CANOpen 系列教程 02

理解 CAN 总线协议

作者：strongerHuang

申明：该文档仅供个人学习使用

归类	CANOpen 系列教程
标签	CAN、 CANOpen、 CanFestival
网站	http://www.strongerhuang.com

目 录

一、写在前面

上一篇文章讲述了 CAN 和 CANOpen，相信大家 CAN 和 CANOpen 有一定理解了。本文说的 **CAN 即是一种总线，也是一种协议**。因此，我们常听见 CAN 总线，也常听见 CAN 协议。

CAN 协议和 CANOpen 协议是两套不同的协议。**从软硬件层次来划分，CAN 协议属于硬件协议，而 CANOpen 属于软件协议**。

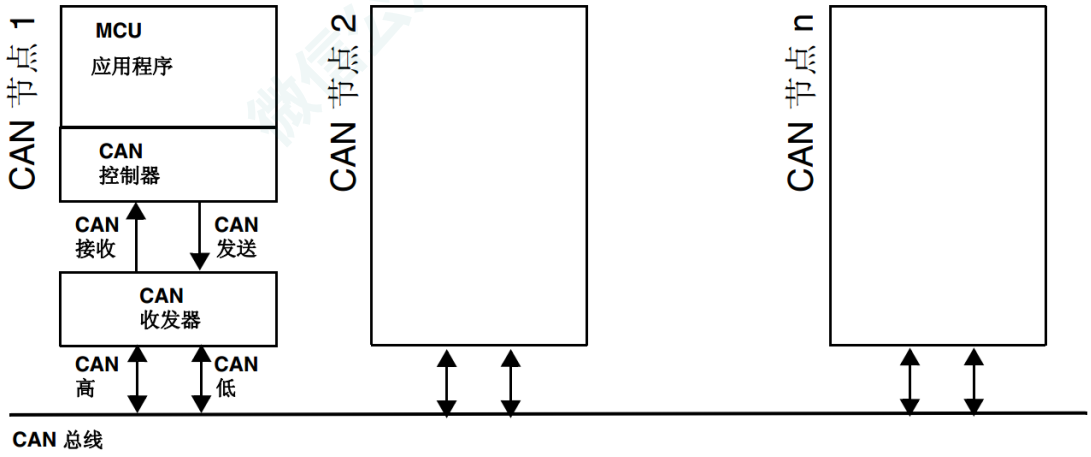
本篇文章先概述一下 CAN 网络，让大家对 CAN 总线协议有一个全局的概念，再到底层的 CAN 总线协议知识。

本文章收录于【CANOpen 系列教程】，在我的博客分类“CANOpen 系列教程”也能查找到。

为了方便大家平时公交、地铁、外出办事也能用手机随时随地查看该教程，该系列教程也同步更新于微信公众号【EmbeddedDevelop】，关注微信公众号回复【CANOpen 系列教程】即可查看。

二、CAN 网络

CAN 网络可以理解为多台 CAN 设备连接在**同一条 CAN 总线**上组合成的网络，**其中的 CAN 设备我们称之为节点**。CAN 网络拓扑结构如下图：



如上图，一个 CAN 节点主要包含三类：**MCU 应用程序、CAN 控制器、CAN 收发器**。

2.1 MCU 应用程序

MCU 应用程序我将其分为三块：**业务逻辑代码、协议层代码、底层驱动代码**。

A.业务逻辑代码：是根据项目需求而定，也很好理解。比如我读取一个传感器数据，并对其做出相应逻辑处理。

B.协议层代码：比如后续要讲述的 CANOpen。

C.底层驱动代码：配置 CAN 总线相应参数、控制收发的代码。

2.2 CAN 控制器

CAN 控制器内部结构还是挺复杂的，**一般现在 CAN 控制器都是与处理器集成在一起**。

其实对于编程的人来说，无非也就是包含一些控制、状态、配置等寄存器。

比如我们看到有些 STM32 芯片带有 CAN，也就是说 CAN 控制器已经集成在 STM32 芯片中了，我们只需要编程操作其中的寄存器即可。

2.3 CAN 收发器

CAN 收发器：将 CAN 收发引脚（CAN_TX 和 CAN_RX）的 **TTL 信号转换成 CAN 总线的电平信号**。

PS：你可以把 CAN 总线通信认为是 UART 通过 485 进行通信：CAN 控制器就如 UART 的控制器，而 CAN 收发器就如 485 转换芯片。

三、ISO 标准化的 CAN 协议

写这一章节的主要目的就是想让大家了解 **CAN 总线位于 OSI 所在层次**。

3.1 ISO/OSI 基本参照模型

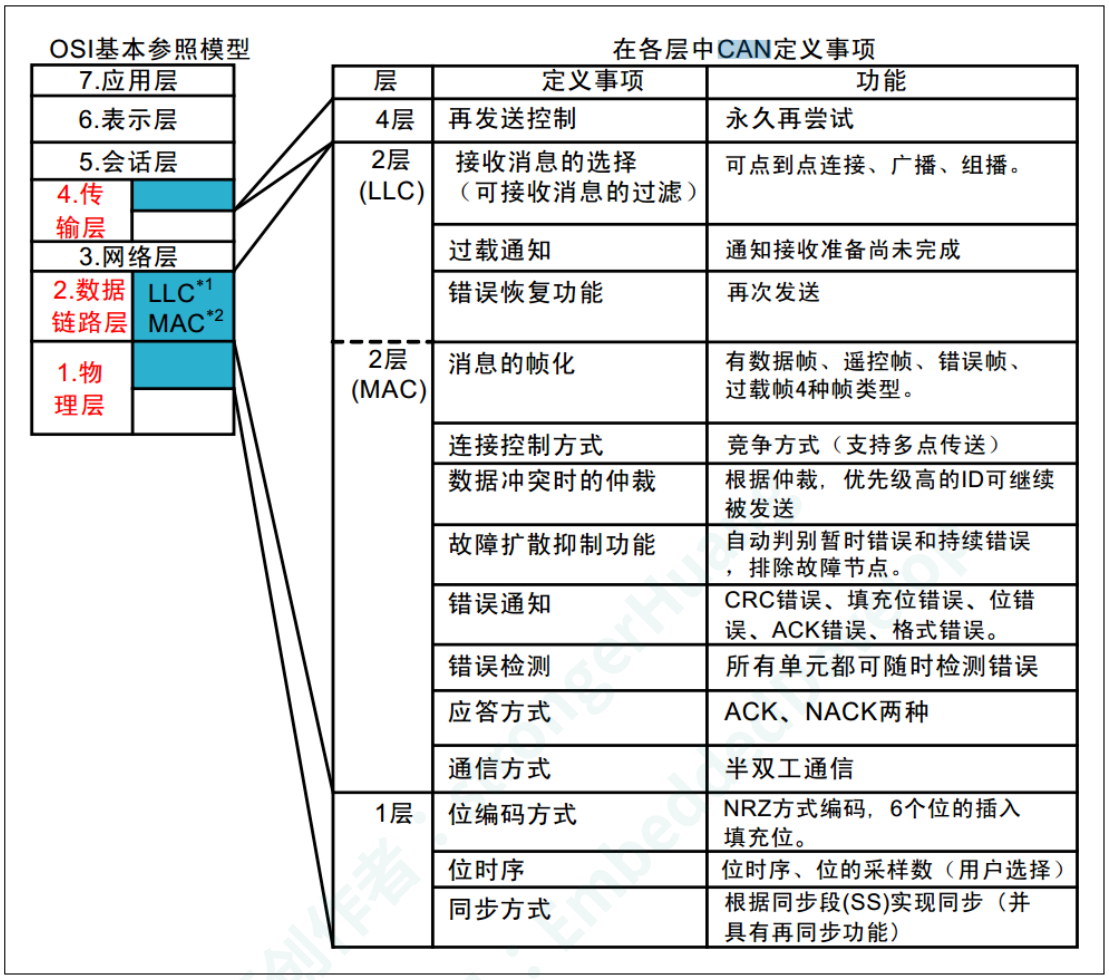
ISO/OSI 基本参照模型		各层定义的主要项目
软件控制	**7 层：应用层**	由实际应用程序提供可利用的服务。
	6 层：表示层	进行数据表现形式的转换。 如：文字设定、数据压缩、加密等的控制
	5 层：会话层	为建立会话式的通信，控制数据正确地接收和发送。
	4 层：传输层	控制数据传输的顺序、传送错误的恢复等，保证通信的品质。 如：错误修正、再传输控制。
	3 层：网络层	进行数据传送的路由选择或中继。 如：单元间的数据交换、地址管理。
硬件控制	**2 层：数据链路层**	将物理层收到的信号（位序列）组成有意义的数据，提供传输错误控制等数据传输控制流程。 如：访问的方法、数据的形式。 通信方式、连接控制方式、同步方式、检错方式。 应答方式、通信方式、包（帧）的构成。 位的调制方式（包括位时序条件）。
	1 层：物理层	规定了通信时使用的电缆、连接器等的媒体、电气信号规格等，以实现设备间的信号传送。 如：信号电平、收发器、电缆、连接器等的形态。

【注】

ISO：International Standardization Organization 国际标准化组织；

OSI：Open Systems Interconnection 开放式系统间互联；

3.2 CAN 在 OSI 模型中的定义



层	定义事项	功能
4层	再发送控制	永久再尝试
2层 (LLC)	接收消息的选择（可接收消息的过滤）	可点到点连接、广播、组播。
	过载通知	通知接收准备尚未完成
	错误恢复功能	再次发送
2层 (MAC)	消息的帧化	有数据帧、遥控帧、错误帧、过载帧4种帧类型。
	连接控制方式	竞争方式（支持多点传送）
	数据冲突时的仲裁	根据仲裁，优先级高的ID可继续被发送
	故障扩散抑制功能	自动判别暂时错误和持续错误，排除故障节点。
	错误通知	CRC错误、填充位错误、位错误、ACK错误、格式错误。
	错误检测	所有单元都可随时检测错误
	应答方式	ACK、NACK两种
	通信方式	半双工通信
1层	位编码方式	NRZ方式编码，6个位的插入填充位。
	位时序	位时序、位的采样数（用户选择）
	同步方式	根据同步段(SS)实现同步（并具有再同步功能）

【注】
LLC：Logical Link Control 逻辑链路控制；
MAC：Medium Access Control 媒介访问控制；

从上图可以知道 CAN 总线底层硬件的内容（CAN 控制器、收发器）主要位于 OSI 的第 1 层和第 2 层。

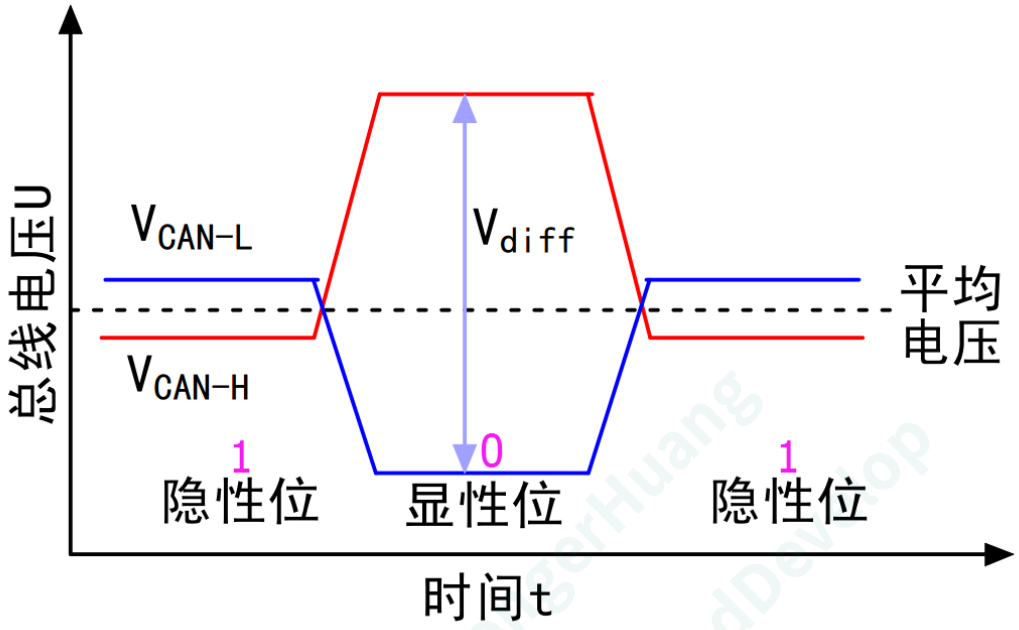
四、概述 CAN 总线协议

CAN 总线协议：就是为了保证通信（收发）数据在 CAN 总线上能稳定传输而制订的一套协议。

CAN 总线协议的内容很多，为方便初学者理解，本文先大概描述一下 CAN 总线协议，后续文章详细讲述 CAN 总线协议的内容。

4.1 总线信号

CAN 总线为「**两线**」「**差分**」**信号**，用隐形代表逻辑 1，显性代表逻辑 0。如下图：



4.2 优先级

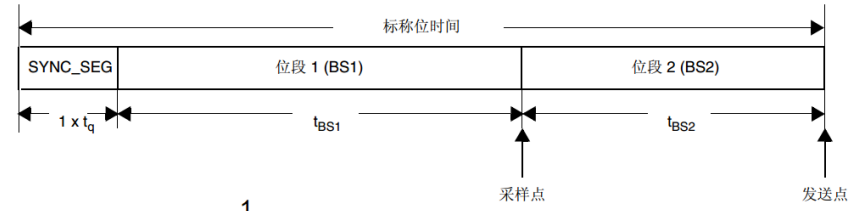
假如某一时刻，一个设备（节点）往总线发 0，一个设备往总线发 1。那么总线会呈现什么现象？

答案：最后总线呈现为**显性**，也就是 0。

4.3 位时序

位时序逻辑将监视串行总线，执行采样并调整采样点，在调整采样点时，需要在起始位边沿进行同步并后续的边沿进行再同步。

简单的说就是对一个 bit 位分几段进行采样，目的就是提高数据传输稳定性。在 STM32 中底层驱动代码就需要进行位时序编程，在 STM32 参考手册中也会发现如下位时序图：



$$BaudRate = \frac{1}{NominalBitTime}$$

$$NominalBitTime = 1 \times t_q + t_{BS1} + t_{BS2}$$

其中：

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

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

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

其中 t_q 为时间片

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。

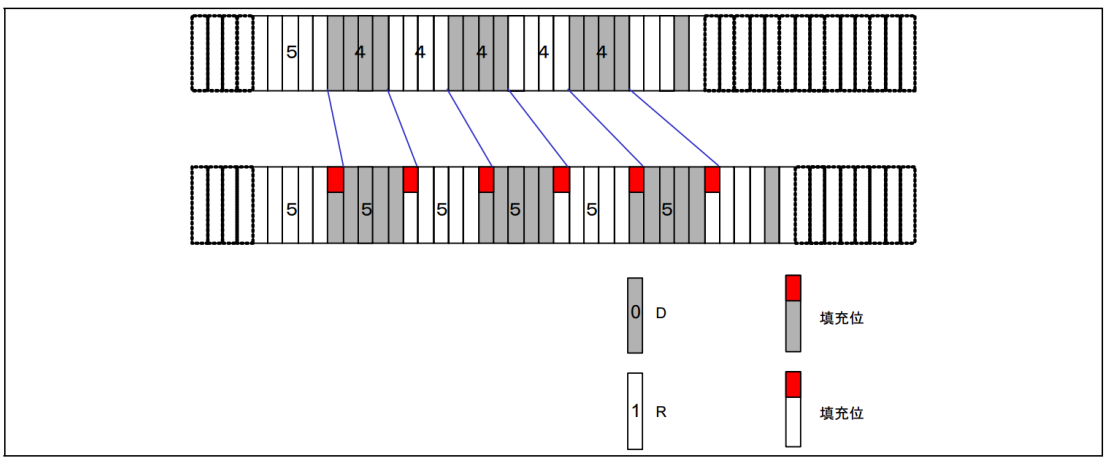
4.4 帧的种类和格式

帧的种类有多种：
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。
帧间隔：用于将数据帧及遥控帧与前面的帧分离开来的帧。

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。

4.5 位填充

位填充是为防止突发错误而设定的功能。**当同样的电平持续 5 位时则添加一个位的反型数据**。如下图：



4.6 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	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） • 遥控帧（SOF~EOF） • 错误帧 • 过载帧	发送单元 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） • 遥控帧（SOF~CRC 顺序）	发送单元 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） • 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） • 遥控帧（CRC 界定符、ACK 界定符、EOF） • 错误界定符 • 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） • 遥控帧（ACK 槽）	发送单元

CAN 总线协议内容很多，初学者先了解这些，后面文章具体到每一个点上，相信大家就会更明白其中的含义。

五、说明

1.该文档部分文字来自网络，仅供个人学习使用，版权所有，禁止商用。

2. 本文由我一个人编辑并整理，难免存在一些错误。

3.本教程收录于微信公众号「嵌入式专栏」，关注微信公众号回复【CANOpen 系列教程】即可查看全系列教程。

六、最后

我的博客：http://www.strongerhuang.com

我的 GitHub：https://github.com/EmbeddedDevelop

我的微信公众号（ID：strongerHuang）还在分享 STM8、STM32、Keil、IAR、FreeRTOS、UCOS、RT-Thread、CANOpen、Modbus...等更多精彩内容，如果想查看更多内容，可以关注我的微信公众号。