CANOpen 系列教程 15

NMT 网络管理和相关源码说明

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

该系列教程 13、14 讲述了移植相关内容，以及提供给大家可直接编译、下载运行的源码工程，想必你已经掌握了一些基础的知识了。

在《CANOpen 系列教程 08_CANOpen 通信接口引导学习》中，引导大家参看《CANopen 轻松入门》通信接口的相关知识。而没有在文中具体描述关于**CANOpen 通信接口**的知识。

原因在于通信接口的内容太多，单纯的去看这些内容，容易让人产生更多困惑。

对于程序员来说，**结合源代码来理解相关理论知识更加容易**。所以，我将其留在了这后面来讲述。

下面结合移植好的源代码，讲述 **CANOpen 网络管理（NMT：Network management）**的相关知识。

二、关于网络管理 NMT

在 CANOpen 网络中，分主站和从站，而通常由主站来管理整个网络。比如：复位、停止等。

《CANopen 轻松入门》中军队的例子说的很好：一个军队，如果没有指挥员来管理，下面士兵岂不乱套了。

例子中**指挥员就是 CANOpen 中的网络管理主站，士兵就是从站**。

重要的一点：**每个 CANopen 从节点的 CANopen 协议栈中，必须具备 NMT 管理的相应代码**。也就是说从站是被接收管理。

当然，我们使用的 Canfestival 这套免费框架，大家不必担心，肯定是具有相关代码，我们需要明白怎么使用这套源码即可。

三、NMT 网络管理节点状态

3.1 6 种节点状态

NMT 网络管理主要包含以下 **6 种状态**：
1.初始化（Initializing）：节点上电后对功能部件包括 CAN 控制器进行初始化；

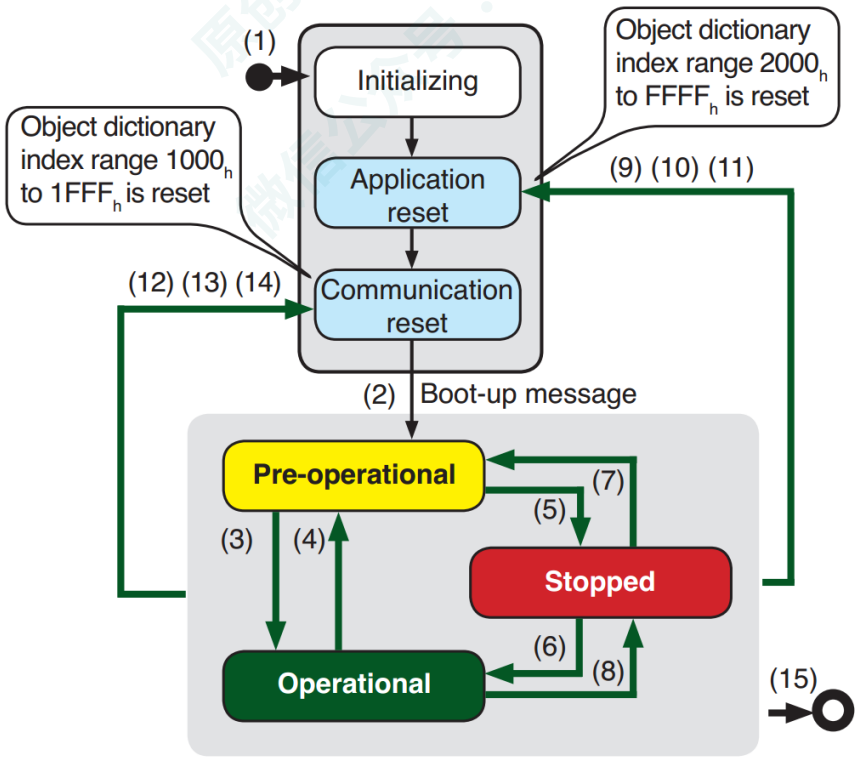
2.应用层复位（Application Reset）：节点中的应用程序复位（开始），比如开关量输出、模拟量输出的初始值；

3.通讯复位（Communication reset）：节点中的 CANopen 通讯复位（开始），从这个时刻起，此节点就可以进行 CANopen 通讯了；

4.预操作状态（Pre-operational）：节点的 CANopen 通讯处于操作就绪状态，此时此节点不能进行 PDO 通信，而可以进行 SDO 进行参数配置和 NMT 网络管理的操作；

5.操作状态（operational）：节点收到 NMT 主机发来的启动命令后， CANopen 通讯被激活， PDO 通信启动后， 按照对象字典里面规定的规则进行传输，同样 SDO 也可以对节点进行数据传输和参数修改；

6.停止状态（Stopped）： 节点收到 NMT 主机发来的停止命令后，节点的 PDO 通信被停止，但 SDO 和 NMT 网络管理依然可以对节点进行操作；



NMT 网络管理 6 种状态如上图所示，其中 1 --- 15 各处代表含义：

(1)：Power on 上电初始化

(2)：Automatic switch to Pre-operational 自动切换预操作状态

(3) 、(6)：NMT switch to Operational 网络管理切换到操作状态

(4)、(7)：NMT switch to Pre-operational 网络管理切换到预操作状态

(5)、(8)：NMT switch to Stopped 网络管理切换到停止状态

(9)、(10)、(11)：NMT switch to Application reset 网络管理切换到应用层复位状态

(12)、(13)、(14)：NMT switch to Communication reset 网络管理切换到通讯复位状态

(15)：Power-off or hardware reset 掉电或硬件复位

3.2 源码说明

通过上面描述，需要知道两点重要内容：**主站进行网络管理，网络各个节点有多种状态**。当然，一个时刻只能一种状态。

在 Canfestival 框架源码中，主站可通过 **masterSendNMTstateChange** 这个函数接口来管理网络节点的状态。可以理解为：**主站控制，或切换从站节点的状态**。

这里需要理解 masterSendNMTstateChange 这个函数接口的用法，也就是说接口含义，以及参数。

比如：主站上电之后，让网络中节点 0x01 复位：

```
static void CANOpen_App_Task(void *pvParameters)
{
  unsigned char nodeID = 0x00;                    //节点ID

  setNodeId(&TestMaster_Data, nodeID);
  setState(&TestMaster_Data, Initialisation);
  setState(&TestMaster_Data, Operational);

  masterSendNMTstateChange(&TestMaster_Data, 0x01, NMT_Reset_Node);

  for(;;)                                          节点ID    节点状态
  {                                                          （复位节点）
    vTaskDelay(500);

    /* 应用代码 */
  }
}
```

这里我们结合《CANOpen 系列教程 13》提供的代码基础上，添加一行管理

节点 0x01 的代码：

masterSendNMTstateChange(&TestMaster_Data, 0x01, NMT_Reset_Node);

第 1 个参数 TestMaster_Data：主站对象字典
第 2 个参数 0x01：节点（从站）ID
第 3 个参数 NMT_Reset_Node：复位节点

第 1,2 个参数很好理解，第 3 个参数是通过宏定义在 def.h 中，总共 5 种状态，如下图：

def.h

```
150 /* NMT Command Specifier, sent by master to change a slave state */
151 /* ------------------------------------------------------------- */
152 /* Should not be modified */
153 #define NMT_Start_Node              0x01
154 #define NMT_Stop_Node               0x02
155 #define NMT_Enter_PreOperational    0x80
156 #define NMT_Reset_Node              0x81
157 #define NMT_Reset_Comunication      0x82
158
```

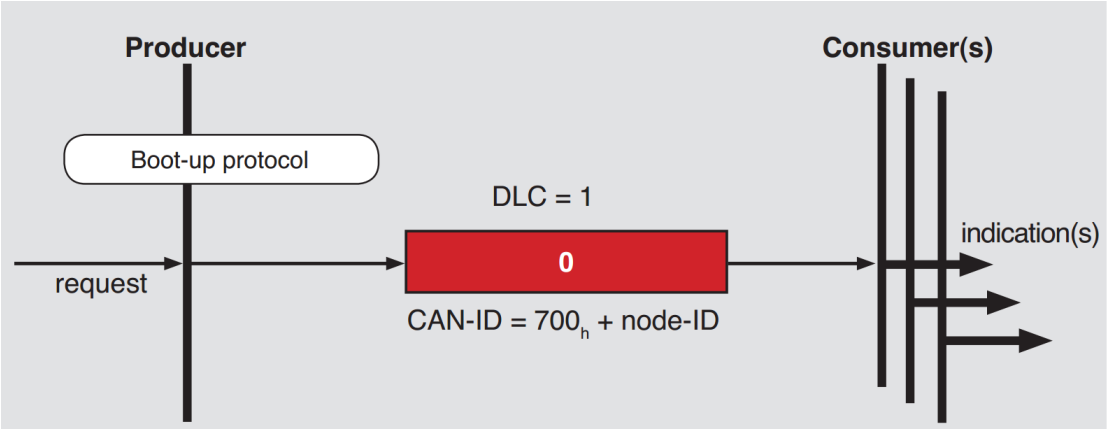
写到这里，相信大家对网络管理节点有一定认识了，初学者可以多结合代码理解。

四、NMT 网络管理节点上线报文

该教程上一篇文章末尾讲述运行效果时，提供了一张运行时 CAN 总线数据的截图：

任何一个 CANopen 从站上线后，为了提示主站它已经加入网络（便于热插拔），或者避免与其他从站 Node-ID 冲突。这个从站必须发出节点上线报文（boot-up）。

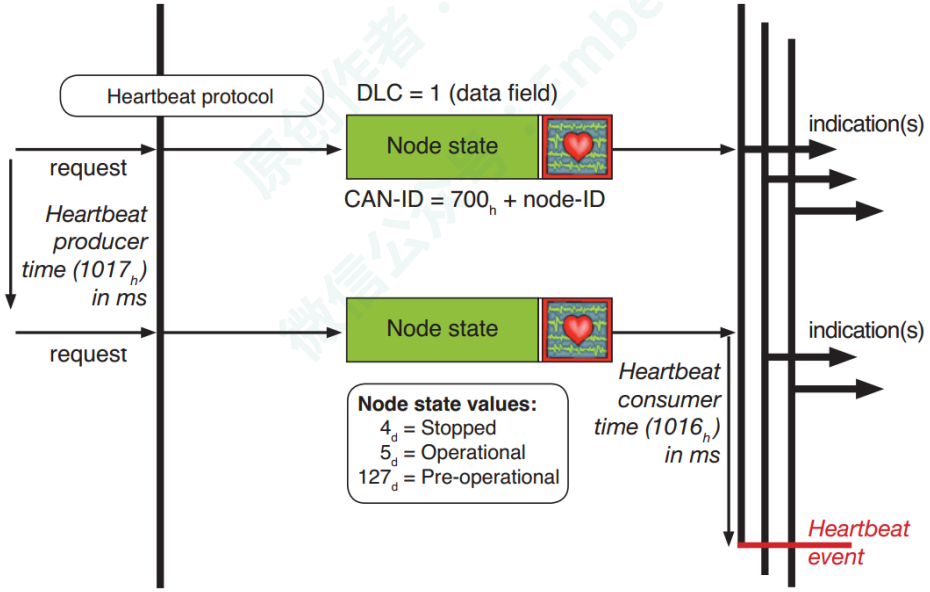
节点上线报文的 ID 为：**700h + Node-ID**，数据规定为 **0**。生产者（Producer）为 CANopen 从站。如下图：



五、NMT 网络管理心跳报文

为了**监控 CANopen 节点是否在线与目前的节点状态**。CANopen 应用中通常都要求在线上电的从站定时发送状态报文（心跳报文），以便于主站确认从站是否故障、是否脱离网络。

格式：
CAN-ID：700h + Node-ID
数据：一字节状态



在《CANOpen 系列教程 13》提供例程中，就是包含一个心跳报文。心跳报文比较简单，请结合上图理解。

注意：
1.CANopen 从站按其对象字典中 **1017h 中填写的心跳生产时间**（ms）进行心跳报文的发送。

2.CANopen 主站则会按其 **1016h 中填写的心跳消费时间进行检查**，假设超过诺干次心跳消费时间没有收到从站的心跳报文，则认为从站已经离线或者损坏。

还有一些网络管理相关内容，大家可以结合这种思路去理解，建议参看：
1.CiA 301 V4.2.0 网络管理章节
2.周立功的 CANOpen 轻松入门

六、说明

1.该文档仅供个人学习使用，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。

七、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。