CANopen 协议介绍

流行欧洲的 CAN-bus 高层协议

目录

1、介绍

从 OSI 网络模型的角度来看同，现场总线网络一般只实现了第 1 层（物理层）、第 2 层（数据链路层）、第 7 层（应用层）。因为现场总线通常只包括一个网段，因此不需要第 3 层（传输层）和第 4 层（网络层），也不需要第 5 层（会话层）第 6 层（描述层）的作用。

CAN（Controller Area Network）现场总线仅仅定义了第 1 层、第 2 层（见 ISO11898 标准）；实际设计中，这两层完全由硬件实现，设计人员无需再为此开发相关软件（Software）或固件（Firmware）。

同时，CAN 只定义物理层和数据链路层，没有规定应用层，本身并不完整，需要一个高层协议来定义 CAN 报文中的 11/29 位标识符、8 字节数据的使用。而且，基于 CAN 总线的工业自动化应用中，越来越需要一个开放的、标准化的高层协议：这个协议支持各种 CAN 厂商设备的互用性、互换性，能够实现在 CAN 网络中提供标准的、统一的系统通讯模式，提供设备功能描述方式，执行网络管理功能。

- 应用层（Application layer）：为网络中每一个有效设备都能够提供一组有用的服务与协议。
- 通讯描述（Communication profile）：提供配置设备、通讯数据的含义，定义数据通讯方式。
- 设备描述（Device proflile）：为设备（类）增加符合规范的行为。

下面的章节将介绍基于 CAN 的高层协议：CAL 协议和基于 CAL 协议扩展的 CANopen 协议。CANopen 协议是 CAN-in-Automation(CiA)定义的标准之一，并且在发布后不久就获得了广泛的承认。尤其是在欧洲，CANopen 协议被认为是在基于 CAN 的工业系统中占领导地位的标准。大多数重要的设备类型，例如数字和模拟的输入输出模块、驱动设备、操作设备、控制器、可编程控制器或编码器，都在称为“设备描述”的协议中进行描述；“设备描述”定义了不同类型的标准设备及其相应的功能。依靠 CANopen 协议的支持，可以对不同厂商的设备通过总线进行配置。

在 OSI 模型中，CAN 标准、CANopen 协议之间的关系如下图所示：

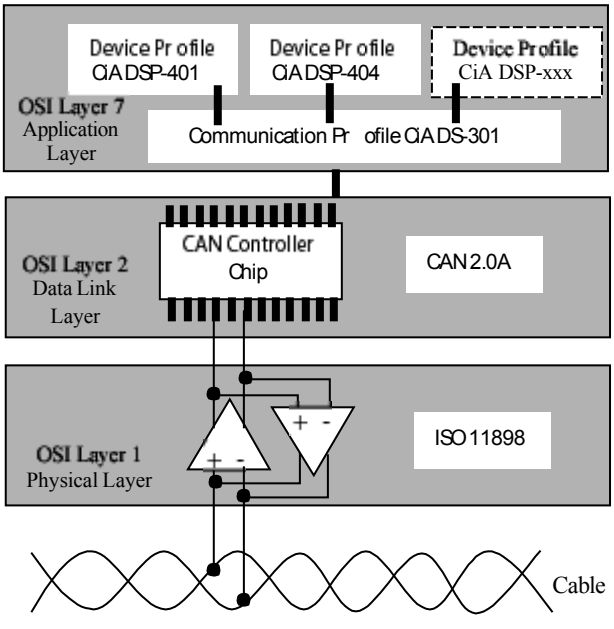

图 1.1　CAN、CANopen 标准在 OSI 网络模型中的位置框图

2、CAL 协议

CAL（CAN Application Layer）协议是目前基于 CAN 的高层通讯协议中的一种，最早由 Philips 医疗设备部门制定。现在 CAL 由独立的 CAN 用户和制造商集团 CiA（CAN in Automation）协会负责管理、发展和推广。

CAL 提供了 4 种应用层服务功能：

- CMS (CAN-based Message Specification)

  CMS 提供了一个开放的、面向对象的环境，用于实现用户的应用。CMS 提供基于变量、事件、域类型的对象，以设计和规定一个设备（节点）的功能如何被访问（例如，如何上载下载超过 8 字节的一组数据（域），并且有终止传输的功能）。

  CMS 从 MMS (Manufacturing Message Specification)继承而来。MMS 是 OSI 为工业设备的远程控制和监控而制定的应用层规范。

- NMT (Network ManagemenT)

  提供网络管理（如初始化、启动和停止节点，侦测失效节点）服务。这种服务是采用主从通讯模式（所以只有一个 NMT 主节点）来实现的。

- DBT (DistriBuTor)

  提供动态分配 CAN ID（正式名称为 COB-ID，Communication Object Identifier）服务。这种服务是采用主从通讯模式（所以只有一个 DBT 主节点）来实现的。

- LMT (Layer ManagemenT)

  LMT 提供修改层参数的服务：一个节点（LMT Master）可以设置另外一个节点（LMT Slave）的某层参数（如改变一个节点的 NMT 地址，或改变 CAN 接口的位定时和波特率）。

CMS 为它的消息定义了 8 个优先级，每个优先级拥有 220 个 COB-ID，范围从 1 到 1760。剩余的标志（0，1761-2031）保留给 NMT，DBT 和 LMT，见表 2-1。

表 2-1　映射到 CAL 服务和对象的 COB-ID(11 位 CAN 标识符)

COB-ID	服务或对象
0	NMT 启动/停止服务
1 - 220	CMS 对象　优先级 0
221 - 440	CMS 对象　优先级 1
441 - 660	CMS 对象　优先级 2
661 - 880	CMS 对象　优先级 3
881 - 1100	CMS 对象　优先级 4
1101 - 1320	CMS 对象　优先级 5
1321 - 1540	CMS 对象　优先级 6
1541 - 1760	CMS 对象　优先级 7
1761 - 2015	NMT 节点保护
2016 - 2031	NMT，LMT，DBT 服务

注意这是 CAN2.0A 标准，11 位 ID 范围[0，2047]，由于历史原因限制在[0，2031]。如果使用 CAN2.0B 标准，29 位 ID 并不改变这个描述；表中的 11 位映射到 29 位 COB-ID 中的最高 11 位，以至于表中的 COB-ID 范围变得增大许多。

3、CANopen

CAL 提供了所有的网络管理服务和报文传送协议，但并没有定义 CMS 对象的内容或者正在通讯的对象的类型（它只定义了 how，没有定义 what）。而这正是 CANopen 切入点。

CANopen 是在 CAL 基础上开发的，使用了 CAL 通讯和服务协议子集，提供了分布式控制系统的一种实现方案。CANopen 在保证网络节点互用性的同时允许节点的功能随意扩展：或简单或复杂。

CANopen 的核心概念是设备对象字典（OD：Object Dictionary），在其它现场总线（Profibus，Interbus-S）系统中也使用这种设备描述形式。注意：对象字典不是 CAL 的一部分，而是在 CANopen 中实现的。

下面先介绍对象字典（OD：Object Dictionary），然后再介绍 CANopen 通讯机制。

3.1 对象字典 OD

对象字典（OD：Object Dictionary）是一个有序的对象组；每个对象采用一个 16 位的索引值来寻址，为了允许访问数据结构中的单个元素，同时定义了一个 8 位的子索引，对象字典的结构参照表 3-1。不要被对象字典中索引值低于 0x0FFF 的‘data types’项所迷惑，它们仅仅是一些数据类型定义。一个节点的对象字典的有关范围在 0x1000 到 0x9FFF 之间。

表 3-1　CANopen 对象字典通用结构

索引	对象
0000	Not used
0001 - 001F	静态数据类型 （标准数据类型，如 Boolean，Integer 16）
0020 - 003F	复杂数据类型 （预定义由简单类型组合成的结构如 PDOCommPar，SDOParameter）
0040 - 005F	制造商规定的复杂数据类型
0060 - 007F	设备子协议规定的静态数据类型
0080 - 009F	设备子协议规定的复杂数据类型
00A0 - 0FFF	Reserved
1000 - 1FFF	通讯子协议区域 (如设备类型，错误寄存器，支持的 PDO 数量)
2000 - 5FFF	制造商特定子协议区域
6000 - 9FFF	标准的设备子协议区域 (例如“DSP-401 I/O 模块设备子协议”：Read State 8 Input Lines 等)
A000 - FFFF	Reserved

CANopen 网络中每个节点都有一个对象字典。对象字典包含了描述这个设备和它的网络行为的所有参数。

一个节点的对象字典是在电子数据文档（EDS：Electronic Data Sheet）中描述或者记录在纸上。不必要也不需要通过 CAN-bus“审问”一个节点的对象字典中的所有参数。如果一个节点严格按照在纸上的对象字典进行描述其行为，也是可以的。节点本身只需要能够提供对象字典中必需的对象（而在 CANopen 规定中必需的项实际上是很少的），以及其它可选择的、构成节点部分可配置功能的对象。

CANopen 由一系列称为子协议的文档组成。

通讯子协议（communication profile），描述对象字典的主要形式和对象字典中的通讯子协议区域中的对象，通讯参数。同时描述 CANopen 通讯对象。这个子协议适用于所有的 CANopen 设备。

还有各种设备子协议（device profile），为各种不同类型设备定义对象字典中的对象。目前已有 5 种不

同的设备子协议，并有几种正在发展。

设备子协议为对象字典中的每个对象描述了它的功能、名字、索引和子索引、数据类型，以及这个对象是必需的还是可选的，这个对象是只读、只写或者可读写等等。

注意：一个设备的通讯功能、通讯对象、与设备相关的对象以及对象的缺省值由电子数据文档（EDS：Electronic Data Sheet）中提供。

单个设备的对象配置的描述文件称作设备配置文件（DCF：Device Configuration File），它和 EDS 有相同的结构。二者文件类型都在 CANopen 规范中定义。

设备子协议定义了对象字典中哪些 OD 对象是必需的，哪些是可选的；必需的对象应该保持最少数目以减小实现的工作量。

可选项－－在通讯部分和与设备相关部分－－可以根据需要增加以扩展 CANopen 设备的功能。如果需要的项超过了设备子协议中可以提供的，在设备子协议中已预留由足够空间提供给厂商的特定功能使用。

对象字典中描述通讯参数部分对所有 CANopen 设备（例如在 OD 中的对象是相同的，对象值不必一定相同）都是一样的。对象字典中设备相关部分对于不同类的设备是不同的。

3．2　CANopen 通讯

前面说明了 CANopen 中对象字典的概念，现在我们来介绍在 CANopen 网络中的通讯消息，它们的内容和功能，换句话：CANopen 通讯模式。

注意：请区分对象字典中的对象（使用对象字典索引和子索引）和通讯对象（或者消息，使用 COB-ID）。

CANopen 通讯模型定义了 4 种报文（通讯对象）：

1．管理报文
   - 层管理，网络管理和 ID 分配服务：如初始化，配置和网络管理（包括：节点保护）。
   - 服务和协议符合 CAL 中的 LMT，NMT 和 DBT 服务部分。这些服务都是基于主从通讯模式：在 CAN 网络中，只能有一个 LMT，NMT 或 DBT 主节点以及一个或多个从节点。

2．服务数据对象 SDO(Service Data Object)
   - 通过使用索引和子索引（在 CAN 报文的前几个字节），SDO 使客户机能够访问设备（服务器）对象字典中的项（对象）。
   - SDO 通过 CAL 中多元域的 CMS 对象来实现，允许传送任何长度的数据（当数据超过 4 个字节时分拆成几个报文）。
   - 协议是确认服务类型：为每个消息生成一个应答（一个 SDO 需要两个 ID）。SDO 请求和应答报文总是包含 8 个字节（没有意义的数据长度在第一个字节中表示，第一个字节携带协议信息）。SDO 通讯有较多的协议规定。

3．过程数据对象 PDO（Process Data Object）
   - 用来传输实时数据， 数据从一个生产者传到一个或多个消费者。数据传送限制在 1 到 8 个字节（例如，一个 PDO 可以传输最多 64 个数字 I/O 值，或者 4 个 16 位的 AD 值）。
   - PDO 通讯没有协议规定。PDO 数据内容只由它的 CAN ID 定义，假定生产者和消费者知道这个 PDO 的数据内容。
   - 每个 PDO 在对象字典中用 2 个对象描述：
     - PDO 通讯参数：包含哪个 COB-ID 将被 PDO 使用，传输类型，禁止时间和定时器周期。
     - PDO 映射参数：包含一个对象字典中对象的列表，这些对象映射到 PDO 里，包括它们的数据长度（in bits）。生产者和消费者必须知道这个映射，以解释 PDO 内容。
   - PDO 消息的内容是预定义的（或者在网络启动时配置的）：

     映射应用对象到 PDO 中是在设备对象字典中描述的。如果设备（生产者和消费者）支持可变 PDO 映射，那么使用 SDO 报文可以配置 PDO 映射参数。
   - PDO 可以有多种传送方式：

- 同步（通过接收 SYNC 对象实现同步）
  - 非周期：由远程帧预触发传送，或者由设备子协议中规定的对象特定事件预触发传送。
  - 周期：传送在每 1 到 240 个 SYNC 消息后触发。
- 异步
  - 由远程帧触发传送。
  - 由设备子协议中规定的对象特定事件触发传送。

表 3-2 给出来了由传输类型定义的不同 PDO 传输模式，传输类型为 PDO 通讯参数对象的一部分，由 8 位无符号整数定义。

表 3-2 PDO 传输类型定义

传输类型	触发 PDO 的条件 (B = both needed O = one or both)			PDO 传输
	SYNC	RTR	Event	
0	B	-	B	同步，非循环
1-240	O	-	-	同步，循环
241-251	-	-	-	Reserved
252	B	B	-	同步，在 RTR 之后
253	-	O	-	异步，在 RTR 之后
254	-	O	O	异步，制造商特定事件
255	-	O	O	异步，设备子协议特定事件

说明：

- SYNC –接收到 SYNC-object。
- RTR　－接收到远程帧。
- Event –例如数值改变或者定时器中断。
- 传输类型为：1 到 240 时，该数字代表两个 PDO 之间的 SYNC 对象的数目）。

- 一个 PDO 可以指定一个禁止时间，即定义两个连续 PDO 传输的最小间隔时间，避免由于高优先级信息的数据量太大，始终占据总线，而使其它优先级较低的数据无力竞争总线的问题。禁止时间由 16 位无符号整数定义，单位 100us。
- 一个 PDO 可以指定一个事件定时周期，当超过定时时间后，一个 PDO 传输可以被触发（不需要触发位）。事件定时周期由 16 位无符号整数定义，单位 1ms。
- PDO 通过 CAL 中存储事件类型的 CMS 对象实现。PDO 数据传送没有上层协议，而且 PDO 报文没有确认（一个 PDO 需要一个 CAN-ID）。每个 PDO 报文传送最多 8 个字节（64 位）数据。

4. 预定义报文或者特殊功能对象
   - 同步（SYNC）
     - 在网络范围内同步（尤其在驱动应用中）：在整个网络范围内当前输入值准同时保存，随后传送（如果需要），根据前一个 SYNC 后接收到的报文更新输出值。
     - 主从模式：SYNC 主节点定时发送 SYNC 对象，SYNC 从节点收到后同步执行任务。
     - 在 SYNC 报文传送后，在给定的时间窗口内传送一个同步 PDO。
     - 用 CAL 中基本变量类型的 CMS 对象实现。
     - CANopen 建议用一个最高优先级的 COB-ID 以保证同步信号正常传送。SYNC 报文可以不传送数据以使报文尽可能短。
   - 时间标记对象（Time Stamp）
     - 为应用设备提供公共的时间帧参考。
     - 用 CAL 中存储事件类型的 CMS 对象实现。

- 紧急事件（Emergency）
  - 设备内部错误触发。
  - 用 CAL 中存储事件类型的 CMS 对象实现。
- 节点/寿命保护（Node/Life guarding）。
  - 主从通讯模式
  - NMT 主节点监控节点状态：称作节点保护（Node guarding）。
  - 节点也可以（可选择）监控 NMT 主节点的状态：称作寿命保护（Life guarding）。当 NMT 从节点接收到 NMT 主节点发送的第一个 Node Guard 报文后启动寿命保护。
  - 检测设备的网络接口错误（不是设备自身的错误）：通过应急指示报告。
  - 根据 NMT 节点保护协议实现： NMT 主节点发送远程请求到一个特定节点，节点给出应答，应答报文中包含了这个节点的状态。
- Boot-UP
  - 主从通讯模式
  - NMT 从节点通过发送这个报文，向 NMT 主节点说明该节点已经由初始化状态进入预操作状态。

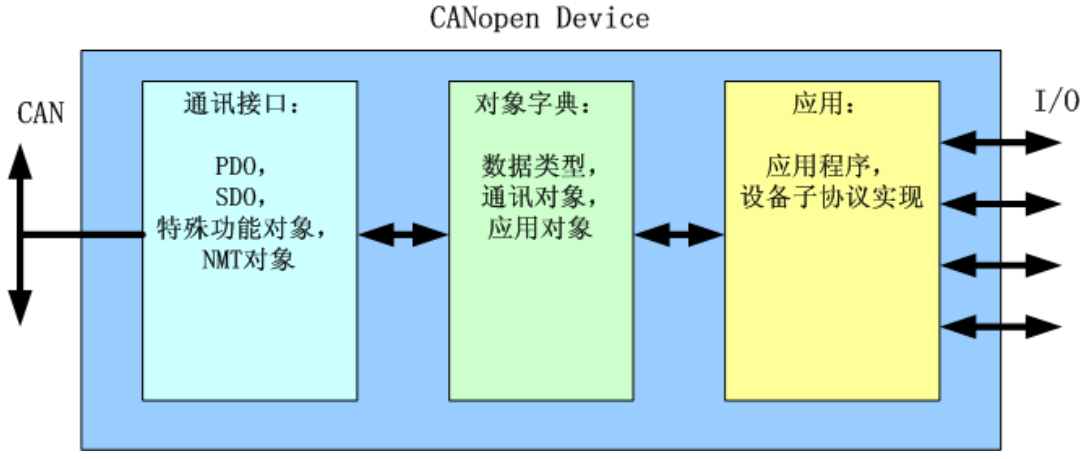

图 3-1 CANopen 设备

上面提到的通讯对象类型中有二个对象用于数据传输。它们采用二种不同的数据传输机制实现：

- SDO 用来在设备之间传输大的低优先级数据，典型的是用来配置 CANopen 网络上的设备。
- PDO 用来传输 8 字节或更少数据，没有其它协议预设定（意味着数据内容已预先定义）。

一个 CANopen 设备必须支持一定数量的网络管理服务（管理报文，administrative messages），需要至少一个 SDO。每个生产或消费过程数据的设备需要至少一个 PDO。所有其它的通讯对象是可选的。一个 CANopen 设备中 CAN 通讯接口、对象字典和应用程序之间的联系如图 3-1 所示。

3．3 CANopen 预定义连接集

为了减小简单网络的组态工作量，CANopen 定义了强制性的缺省标识符（CAN-ID）分配表。这些标志符在预操作状态下可用，通过动态分配还可修改他们。CANopen 设备必须向它所支持的通讯对象的提供相应的标识符。

缺省 ID 分配表是基于 11 位 CAN－ID，包含一个 4 位的功能码部分和一个 7 位的节点 ID(Node-ID)部分。如图 3-2 所示。

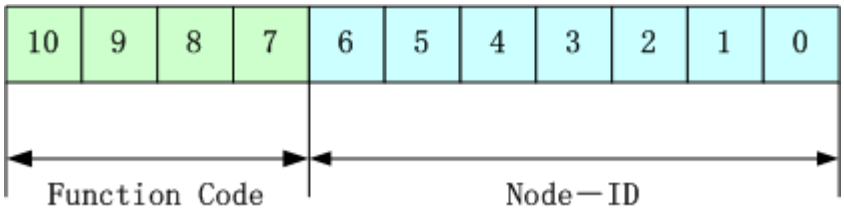

图 3-2　预定义连接集 ID

Node-ID 由系统集成商定义，例如通过设备上的拨码开关设置。Node-ID 范围是 1~127（0 不允许被使用）。

预定义的连接集定义了 4 个接收 PDO（Receive－PDO），4 个发送 PDO（Transmit－PDO），1 个 SDO（占用 2 个 CAN-ID），1 个紧急对象和 1 个节点错误控制(Node-Error-Control)ID。也支持不需确认的 NMT-Module-Control 服务，SYNC 和 Time Stamp 对象的广播。

缺省 ID 分配表如表 3-3 所示。

表格 3-3　CANopen 预定义主/从连接集 CAN 标识符分配表

CANopen 预定义主/从连接集的广播对象			
对象	功能码 （ID-bits 10-7）	COB-ID	通讯参数在 OD 中的索引
NMT Module Control	0000	000H	-
SYNC	0001	080H	1005H，1006H，1007H
TIME SSTAMP	0010	100H	1012H，1013H
CANopen 主/从连接集的对等对象			
对象	功能码 功能码 （ID-bits 10-7）	COB-ID	通讯参数在 OD 中的索引
紧急	0001	081H-0FFH	1024H，1015H
PDO1(发送)	0011	181H-1FFH	1800H
PDO1(接收)	0100	201H-27FH	1400H
PDO2(发送)	0101	281H-2FFH	1801H
PDO2(接收)	0110	301H-37FH	1401H
PDO3(发送)	0111	381H-3FFH	1802H
PDO3(接收)	1000	401H-47FH	1402H
PDO4(发送)	1001	481H-4FFH	1803H
PDO4(接收)	1010	501H-57FH	1403H
SDO(发送/服务器)	1011	581H-5FFH	1200H
SDO(接收/客户)	1100	601H-67FH	1200H
NMT Error Control	1110	701H-77FH	1016H-1017H

注意：

- PDO/SDO 发送/接收是由（slave）CAN 节点方观察的。
- NMT 错误控制包括节点保护（Node Guarding），心跳报文（Heartbeat）和 Boot-up 协议。

3．4　CANopen 标识符分配

ID 地址分配表与预定义的主从连接集（set）相对应，因为所有的对等 ID 是不同的，所以实际上只有一个主设备(知道所有连接的节点 ID)能和连接的每个从节点（最多 127 个）以对等方式通讯。两个连接在一起的从节点不能够通讯，因为它们彼此不知道对方的节点 ID。

比较上表的 ID 映射和 CAL 的映射，显示了具有特定功能的 CANopen 对象如何映射到 CAL 中一般的 CMS 对象。

CANopen 网络中 CAN 标识符（或 COB-ID）分配 3 种不同方法：

- 使用预定义的主从连接集。ID 是缺省的，不需要配置。如果节点支持，PDO 数据内容也可以配置。
- 上电后修改 PDO 的 ID（在预操作状态），使用（预定义的）SDO 在节点的对象字典中适当位置进行修改。
- 使用 CAL　DBT 服务：节点或从节点最初由它们的配置 ID 指称。节点 ID 可以由设备上的拨码开关配置，或使用 CAL　LMT 服务进行配置。当网络初始化完毕，并且启动后，主节点首先通过”Connect_Remote_Node”报文（是一个 CAL　NMT 服务）和每个连接的从设备建立一个对话。一旦这个对话建立，CAN 通讯 ID（SDO 和 PDO）用 CAL DBT 服务分配好，这需要节点支持扩展的 boot-up。

3．5　CANopen boot-up 过程

在网络初始化过程中，CANopen 支持扩展的 boot-up，也支持最小化 boot-up 过程。

扩展 boot-up 是可选的，最小 boot-up 则必须被每个节点支持。两类节点可以在同一个网络中同时存在。

如果使用 CAL 的 DBT 服务进行 ID 分配，则节点必须支持扩展 boot-up 过程。

可以用节点状态转换图表示这两种初始化过程，如图 3-3 所示。扩展 boot-up 的状态图在预操作和操作状态之间比最小化 boot-up 多了一些状态。

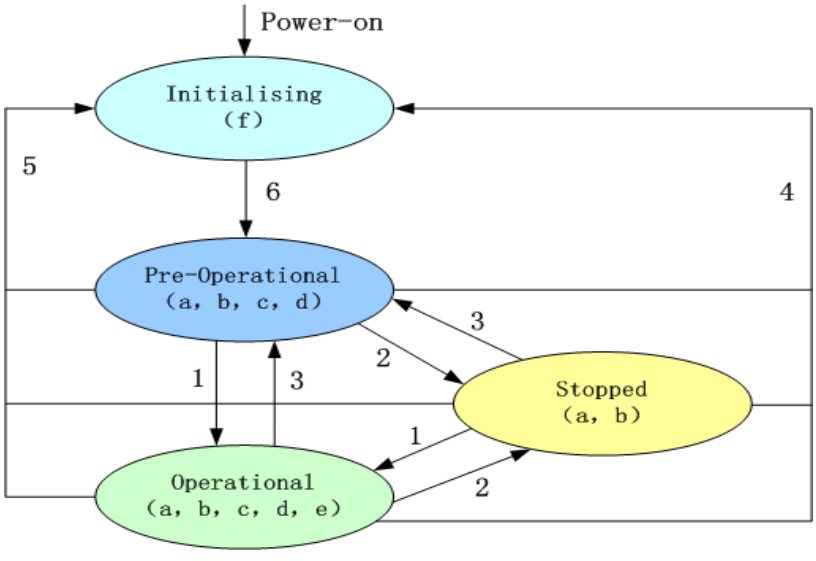

图 3-3 CANopen 最小化 boot-up 节点状态转换图

注意：

- 图 3-3 中括号内的字母表示处于不同状态那些通讯对象可以使用。

a. NMT ， b. Node Guard ， c. SDO ， d. Emergency ， e. PDO ， f. Boot-up

- 状态转移（1－5 由 NMT 服务发起），NMT 命令字（在括号中）：

1： Start_Remote_node (0x01)

2：Stop_Remote_Node (0x02)

3： Enter_Pre-Operational_State (0x80)

4： Reset_Node (0x81)

5：Reset_Communication (0x82)

6：设备初始化结束，自动进入 Pre_Operational 状态，发送 Boot-up 消息

在任何时候 NMT 服务都可使所有或者部分节点进入不同的工作状态。NMT 服务的 CAN 报文由 CAN 头(COB-ID=0)和两字节数据组成；第一个字节表示请求的服务类型('NMT command specifier')，第二个字节是节点 ID，或者 0（此时寻址所有节点）。

仅支持最小化 boot-up 的设备叫最小能力设备。最小能力设备在设备初始化结束后自动进入预操作 1 状态。在这个状态，可以通过 SDO 进行参数配置和进行 COB-ID 分配。

设备进入准备状态后，除了 NMT 服务和节点保护服务（如果支持并且激活的话）外，将停止通讯。（因此'Stopped'是描述这个状态的一个好名字）

3．6 CANopen 消息语法细节

在以下部分中 COB－ID 使用的是 CANopen 预定义连接集中已定义的缺省标志符。

3．6．1 NMT 模块控制（NMT Module Control）

只有 NMT-Master 节点能够传送 NMT Module Control 报文。所有从设备必须支持 NMT 模块控制服务。NMT Module Control 消息不需要应答。NMT 消息格式如下：

NMT-Master ➔ NMT-Slave(s)

COB-ID	Byte 0	Byte 1
0x000	CS	Node-ID

当 Node-ID=0，则所有的 NMT 从设备被寻址。CS 是命令字，可以取如下值：

命令字	NMT 服务
1	Start Remote Node
2	Stop Remote Node
128	Enter Pre-operational State
129	Reset Node
130	Reset Communication

3．6．2 MNT 节点保护（NMT Node Guarding）

通过节点保护服务，MNT 主节点可以检查每个节点的当前状态，当这些节点没有数据传送时这种服务尤其有意义。

NMT-Master 节点发送远程帧（无数据）如下：

NMT-Master ➔ NMT-Slave

COB-ID
0x700+Node_ID

NMT-Slave 节点发送如下报文应答：

NMT-Master ➔ NMT-Slave

COB-ID	Byte0
0x700 + Node_ID	Bit 7 : toggle Bit6-0 : 状态

数据部分包括一个触发位（bit7），触发位必须在每次节点保护应答中交替置“0”或者“1”。触发位在第一次节点保护请求时置为“0”。位 0 到位 6（bits0～6）表示节点状态，可为下表中的数值。

Value	状态
0	Initialising
1	Disconnected *
2	Connecting *
3	Preparing *
4	Stopped
5	Operational
127	Pre-operational

注意：带＊号的状态只有支持扩展 boot-up 的节点才提供。注意状态 0 从不在节点保护应答中出现，因为一个节点在这个状态时并不应答节点保护报文。

或者，一个节点可被配置为产生周期性的被称作心跳报文（Heartbeat）的报文。

Heartbeat Producer ➔ Consumer(s)

COB-ID	Byte 0
0x700 + Node_ID	状态

状态可为下表种的数值：

状态	意义
0	Boot-up
4	Stopped
5	Operational
127	Pre-operational

当一个 Heartbeat 节点启动后它的 Boot-up 报文是其第一个 Heartbeat 报文。Heartbeat 消费者通常是 NMT-Master 节点，它为每个 Heartbeat 节点设定一个超时值，当超时发生时采取相应动作。

一个节点不能够同时支持 Node Guarding 和 Heartbeat 协议。

3．6．3 NMT Boot-up

NMT-slave 节点发布 Boot-up 报文通知 NMT-Master 节点它已经从 initialising 状态进入 pre-operational 状态。

NMT-Master ➔ NMT-Slave

COB-ID	Byte 0
0x700 + Node_ID	0

3．6．4 过程数据对象（PDO）

作为一个例子，假定第二个 transmit-PDO 映射如下（在 CANopen 中用对象字典索引 0x1A01 描述）：

对象 0x1A01：第二个 Transmit-PDO 映射		
子索引	值	意义
0	2	2 个对象映射到 PDO 中
1	0x60000208	对象 0x6000，子索引 0x02，由 8 位组成
2	0x64010110	对象 0x6401，子索引 0x01，由 16 位组成

在 CANopen I/O 模块的设备子协议（CiA DSP-401）定义中，对象 0x6000 子索引 2 是节点的第 2 组 8 位数字量输入，对象 0x6401 子索引 0x01 是节点的第 1 组 16 位模拟量输入。

这个 PDO 报文如果被发送(可能由输入改变，定时器中断或者远程请求帧等方式触发，和 PDO 的传输类型相一致，可以在对象 0x1801 子索引 2 中查找)，则由 3 字节数据组成，格式如下：

PDO-producer ➔ PDO-consumer(s)

COB-ID	Byte 0	Byte 1	Byte 2
0x280+Node_ID	8 位数据量输入	16 位模拟量输入 （低 8 位）	16 位模块量输入 （高 8 位）

通过改变对象 0x1A01 的内容，PDO 的内容可被改变（如果节点支持（可变 PDO 映射））。

注意在 CANopen 中多字节参数总是先发送 LSB（little endian）。

不允许超过 8 个字节的数据映射到某一个 PDO 中。

在 CANopen Application Layer and Communication Profile （CiA DS 301 V 4.02 ）中定义了 MPDO(multiplexor PDO)，允许一个 PDO 传输大量变量，通过在报文数据字节中包含源或目的节点 ID、OD 中的索引和子索引来实现。举个例子：如果没有这个机制，当一个节点有 64 个 16 位的模拟通道时，就需要 16 个不同的 Transmit-PDOs 来传送数据，

3．6．5 服务数据对象（SDO）

SDO 用来访问一个设备的对象字典。访问者被称作客户 (client)，对象字典被访问且提供所请求服务的 CANopen 设备别称作服务器(server)。客户的 CAN 报文和服务器的应答 CAN 报文总是包含 8 字节数据（尽管不是所有的数据字节都一定有意义）。一个客户的请求一定有来自服务器的应答。

SDO 有 2 种传送机制：

- 加速传送（Expedited transfer） ： 最多传输 4 字节数据
- 分段传送（Segmented transfer） ： 传输数据长度大于 4 字节

SDO 的基本结构如下：

Client ➔ Server / Server ➔ Client

Byte 0	Byte 1-2	Byte 3	Byte 4-7
SDO Command Specifier	对象索引	对象子索引	**

（**最大 4 字节数据(expedited transfer)或 4 字节字节计数器（segmented transfer）或关于 block transfer 参数)

或者

Client ➔ Server / Server ➔ Client

Byte 0	Byte 1-70
SDO 命令字	最大 7 字节数据（segmented transfer）

SDO 命令字包含如下信息：

- 下载/上传（Download / upload）
- 请求/应答（Request /response）
- 分段/加速传送（Segmented / expedited transfer）
- CAN 帧数据字节长度
- 用于后续每个分段的交替清零和置位的触发位（toggle bit）

SDO 中实现了 5 个请求/应答协议：启动域下载 （Initiate Domain Download），域分段下载（Download Domain Segment）， 启动域上传 （Initiate Domain Upload），域分段上传 （Upload Domain Segment） 和域传送中止（Abort Domain Transfer）。

在 CANopen 通讯协议的最新版本中，引入了一种新的 SDO 传送机制：

块传送（Block transfer）：当传送数据长度大于 4 字节时，多个分段只由 1 个确认报文应答（如果是下载，则由服务器启动传送，如果是上传，则由客户启动传送）以增加总线吞吐量。

相应的协议为：启动块下载 （Initiate Block Download），块分段下载 （Download Block Segment），块下载结束（End Block Download），启动块上传 （Initiate Block Upload），块分段上传（Upload Block

Segment） 和 块上传结束（End Block Upload）。

下载（Download）是指对对象字典进行写操作，上传（Upload）指对对象字典进行读操作。

这些协议的 SDO 命令字（SDO CAN 报文的第一个字节）语法和细节在下面部分说明：（‘—’表示不相关，应为 0）。

启动域下载（Initiate Domain Download）								
Bit	7	6	5	4	3	2	1	0
Client➔	0	0	1	-	n		e	s
➔Server	0	1	1	-	-	-	-	-

说明：

- n ： 如果 e=1，且 s=1，则有效，否则为 0；表示数据部分中无意义数据的字节数（字节 8—n 到 7 数据无意义）。
- e ： 0 = 正常传送，1 = 加速传送。
- s ： 是否指明数据长度，0 = 数据长度未指明，1 = 数据长度指明。
- e = 0， s = 0： 由 CiA 保留。
- e = 0， s = 1 ： 数据字节为字节计数器，byte 4 是数据低位部分（LSB），byte 7 是数据高位部分（MSB）。
- e = 1 ： 数据字节为将要下载（download）的数据。

域分段下载（Download Domain Segment）								
Bit	7	6	5	4	3	2	1	0
Client➔	0	0	0	t	n			c
➔Server	0	0	1	t	-	-	-	-

说明：

- n ：无意义的数据字节数。如果没有指明段长度，则为 0。
- c ： 0 = 有后续分段需要 download，1 = 最后一个段。
- t ： 触发位，后续每个分段交替清零和置位（第一次传送为 0，等效于 request/response）。

启动域上传（Initiate Domain Upload）								
Bit	7	6	5	4	3	2	1	0
Client➔	0	1	0	-	-	-	-	-
➔Server	0	1	0	-	n		e	s

说明：n，e，s： 与启动域下载相同。

域分段上传（Upload Domain Segment）								
Bit	7	6	5	4	3	2	1	0
Client➔	0	1	1	t	-	-	-	-
➔Server	0	0	0	t	n			c

说明：n，c，t ： 与域分段下载相同。

SDO 客户或服务器通过发出如下格式的报文来中止 SDO 传送：

域传送中止（Abort Domain Transfer）								
Bit	7	6	5	4	3	2	1	0
C➔/➔S	1	0	0	-	-	-	-	-

在域传送中止报文中，数据字节 0 和 1 表示对象索引，字节 2 表示子索引，字节 4 到 7 包含 32 位中止码，描述中止报文传送原因，见表 3-4 所示。

表 3-4　域传送中止 SDO：16 进制中止代码表（字节 4 到 7）

中止代码	代码功能描述
0503 0000	触发位没有交替改变
0504 0000	SDO 协议超时
0504 0001	非法或未知的 Client/Server 命令字
0504 0002	无效的块大小（仅 Block Transfer 模式）
0504 0003	无效的序号（仅 Block Transfer 模式）
0503 0004	CRC 错误（仅 Block Transfer 模式）
0503 0005	内存溢出
0601 0000	对象不支持访问
0601 0001	试图读只写对象
0601 0002	试图写只读对象
0602 0000	对象字典中对象不存在
0604 0041	对象不能够映射到 PDO
0604 0042	映射的对象的数目和长度超出 PDO 长度
0604 0043	一般性参数不兼容
0604 0047	一般性设备内部不兼容
0606 0000	硬件错误导致对象访问失败
0606 0010	数据类型不匹配，服务参数长度不匹配
0606 0012	数据类型不匹配，服务参数长度太大
0606 0013	数据类型不匹配，服务参数长度太短
0609 0011	子索引不存在
0609 0030	超出参数的值范围(写访问时)
0609 0031	写入参数数值太大
0609 0032	写入参数数值太小
0609 0036	最大值小于最小值
0800 0000	一般性错误
0800 0020	数据不能传送或保存到应用
0800 0021	由于本地控制导致数据不能传送或保存到应用
0800 0022	由于当前设备状态导致数据不能传送或保存到应用
0800 0023	对象字典动态产生错误或对象字典不存在 （例如，通过文件生成对象字典，但由于文件损坏导致错误产生）

启动块下载（Initiate Block Download）								
Bit	7	6	5	4	3	2	1	0
Client➔	1	1	0	-	-	cc	s	0
➔Server	1	0	1	-	-	sc	-	0

说明：

- cc ：客户数据是否使用 CRC 校验， 0 = no，1 = yes。
- sc ：服务器数据是否使用 CRC 校验， 0 = no， 1 = yes。

- s ： 是否指明数据集长度，0=数据集长度未指明，1＝数据集长度指明。
- s=0 ：由 CiA 保留。
- s=1 ： 数据字节为字节计数器，字节 4：LSB，字节 7：MSB。
- 服务器字节 4 表示 blksize，即每块中分段的数目，0<blksize<128。

块分段下载（Download Block Segment）								
Bit	7	6	5	4	3	2	1	0
Client➔	c	0						
Client➔	c	1						
…etc…	c	seqno						
➔Server	1	0	1	-	-	-	1	0

说明：

- c ： 是否有后续分段需要 download，0＝yes，1=no。
- seqno ： 分段号，0<seqno<128。
- 客户数据字节：每分段至多包括 7 字节数据被 download。
- 服务器字节 1：表示最后一个被成功接收的分段号；如果为 0，表示分段号为 1 的分段未正确接收，所有段必须重传。
- 服务器字节 2：包含 blksize，每个块中分段的数目，客户机必须使用它进行下一次块下载，0<blksize<128。

块下载结束（End Block Download）								
Bit	7	6	5	4	3	2	1	0
Client➔	1	1	0	n			-	1
➔Server	1	0	1	-	-	-	-	1

说明：

- n ： 指示最后一个块的最后一个段中无意义数据的字节数。
- 客户数据 bytes1+2 为整个数据集的 16 位 CRC；只有当启动块下载报文中 cc 和 sc 同时为 1 时 CRC 才有效。

启动块上传（Initiate Block Upload）								
Bit	7	6	5	4	3	2	1	0
Client➔	1	0	1	-	-	cc	0	0
➔Server	1	1	0	-	-	sc		0
Client➔	1	0	1	-	-	-	1	1

说明：

- cc ： 客户数据是否使用 CRC 校验， 0＝no ， 1=yes。
- sc ： 服务器数据是否使用 CRC 校验， 0=no， 1=yes。
- s ： 是否指明数据集长度，0=数据集长度未指明，1＝数据集长度指明。
- s=0： 由 CiA 保留。
- s=1：数据字节为字节计数器，字节 4：LSB，字节 7：MSB。
- 客户数据字节 4：表示 blksize，即每块中分段的数目，0<blksize<128。
- 客户数据字节 5：表示协议转换字 （pst：Protocol Switch Threshold），用于是否允许改变 SDO 传送协议，0=不允许改变，1=允许改变（如果要 upload 的数据字节数少于或等于 pst，Server 可以通过 Initiate Domain Upload 协议应答改变到 Upload Domain 协议）。

块分段上传（Upload Block Segment）								
Bit	7	6	5	4	3	2	1	0
←Server	c	0						
←Server	c	1						
..etc…	c	seqno						
Client→	1	1	0	-	-	-	1	0

说明：

- c ： 是否有后续分段需要 download，0＝yes，1=no。
- seqno ： 分段号，0<seqno<128。
- 服务器数据字节：每分段至多包括 7 字节数据被 download。
- 客户字节 1：表示最后一个被成功接收的分段号；如果为 0，表示分段号为 1 的分段未正确接收，所有段必须重传。
- 客户字节 2：包含 blksize，每个块中分段的数目，客户机必须使用它进行下一次块下载，0<blksize<128。

块上传结束（End Block Upload）								
Bit	7	6	5	4	3	2	1	0
←Server	1	1	0	n			-	1
Client→	1	0	1	-	-	-	-	1

说明：

- n ： 指示最后一个块的最后一个段中无意义数据的字节数。
- 服务器数据 bytes1+2 为整个数据集的 16 位 CRC；只有当启动块上传报文中 cc 和 sc 同时为 1 时 CRC 才有效。

下面给出几个例子说明如何使用 SDO 来访问一个节点的对象字典。

使用下面的 SDO 消息，值 0x3FE 将写到节点 ID 为 2 的对象字典中索引为 0x1801，子索引为 3 的对象中去，使用启动域下载协议，加速传输（2 字节数据）：

Client → Server (节点#2)

COB-ID	Byte						
	0	1	2	3	4	5	6-7
602	2B	01	18	03	FE	03	-
Client ← Server(节点＃2)							
582	60	01	18	03	-	-	-

使用下面的 SDO 消息，同样的对象字典中索引为 0x1801，子索引为 3 的对象将被读出，使用启动域上传协议，服务器使用加速传输方式应答（2 字节数据）：

Client → Server(节点＃2)

COB-ID	Byte						
	0	1	2	3	4	5	6-7
602	40	01	18	03	-	-	-
Client ← Server(节点＃2)							
582	4B	01	18	03	FE	03	-

3．6．6 应急指示对象(Emergency Object)

应急指示报文由设备内部出现的致命错误触发，由相关应用设备已最高优先级发送到其它设备。适用于中断类型的错误报警信号。

一个应急报文由 8 字节组成，格式如下：

sender ➔ receiver(s)

COB-ID	Byte 0-1	Byte 2	Byte 3-7
0x080+Node_ID	应急错误代码	错误寄存器 (对象 0x1001)	制造商特定的错误区域

16 进制的应急错误代码如下表 3-5 所示。应急错误代码中‘xx’部分由相应的设备子协议定义。

表 3-5 应急错误代码（16 进制）

应急错误代码	代码功能描述
00xx	Error Reset 或 No Error
10xx	Generic Error
20xx	Current
21xx	Current，device input side
22xx	Current，inside the device
23xx	Current，device output side
30xx	Voltage
31xx	Mains voltage
32xx	Voltage inside the device
33xx	Output voltage
40xx	Temperature
41xx	Ambient temperature
42xx	Device tempearture
50xx	Device hardware
60xx	Device software
61xx	Internal software
62xx	User software
63xx	Data set
70xx	Additional modules
80xx	Monitoring
81xx	communication
8110	CAN overrun
8120	Error Passive
8130	Life Guard Error 或 Heartbeat Error
8140	Recovered from Bus-Off
82xx	Protocol Error
8210	PDO no processed Due to length error
8220	Length exceedd
90xx	External error
F0xx	Additional functions
FFxx	Device specific

错误寄存器(Error Register)在设备的对象字典（索引 0x1001）中，表 3-6 说明了错误寄存器的位定义。设备可以将内部错误映射到这个状态字节中，并可以快速查看当前错误。

表 3-6 8 位错误寄存器位定义

Bit	错误类型
0	Generic
1	Current
2	Voltage
3	Temperature
4	Communication
5	Device profile specific
6	Reserved(=0)
7	Manufacturer specific

制造商特定错误区域可能包含与设备相关的其它的错误信息。

4、总结

基于 CAN 总线的 CANopen 网络通讯具有以下特点：

- 使用对象字典（OD：Object Dictionary）对设备功能进行标准化的描述。
- 使用 ASCII 文档：电子数据文档（EDS）和设备配置文件（DCF）对设备及其配置进行标准化的描述。
- CANopen 网络的数据交换和系统管理基于 CAL 中 CMS 服务。
- 系统 boot-up 和节点保护（Node Guarding）的标准基于 CAL 中 NMT 服务。
- 定义了整个系统的同步操作。
- 定义了节点特定的应急报文。

为与 CANopen 通讯协议和相应的设备子协议保持一致，以使制造商的产品能够用于任何 CANopen 网络，以下 3 种层次的兼容性要求需要满足（对日益增长的设备兼容性的要求）：

- 一致性：

设备连接到 CANopen 网络后不能影响其他设备的通讯：应用层的一致性。

- 互用性：

设备能够同网络上的其它节点交换数据：通讯协议的一致性。

- 互换性：

设备能够代替另外一个同类设备：设备子协议的一致性。

一个 CANopen 设备至少应该具有（最小能力设备）：

- 一个节点 ID，
- 一个对象字典（内容由设备功能决定），
- 一个 SDO，能够访问对象字典中必需的对象（只读），
- 支持下列 NMT 从设备服务：
  Reset_Node， Enter_Preoperational_State， Start_Remote_Node，
  Stop_Remote_Node， Reset_Communication，
- 缺省的标识符分配。

5、说明

本文章的大部分内容翻译自国外介绍 CANopen 协议的一篇文章《CANopen：high-level protocol for CAN-bus》，该文章比较全面地介绍了 CANopen 协议。

CANopen 协议是基于 CAN-bus 的一种高层协议，在欧洲应用较为广泛，适合于电梯电气、越野汽车、航海电子、医疗电器、工程机械、铁路机车等领域，且协议针对行业应用，实现比较简洁。

翻译此文的目的在于向一些特定行业的 CAN-bus 用户提供关于 CANopen 协议的有用信息，希望能够对这些行业的 CANopen 协议产品设计起一定的指导作用，从而使我国的 CAN-bus 应用及早与国际接轨。

原文来自荷兰 NIKHEF 公司网站，作者为：H. Boterenbrood。