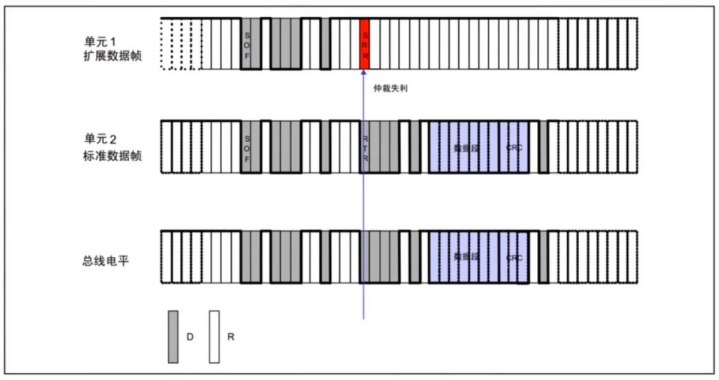



图 30. 标准格式与扩展格式的仲裁过程

## 错误处理

### 5 错误类型

江协科技 jiangxiekeji.com

- 错误共有5种： 位错误、填充错误、CRC错误、格式错误、应答错误

表 9. 错误的种类

| 错误的种类 | 错误的内容 | 错误的检测帧（段） | 检测单元 |
|---|---|---|---|
| 位错误 回读 | 比较输出电平和总线电平（不含填充位），当两电平不一样时所检测到的错误。 | • 数据帧（SOF~EOF）<br>• 遥控帧（SOF~EOF）<br>• 错误帧 元仲裁级<br>• 过载帧 | 发送单元<br>接收单元 |
| 填充错误 | 在需要位填充的段内，连续检测到 6 位相同的电平时所检测到的错误。 | • 数据帧（SOF~CRC 顺序）<br>• 遥控帧（SOF~CRC 顺序） | 发送单元<br>接收单元 |
| CRC 错误 | 从接收到的数据计算出的 CRC 结果与接收到的 CRC 顺序不同时所检测到的错误。 | • 数据帧（CRC 顺序）<br>• 遥控帧（CRC 顺序） | 接收单元 |
| 格式错误 | 检测出与固定格式的位段相反的格式时所检测到的错误。 | • 数据帧（CRC 界定符、ACK 界定符、EOF）<br>• 遥控帧（CRC 界定符、ACK 界定符、EOF）<br>• 错误界定符<br>• 过载界定符 | 接收单元 |
| ACK 错误 | 发送单元在 ACK 槽(ACK Slot)中检测出隐性电平时所检测到的错误（ACK 没被传送过来时所检测到的错误）。 | • 数据帧（ACK 槽）<br>• 遥控帧（ACK 槽） | 发送单元 |

为了防止有一个设备损坏，不管数据好坏统统破坏掉，于是有下面的机制

### 5 错误状态 （有一部分在前面的帧格式）

江协科技 jiangxiekeji.com

- 主动错误状态的设备正常参与通信并在检测到错误时发出主动错误帧
- 被动错误状态的设备正常参与通信但检测到错误时只能发出被动错误帧
- 总线关闭状态的设备不能参与通信
- 每个设备内部管理一个TEC和REC，根据TEC和REC的值确定自己的状态

主动错误状态，可以破坏别的设备报文
被动错误只影响自己
总线关闭：直接将频繁报错的设备闭嘴

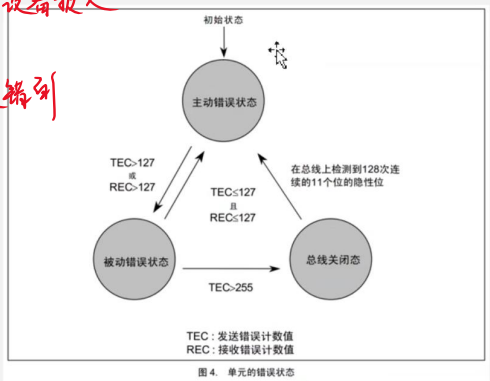


图 4. 单元的错误状态

TEC：发送时每发现一个错误就增加一次，进行一次正常的就减小一次

REC：接收时每发现一个错误就增加一次，进行一次正常的就减小一次

## 5 错误计数器

江协科技 jiangxiekeji.com

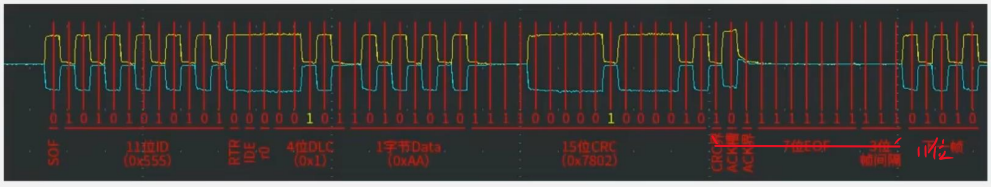
表 2. 错误计数值的变动条件

| | 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。<br>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。<br>之后，每检测出连续的 8 个位的显性位时。 | 发送时<br>+8 | 接收时<br>+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<br>+8 | 接收时<br>+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<br>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<br>REC=0 时±0<br>REC>127 时<br>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 |

## 5 波形示例

江协科技 jiangxiekeji.com

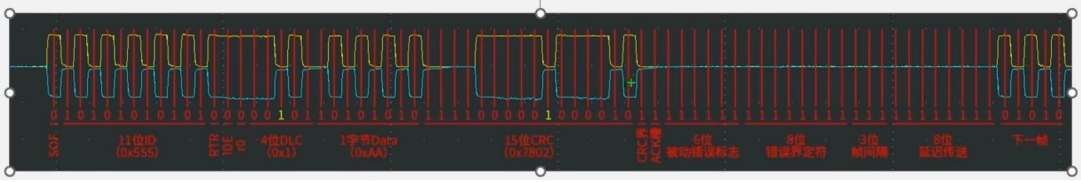
- 设备处于主动错误状态，发送标准数据帧，正常传输




- 设备处于主动错误状态，发送标准数据帧，检测到ACK错误

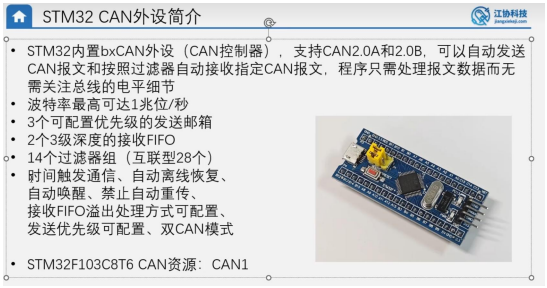


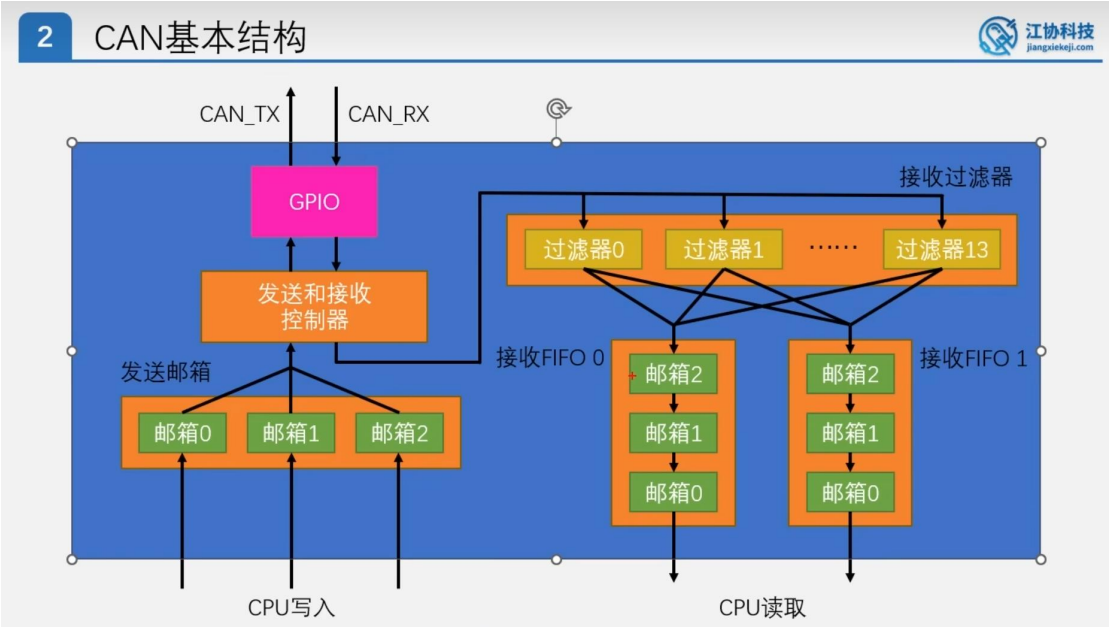

- 设备处于被动错误状态，发送标准数据帧，检测到ACK错误




这个 8 位的延迟传送就是让他降低优先级

**STM32 CAN 外设：**


STM32 CAN外设简介
江协科技
• STM32内置bxCAN外设（CAN控制器），支持CAN2.0A和2.0B，可以自动发送CAN报文和按照过滤器自动接收指定CAN报文，程序只需处理报文数据而无需关注总线的电平细节
• 波特率最高可达1兆位/秒
• 3个可配置优先级的发送邮箱
• 2个3级深度的接收FIFO
• 14个过滤器组（互联型28个）
• 时间触发通信、自动离线恢复、自动唤醒、禁止自动重传、接收FIFO溢出处理方式可配置、发送优先级可配置、双CAN模式
• STM32F103C8T6 CAN资源：CAN1

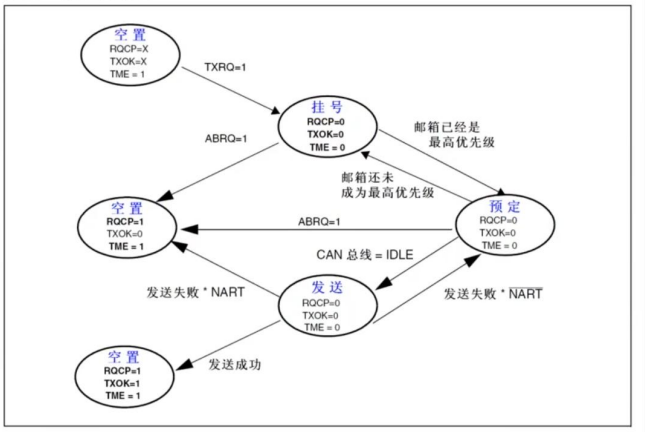

2
CAN基本结构
江协科技
jiangxiekeji.com
CAN_TX
CAN_RX
GPIO
发送和接收控制器
发送邮箱
邮箱0
邮箱1
邮箱2
接收过滤器
过滤器0
过滤器1
……
过滤器13
接收FIFO 0
邮箱2
邮箱1
邮箱0
接收FIFO 1
邮箱2
邮箱1
邮箱0
CPU写入
CPU读取

## 2 发送过程

江协科技 jiangxiekeji.com

- 基本流程：选择一个空置邮箱→写入报文 →请求发送

图200 发送邮箱状态




RQCP：请求完成　x 表示任意值　　TXRQ：发送请求控制位　=1 表示产生发送请求

TXOK：发送成功　x 表示任意值　　NART: 禁止自动重传　=0 表示使用自动重传反之禁止 g

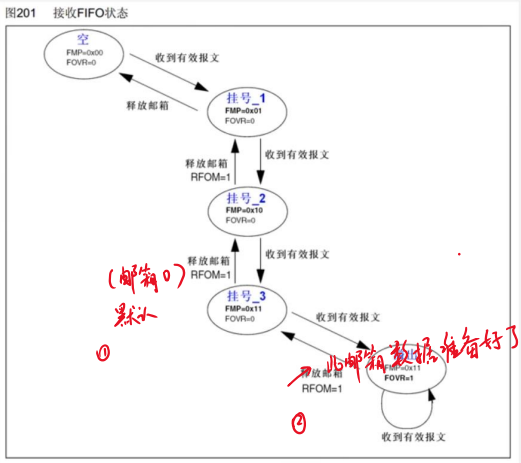
TME:　发送邮箱　=1 代表空　　ABRQ：终止发送

## 2 接收过程

江协科技 jiangxiekeji.com

- 基本流程：接收到一个报文→匹配过滤器后进入FIFO 0或FIFO 1→CPU读取

图201 接收FIFO状态

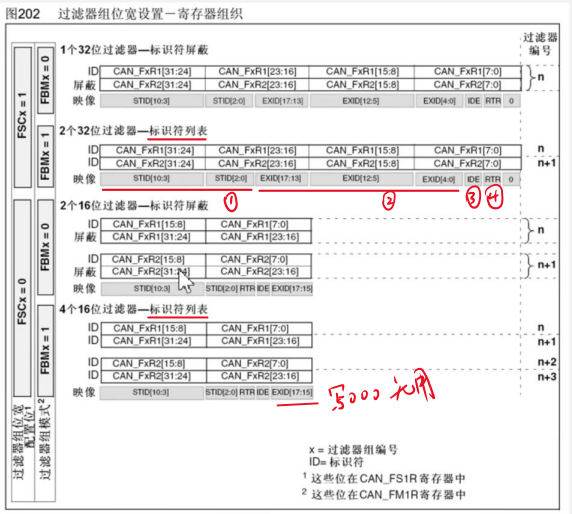



FMP:报文树木 0x00 意思是为 0　　FOVR:FIFO 溢出　=0 为无溢出

## 2 标识符过滤器

- 每个过滤器的核心由两个32位寄存器组成：R1[31:0]和R2[31:0]
- FSCx：位宽设置<br>置0，16位；置1，32位
- FBMx：模式设置<br>置0，屏蔽模式；置1，列表模式
- FFAx：关联设置<br>置0，FIFO 0；置1，FIFO 1
- FACTx：激活设置<br>置0，禁用，置1，启用

0-31有32位 图中问理




一共有 13 个过滤器，过滤器是几 x 就是几，如 FBM11

列表模式

列表模式 就是直接把 id 写在过滤器里，在列表中的 id 就过去，不在就过不去（32 位只能存 2 个）

① 高 11 位存入标准格式 ID 号 STID

② 18 位 扩展格式的 ID 号 EXID

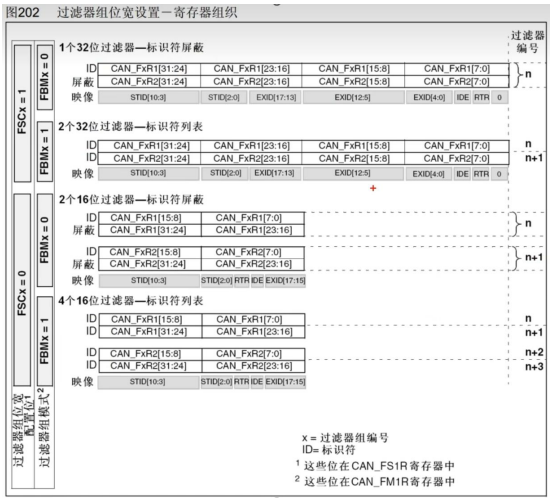
③ 要是想要过滤扩展 ID，IDE 就写 1 反之 0

④ 过滤遥控帧 RTR 写 1 数据帧写 0

有时候用不到扩展的也就是浪费，就可以用 16 位的，可以同时在

| 场景 | R1 的 RTR 位 | R2 的 RTR 位 | 结果 |
|---|---|---|---|
| 只要数据 | 0 | 1 | 只有 ID 匹配且是数的包能进来。 |
| 只要遥控 | 1 | 1 | 只有 ID 匹配且是遥的包能进来。 |
| 全都要 (最常用) | 无所谓 (通常给0) | 0 | 只要 ID 匹配，数据遥控帧通杀。 |

5-15 4 3 0-2




**R1 (Identifier Register)：设定目标值**

你在 R1 的 RTR 位写什么，就表示你**期望**收到什么样的帧。

- 你写 0，说明你倾向于接收“数据帧”。
- 你写 1，说明你倾向于接收“遥控帧”。

**R2 (Mask Register)：设定严格程度**

R2 里的 RTR 位（屏蔽位）决定了硬件是否真的要去查 R1 里的那个设定。

- **R2 的 RTR 位写 1 (Must Match)**：“死命令”。进来的帧，其 RTR 位必须跟 R1 一模一样。
- **R2 的 RTR 位写 0 (Don't Care)**：“无所谓”。管你是送货的还是催单的，只要 ID 对得上，通通放行。

R1 寄存器写入 ID 号，R2 写入屏蔽位（掩码）

假设过滤出 0x1 开头所有 ID 号,11 位的 2 进制表示就是 001 xxxx xxxx,R1 的 ID 就是 001 0000 0000 EXID 全是 0 IDE 位 0 RTR 位 0

R2 写入（如果 R2 这个位置写 1，那么这个位的 ID 必须与 R1 的一致，写 0 表示 R1 里的 1 与 0 均可）111 0000 0000， EXID 0

IDE 必须为 1。对于 RTR,R1 为 0,R2 为 1，表示必须匹配数据帧，若必须匹配遥控帧，则都为 1，要是都想要就是上面无所谓，下面给 0

于是过滤出来 001 xxxx xxxx 的 ID

## 2 过滤器配置示例

江协科技 jiangxiekeji.com

| 总线上存在的ID | 想要接收的ID | 过滤器模式 | R1[31:0]配置值 | R2[31:0]配置值 |
|---|---|---|---|---|
| 0x123, 0x234, 0x345, 0x456, 0x567, 0x678 | 0x234, 0x345, 0x567 | 16位/列表 | ID: R1[15:0]=0x234<<5<br>ID: R1[31:16]=0x345<<5 | ID: R2[15:0]=0x567<<5<br>ID: R2[31:16]=0x000<<5 |
| 0x100~0x1FF, 0x200~0x2FF, 0x310~0x31F, 0x320~0x32F | 0x200~0x2FF, 0x320~0x32F | 16位/屏蔽 | ID: R1[15:0]=0x200<<5<br>Mask: R1[31:16]=<br>(0x700<<5)\|0x10\|0x8 | ID: R2[15:0]=0x320<<5<br>Mask: R2[31:16]=<br>(0x7F0<<5)\|0x10\|0x8 |
| 0x123, 0x234, 0x345, 0x456, 0x12345678, 0x0789ABCD | 0x123, 0x12345678 | 32位/列表 | ID: R1[31:0]=<br>0x123<<21 | ID: R2[31:0]=<br>(0x12345678<<3)\|0x4 |
| 0x12345600~0x123456FF, 0x0789AB00~0x0789ABFF | 0x12345600~0x123456FF | 32位/屏蔽 | ID: R1[31:0]=<br>(0x12345600<<3)\|0x4 | Mask: R2[31:0]=<br>(0x1FFFF00<<3)\|0x4\|0x2 |
| 任意ID | 只要遥控帧 | 32位/屏蔽 | ID: R1[31:0]=0x2 | Mask: R2[31:0]=0x2 |
| 任意ID | 所有帧 | 32位/屏蔽 | ID: R1[31:0]=随意 | Mask: R2[31:0]=0 |

**第一步：解开“左移 5 位（`<< 5`）”的千古之谜**

为什么不能直接把 ID 放进去？**因为 STM32 的 16 位过滤器寄存器，不是全拿来装 ID 的。**

你可以把这个 16 位的寄存器想象成一个有特定格子的收纳盒：

- **第 15 位 ~ 第 5 位（共 11 位）：** 存放**标准 ID (StdId)**。
- **第 4 位：** RTR 位（遥控帧标志，你上个问题刚问过）。
- **第 3 位：** IDE 位（扩展帧标志）。
- **第 2 位 ~ 第 0 位：** 扩展 ID 的一部分（标准帧里不用管）。

**结论：** 你的 11 位标准 ID 必须放在这个寄存器的**高 11 位**。如果你直接写 `0x234`，它会落在寄存器的低位，跟 RTR、IDE 混在一起。所以，**必须把你的 ID 整体向左推 5 个格子**，也就是 `<< 5`，让它对齐到 [15:5] 的位置。

**第二步：看懂第一行 —— 16位列表模式 (List Mode)**

**模式逻辑：** 类似于“VIP 白名单”。只要有空位，把你想放行的 ID 挨个填进去就行。绝对匹配，差一位都不行。

在 16 位模式下，STM32 会把原本的两个 32 位寄存器（R1 和 R2）**从中间劈开**，变成 4 个 16 位的“槽位”：

- `R1[15:0]`：R1 的低 16 位（槽位 1）
- `R1[31:16]`：R1 的高 16 位（槽位 2）
- `R2[15:0]`：R2 的低 16 位（槽位 3）
- `R2[31:16]`：R2 的高 16 位（槽位 4）

**对应图中的配置：**

你要接收 `0x234`，`0x345`，`0x567`。直接塞进前三个槽位，别忘了左移 5 位对齐：

- 槽位 1：`R1[15:0] = 0x234 << 5`
- 槽位 2：`R1[31:16] = 0x345 << 5`
- 槽位 3：`R2[15:0] = 0x567 << 5`
- 槽位 4：因为没有第四个 ID 要收了，所以填 `0x000 << 5` 占位。

*（第一行其实非常简单粗暴，难的是第二行。）*

**第三步：看懂第二行 —— 16位屏蔽模式 (Mask Mode)**

**模式逻辑：** 类似于“保安查号段”。保安手里拿两张纸：

1. **基准 ID：** 规定号段的基础样子。
2. **屏蔽码 (Mask)：** 告诉保安，“哪几位必须严查（写1），哪几位无所谓（写0）”。

在 16 位屏蔽模式下，原本劈开的槽位有了分工：

- `R1[15:0]` 放**基准 ID**，`R1[31:16]` 放**屏蔽码**。
- `R2[15:0]` 放**基准 ID**，`R2[31:16]` 放**屏蔽码**。

**核心难点：怎么算出 `0x700` 和 `0x7F0` 的?**

我们要把 16 进制转成 2 进制，找“共同点”。

**例子 A：要接收 `0x200 ~ 0x2FF`**

1. 把范围两端转成 11 位的二进制（这是最关键的一步）：
   - `0x200` = `010 0000 0000`
   - `0x2FF` = `010 1111 1111`
2. 找规律：
   - 前 3 位（`010`）是**固定不变**的。
   - 后 8 位（`0000 0000` 到 `1111 1111`）是**随便变**的。
3. 生成屏蔽码 (Mask)：
   - 不变的位写 1（必须查），随便变的位写 0（不用查）。
   - Mask = `111 0000 0000`
4. 转回 16 进制：`111 0000 0000` 换算成 16 进制正好是 `0x700`！
5. **配置结论：** 基准 ID 设为 `0x200`，Mask 设为 `0x700`。（别忘了都要 `<< 5`）。这就对应了图中 R1 的配置。

**例子 B：要接收 `0x320 ~ 0x32F`**

1. 转二进制：
   - `0x320` = `011 0010 0000`
   - `0x32F` = `011 0010 1111`
2. 找规律：
   - 前 7 位 ( `011 0010` ) 是**固定不变**的。
   - 后 4 位 是**随便变**的。
3. 生成屏蔽码 (Mask)：
   - Mask = `111 1111 0000`
4. 转回 16 进制：二进制 `111 1111 0000` 换算出来就是 `0x7F0` ！
5. **配置结论：** 基准 ID 设为 `0x320`，Mask 设为 `0x7F0` 。这就对应了图中 R2 的配置。

**第四步：那个诡异的 `|0x10|0x8` 是什么鬼?**

看图里，它的 Mask 配置写的是 `(0x700<<5) | 0x10 | 0x8` 。

回到我们第一步说的收纳盒：

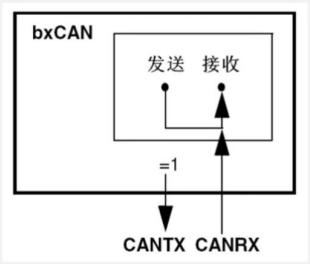
- **第 4 位是 RTR**。十六进制 `0x10` 的二进制是 `0001 0000` ，正好第 4 位是 1。
- **第 3 位是 IDE**。十六进制 `0x08` 的二进制是 `0000 1000` ，正好第 3 位是 1。

这里使用**按位或（`|`）**，意思是：
除了要求 ID 的前几位匹配之外，我还强硬要求：**进来的这帧数据，它的 RTR 位和 IDE 位，必须跟我的基准 ID 一模一样！**
（因为基准 ID 没有设这两个位，默认为 0，所以它的潜台词是：“我只收标准帧、数据帧，哪怕 ID 在范围内，遥控帧和扩展帧我也不要！”）

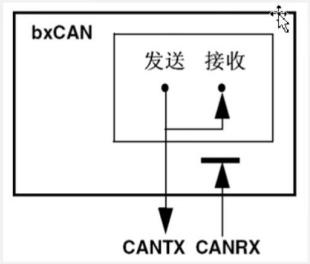
## 3 测试模式

江协科技 jiangxiekeji.com

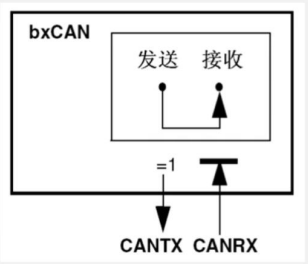
- 静默模式：用于分析CAN总线的活动，不会对总线造成影响
- 环回模式：用于自测试，同时发送的报文可以在CAN_TX引脚上检测到
- 环回静默模式：用于热自测试，自测的同时不会影响CAN总线




静默模式




环回模式



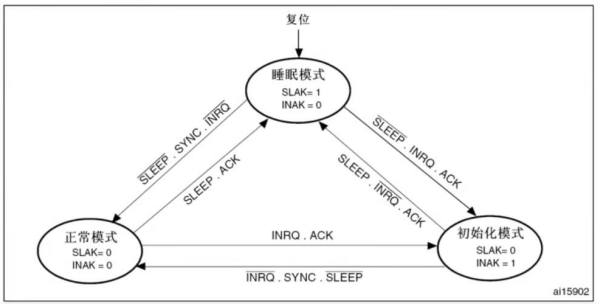

环回静默模式

## 3 工作模式

江协科技 jiangxiekeji.com

- 初始化模式：用于配置CAN外设，禁止报文的接收和发送
- 正常模式：配置CAN外设后进入正常模式，以便正常接收和发送报文
- 睡眠模式：低功耗，CAN外设时钟停止，可使用软件唤醒或者硬件自动唤醒
- AWUM：置1，自动唤醒，一旦检测到CAN总线活动，硬件就自动清零SLEEP，唤醒CAN外设；置0，手动唤醒，软件清零SLEEP，唤醒CAN外设

图196 bxCAN工作模式




SLAK:睡眠确认状态位 =1 进入睡眠

INAK：初始化确认位 =0 没有进入初始化模式

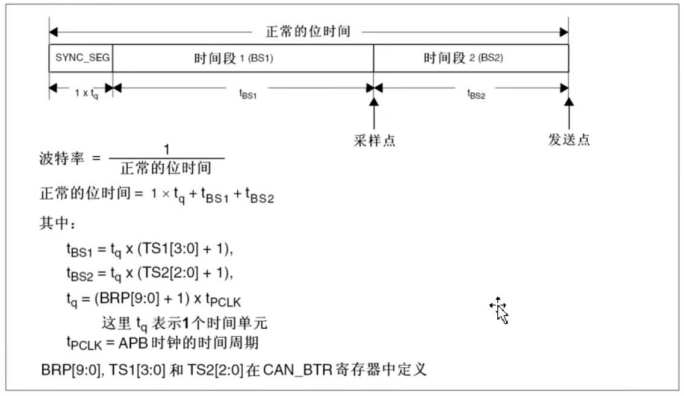
SLEEP:=1 请求进入睡眠 =0 请求退出睡眠（头上一横线表示 0）

SYNC 总线空闲

## 3 位时间特性

江协科技 jiangxiekeji.com

图206 位时序




$$波特率 = \frac{1}{正常的位时间}$$

$$正常的位时间 = 1 \times t_q + t_{BS1} + t_{BS2}$$

其中：

$t_{BS1} = t_q \times (TS1[3:0] + 1)$,

$t_{BS2} = t_q \times (TS2[2:0] + 1)$,

$t_q = (BRP[9:0] + 1) \times t_{PCLK}$

这里 $t_q$ 表示1个时间单元

$t_{PCLK}$ = APB 时钟的时间周期

BRP[9:0], TS1[3:0] 和 TS2[2:0]在CAN_BTR寄存器中定义

- SS = 1Tq
- BS1 = 1~16Tq
- BS2 = 1~8Tq
- SJW=1~4Tq

BS1 = PTS1 + PBS1

- 波特率 = APB1时钟频率 / 分频系数 / 一位的Tq数量
  = 36MHz / (BRP[9:0]+1) / (1 + (TS1[3:0]+1) + (TS2[2:0]+1))

1M = 1000K

= 时钟频率 / 分频系数 / (1 + BS1的Tq + BS2的Tq)

## 3 中断

江协科技
jiangxiekeji.com

- CAN外设占用4个专用的中断向量
- 发送中断：发送邮箱空时产生
- FIFO 0中断：收到一个报文/FIFO 0满/FIFO 0溢出时产生
- FIFO 1中断：收到一个报文/FIFO 1满/FIFO 1溢出时产生
- 状态改变错误中断：出错/唤醒/进入睡眠时产生

图208 事件标志和中断产生

