2018年11月4日 22:58

CAN通信

2018年11月4日
10:21

——简介：

CAN总线也称控制器局域网，属于现场总线的范畴

它是一种有效支持分布控制或实时控制的串行通信网络

——CAN总线的特点：

1、CAN为多主工作方式

网络上任何一个节点均可以在任意时刻主动地向网络上其他节点发送信息，而不分主从

2、消息的发送

在 CAN 协议中，所有的消息都以固定的格式发送

总线空闲时，所有与总线相连的单元都可以开始发送新消息

两个以上的单元同时开始发送消息时，根据标识符（Identifier 以下称为 ID）决定优先级

ID 并不是表示发送的目的地址，而是表示访问总线的消息的优先级

两个以上的单元同时开始发送消息时，对各消息 ID 的每个位进行逐个仲裁比较

仲裁获胜（被判定为优先级最高）的单元可继续发送消息，仲裁失利的单元则立刻停止发送而进行接收工作

3、CAN采用非破坏总线仲裁技术

当多个节点同时向总线发送信息出现冲突时，优先级低地节点会主动退出发送

而优先级高地节点可以不受影响地继续传输数据，从而大大节省了总线冲突地仲裁时间

尤其在网络负载很重地情况下，也不会出现网络瘫痪情况（以太网则可能）

4、通信速度

根据整个网络的规模，可设定适合的通信速度

在同一网络中，所有单元必须设定成统一的通信速度

即使有一个单元的通信速度与其它的不一样，此单元也会输出错误信号，妨碍整个网络的通信

5、错误检测功能·错误通知功能·错误恢复功能

所有的单元都可以检测错误（错误检测功能）

检测出错误的单元会立即同时通知其他所有单元（错误通知功能）

正在发送消息的单元一旦检测出错误，会强制结束当前的发送

强制结束发送的单元会不断反复地重新发送此消息直到成功发送为止（错误恢复功能）

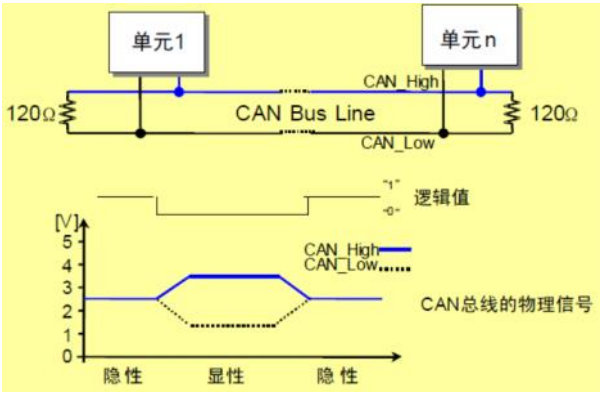
6、故障封闭

CAN 可以判断出错误的类型是总线上暂时的数据错误（如外部噪声等）还是持续的数据错误（如单元内部故障、驱动器故障、断线等）

由此功能，当总线上发生持续数据错误时，可将引起此故障的单元从总线上隔离出去

——CAN物理层特性：

图示：



由图可知：

显性电平对应逻辑 0， CAN_H 和 CAN_L 之差为 2.5V 左右

而隐性电平对应逻辑 1， CAN_H 和 CAN_L 之差为 0V

在总线上显性电平具有优先权，只要有一个单元输出显性电平，总线上即为显性电平

而隐形电平则具有包容的意味，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）

另外，在 CAN 总线的起止端都有一个 120Ω的终端电阻，来做阻抗匹配，以减少回波反射

——CAN 协议是通过以下 5 种类型的帧进行的：

帧类型	帧用途

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
间隔帧	用于将数据帧及遥控帧与前面的帧分离开来的帧

注：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

——数据帧：

（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。

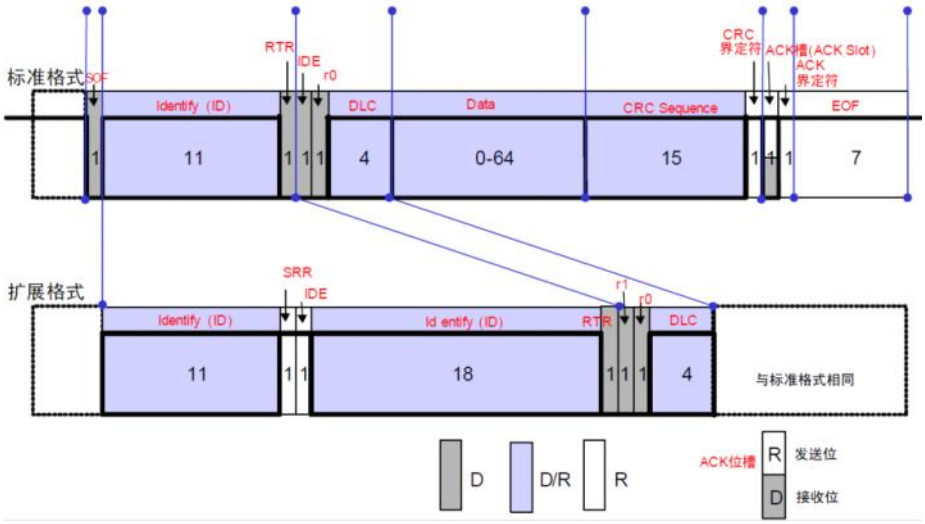
（4） 数据段：数据的内容，一帧可发送 0~8 个字节的数据。

（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。

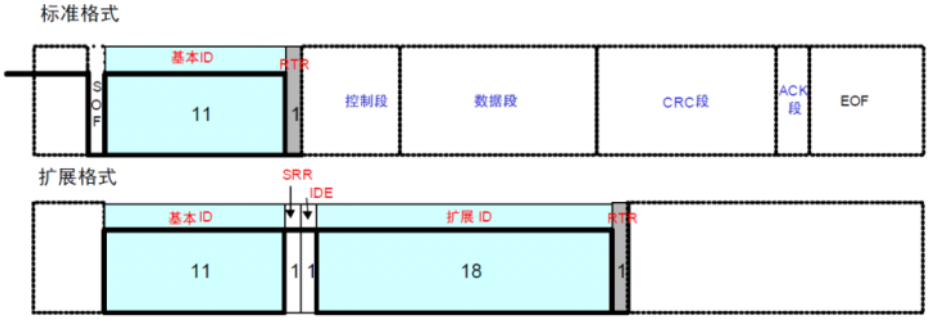
（7） 帧结束：表示数据帧结束的段

图示：（图中 D 表示显性电平， R 表示隐形电平 ）



帧起始：标准帧和扩展帧都是由 1 个位的显性电平表示帧起始

仲裁段：表示数据优先级的段，标准帧和扩展帧格式在本段有所区别



标准格式的 ID 有 11 个位，从 ID28 到 ID18 被依次发送

禁止高 7 位都为隐性（禁止设定： ID=1111111XXXX）

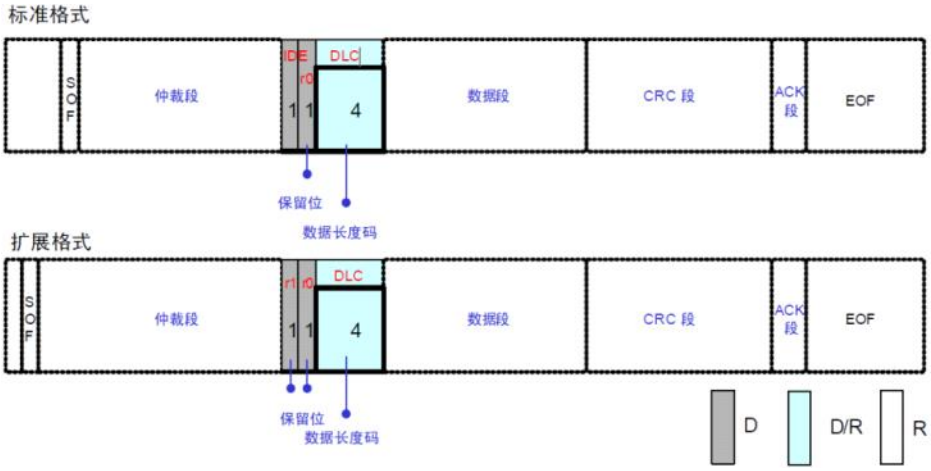
扩展格式的 ID 有 29 个位，基本 ID 从 ID28 到 ID18，扩展 ID 由ID17 到 ID0 表示，基本 ID 和标准格式的 ID 相同，禁止高 7 位都为隐性

其中 RTR 位用于标识是否是远程帧（0，数据帧； 1，远程帧）

IDE 位为标识符选择位（0，使用标准标识符； 1，使用扩展标识符）

SRR 位为代替远程请求位，为隐性位，它代替了标准帧中的 RTR 位

控制段：由 6 个位构成，表示数据段的字节数，标准帧和扩展帧的控制段稍有不同



上图中，r0 和 r1 为保留位，必须全部以显性电平发送，但是接收端可以接收显性、隐性及任意组合的电平

DLC 段为数据长度表示段，高位在前， DLC 段有效值为 0~8，但是接收方接收到 9~15 的时候并不认为是错误

数据段：该段可包含 0~8 个字节的数据，从最高位（MSB）开始输出，标准帧和扩展帧在这个段的定义都是一样的

CRC段：该段用于检查帧传输错误，由 15 个位的 CRC 顺序和 1 个位的 CRC 界定符（用于分隔的位）组成，标准帧和扩展帧在这个段的格式也是相同的

此段 CRC 的值计算范围包括：帧起始、仲裁段、控制段、数据段

接收方以同样的算法计算 CRC 值并进行比较，不一致时会通报错误

ACK段：此段用来确认是否正常接收，由 ACK 槽(ACK Slot)和 ACK 界定符 2 个位组成，标准帧和扩展帧在这个段的格式也是相同的

发送单元的 ACK，发送 2 个位的隐性位

而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，这个过程叫发送 ACK/返回 ACK

帧结束：这个段也比较简单，标准帧和扩展帧在这个段格式一样， 由 7 个位的隐性位组成

——位时序：（1 位分为 4 个段，每个段又由若干个 Tq 构成，这称为位时序 ）

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（以下称为 Tq）的最小时间单位构成

1 位由多少个 Tq 构成、每个段又由多少个 Tq 构成等，可以任意设定位时序

通过设定位时序，多个单元可同时采样，也可任意设定采样点

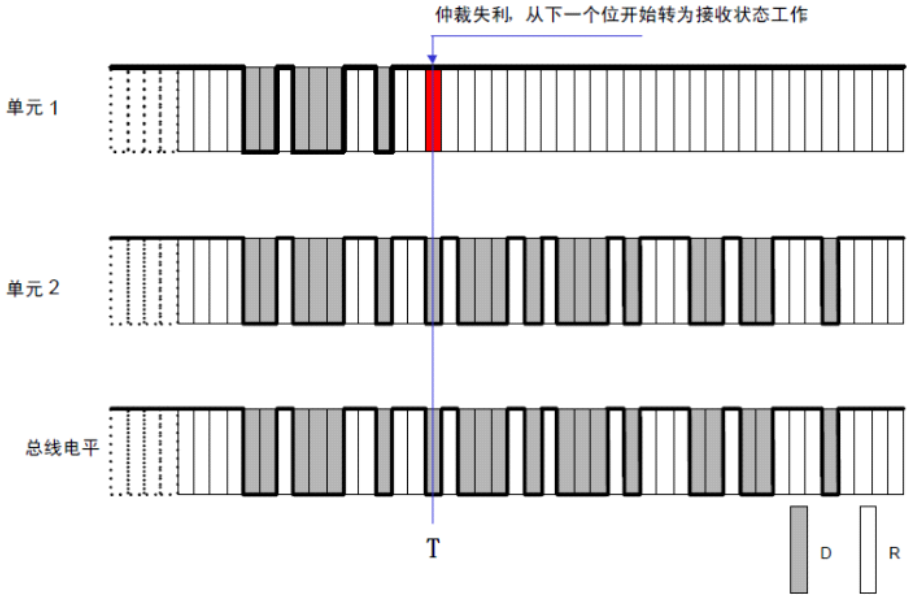
采样点：是指读取总线电平，并将读到的电平作为位值的点（位置在 PBS1 结束处）

段名称	段的作用	Tq 数	
同步段 （SS: Synchronization Segment）	多个连接在总线上的单元通过此段实现时序调整，同步进行接收和发送的工作。由隐性电平到显性电平的边沿或由显性电平到隐性电平边沿最好出现在此段中。	1Tq	8～25Tq
传播时间段 （PTS: Propagation Time Segment）	用于吸收网络上的物理延迟的段。 所谓的网络的物理延迟指发送单元的输出延迟、总线上信号的传播延迟、接收单元的输入延迟。 这个段的时间为以上各延迟时间的和的两倍。	1～8Tq	
相位缓冲段 1 (PBS1: Phase Buffer Segment 1)	当信号边沿不能被包含于 SS 段中时，可在此段进行补偿。	1～8Tq	
相位缓冲段 2 (PBS2: Phase Buffer Segment 2)	由于各单元以各自独立的时钟工作，细微的时钟误差会累积起来，PBS 段可用于吸收此误差。 通过对相位缓冲段加减 SJW 吸收误差。 SJW 加大后允许误差加大，但通信速度下降。	2～8Tq	
再同步补偿宽度 (SJW: reSynchronization Jump Width)	因时钟频率偏差、传送延迟等，各单元有同步误差。SJW 为补偿此误差的最大值。	1～4Tq	

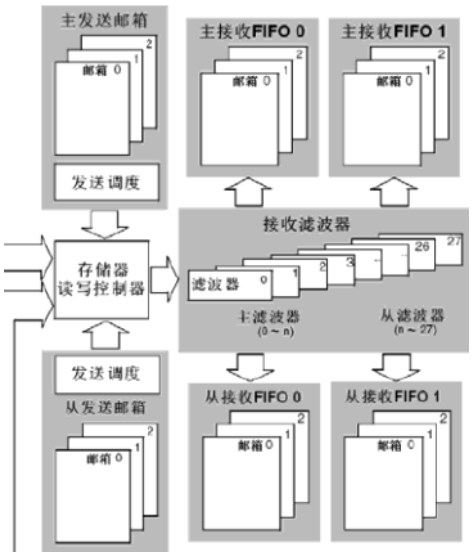
——CAN协议的仲裁功能：

1、在总线空闲态，最先开始发送消息的单元获得发送权

2、当多个单元同时开始发送时，各发送单元从仲裁段的第一位开始进行仲裁。连续输出显性电平最多的单元可继续发送



——CAN控制器：



如图：两个 CAN 都分别拥有自己的发送邮箱和接收 FIFO，但是他们共用 28 个滤波器

通过 CAN_FMR 寄存器的设置，可以设置滤波器的分配方式

STM32 每个过滤器组的位宽都可以独立配置，以满足应用程序的不同需求

根据位宽的不同，每个过滤器组可提供：

- 1 个 32 位过滤器，包括： STDID[10:0]、 EXTID[17:0]、 IDE 和 RTR 位
- 2 个 16 位过滤器，包括： STDID[10:0]、 IDE、 RTR 和 EXTID[17:15]位

此外过滤器可配置为，屏蔽位模式和标识符列表模式：

在屏蔽位模式下，标识符寄存器和屏蔽寄存器一起，指定报文标识符的任何一位，应该按照“必须匹配”或“不用关心”处理

在标识符列表模式下，屏蔽寄存器也被当作标识符寄存器用

因此，不是采用一个标识符加一个屏蔽位的方式，而是使用 2 个标识符寄存器

接收报文标识符的每一位都必须跟过滤器标识符相同

为了过滤出一组标识符，应该设置过滤器组工作在屏蔽位模式。

为了过滤出一个标识符，应该设置过滤器组工作在标识符列表模式

图示：

举例：

我们设置过滤器组 0 工作在： 1 个 32 为位过滤器-标识符屏蔽模式

然后设置 CAN_F0R1=0XFFFF0000， CAN_F0R2=0XFF00FF00

其中存放到 CAN_F0R1 的值就是期望收到的 ID，即我们希望收到的映像（STID+EXTID+IDE+RTR）最好是： 0XFFFF0000

而0XFF00FF00 就是设置我们需要必须关心的 ID，表示收到的映像

其位[31:24]和位[15:8]这 16个位的必须和 CAN_F0R1 中对应的位一模一样，而另外的 16 个位则不关心，一样/不一样，都认为是正确的 ID

即收到的映像必须是 0XFFxx00xx，才算是正确的（x 表示不关心）

——CAN发送接收流程：

发送：

程序选择 1 个空置的邮箱（TME=1）

→ 设置标识符（ID），数据长度和发送数据

→ 设置 CAN_TIxR 的 TXRQ 位为 1，请求发送

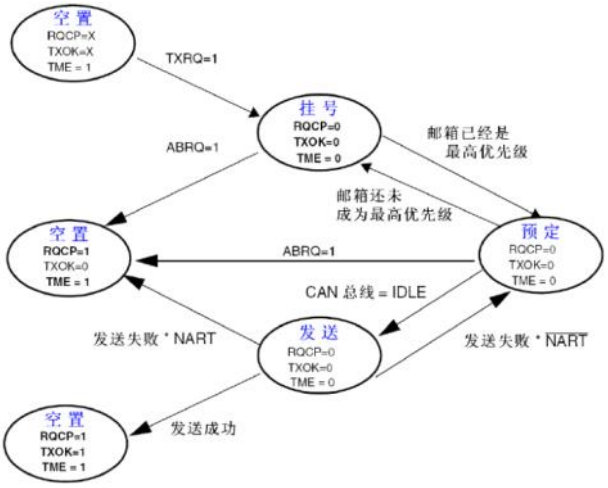
→ 邮箱挂号（等待成为最高优先级）

→预定发送（等待总线空闲）

→发送

→邮箱空置

图示：



接收：

CAN 接收到的有效报文，被存储在 3 级邮箱深度的 FIFO 中

FIFO 空

→收到有效报文

→挂号_1（存入 FIFO 的一个邮箱，这个由硬件控制，我们不需要理会）

→收到有效报文

→挂号_2

→收到有效报文

→挂号_3

→收到有效报文

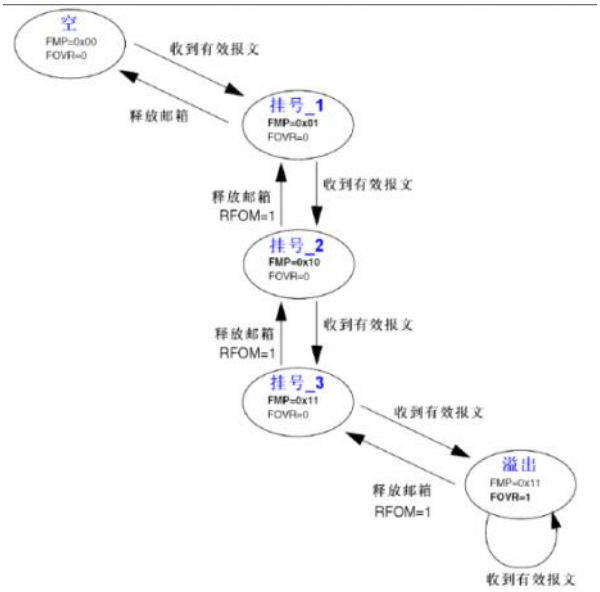
→溢出

这里，我们没有考虑从 FIFO 读出报文的情况

实际情况是：我们必须在 FIFO 溢出之前，读出至少 1 个报文，否则下个报文到来，将导致 FIFO 溢出，从而出现报文丢失

每读出 1 个报文，相应的挂号就减 1，直到 FIFO 空

图示：

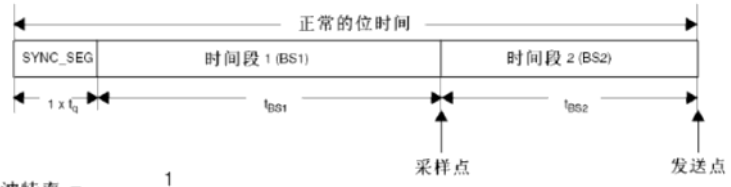


——CAN 位时间特性：

STM32 把传播时间段和相位缓冲段 1（STM32 称之为时间段 1）合并了

所以 STM32 的 CAN 一个位只有 3 段：同步段（SYNC_SEG）、时间段 1（BS1）和时间段2（BS2）

STM32 的 BS1 段可以设置为 1~16 个时间单元，刚好等于我们上面介绍的传播时间段和相位缓冲段 1 之和



$$波特率 = \frac{1}{正常的位时间}$$

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

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

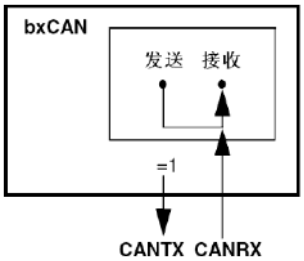
BRP[9:0], TS1[3:0] 和 TS2[2:0] 在 CAN_BTR 寄存器中定义

eg：

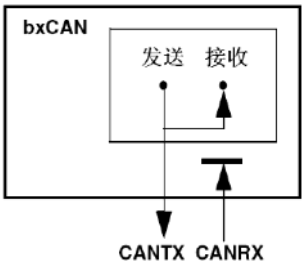
比如设置 TS1=6、 TS2=7 和 BRP=4，在 APB1 频率为 36Mhz 的条件下，即可得到 CAN 通信的波特率=36000/[(7+8+1)*5]=450Kbps

——CAN工作模式：

静默模式：（不发送）



环回模式：（不接收外部信息，向外发送+自发自收）



环回模式可用于自测试

在环回模式下， bxCAN 在内部把 Tx 输出回馈到 Rx 输入上，而完全忽略 CANRX 引脚的实际状态

发送的报文可以在 CANTX 引脚上检测到

静默环回模式：（只自发自收）