AN0269
应用笔记

CANopen on AT32 MCU

前言

本应用笔记介绍了如何将CANopenNode协议栈移植到AT32单片机方法。本文档提供的源代码演示了使用CANopen的应用程序。单片机（作为CANopen从机）通过收发器电路板连接到CAN总线，上位机软件CANopen Device Explorer（作为CANopen主机）通过PCAN-USB适配器也连接到同一个CAN总线上，与单片机建立一个最基本的CANopen通讯网络。另外，还会介绍一下使用CANopen的对象字典编辑工具来配置参数，可方便生成源代码和配置文件等。

本文将以AT32M412单片机为示例详细论述其移植、演示的过程，同时也有提供AT32F403A单片机的例程以供参考。

注：本应用笔记对应的代码是基于雅特力提供的V2.x.x 板级支持包（BSP）而开发，对于其他版本BSP，需要注意使用上的区别。

支持型号列表：

支持型号	AT32M412 系列
	AT32F403A 系列

目录

表目录

图目录

1 概述

CANopen协议是在20世纪90年代末，由总部位于德国纽伦堡的CiA（CAN in Automation）组织，在CAL（CAN Application Layer）的基础上发展而来。

经过对CANopen协议规范文本的多次修改，使得CANopen协议的稳定性、实时性、抗干扰性都得到了进一步的提高。并且CiA在CANopen的基础协议——CiA 301之上，对各个行业不断推出设备子协议，使CANopen协议在各个行业得到更快的发展与推广。

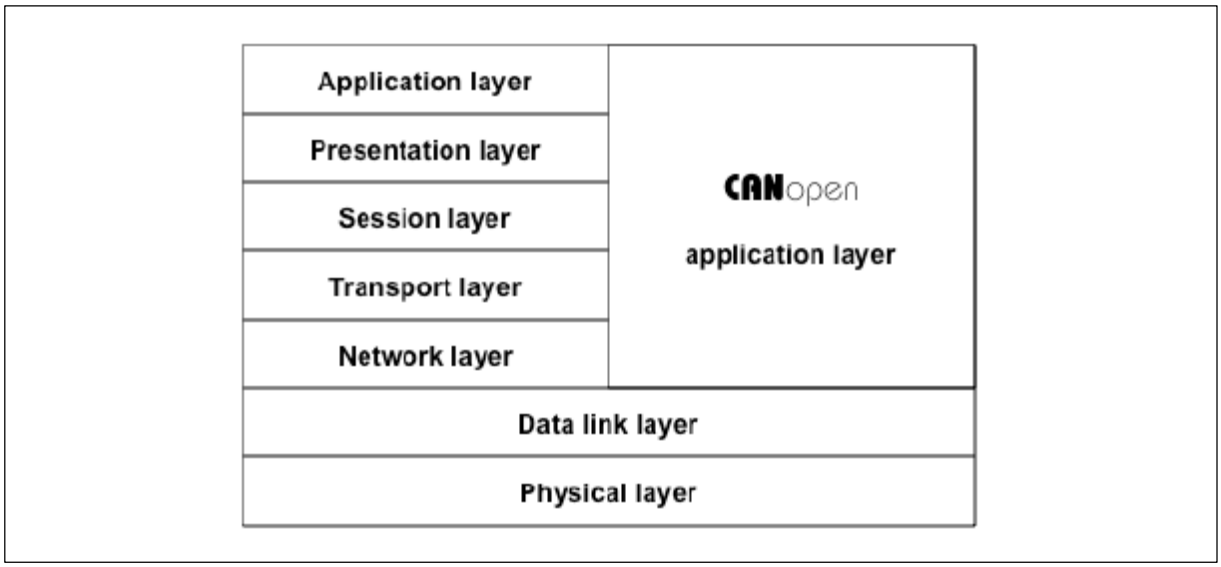
1.1 关于 CANopen 协议

下面将简单介绍一下关于CANopen的基础协议——CiA 301（CANopen应用层和通讯描述协议）。

1.1.1 通信参考模型

CANopen的通信概念符合ISO-OSI的参考模型（即ISO 7498-1定义的基本模式），如下图所示。

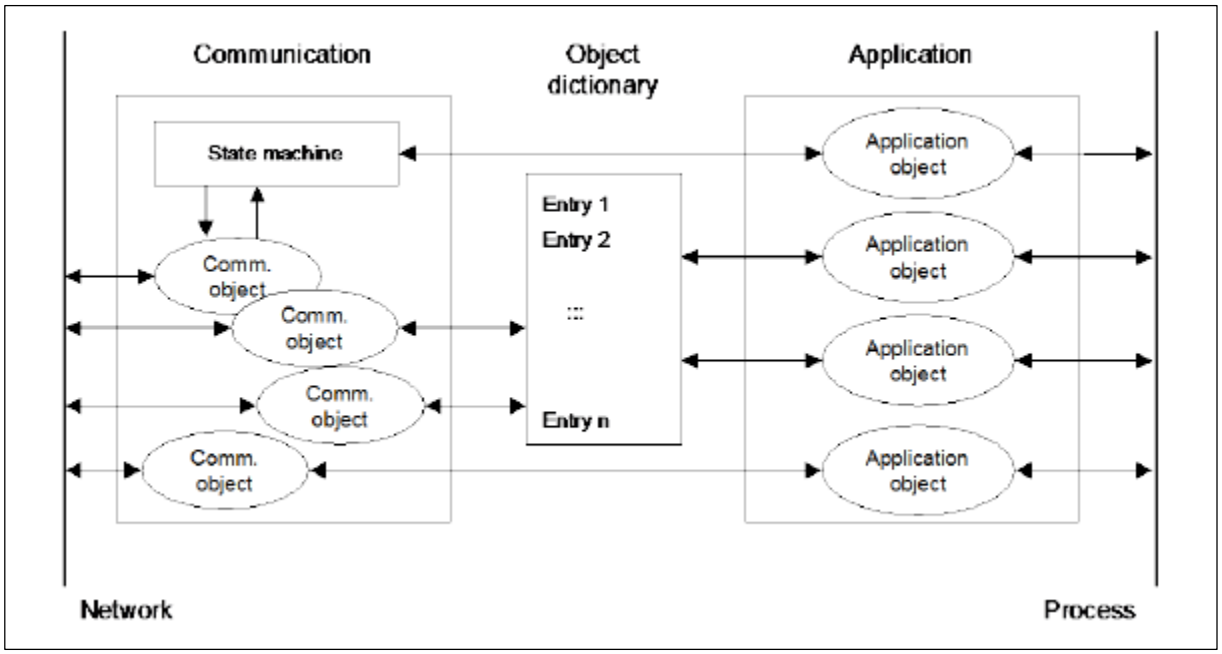
图 1. CANopen 通信参考模型



1.1.2 设备模型

CANopen的设备结构通常分为用户应用层、对象字典以及通信三个部分，如下图所示。

图 2. CANopen 设备结构



1、通信：提供通信对象和与之相应的通过底层网络结构传输数据的能力。

2、对象字典：集合了设备上所有影响应用程序对象、通信对象和状态机行为的数据项。

3、应用：包括与过程环境能够产生相互作用的设备功能。

因此，对象字典作为一个服务于通信与应用之间的一个接口。其中最为核心的就是对象字典，它描述了应用对象和CANopen报文之间的关系。后续还会介绍针对与CANopenNode协议栈配套使用的对象字典的开源编辑工具，以方便开发测试。

1.1.3 预定义报文的 ID 分类

CANopen主要是有突发发送的实时性优势，所以在设计CANopen时，对其定义为小网络、控制信号的实时通讯，总结主要有以下的通信定义：

1、报文传输采用CAN标准帧格式。即11 bit的ID域，以尽量减小传输时间。

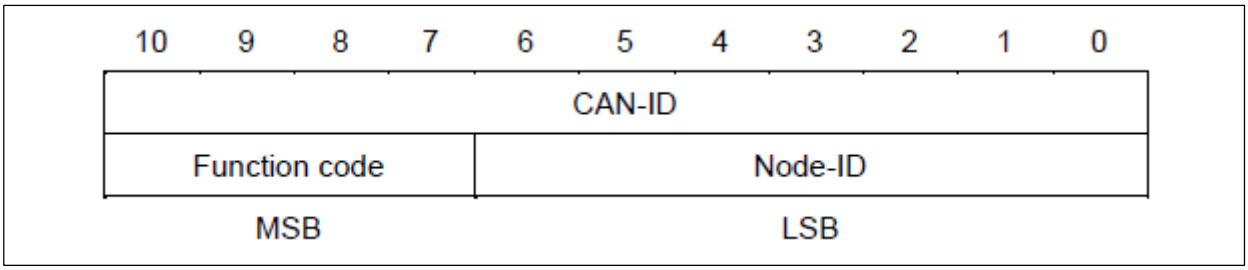
2、网络控制报文均采用数据最小字节数。比如心跳报文，只有1个字节数据。

3、实时更新的过程数据无需接收方报文应答。即采用生产消费模型，降低总线负载。

4、需要接收方确认的配置参数一般都是采用快速单字传输。即1个报文最多传送1个32字节的参数变量，避免了分帧引起的实时性降低。

以上这些定义都是为了节约时间开销，最大限度保证实时性。同时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帧ID）分配表，该分配表是基于11 位CAN-ID 的标准帧格式。将其划分为4 位的功能码（Function-ID）和7 位的节点号（Node-ID），如下图所示。

图 3. 预定义连接集 CAN-ID 分配方案设置



在CANopen里也通常把CAN-ID 称为COB-ID（通信对象编号）。所以我们可以分清楚两个易于混淆的名称：

- COB-ID：通信对象ID号，即CANopen中对某种通讯对象的报文帧ID，即CAN报文的11位ID。代表了一种通讯含义。
- Node-ID：节点ID号，即CANopen网络中的节点地址，CANopen规定了逻辑上最大128个节点（包含一个主机节点），所以Node-ID最大为127（7位）。

COB-ID和Node-ID无必然联系，但在点对点的通信中，COB-ID中包含了Node-ID。

CAN-ID分配方案（如下表）包括功能部分，它决定了对象的优先级和node-ID部分，它用于区分CANopen设备。这将允许在单一主机和多至127个从机间进行点对点通信。同时还支持无应答的NMT，SYNC和TIME广播消息。广播的Node-ID为零。

表 1. 通用预定义连接集的广播对象

COB	Function code	Resulting CAN-ID
NMT	0000b	0（000h）
SYNC	0001b	128（080h）
TIME	0010b	256（100h）

表 2. 通用预定义连接集的点对点对象

COB	Function code	Resulting CAN-ID
EMCY	0001b	129（001h）~ 255（0FFh）
PDO1（Tx）	0011b	385（181h）~ 511（1FFh）
PDO1（Rx）	0100b	513（201h）~ 639（27Fh）
PDO2（Tx）	0101b	641（281h）~ 767（2FFh）
PDO2（Rx）	0110b	769（301h）~ 895（37Fh）
PDO3（Tx）	0111b	897（381h）~ 1023（3FFh）
PDO3（Rx）	1000b	1025（401h）~ 1151（47Fh）
PDO4（Tx）	1001b	1153（481h）~ 1279（4FFh）
PDO4（Rx）	1010b	1281（501h）~ 1407（57Fh）
SDO（Tx）	1011b	1409（581h）~ 1535（5FFh）
SDO（Rx）	1100b	1537（601h）~ 1663（67Fh）
NMT error control	1110b	1793（701h）~ 1919（7FFh）

1.1.4 通信对象概述

通信对象被描述为服务和协议。

所有服务以表格形式给出，包含为服务定义的每个原始参数。特定的服务类型包含不同的参数（如无

应答、应答等等）。

下面针对通信对象作一个简单的介绍：

1、服务数据对象（SDO）

SDO提供了直接访问CANopen 设备对象字典的入口。入口条件包括数据类型和大小。SDO可用于客户和服务器之间传输多个数据集（每个包含任意大小数据块）。客户端通过寻址（对象字典的索引和子索引）来控制传输哪个数据集。数据集的内容定义在对象字典中。

SDO属于服务数据，有指定被接收节点的地址（Node-ID），并且需要指定的接收节点回应CAN报文来确认已经接收，如果超时没有确认，则发送节点将会重新发送原报文。这种通讯方式属于常见的“服务器客户端”的通信模型。

2、过程数据对象（PDO）

实时的数据传输通过“过程数据对象（PDO）”完成。PDO传输无协议开销。

由对象字典提供PDO数据和配置的接口。数据字典中对应的映射结构决定了一个PDO的数据类型和映射关系。如果CANopen设备支持可变映射PDO，则在配置过程中，可通过SDO实现对PDO在对象字典中对应的配置进行修改。

CANopen设备的PDO数量和长度可由应用规范、设备协议或应用协议指定。

PDO分两种用法，发送和接收，分别是Transmit-PDO（TPDO）和Receive-PDO（RPDO）。支持TPDO的CANopen设备成为PDO生产者，支持RPDO的称为PDO消费者。

PDO由PDO通讯参数和PDO映射参数描述。PDO通讯参数描述了PDO的通信功能。PDO映射参数包含了PDO传输内容信息。每个PDO的通信参数和映射参数都是必不可少的。

PDO属于过程数据，即单向传输，无需接收节点回应CAN报文来确认，从通讯术语上来说是属于“生产消费”模型。

3、同步对象（SYNC）

同步生产者定期广播同步对象。SYNC提供基本的网络同步机制。SYNC周期由标准通信循环周期参数定义，可由配置工具在CANopen设备引导过程中写入。

使用可选的计数器参数，可以在当前SYNC循环和PDO通信之间建立显示关系（详见协议中PDO通信参数：SYNC启动值）

为了保证及时的网络访问，SYNC的CAN-ID优先级较高。CANopen设备通过同步操作来同步自身与同步对象生产者。

4、时间戳对象（TIME）

TIME生产者广播时间戳对象。TIME提供了简单的网络时钟。

为了保证及时的网络访问，SYNC的CAN-ID优先级较高。CANopen设备通过时间戳对象生产者的TIME对象校准本地时间。

5、应急对象（EMCY）

EMCY由CANopen设备的内部错误触发并且由CANopen设备应急生产者对象负责发起。应急对象适合中断类型的错误报警。一个“错误事件”仅触发一次应急对象通信。CANopen设备无新错误不会再产生应急对象。

6、网络管理（NMT）

CANopen设备网络管理（NMT）为主从结构模型。NMT对象用于执行NMT服务。通过NMT服务，CANopen设备才能够执行初始化、启动、监控、复位或停止等行为。所有的CANopen设备

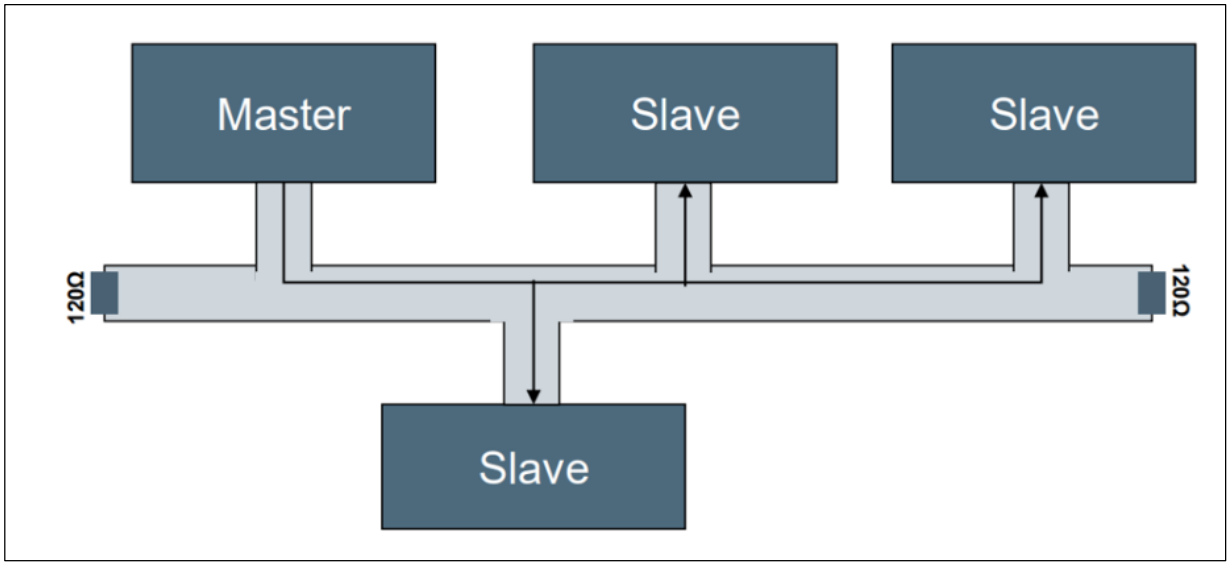
被视为NMT从站。NMT从站使用Node-ID标识其身份，取值范围[1 ~ 127]。

网络中须有一个CANopen设备满足NMT主站功能需求。本案例中会用到上位机软件CANopen Device Explorer通过连接到PCAN-USB适配器，来作为CANopen网络的一个主站。

1.1.5 网络管理和系统启动

网络管理采用的主从方式，主机节点可以向从节点发送和请求数据。同一个网络中只能有一个主节点，一个或多个从节点，如下图所示。

图 4. NMT 结构图



网络管理涉及到一个CANopen节点从上电开始的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。在本案列中，会直接操作MCU复位。

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。在本案列中，会操作与通信相关的CAN、TMR复位。

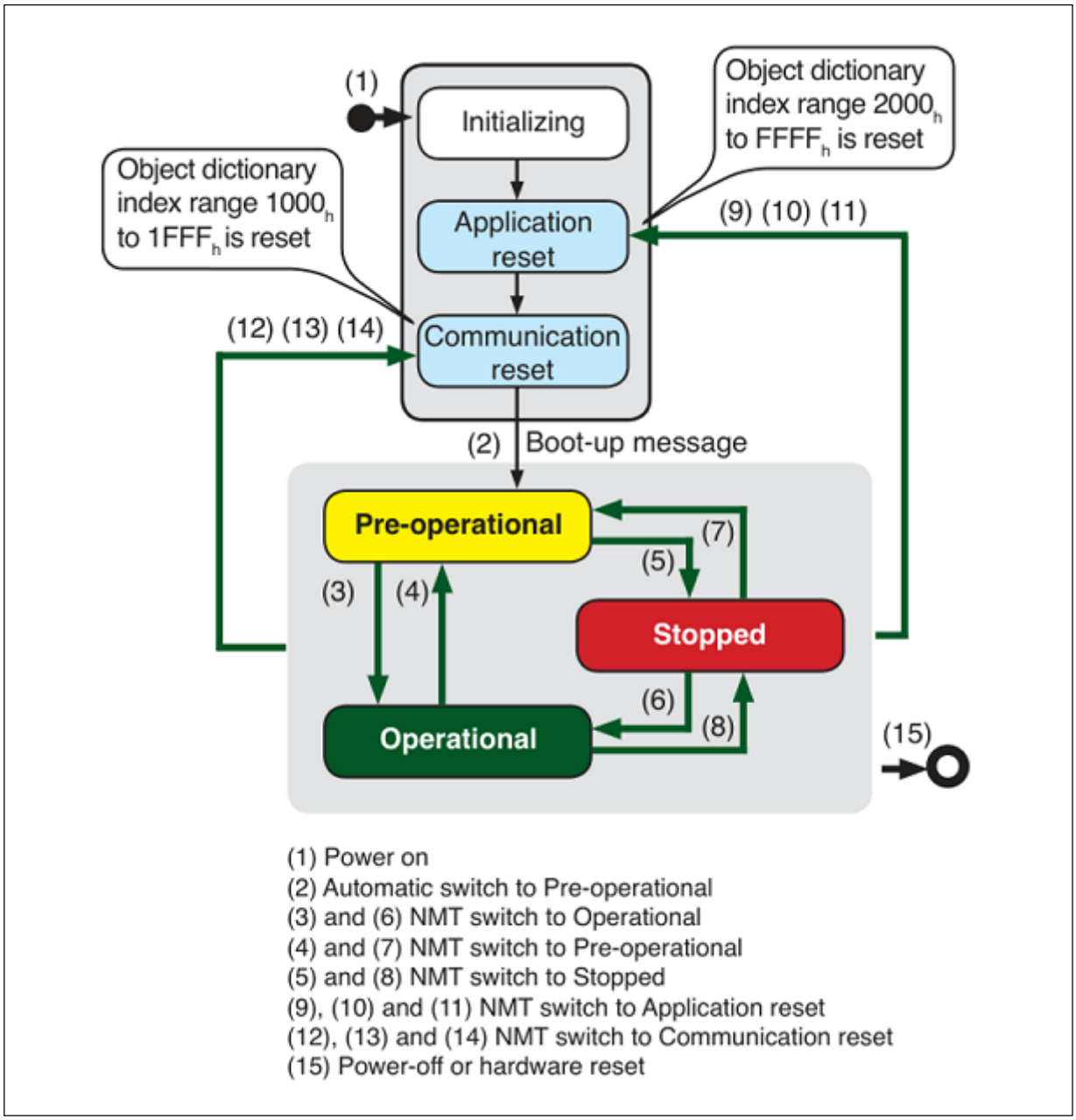
4、预操作状态（Pre-operational）：节点的CANopen通讯处于操作就绪状态，此时此节点不能进行PDO通信，而可以进行SDO进行参数配置和NMT网络管理的操作。

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主机通过NMT命令可以让网络中任意一个的CANopen从节点进行其他5种状态的切换，如下图所示。在本案例中，CANopen从站在初始化成功后会直接进入到操作状态（operational）。

图 5. NMT 状态转换图



下表中规定了CANopen节点状态与通信对象之间的关系。列举的通信对象中的服务只在CANopen设备处在特定的状态下才能被执行。

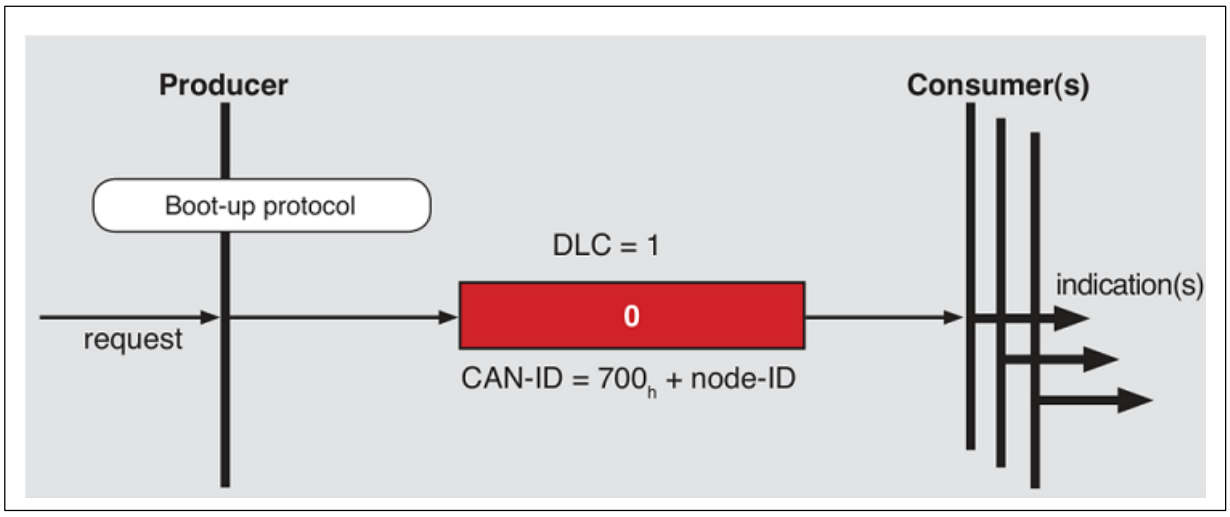
表 3. 通信对象和 NMT 状态关系

Communication Objects	Pre-operational	Operational	Stopped
PDO		√	
SDO	√	√	
SYNC	√	√	
TIME	√	√	
EMCY	√	√	
Node control and error control	√	√	√

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 message），如下图所示，节点上线报

文的ID为700h+Node-ID，数据为1个字节0。生产者为CANopen从站。

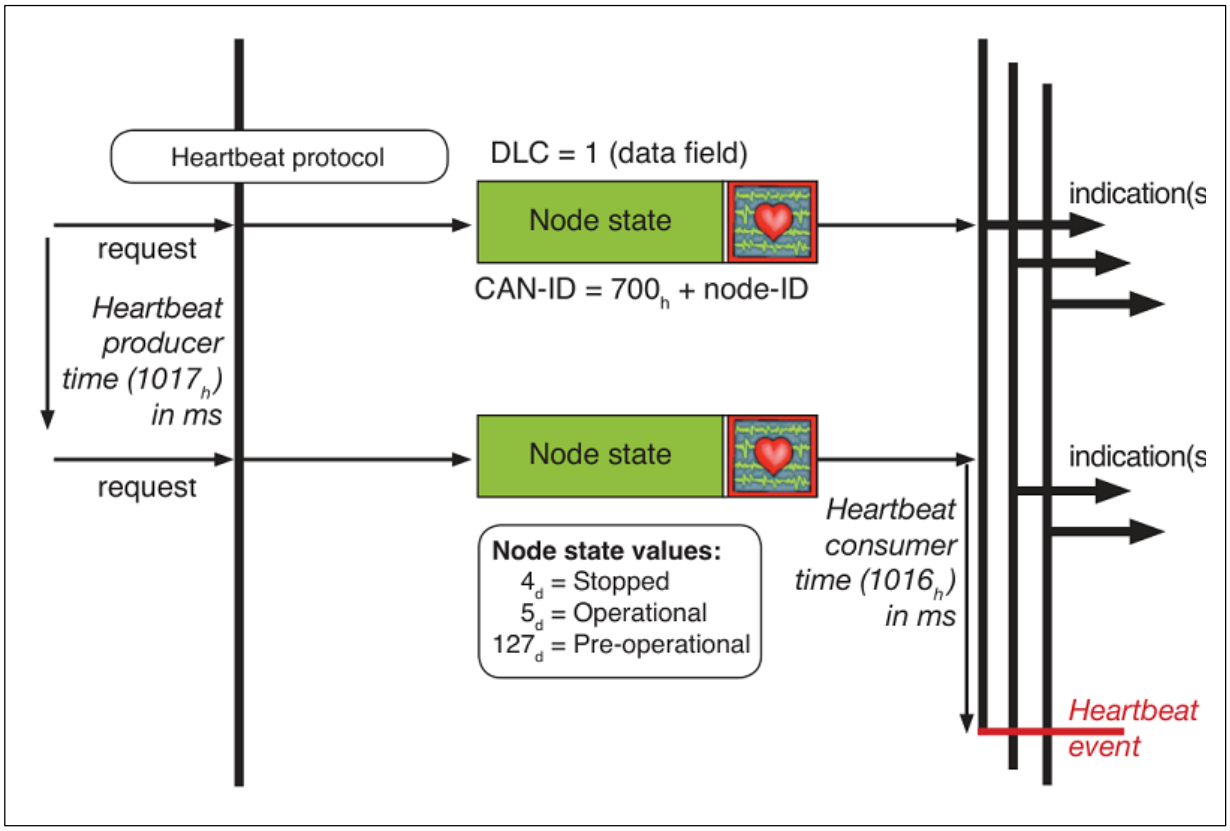
图 6. 节点上线报文



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。

如下图所示，为心跳报文发送的格式，CAN-ID与节点上线报文相同为700h+Node-ID，数据为1个字节，代表节点目前的状态，04h为停止状态，05h为操作状态，7Fh为预操作状态。

图 7. 节点状态与心跳报文



NMT网络管理中，最核心的就是NMT节点状态切换命令，这是NMT主站所进行网络管理的“命令”报文。使用者必须牢记这些命令。

CANID均为000h，具备最高的CAN优先级。数据为2个字节：

- 第1个字节代表命令类型：

01h为启动命令（让节点进入操作状态）；

02h为停止命令（让节点进入停止状态）；

80h为进入预操作状态（让节点进入预操作状态）；

81h为复位节点应用层（让节点的应用恢复初始状态，本案例中会直接复位MCU）；

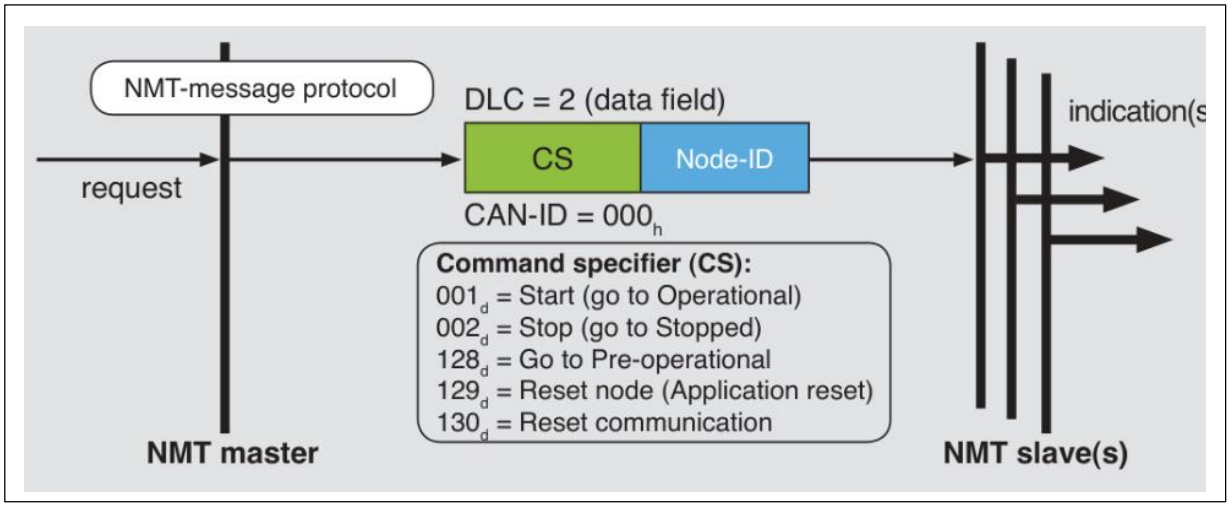
82h为复位节点通讯（让节点的CAN和CANopen通讯重新初始化，一般用于总线收到干扰，导致节点总线错误被动，或者总线关闭时，本案例中会复位CAN、TMR等IP）。

- 第2个字节代表被控制的节点Node-ID。

如果要对整个网络所有节点同时进行控制，则这个数值为0即可。

如下图所示：

图 8. 节点状态切换命令



1.1.6 对象字典概述

对象字典（OD: Object Dictionary）是CANopen协议最为核心的概念。所谓的对象字典就是一个有序的对象组，描述了对应CANopen节点的所有参数，包括通讯数据的存放位置也列入其索引，这个表变成可以传递形式就叫做EDS文件（电子数据文档Electronic Data Sheet）。

每个接口对象通过一个16位的索引和一个8位的子索引寻址。

每个CANopen设备都有一个对象字典，使用电子数据文档（EDS文件）来记录这些参数。

CANopen对象字典中的项由一系列子协议来描述。子协议为对象字典中的每个对象都描述了它的功能、名字、索引、子索引、数据类型，以及这个对象是否必需、读写属性等等，这样可保证不同厂商的同类型设备兼容。

CANopen协议的核心描述协议是DS301，其包括了CANopen协议应用层及通信结构描述，其它的子协议都是对DS301 协议描述文本的补充与扩展。在不同的应用行业都会起草一份CANopen设备子协议，子协议编号一般是DS4xx。

对象字典容量65536（0000h ~ FFFFh），每个对象可包含多至256个子项（00h ~ FFh）。

如下表所示，为对象字典索引区域定义，其中通讯对象协议区（Communication profile area）和制造商特定协议区（Manufacturer-specific profile area）是用户需要关注的区域。

表 4. 对象字典总述

Index range	Description
0000h	Reserved
0001h to 025Fh	Data types
0260h to 0FFFh	Reserved
1000h to 1FFFh	Communication profile area
2000h to 5FFFh	Manufacturer-specific profile area
6000h to 9FFFh	Standardized profile area
A000h to AFFFh	Network variables
B000h to BFFFh	System variables
C000h to FFFFh	Reserved

这里只列举几个较常用的对象字典索引区域作一个简单的介绍：

1、索引范围（1000h ~ 1FFFh）：通讯协议区（Communication profile area）

其中索引范围（1000h ~ 1029h）为通用通讯对象，所有CANopen节点都必须具备这些索引，否则将无法加入CANopen网络。其他索引根据实际情况进行分配与定义，如下表所示。

表 5. 通讯对象协议区

Index range	Description
1000h to 1029h	General communication objects
1200h to 12FFh	SDO parameter objects
1300h to 13FFh	CANopen safety objects
1400h to 1BFFh	PDO parameter objects
1F00h to 1F11h	SDO manager objects
1F20h to 1F27h	Configuration manager objects
1F50h to 1F54h	Program control object
1F80h to 1F89h	NMT master objects

2、索引范围（2000h ~ 5FFFh）：制造商指定协议区（Manufacturer-specific profile area）

通常是存放厂商定制功能所需的数据。而上文所描述的通讯协议区（Communication profile area）是存放这些应用数据的通信参数。

例如，本例程中定义了：

- RPDO的通信参数存放在（1400h ~ 15FFh），映射参数存放在（1600h ~ 17FFh）且该参数中所存放的数据指向索引为2600h 之后的厂商自定义区；
- TPDO的通信参数存放在（1800h ~ 19FFh），映射参数存放在（1A00h ~ 1BFFh）且该参数中所存放的数据指向索引为2A00h 之后的厂商自定义区。

3、索引范围（6000h ~ 9FFFh）：标准化设备协议区（Standardized profile area）

标准化设备子协议，为各种行业不同类型的标准设备定义对象字典中的对象。目前已有十几种为不同类型的设备定义的子协议，例如DS401、DS402、DS406等。同样，这个区域对于不同的标准化设备子协议来说，相同的对象字典项其定义不一定相同。这部分本文不做叙述。

以上是关于CANopen的基础协议的简单介绍，详细信息请参见CiA 301（CANopen应用层和通讯描述协议）或访问官网：https://www.can-cia.org/canopen/。

1.2 关于 CANopenNode 协议栈

CANopenNode是免费开源的CANopen协议栈。

CANopen是建立在CAN基础上的用于嵌入式控制系统的国际标准化(EN 50325-4) (CiA301)高层协议。CANopenNode是用ANSI C语言以面向对象的方式编写的。它运行在不同的微控制器上，作为裸机应用或带RTOS应用。通信变量、设备变量、自定义变量收集在CANopen对象字典中，可以从C代码和CANopen网络访问。

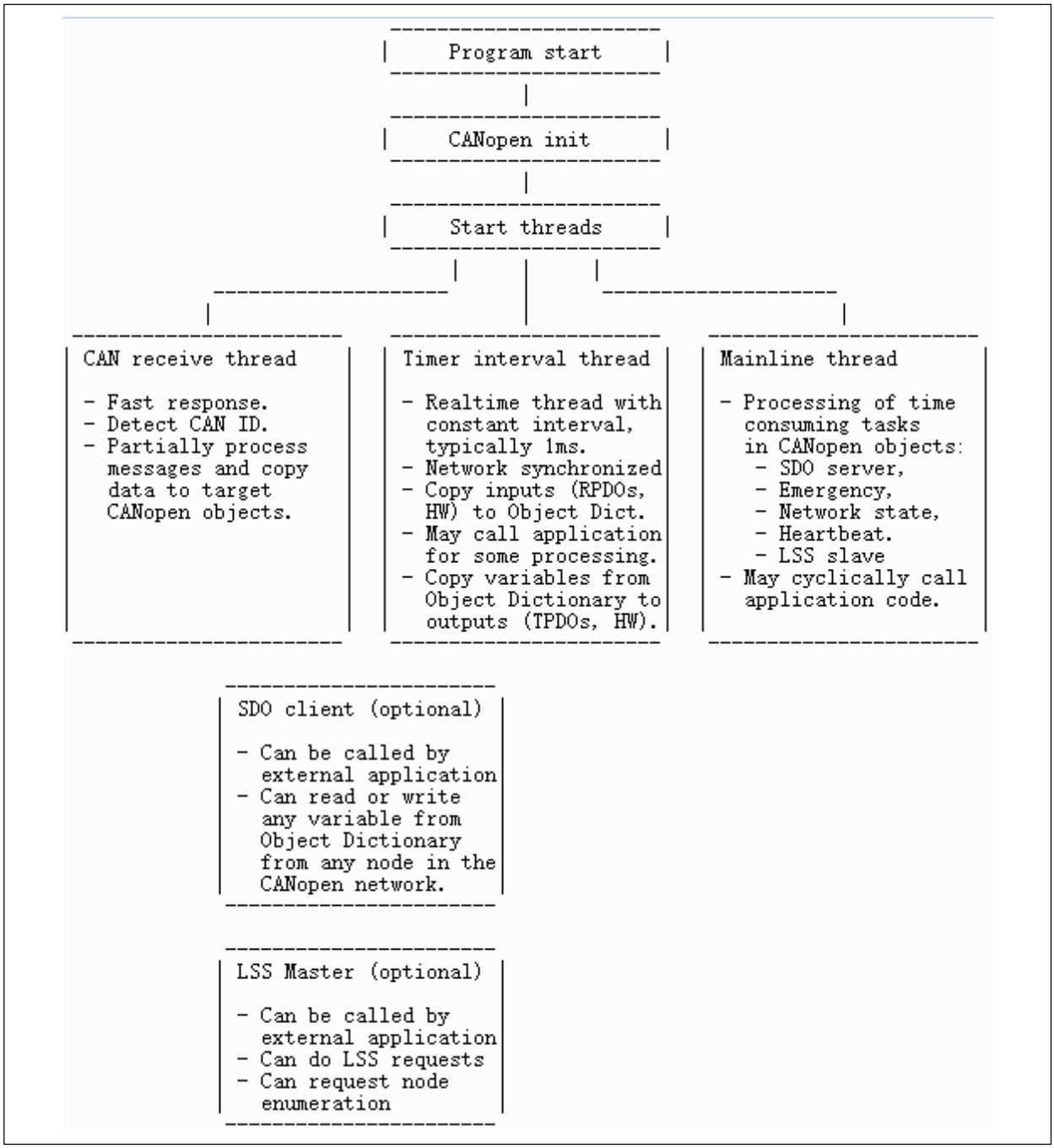
CANopenNode的主页为https://github.com/CANopenNode/CANopenNode。

主要特征如下：

- Object Dictionary 提供清晰和灵活的组织任何变量。变量可以直接访问或通过读/写函数访问。
- NMT 从机可启动、停止、复位设备。以及简单的 NMT 主机。
- Heartbeat 用于监控 CANopen 设备的心跳生产者/消费者错误控制。
- PDO 用于广播处理数据，具有高优先级，没有协议开销。对象字典中的变量可以动态映射到TPDO上，TPDO按照通信规则进行传输，并被另一台设备作为RPDO接收。
- SDO 服务器支持对CANopen设备内所有对象字典变量的快速、分段和块传输访问。
- SDO 客户机可以访问网络内任何CANopen设备上的任何对象字典变量。
- Emergency 消息生产者/使用者。
- Sync 生产者/消费者允许网络同步传输PDO对象等等。
- Time-stamp 生产者/消费者支持以毫秒分辨率进行日期和时间同步。
- LSS CANopen 节点ID号和比特率设置，主站和从站，LSS 快速扫描。
- CANopen gateway 用于 NMT主站、LSS主站和 SDO客户端的 CiA309-3 Ascii 命令接口。
- CANopen Safety, EN 50325-5, CiA304，安全相关网络中的“类PDO”通信。

CANopenNode协议栈实现流程图如下：

图 9. CANopenNode 实现流程图



CANopenNode协议栈文件结构如下：

- 301/ - CANopen 应用层和通信配置文件

  CO_config.h - CANopenNode 的配置宏。

  CO_driver.h - CAN 硬件和 CANopenNode 之间的接口。

  CO_ODinterface.h/.c - CANopen 对象字典接口。

  CO_Emergency.h/.c - CANopen 紧急协议。

  CO_HBconsumer.h/.c - CANopen 心跳消费者协议。

  CO_NMT_Heartbeat.h/.c - CANopen 网络管理和心跳生产者协议。

  CO_PDO.h/.c - CANopen 过程数据对象协议。

CO_SDOclient.h/.c - CANopen 服务数据对象 - 客户端协议（主站功能）。

CO_SDOserver.h/.c - CANopen 服务数据对象 - 服务器协议。

CO_SYNC.h/.c - CANopen 同步协议（生产者和消费者）。

CO_TIME.h/.c - CANopen 时间戳协议。

CO_fifo.h/.c - 用于 SDO 和网关数据传输的 Fifo 缓冲区。

crc16-ccitt.h/.c - CRC 16 CCITT 多项式的计算。

- 303/ - CANopen 推荐

  CO_LEDs.h/.c - CANopen LED 指示灯

- 304/ - CANopen 安全

  CO_SRDO.h/.c - CANopen 安全相关数据对象协议。

  CO_GFC.h/.c - CANopen 全局故障安全命令（生产者和消费者）。

- 305/ - CANopen 层设置服务 (LSS) 和协议

  CO_LSS.h - CANopen 层设置服务协议（通用）。

  CO_LSSmaster.h/.c - CANopen 层设置服务 - 主协议。

  CO_LSSslave.h/.c - CANopen 层设置服务 - 从站协议。

- 309/ - 从其他网络访问 CANopen

  CO_gateway_ascii.h/.c - Ascii 映射：NMT 主站、LSS 主站、SDO 客户端。

- storage/ - 存储

  CO_storage.h/.c - CANopen 数据存储基础对象。

  CO_storageEeprom.h/.c - CANopen 数据存储对象，用于将数据存储到块设备 (eeprom) 中。

  CO_eeprom.h - 与 CO_storageEeprom 一起使用的 eeprom 接口，功能是特定于目标系统的。

- extra/ - 附加项

  CO_trace.h/.c - 用于随时间记录变量的 CANopen 跟踪对象。

- example/ - 包含基本示例的目录，应在任何系统上编译

  CO_driver_target.h - CANopenNode 的示例硬件定义。

  CO_driver_blank.c - CANopenNode 的示例空白接口。

  main_blank.c - 主线和其他线程 - 示例模板。

  CO_storageBlank.h/.c - 数据存储到非易失性存储器的示例空白演示。

  Makefile – Makefile for example。

  DS301_profile.xpd - DS301 的 CANopen 设备描述文件。 它还包括 CANopenNode 的特定属性。 该文件在对象字典编辑器的配置文件中也可用。

  DS301_profile.eds、DS301_profile.md - 标准 CANopen EDS 文件和降价文档文件，从 DS301_profile.xpd 自动生成。

  OD.h/.c - CANopen 对象字典源文件，从 DS301_profile.xpd 自动生成。

- doc/ - 文档目录

  CHANGELOG.md - 更改日志文件。

deviceSupport.md - 有关受支持设备的信息。

objectDictionary.md - CANopen 对象字典接口的描述。

CANopenNode.png - 小图标。

html - 带有文档的目录 - 必须由 Doxygen 生成。

- CANopen.h/.c - CANopen 对象的初始化和处理，适用于通用配置
- Doxyfile - 文档生成器 doxygen 的配置文件
- LICENSE - 许可证
- README.md - 自述文件

本应用指南将介绍如何在AT32M412单片机上，通过CANopenNode协议栈实现CANopen从机节点的主要功能，并提供基于AT32M412_416_StdPeriph_Lib和CANopenNode协议栈的源代码。如结合AT32-Comm-EV Board和AT-START Board可以方便快速的搭建起基于CAN2.0网络的CANopen从机节点。

1.3 关于 CANopen 对象字典编辑软件

对象字典是 CANopen 最重要的部分之一。要自定义对象字典，必须使用外部应用程序：CANopenEditor。它的主页是https://github.com/CANopenNode/CANopenEditor。有提供了最新的预编译二进制文件。只需提取 zip 文件并运行 EDSEditor.exe。 在 Linux 中，它使用 mono 运行，这在 Ubuntu 上默认可用。只需将文件权限设置为“可执行”，然后执行程序即可。

CANopen对象字典编辑器主要有以下3点特征：

1、 可导入：CANopen的电子数据文档（EDS或XDD的文件格式）。

2、 可导出：CANopen的电子数据文档（EDS或XDD的文件格式）、文件资料、CANopenNode协议栈的C源文件。

3、 为CANopen的对象字典、设备信息等提供图形化的编辑界面。

在程序中，在首选项中，将导出器设置为“CANopenNode_V4”。然后开始新项目或打开现有的项目文件。

支持许多项目文件类型，EDS、XDD v1.0、XDD v1.1、旧的自定义 XML 格式。 然后可以将生成的项目文件保存为 XDD v1.1 文件格式 (xmlns=“http://www.canopen.org/xml/1.1”)。项目文件也可以导出为其他格式，可用于生成对象字典的文档（EDS文件）和CANopenNode源文件（OD.c/.h文件）。

如果启动了新项目，则可能会插入 DS301_profile.xpd。如果编辑现有（旧）项目，则可能会删除现有的通信特定参数，然后可能会插入新的 DS301_profile.xpd。另一种方法是通过观察 objectDictionary.md 中 CANopenNode 的对象字典要求来编辑现有的通信参数。

要克隆、添加或删除，请选择对象并使用右键单击。正确设置自定义对象字典需要一些 CANopen 知识。也可以从另一个项目中插入单独的对象。

CANopenNode 在标准项目文件中包含一些自定义属性。有关详细信息，请参阅 objectDictionary.md。

本应用指南中将下载CANopenEditor在PC机上运行EDSEditor.exe，来编辑CANopen demo的对象字典，并由同样的参数配置生成“EDS文件”和“OD.c/.h文件”，分别供“上位机调试软件”和“CANopen协议栈”使用。

1.4 关于 CANopen DeviceExplorer 调试软件

CANopen DeviceExplorer是德国emotas公司推出的一款用于开发、测试、诊断等服务的多功能工具。它提供CANopen主站功能性并且可分析和配置CANopen设备。每个CANopen设备的有关信息可从设备 (EDS或XDD格式)的电子表格中读取，或者直接从设备上扫描获取。利用标准化设备配置文件 (DCF) 可保存和导入。

另外整个CANopen网络数据可以保存到一个项目文件中。基于QtScript设计的内建脚本功能为用户提供创建自定义测试、控制应用等可能性，可以大大提高效率。

CAN interpreter 包含在标准版本内，作为CAN分析工具。它可实时监视CAN总线并且支持启动和诊断CAN网络。用户能够看到报文传输到CAN总线为原始CAN报文（CANopen通信第2层）或者解释为CANopen报文（CANopen通信第7层）。可选记录数据传输可以后脱线分析。

本应用指南中将下载安装CANopen DeviceExplorer在PC机上连接PCAN-USB适配器作为CANopen的NMT主机，与AT-START Board（作为CANopen从机）通过AT32-Comm-EV的收发器模块相连，实现一个完整的、可测试的CANopen的基本通信网络。

注意：

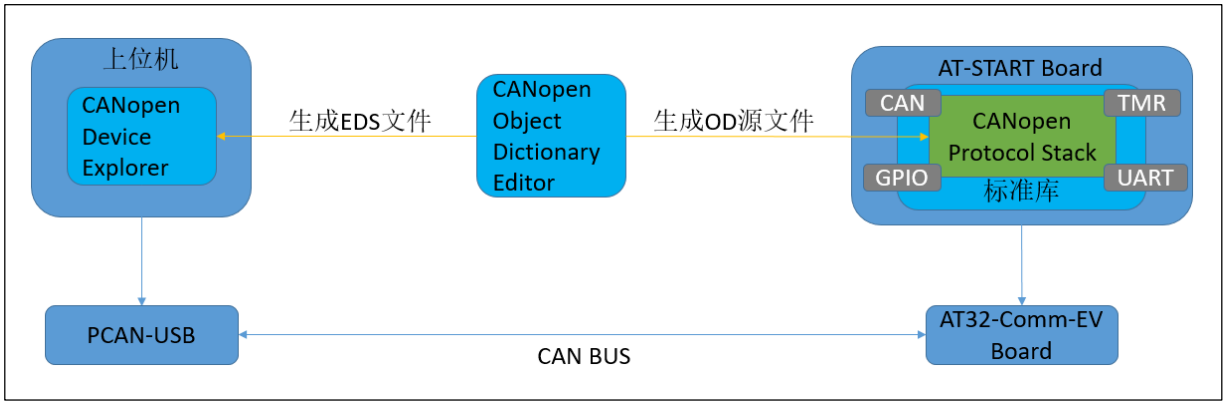
在本案例中，我们使用的是CANopen DeviceExplorer软件的一个评估版本。因该版本有诸多限制，即使用一次的许可时间仅一个小时；波特率固定为125 kBit/s；CANopen的节点ID号仅有1、32和64可使用。鉴于此，我们CANopen从站的demo中波特率配置为125 kBit/s；节点ID号固定为1。

2 AT32 硬件准备

硬件主要由PCAN-USB适配器、AT32-Comm-EV Board和AT-START Board组成。

本应用指南提供的demo使用到的外设资源有CAN、TMR、USART、GPIO等，用户可根据具体需要灵活配置和修改，使用AT32-Comm-EV Board的收发器与PCAN-USB适配器相连来实现CANopen网络的物理层。

图 10. AT32 CANopen 结构原理图

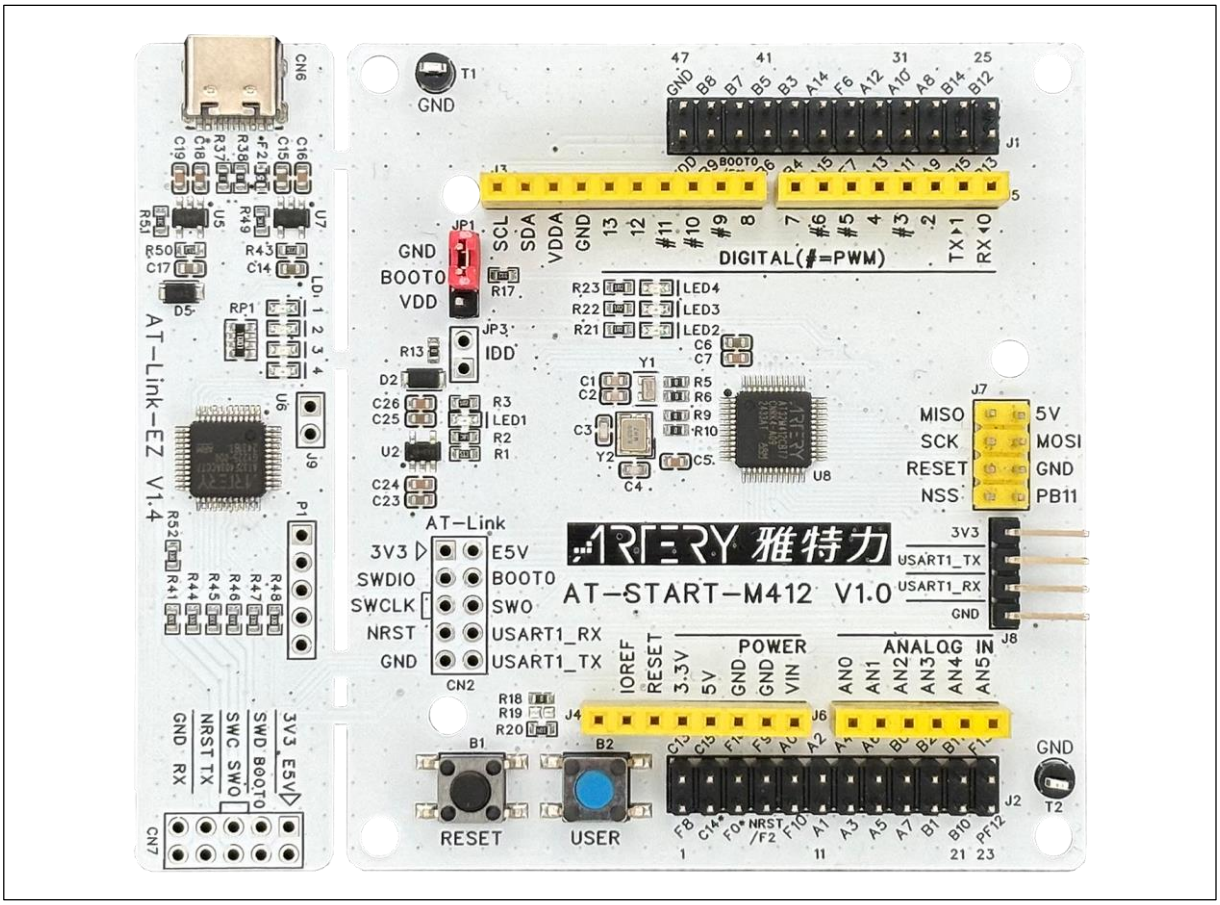


另外，这里还需要使用CANopen对象字典编辑器，来自定义对象字典，比如本案例中有插入新的DS301_profile.xpd 文件所定义的对象，并有新定义PDO映射参数所寻址到的对象。在CANopen的对象字典及参数都配置完成后，再分别生成文件，供上位机软件和AT32工程的协议栈使用。

- AT-START Board

当前提供例程基于AT-START-M412。可提供基于CAN2.0的通信。

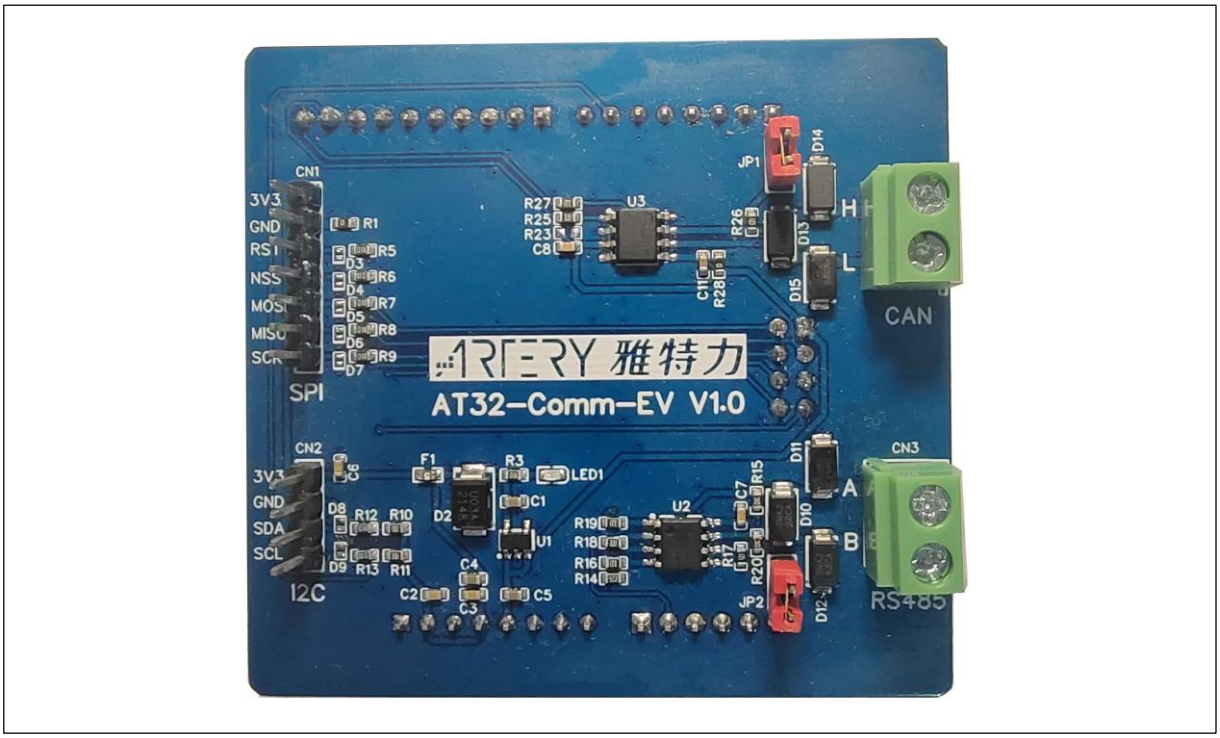
图 11. AT-START-M412 实验板



■ AT32-Comm-EV Board

可提供例程基于CAN2.0的CANopen通信支持。

图 12. AT32-Comm-EV 转接板



3 将CANopenNode移植到AT32上

3.1 基础工程准备

下载最新版本BSP&PACK文件，按照其应用指南进行安装及配置，本文档及例程均基于AT32xxx_Firmware_Library_V2.x.x的BSP&PACK文件进行开发。可借用at_start_m412文件夹下的temple工程来进行修改，更改文件夹及工程名为canopen_for_at32，并准备在该工程内添加CANopenNode源码。

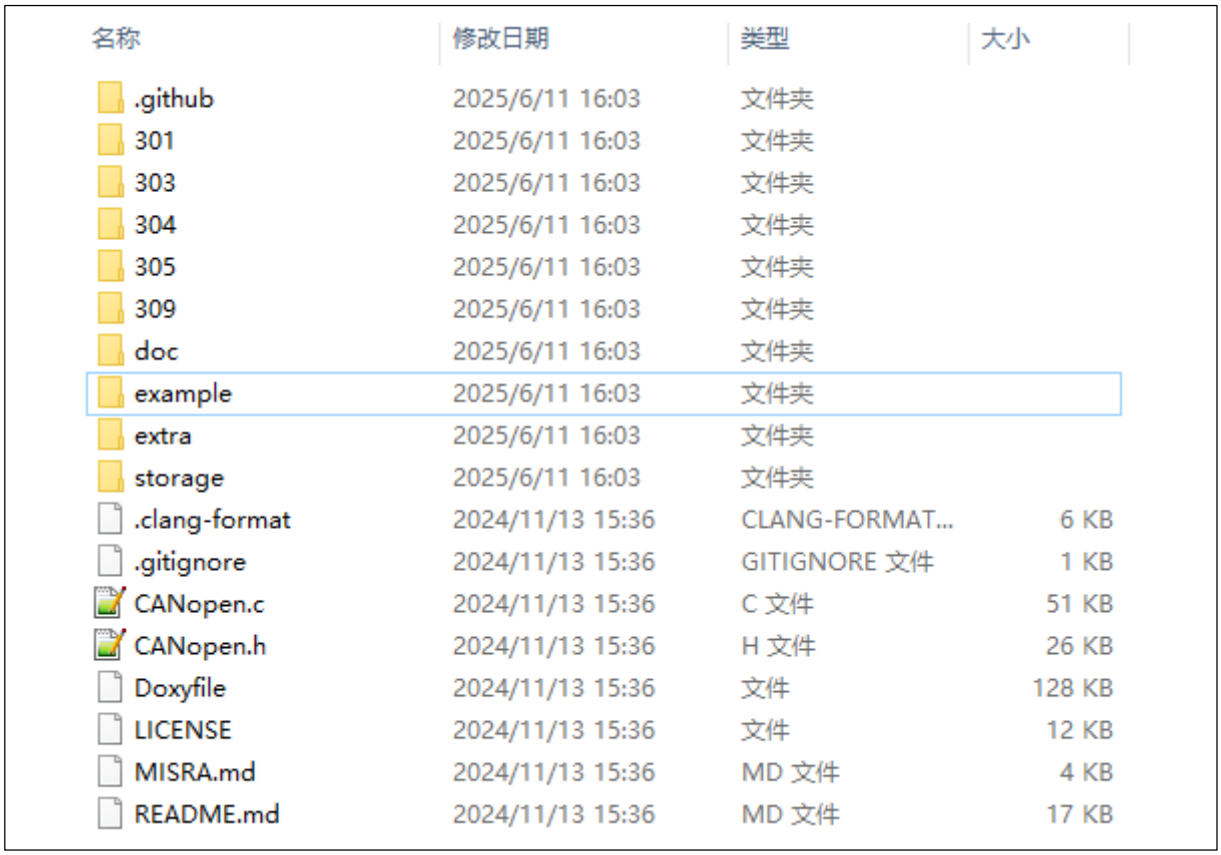
本案例中有提供AT32 IDE和Keil MDK工程的例程，下面将以Keil MDK工程为例加以说明。

3.2 工程内添加 CANopenNode 源码

用户需前往CANopenNode官网即Github上下载最新版源码。源码包解压后，内有如下文件。

本文及所有移植例程均基于CANopenNode-v4版本进行移植。

图 13. CANopenNode 源码文件



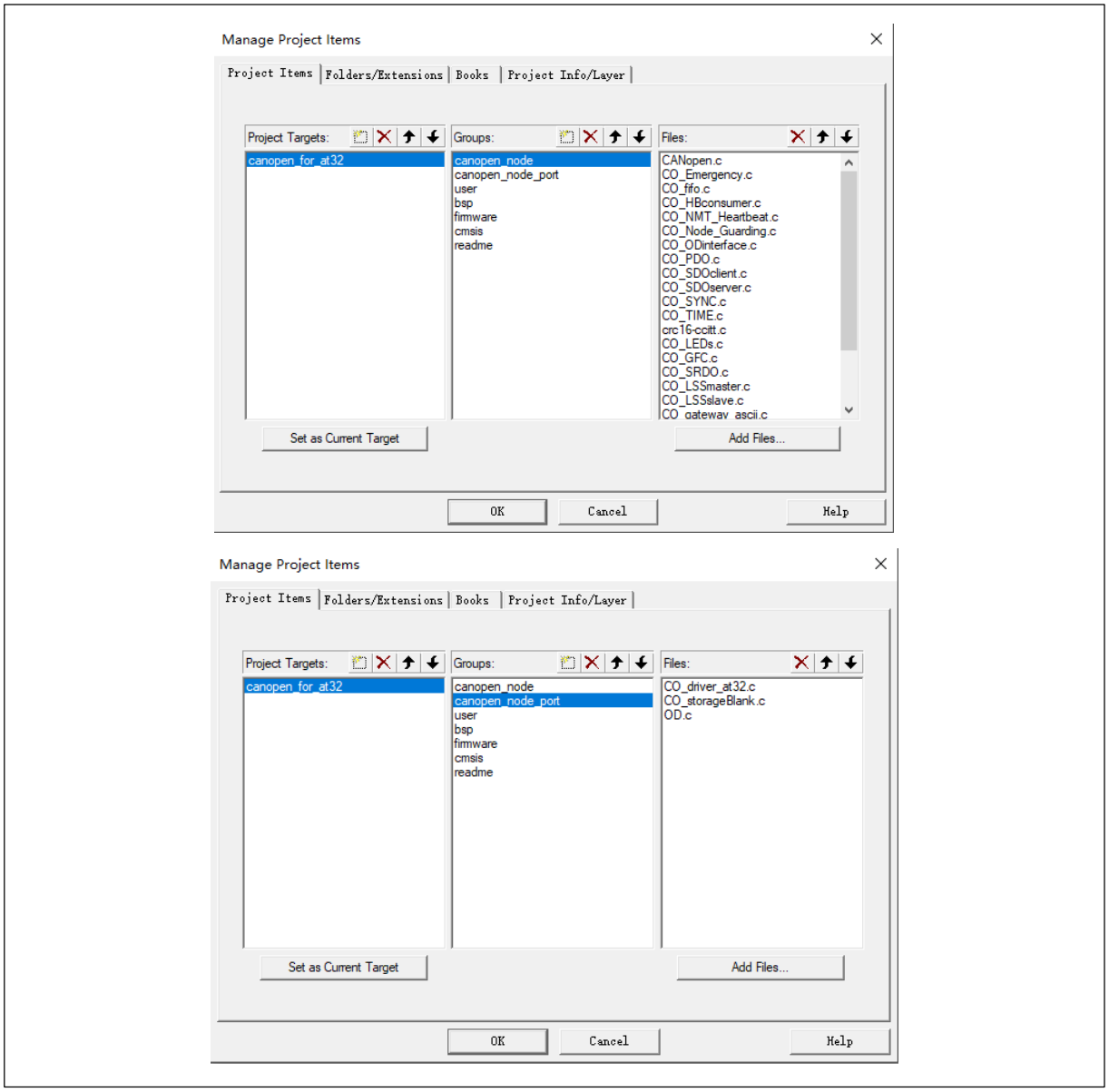
将源码包解压后，复制CANopenNode文件夹和CANopenNode\example文件夹到前面的基础工程canopen_for_at32目录下面，将CANopenNode文件夹改名为canopen_node，再将example文件夹改名为canopen_node_port，该文件夹里的文件有重命名，具体参考本案例例程。如下图所示。

图 14. canopen_for_at32 工程目录

打开工程文件，并按以下2个步骤添加到工程文件内。可参考AT提供的例程文件进行添加。

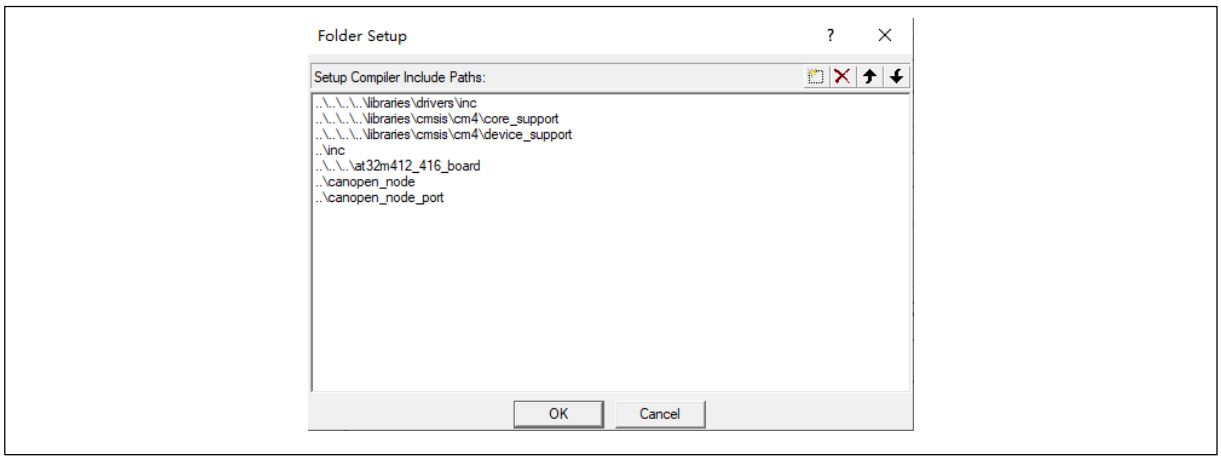
1、添加canopen_node和canopen_node_port内的所有.c文件到工程项目内。

图 15. canopen_for_at32 工程的项目



2、需将添加的.c文件所对应的.h文件的路径添加到工程的文件夹设置内。

图 16. canopen_for_at32 工程的文件夹设置



3.3 工程代码的修改

1、修改“CO_driver_target.h”文件。在该文件中添加包含关于AT32 MCU的“at32m412_416_board.h”和“at32m412_416_clock.h”头文件。补全用于互斥操作的开关中断宏定义。声明重定义的CAN中断回调函数。

2、修改“CO_driver_blank.c”并重命名为“CO_driver_at32.c”文件。在该文件中添加关于CAN和TMR外设的底层驱动部分代码。用户可根据自己的硬件环境来自行修改，包括CAN参数的初始化配置、CAN的收发调用、TM R的定时初始化配置等等，具体可参考AT所提供的例程文件。

3、修改“main_blank.c”并重命名为“main.c”文件。在该文件中添加BSP中关于MCU的基本初始化配置代码，包括时钟、串口打印、LED、IOMUX、NVIC的初始化配置等等，具体可参考AT所提供的例程文件。

4、修改“startup_at32m412_416.s”文件将“Heap_Size”的值设置为“0x00001600”，在Keil工程中必须选择AC6来编译代码。

3.4 对象字典的生成

接下来就是要修改与对象字典相关的“OD.c”和“OD.h”文件，以满足实际应用的需要，并将与此对象字典参数配置一致的“EDS”文件导入到上位机软件CANopen DeviceExplorer中。

本案例中将借用CANopenEditor工具来配置对象字典及参数，须在同样的应用配置下，生成“EDS文件”和“OD.c/.h文件”，供上位机软件和AT32工程的协议栈使用。

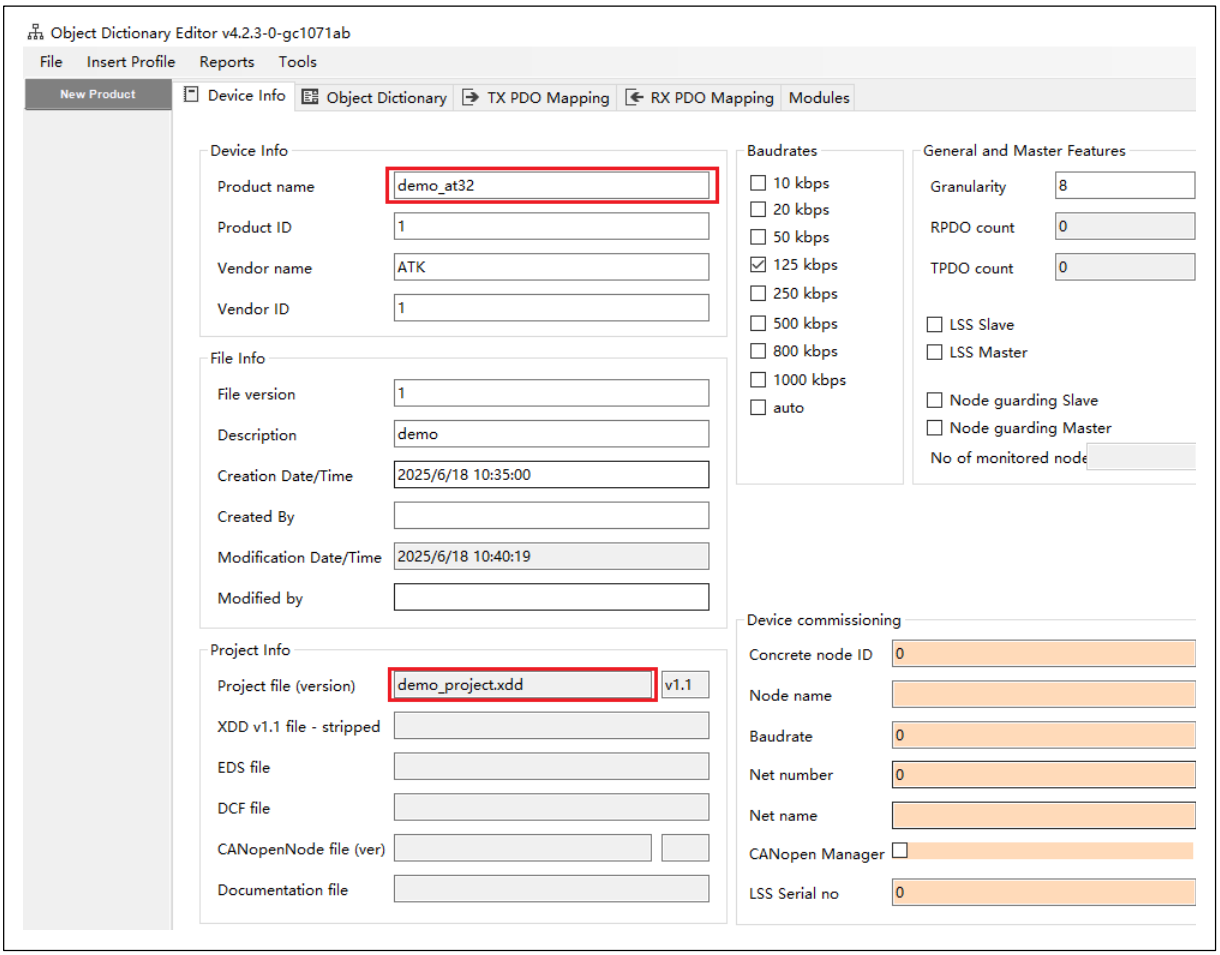
操作流程如下：

1、提取 CANopenEditor-v4.2.3-binary.zip 文件并运行 EDSEditor.exe。新建一个工程后，再在“Device Info”选择框中将产品名重命令为demo_at32，其他选项可根据需要自行修改。

具体流程：

单击“File”→ “New”→ 在“Product name”中修改为“New Product”为“demo_at32”→ 将该工程保存为“demo_project.xdd”。

图 17. 新建对象字典工程

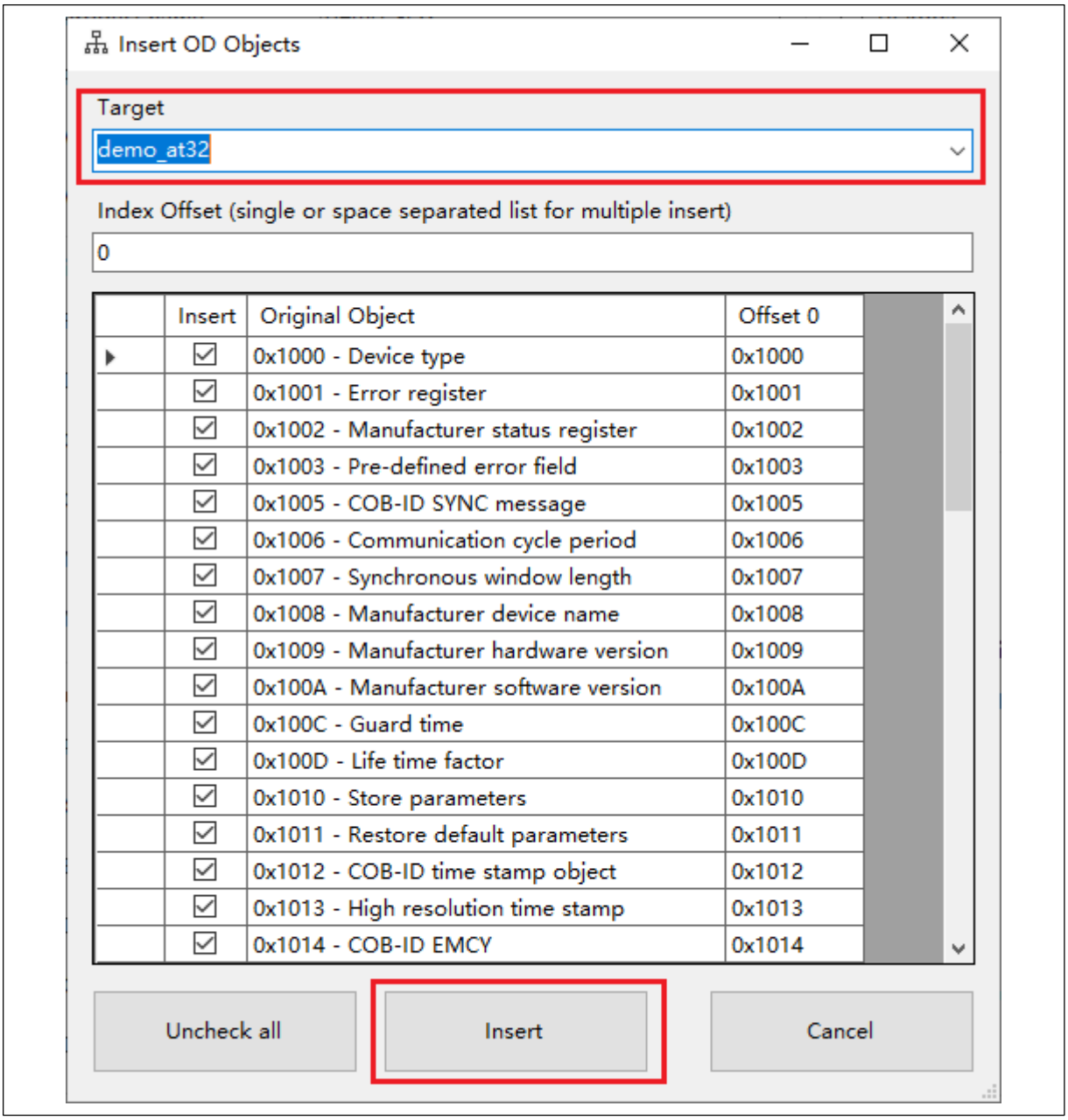


2、 需再插入一个协议的配置文件，这里我们选择插入DS301_profile.xpd（即DS301协议所默认定义的配置文件），再选择刚建立的目标工程demo_at32，其他参数可保持默认值。

具体流程：

单击“Insert Profile”→ 选择“DS301_profile.xpd”→ 弹出“Insert OD Objects”对话框中 →“Target”中选择“demo_at32”工程 → 单击“Insert”。

图 18. 插入对象对话框

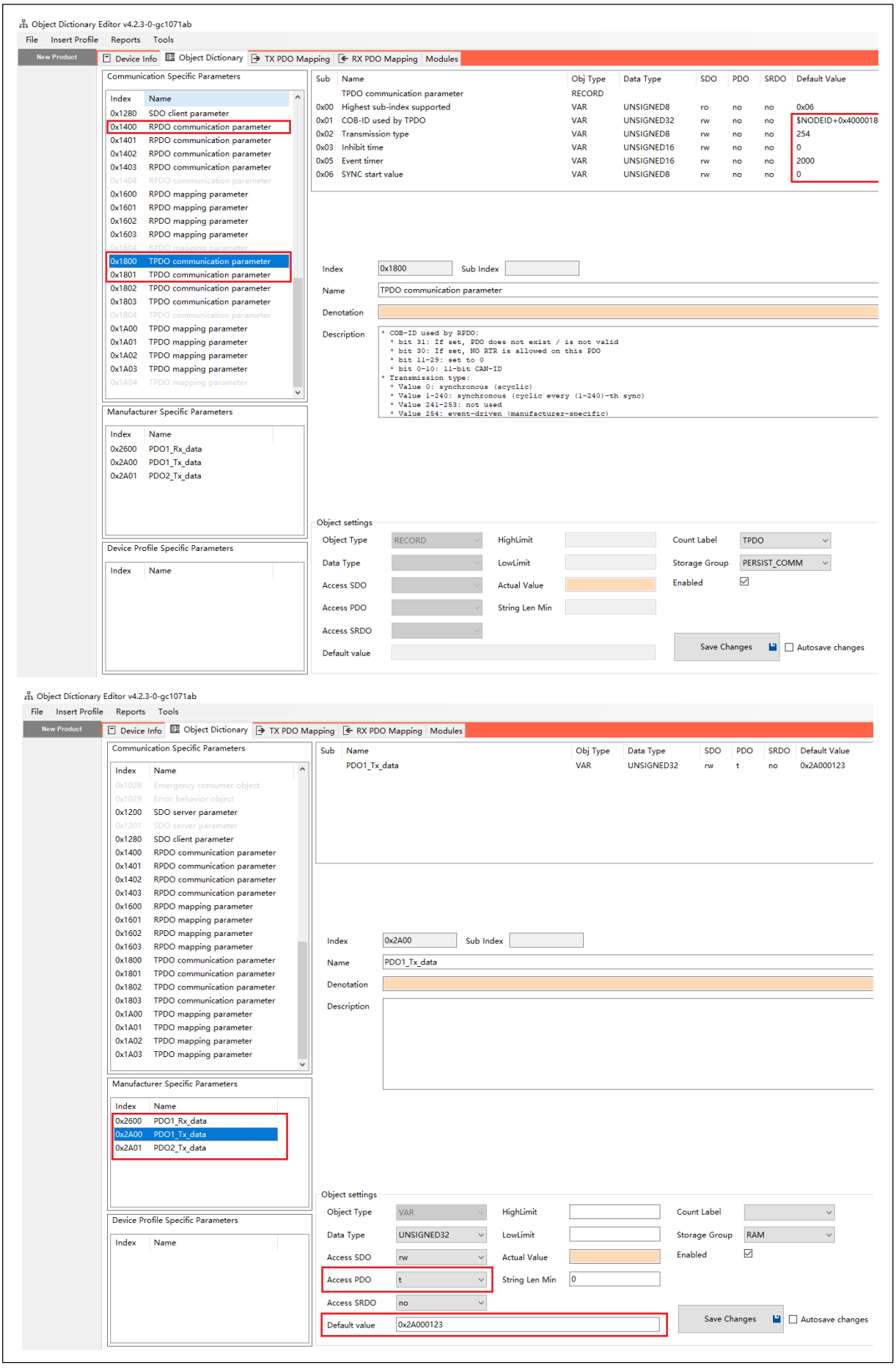


3、 在“Object Dictionary”窗口中需自行配置RPDO和TPDO的通信参数，再在“Manufacturer Specific Parameters”中需新建映射参数所指向的对象。

具体流程：

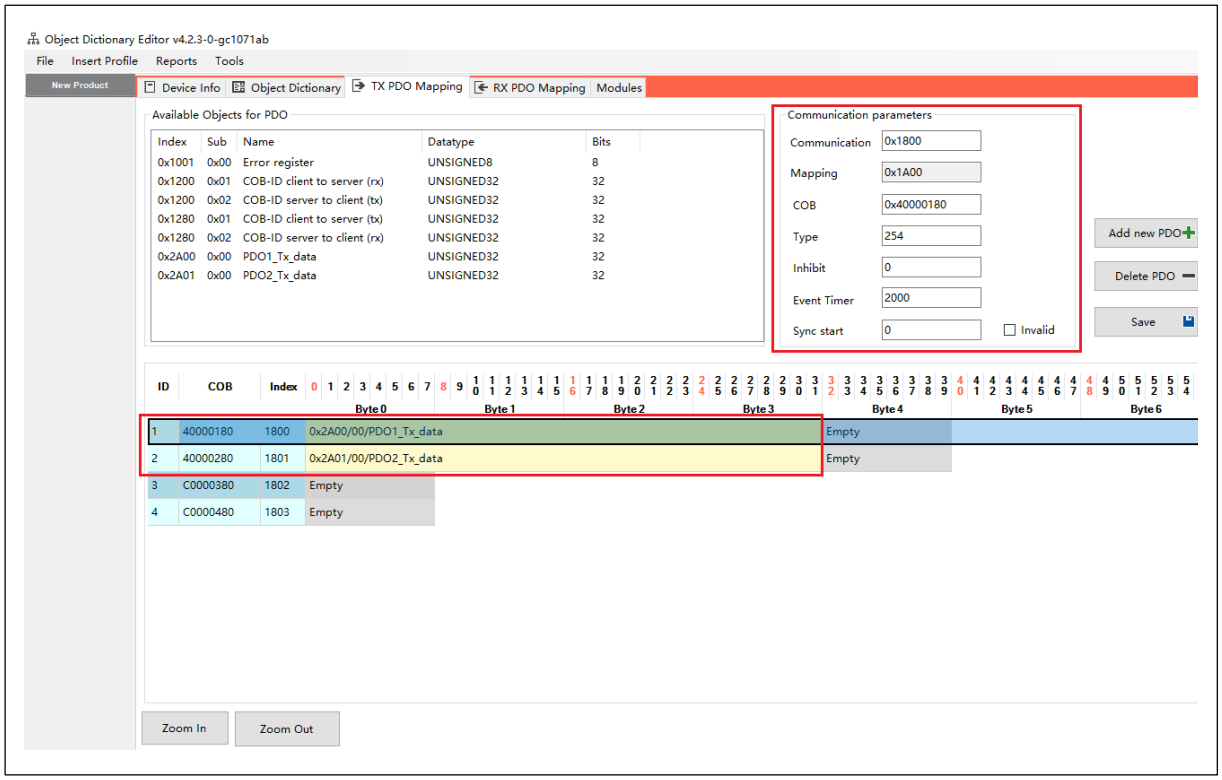
单击“Object Dictionary”窗口 → 自行选择各个RPDO和TPDO去配置通信参数 → 左击“Manufacturer Specific Parameters”中的“Index”或“Name”选择“Add…”来新建自定义的对象 → 在自定义的对象中写入默认值并更改访问权限。

图 19. 对象字典窗口


Object Dictionary Editor v4.2.3-0-gc1071ab
File
Insert Profile
Reports
Tools
New Product
Device Info
Object Dictionary
TX PDO Mapping
RX PDO Mapping
Modules
Communication Specific Parameters
Index
Name
0x1280 SDO client parameter
0x1400 RPDO communication parameter
0x1401 RPDO communication parameter
0x1402 RPDO communication parameter
0x1403 RPDO communication parameter
0x1600 RPDO mapping parameter
0x1601 RPDO mapping parameter
0x1602 RPDO mapping parameter
0x1603 RPDO mapping parameter
0x1800 TPDO communication parameter
0x1801 TPDO communication parameter
0x1802 TPDO communication parameter
0x1803 TPDO communication parameter
0x1A00 TPDO mapping parameter
0x1A01 TPDO mapping parameter
0x1A02 TPDO mapping parameter
0x1A03 TPDO mapping parameter
Manufacturer Specific Parameters
0x2600 PDO1_Rx_data
0x2A00 PDO1_Tx_data
0x2A01 PDO2_Tx_data
Device Profile Specific Parameters
Sub
Name
Obj Type
Data Type
SDO
PDO
SRDO
Default Value
TPDO communication parameter RECORD
0x00 Highest sub-index supported VAR UNSIGNED8 ro no no 0x06
0x01 COB-ID used by TPDO VAR UNSIGNED32 rw no no $NODEID+0x4000018
0x02 Transmission type VAR UNSIGNED8 rw no no 254
0x03 Inhibit time VAR UNSIGNED16 rw no no 0
0x05 Event timer VAR UNSIGNED16 rw no no 2000
0x06 SYNC start value VAR UNSIGNED8 rw no no 0
Index 0x1800
Sub Index
Name TPDO communication parameter
Denotation
Description
* COB-ID used by RPDO:
* bit 31: If set, PDO does not exist / is not valid
* bit 30: If set, NO RTR is allowed on this PDO
* bit 11-29: set to 0
* bit 0-10: 11-bit CAN-ID
* Transmission type:
* Value 0: synchronous (acyclic)
* Value 1-240: synchronous (cyclic every (1-240)-th sync)
* Value 241-253: not used
* Value 254: event-driven (manufacturer-specific)
Object settings
Object Type RECORD
Data Type
Access SDO
Access PDO
Access SRDO
Default value
HighLimit
LowLimit
Actual Value
String Len Min
Count Label TPDO
Storage Group PERSIST_COMM
Enabled
Save Changes
Autosave changes
Object Dictionary Editor v4.2.3-0-gc1071ab
File
Insert Profile
Reports
Tools
New Product
Device Info
Object Dictionary
TX PDO Mapping
RX PDO Mapping
Modules
Communication Specific Parameters
Index
Name
0x1028 Emergency consumer object
0x1029 Error behavior object
0x1200 SDO server parameter
0x1201 SDO server parameter
0x1280 SDO client parameter
0x1400 RPDO communication parameter
0x1401 RPDO communication parameter
0x1402 RPDO communication parameter
0x1403 RPDO communication parameter
0x1600 RPDO mapping parameter
0x1601 RPDO mapping parameter
0x1602 RPDO mapping parameter
0x1603 RPDO mapping parameter
0x1800 TPDO communication parameter
0x1801 TPDO communication parameter
0x1802 TPDO communication parameter
0x1803 TPDO communication parameter
0x1A00 TPDO mapping parameter
0x1A01 TPDO mapping parameter
0x1A02 TPDO mapping parameter
0x1A03 TPDO mapping parameter
Manufacturer Specific Parameters
0x2600 PDO1_Rx_data
0x2A00 PDO1_Tx_data
0x2A01 PDO2_Tx_data
Device Profile Specific Parameters
Sub
Name
Obj Type
Data Type
SDO
PDO
SRDO
Default Value
PDO1_Tx_data VAR UNSIGNED32 rw t no 0x2A000123
Index 0x2A00
Sub Index
Name PDO1_Tx_data
Denotation
Description
Object settings
Object Type VAR
Data Type UNSIGNED32
Access SDO rw
Access PDO t
Access SRDO no
Default value 0x2A000123
HighLimit
LowLimit
Actual Value
String Len Min 0
Count Label
Storage Group RAM
Enabled
Save Changes
Autosave changes

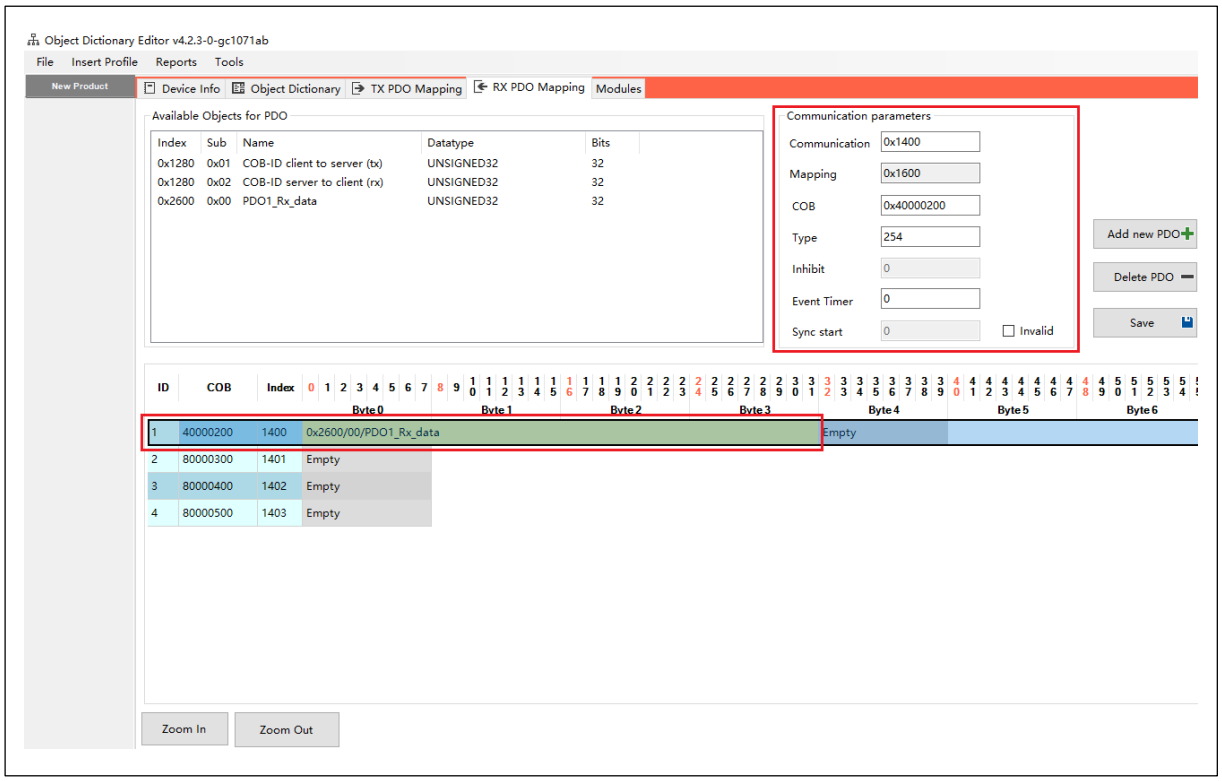
4、在“TX PDO Mapping” 窗口中去配置TPDO的映射参数所寻址的对象（即自定义的对象）。

图 20. 发送 PDO 映射窗口



5、在“RX PDO Mapping” 窗口中去配置RPDO的映射参数所寻址的对象（即自定义的对象）。

图 21. 接收 PDO 映射窗口



6、当对象字典的所有对象及参数都配置完成后，就可以导出相关文件，供上位机软件和AT32工程的协议栈使用。

具体流程：

1、导出EDS文件：单击“File”→ “Export…”→ 在“另存为”对话框中选择“保存类型”为（*.eds）→ 单击“保存”将该文件名默认为“demo_project.eds”。

2、导出OD源文件：单击“File”→ “Export CanOpenNode…”→ 在“另存为”对话框中选择路径为AT32工程下的canopen_node_port文件夹 → 将生成的OD源文件替换原有文件。

3.5 调试软件的配置

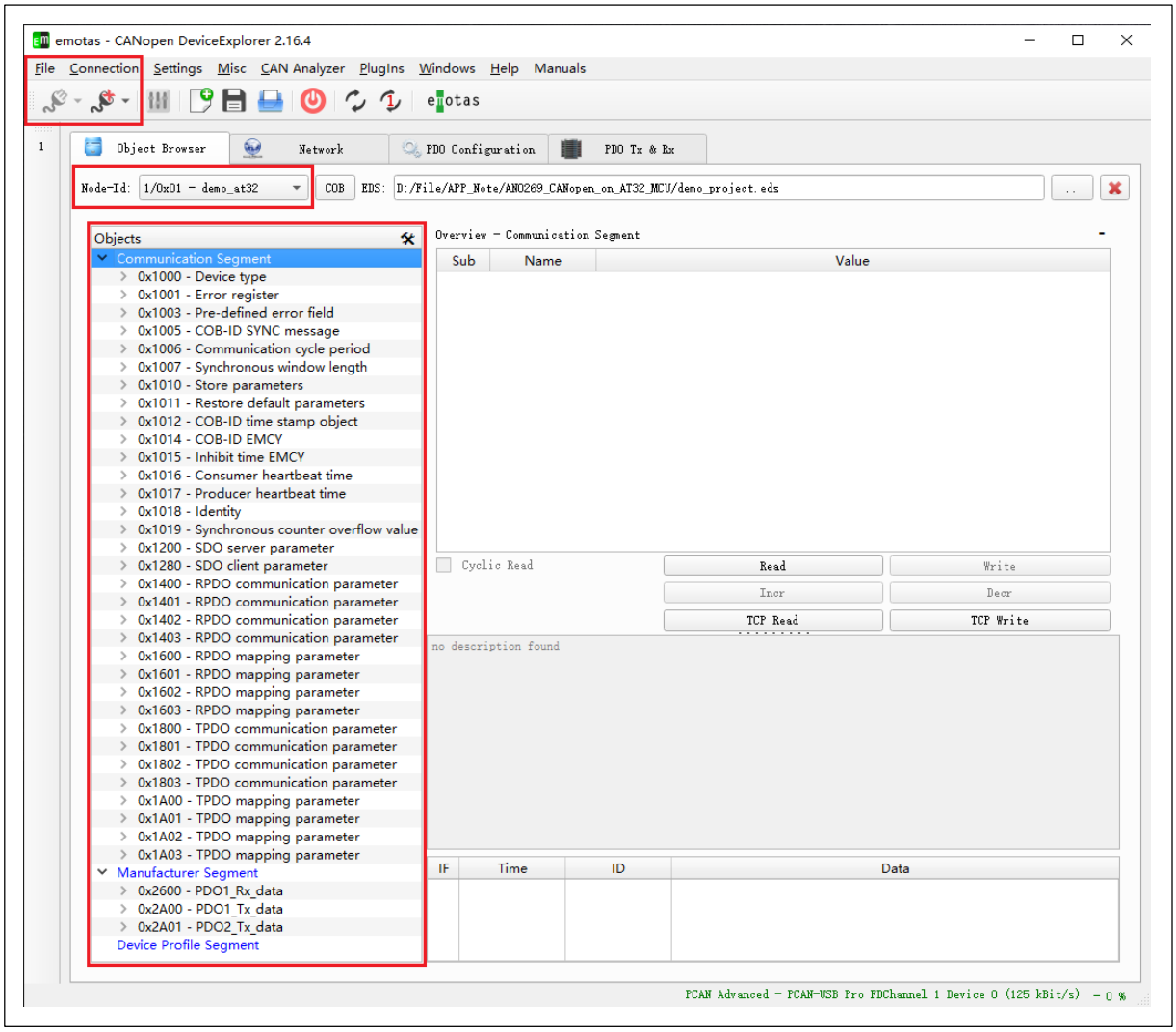
本案例中将借用CANopen DeviceExplorer的上位机调试软件与AT32工程通信来模拟CANopen的实际应用，但在这之前需导入与AT32工程有相同对象及参数配置的EDS文件并连接配置参数等。

具体操作流程如下：

先提取 setup-emotas-cde-2_16_4.zip 中的可执行文件并安装，安装完成后再运行该软件；再单击“File”，选择“Load EDS”，在弹出的对话框中选中之前生成的“demo_project.eds”文件；再单击“Connection”连接到PCAN-USB适配器。这里的Node-Id需选择与AT32工程里写入的Id一致。

如下图是已配置完成的CANopen DeviceExplorer软件界面。

图 22. CANopen DeviceExplorer 软件界面



3.6 设备功能的实现

1、在“main.c”文件中编写CAN接收和发送的中断函数去调用协议栈的回调函数，来实现CANopen底层的接收和发送功能。

```
void CAN1_RX_IRQHandler(void)
{
  /* Clear interrupt flag */
  if (can_interrupt_flag_get(CO->CANmodule->CANptr, CAN_RIF_FLAG) == SET) {
      can_flag_clear(CO->CANmodule->CANptr, CAN_RIF_FLAG);
  }
  else{
      return;
  }
  CO_CANinterrupt_RX(CO->CANmodule);
}

void CAN1_TX_IRQHandler(void)
{
  /* Clear interrupt flag */
  if (can_interrupt_flag_get(CO->CANmodule->CANptr, CAN_TPIF_FLAG) == SET) {
      can_flag_clear(CO->CANmodule->CANptr, CAN_TPIF_FLAG);
  }
  else{
      return;
  }
  CO_CANinterrupt_TX(CO->CANmodule);
}
```

2、在“main.c”文件中编写TMR定时的中断函数去调用协议栈的定时线程函数。

```
void TMR7_GLOBAL_IRQHandler(void)
{
  /* Clear interrupt flag */
  if (tmr_interrupt_flag_get(TMR7, TMR_OVF_FLAG) == SET) {
      tmr_flag_clear(TMR7, TMR_OVF_FLAG);
  }
  else{
      return;
  }
  tmrTask_thread();
}
```

3、在main()函数中，若该节点未被通信或应用复位，则会每间隔500us跑一次协议栈的主线程，并会根据当前该节点的状态来改变红绿LED灯的亮灭。

```
        while (reset == CO_RESET_NOT) {
            /* loop for normal program execution ******************************************/
            /* get time difference since last function call */
            uint32_t timeDifference_us = 500;

            /* CANopen process */
            reset = CO_process(CO, false, timeDifference_us, NULL);
            LED_red = CO_LED_RED(CO->LEDs, CO_LED_CANopen);
            LED_green = CO_LED_GREEN(CO->LEDs, CO_LED_CANopen);
            LED_red ? at32_led_on(LED2) : at32_led_off(LED2);
            LED_green ? at32_led_on(LED4) : at32_led_off(LED4);

            /* Nonblocking application code may go here. */

            /* Process automatic storage */

            /* optional sleep for short time */
            delay_us(timeDifference_us);
        }
```

4 案例演示

4.1 功能简介

实现 CANopen 网络通信的一些基本功能：

1、网络管理及心跳

2、对象字典的读写

3、PDO 收发及配置

4.2 资源准备

1) 硬件环境：

参考“第 2 章节——AT32 硬件准备”。

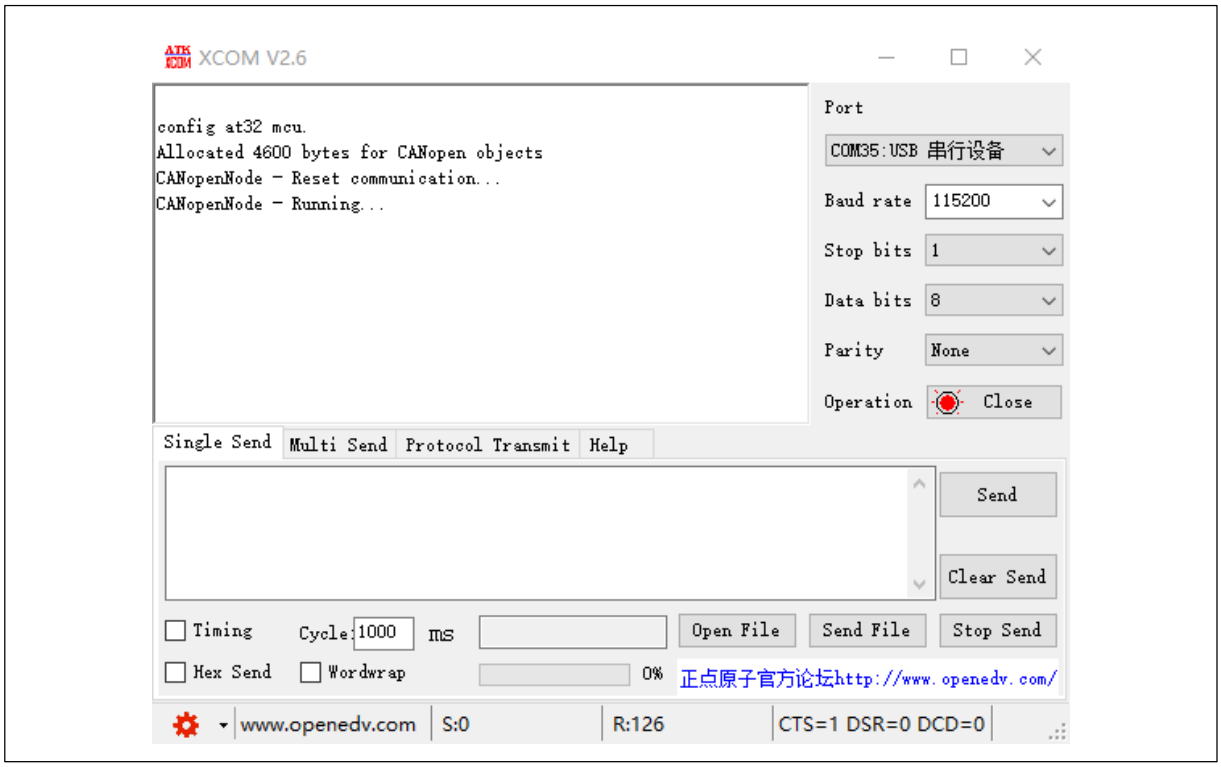
2) 软件环境：

AN0269_SourceCode_V2.0.0\canopen_demo_for_at32m412\project\at_start_m412\canopen_for_at32

4.3 测试效果

至此，官方例程移植完毕，编译并下载，开打与AT-Link相连的串口，可看到如下打印信息。

图 23. 串口打印信息



从打印信息可以看到，设备已经正常的运行起来。

这时我们需要将此设备与PCAN-USB适配器相连接，再打开CANopen DeviceExplorer软件，模拟CANopen主站来进行通信，下面将详细介绍一下测试的功能。

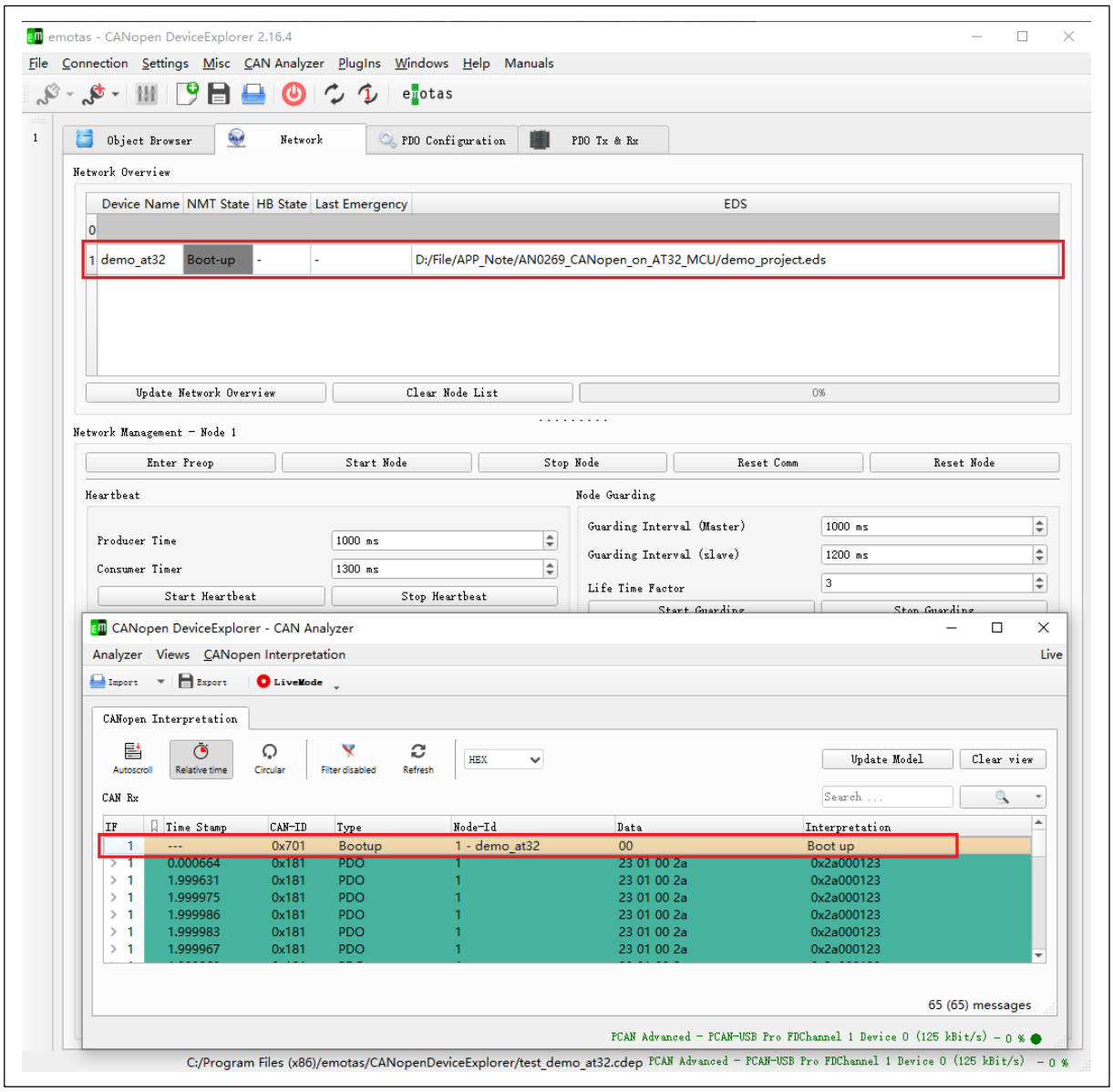
4.3.1 网络管理及心跳测试

网络状态管理是作为CANopen节点的一个最基本的功能，主机通过NMT命令可以让网络中任意一个的CANopen从节点进行网络状态的切换。同时主机通过SDO命令去改写某个CANopen从节点心跳周期的值，该节点会根据设定的周期值定期发送心跳报文，并返回该节点当前的网络状态。

测试的具体步骤如下：

1. 首先打开CANopen DeviceExplorer软件的“Network”窗口；再依次单击“CAN Analyzer” → “CANopen Interpretation”，此时会弹出一个“CAN Analyzer”的子窗口，这里可以查看传输到CAN总线的底层数据并解释为CANopen的报文，可便于测试分析。
2. 然后给预先下载了工程代码的AT-START Board上电复位后，当该节点会在进入到操作状态（operational）之前，发送节点上线报文（Boot-up message）到总线上，此时可以在调式软件的窗口上看到此报文，如下图所示。

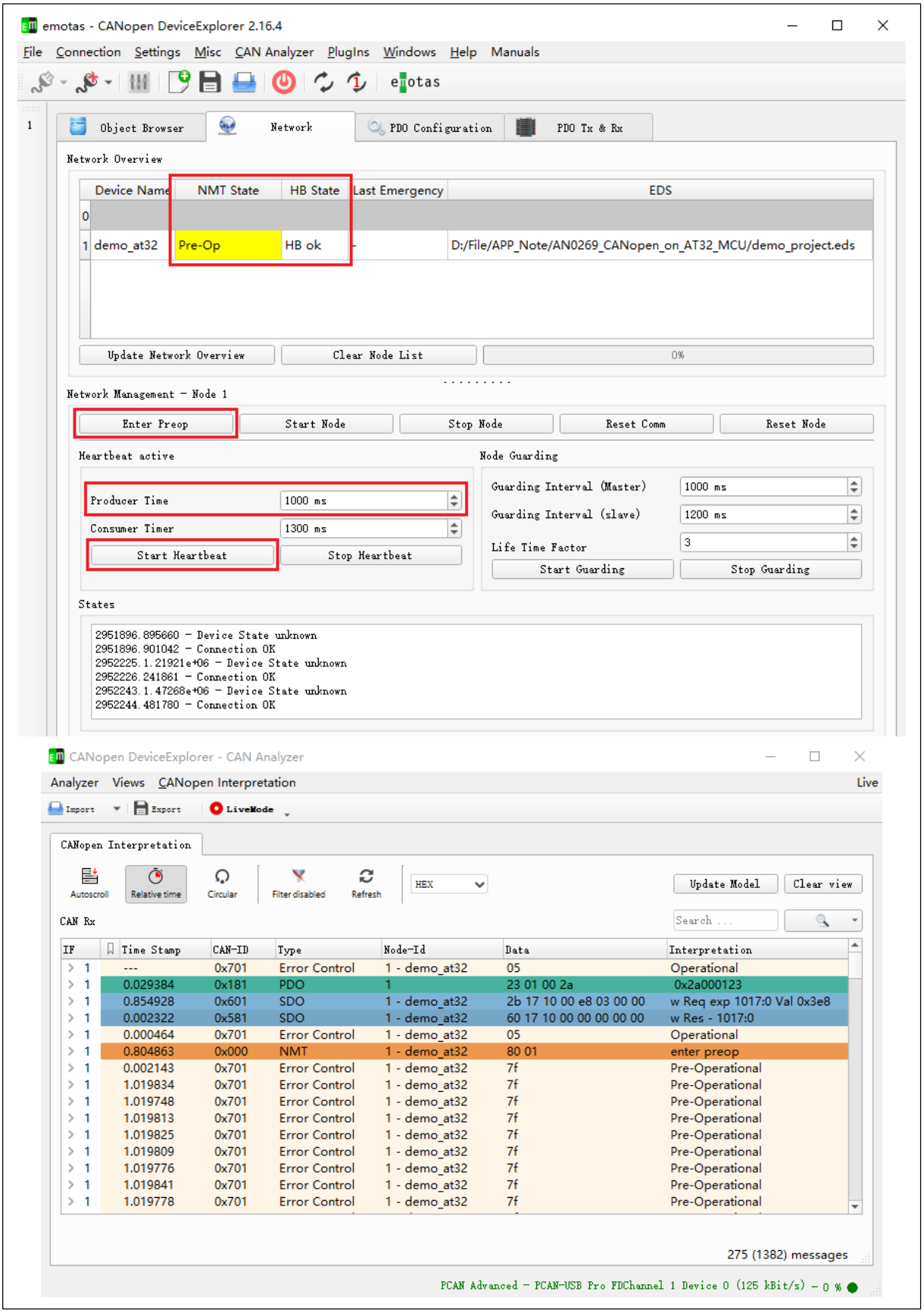
图 24. 节点上线



3. 那接下来就可以去配置该节点的一个心跳周期值，即“Network”窗口中的“Producer Time”框中的值（默认1000ms），单击“Start Heartbeat”，主站就会发送命令给该节点，节点会每隔

1s发送一个心跳报文。如此时点击“Enter Preop”，该节点会进入到预操作状态，并停止PDO通信，心跳报文也会实时反馈该节点的状态，如下图所示。

图 25. 节点状态及心跳



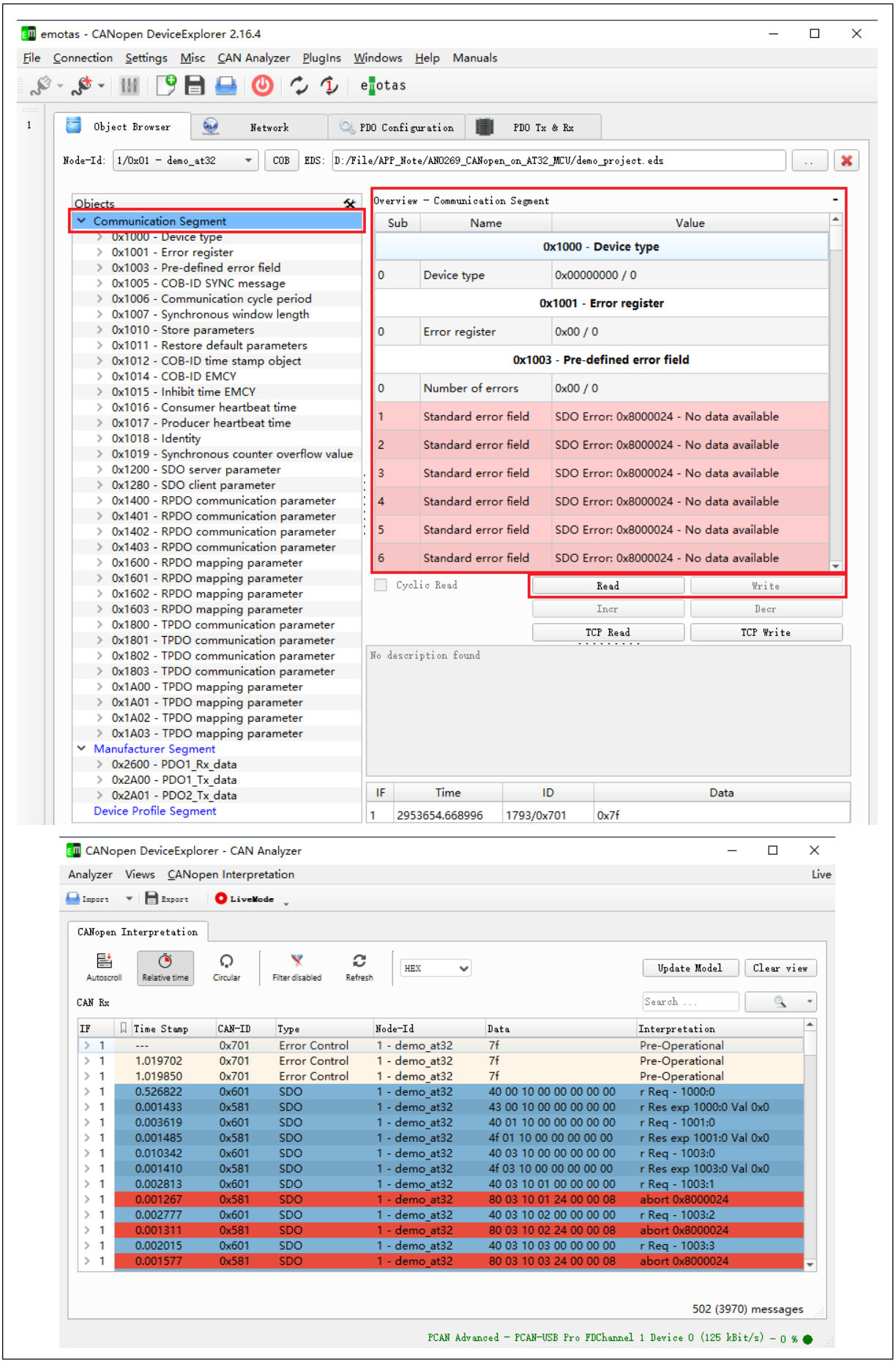
4.3.2 对象字典的读写测试

对象字典是CANopen协议最为核心的概念，也是最重要的部分之一。根据对象字典中预先设定好的SDO读写的访问权限，主机通过SDO命令可以去读写在访问权限内的对象及参数等。

测试的具体步骤如下：

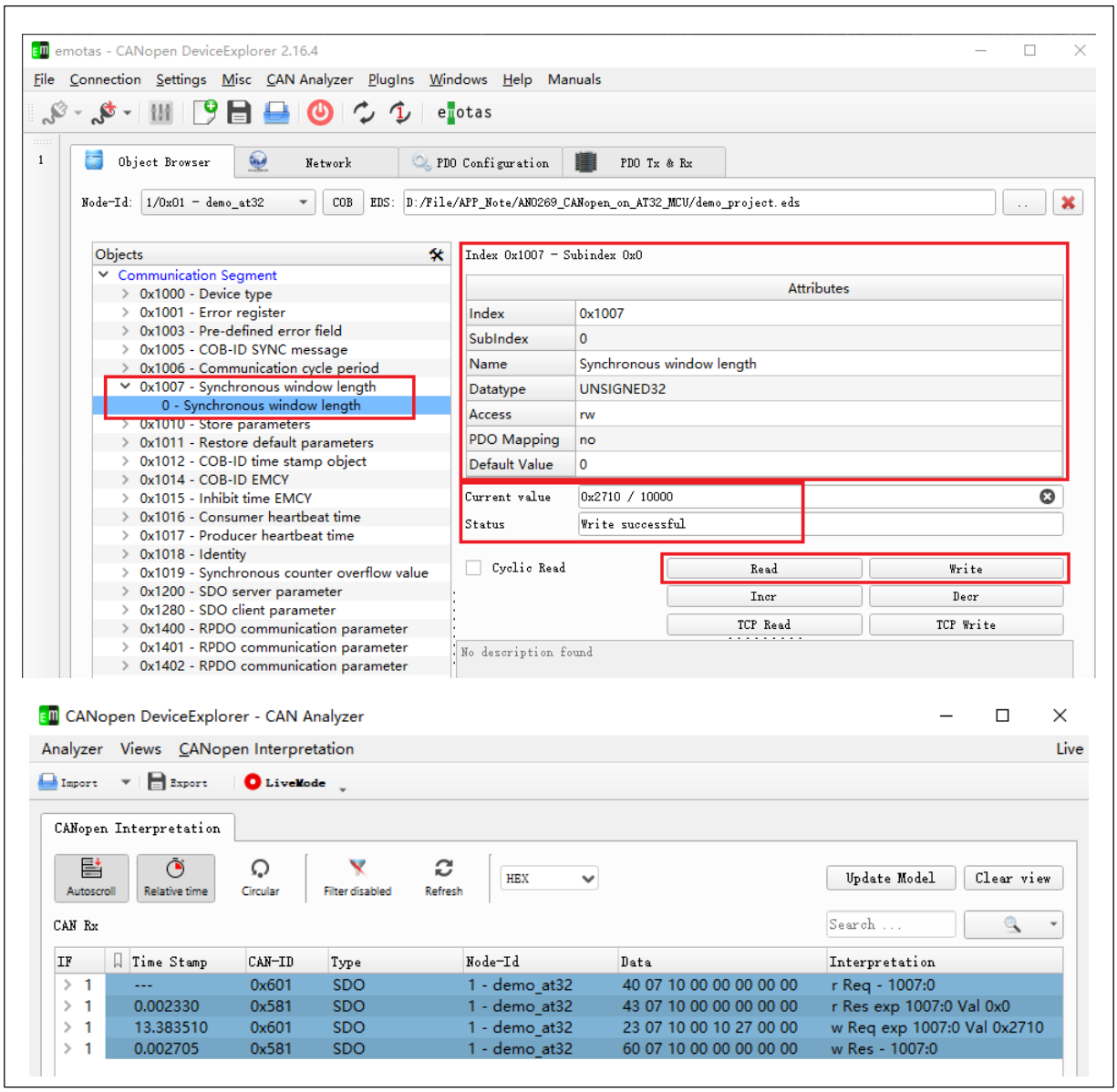
1. 首先打开CANopen DeviceExplorer软件的“Object Browser”窗口；再单击“CAN Analyzer” → “CANopen Interpretation”，此时会弹出一个“CAN Analyzer”的子窗口，这里可以查看传输到CAN总线的底层数据并解释为CANopen的报文，可便于测试分析。
2. 选中“Object Browser”窗口中“Objects”框内的“Communication Segment”，再单击“Read”按键，主站就会依次发送SDO读取命令给该节点，直到读完整个通信区域中的所有对象及参数，在右边的“Overview”窗口中可以查看到已读取的数据，如下图所示。

图 26. 读对象字典


emotas - CANopen DeviceExplorer 2.16.4
File Connection Settings Misc CAN Analyzer PlugIns Windows Help Manuals
Object Browser
Network
PDO Configuration
PDO Tx & Rx
Node-Id: 1/0x01 - demo_at32
COB
EDS: D:/File/APP_Note/AN0269_CANopen_on_AT32_MCU/demo_project.eds
Objects
Communication Segment
0x1000 - Device type
0x1001 - Error register
0x1003 - Pre-defined error field
0x1005 - COB-ID SYNC message
0x1006 - Communication cycle period
0x1007 - Synchronous window length
0x1010 - Store parameters
0x1011 - Restore default parameters
0x1012 - COB-ID time stamp object
0x1014 - COB-ID EMCY
0x1015 - Inhibit time EMCY
0x1016 - Consumer heartbeat time
0x1017 - Producer heartbeat time
0x1018 - Identity
0x1019 - Synchronous counter overflow value
0x1200 - SDO server parameter
0x1280 - SDO client parameter
0x1400 - RPDO communication parameter
0x1401 - RPDO communication parameter
0x1402 - RPDO communication parameter
0x1403 - RPDO communication parameter
0x1600 - RPDO mapping parameter
0x1601 - RPDO mapping parameter
0x1602 - RPDO mapping parameter
0x1603 - RPDO mapping parameter
0x1800 - TPDO communication parameter
0x1801 - TPDO communication parameter
0x1802 - TPDO communication parameter
0x1803 - TPDO communication parameter
0x1A00 - TPDO mapping parameter
0x1A01 - TPDO mapping parameter
0x1A02 - TPDO mapping parameter
0x1A03 - TPDO mapping parameter
Manufacturer Segment
0x2600 - PDO1_Rx_data
0x2A00 - PDO1_Tx_data
0x2A01 - PDO2_Tx_data
Device Profile Segment
Overview - Communication Segment
Sub
Name
Value
0x1000 - Device type
0
Device type
0x00000000 / 0
0x1001 - Error register
0
Error register
0x00 / 0
0x1003 - Pre-defined error field
0
Number of errors
0x00 / 0
1
Standard error field
SDO Error: 0x8000024 - No data available
2
Standard error field
SDO Error: 0x8000024 - No data available
3
Standard error field
SDO Error: 0x8000024 - No data available
4
Standard error field
SDO Error: 0x8000024 - No data available
5
Standard error field
SDO Error: 0x8000024 - No data available
6
Standard error field
SDO Error: 0x8000024 - No data available
Cyclic Read
Read
Write
Incr
Decr
TCP Read
TCP Write
No description found
IF
Time
ID
Data
1
2953654.668996
1793/0x701
0x7f
CANopen DeviceExplorer - CAN Analyzer
Analyzer Views CANopen Interpretation
Live
Import
Export
LiveMode
CANopen Interpretation
Autoscroll
Relative time
Circular
Filter disabled
Refresh
HEX
Update Model
Clear view
CAN Rx
Search ...
IF
Time Stamp
CAN-ID
Type
Node-Id
Data
Interpretation
1 --- 0x701 Error Control 1 - demo_at32 7f Pre-Operational
1 1.019702 0x701 Error Control 1 - demo_at32 7f Pre-Operational
1 1.019850 0x701 Error Control 1 - demo_at32 7f Pre-Operational
1 0.526822 0x601 SDO 1 - demo_at32 40 00 10 00 00 00 00 00 r Req - 1000:0
1 0.001433 0x581 SDO 1 - demo_at32 43 00 10 00 00 00 00 00 r Res exp 1000:0 Val 0x0
1 0.003619 0x601 SDO 1 - demo_at32 40 01 10 00 00 00 00 00 r Req - 1001:0
1 0.001485 0x581 SDO 1 - demo_at32 4f 01 10 00 00 00 00 00 r Res exp 1001:0 Val 0x0
1 0.010342 0x601 SDO 1 - demo_at32 40 03 10 00 00 00 00 00 r Req - 1003:0
1 0.001410 0x581 SDO 1 - demo_at32 4f 03 10 00 00 00 00 00 r Res exp 1003:0 Val 0x0
1 0.002813 0x601 SDO 1 - demo_at32 40 03 10 01 00 00 00 00 r Req - 1003:1
1 0.001267 0x581 SDO 1 - demo_at32 80 03 10 01 24 00 00 08 abort 0x8000024
1 0.002777 0x601 SDO 1 - demo_at32 40 03 10 02 00 00 00 00 r Req - 1003:2
1 0.001311 0x581 SDO 1 - demo_at32 80 03 10 02 24 00 00 08 abort 0x8000024
1 0.002015 0x601 SDO 1 - demo_at32 40 03 10 03 00 00 00 00 r Req - 1003:3
1 0.001577 0x581 SDO 1 - demo_at32 80 03 10 03 24 00 00 08 abort 0x8000024
502 (3970) messages
PCAN Advanced - PCAN-USB Pro FDChannel 1 Device 0 (125 kBit/s) - 0 %

3. 选中“Communication Segment”框内的“0x1007 – Synchronous window length”这个对象，以此为例，测试写对象的参数。先点击该对象下的子索引（0），可在右边的窗口中看到该对象的详细参数（索引、子索引、名称、数据类型、访问权限、默认值等），再点击“Read”读出当前值为（0），再在“Current value”框内写入新值（例如10000），再点击“Write”后可在“Status” 框内查看当前更新的状态，如下图所示。

图 27. 写对象字典



4.3.3 PDO 收发及配置测试

应用上，实时的数据传输通过“过程数据对象（PDO）”完成。在测试前，需预先在对象字典中设定好PDO发送和接收的通信参数对象、映射参数对象、以及映射参数中所指向的自定义对象。

本例程将列举PDO1发送、PDO2发送、PDO1接收来测试“过程数据对象（PDO）”的数据传输。

下面将简单列举一下本例程的对象字典中已预先设定好的这些PDO对象所用到的部分配置参数，如下表所示。

表 6. 本例程中 PDO 通信参数对象

Index	COB	Resulting COB-ID	Transmission type	Event timer
0x1800	PDO1（Tx）	180h + Node-ID = 181h	254（event-driven）	2000（ms）
0x1801	PDO2（Tx）	280h + Node-ID = 281h	2（synchronous）	NA
0x1400	PDO1（Rx）	200h + Node-ID = 201h	NA	NA

表 7. 本例程中 PDO 映射参数对象

Index	COB	Number of mapped application objects（0）	Application object (1)
0x1A00	PDO1（Tx）	1	0x2A000020
0x1A01	PDO2（Tx）	1	0x2A010020
0x1600	PDO1（Rx）	1	0x26000020

表 8. 本例程中 PDO 映射所指向的自定义对象

Index	Name	Object Type	Data Type	Default value
0x2A00	PDO1_Tx_data	VAR	UNSIGNED32	0x2A000123
0x2A01	PDO2_Tx_data	VAR	UNSIGNED32	0x2A010123
0x2600	PDO1_Rx_data	VAR	UNSIGNED32	0x26006666

根据以上对象字典的参数配置，再结合上位机的调试软件来测试验证。

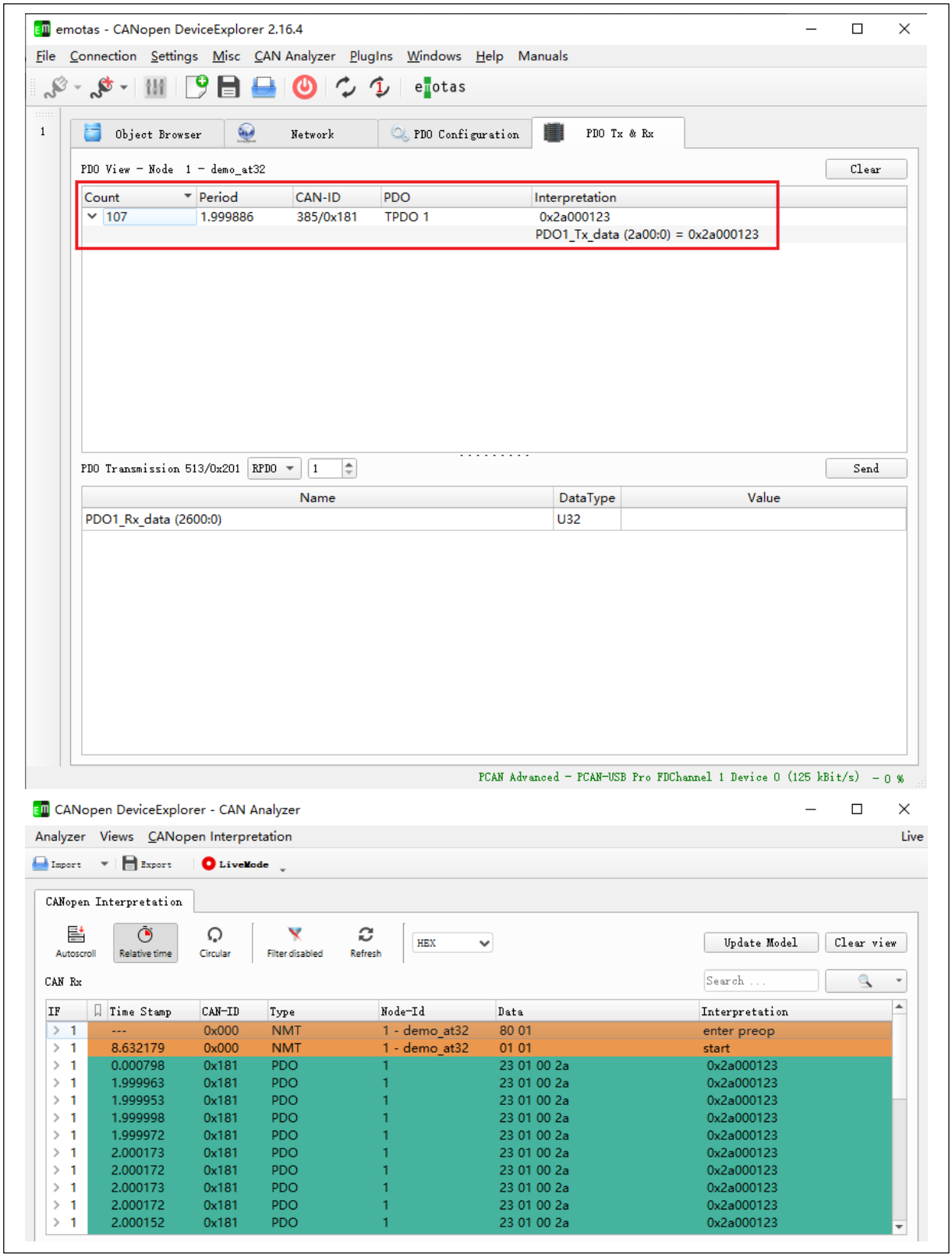
下面将分别说明一下这3个PDO对象的测试过程。

1. PDO1发送对象（PDO1（Tx））：

   此对象中的传输类型为事件驱动（即由节点内部的定时器所驱动），事件定时为2000ms，若节点处于操作状态时，该对象就会按定时时间间隔循环发送映射所指向的自定义对象（PDO1_Tx_data）的值。

   在调试软件的“PDO Tx & Rx”和“CANopen Interpretation”窗口中，均可以看到PDO1发送对象的报文，如下图所示。

图 28. 测试 PDO1 发送对象

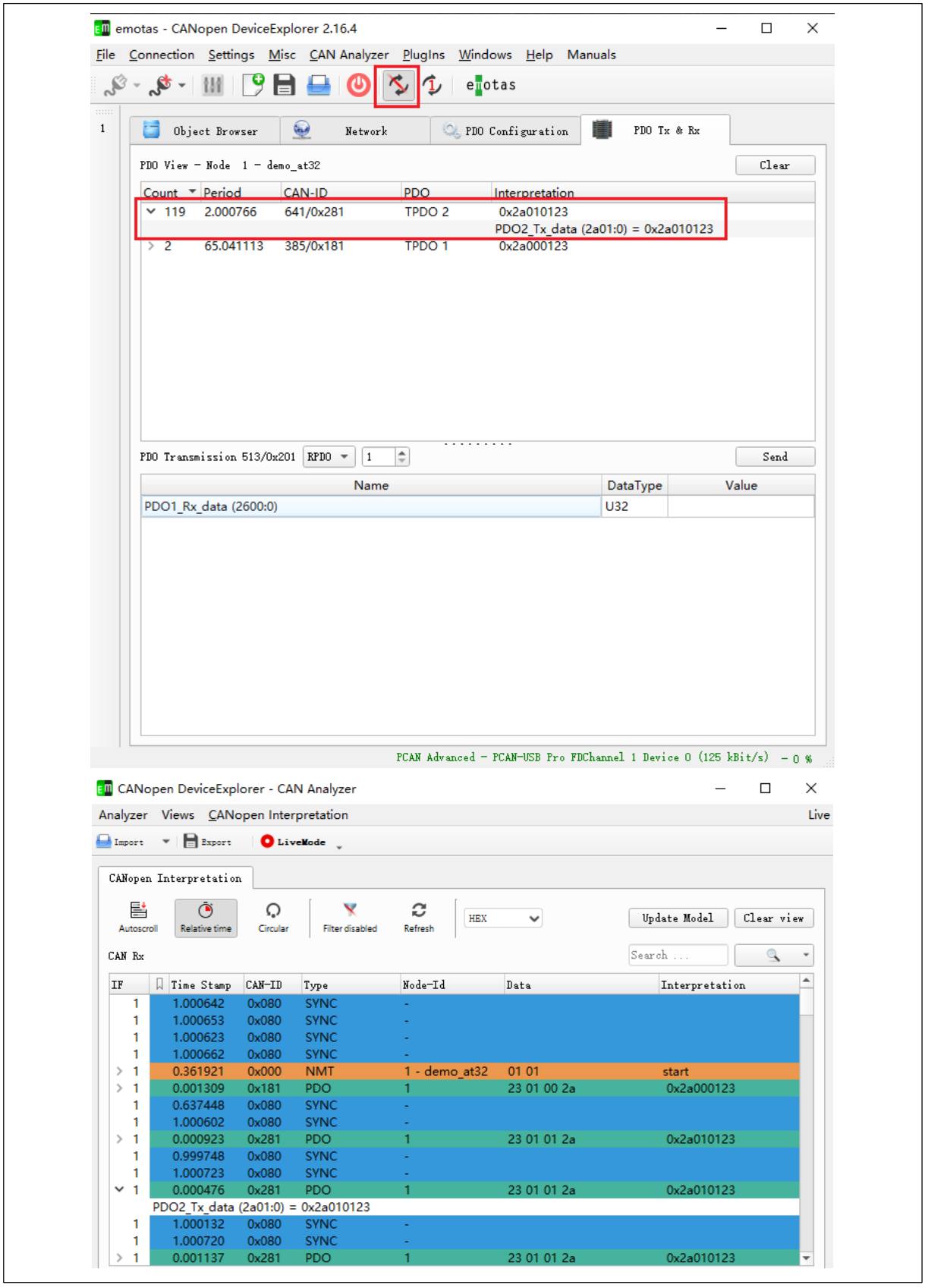


2. PDO2发送对象（PDO2（Tx））：

此对象中的传输类型为同步类型2（即每2个SYNC一周期），若节点处于操作状态时，该对象就会在收到2个SYNC报文后发送映射所指向的自定义对象（PDO2_Tx_data）的值。

点击调试软件的“Start SYNC”的图标，主站就会一直循环发送SYNC报文，并在 “PDO Tx & Rx”和“CANopen Interpretation”窗口中，均可以看到PDO2发送对象的报文，如下图所示。

图 29. 测试 PDO2 发送对象



3. PDO1接收对象（PDO1（Rx））：

此对象不用关心传输类型，若节点处于操作状态时，如总线上接收到该对象的数据，就会将该数据写入到对象映射所指向的自定义对象（PDO1_Rx_data）的值。

测试的具体步骤如下：

1、先在 “PDO Tx & Rx”窗口的“PDO Transmission”栏下“Value”框中写入需要发送的值，再点击“Send”，这时主站会发送一帧针对PDO1（Rx）对象的数据；

2、然后选中“Object Browser”窗口中“Objects”框内的“Manufacturer Segment”，再单击“Read”按键，主站就会依次发送命令去读取自定义对象的值，可以此时看到PDO1（Rx）所映射对象的值已改变，如下图所示。

图 30. 测试 PDO1 接收对象

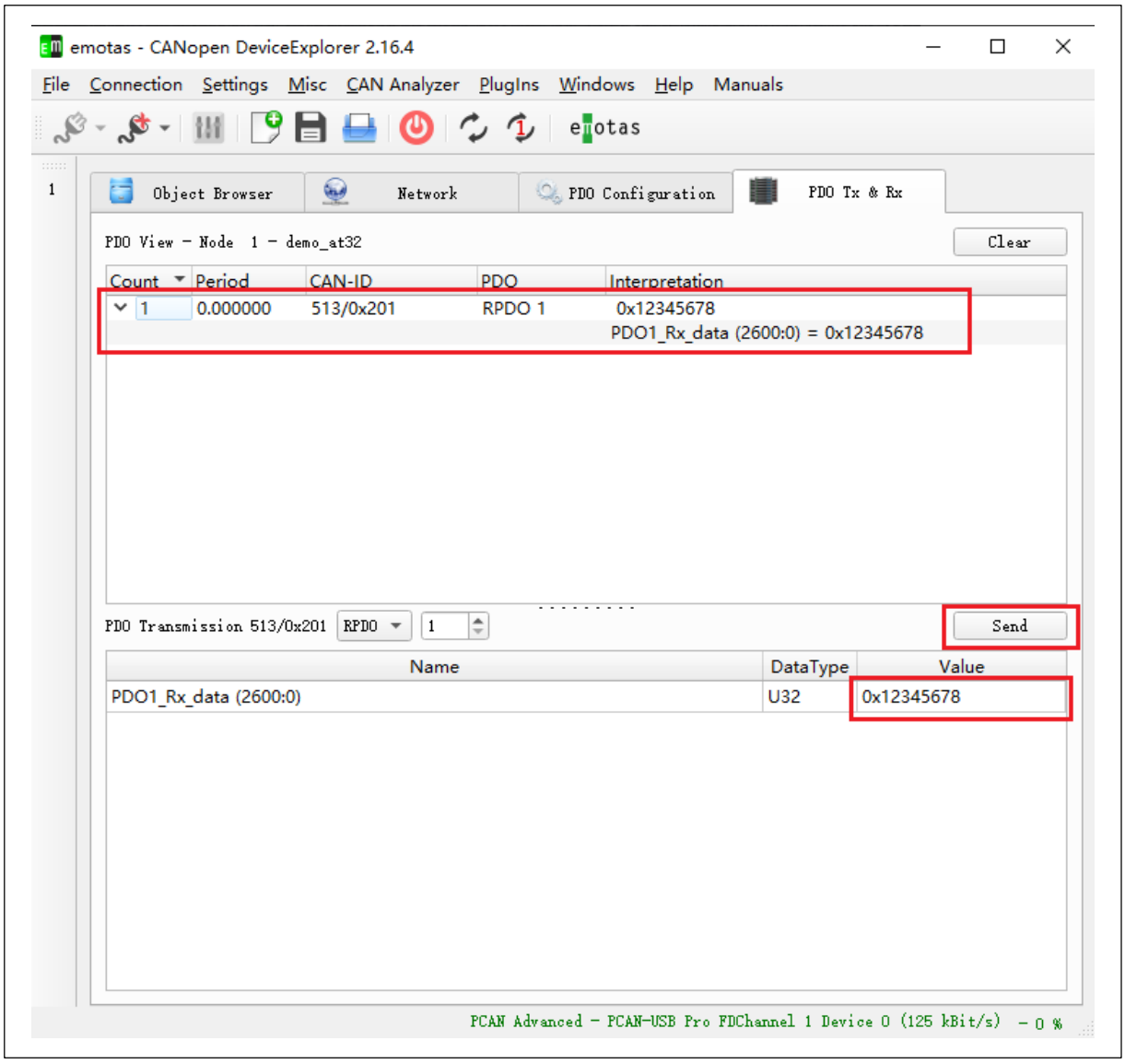
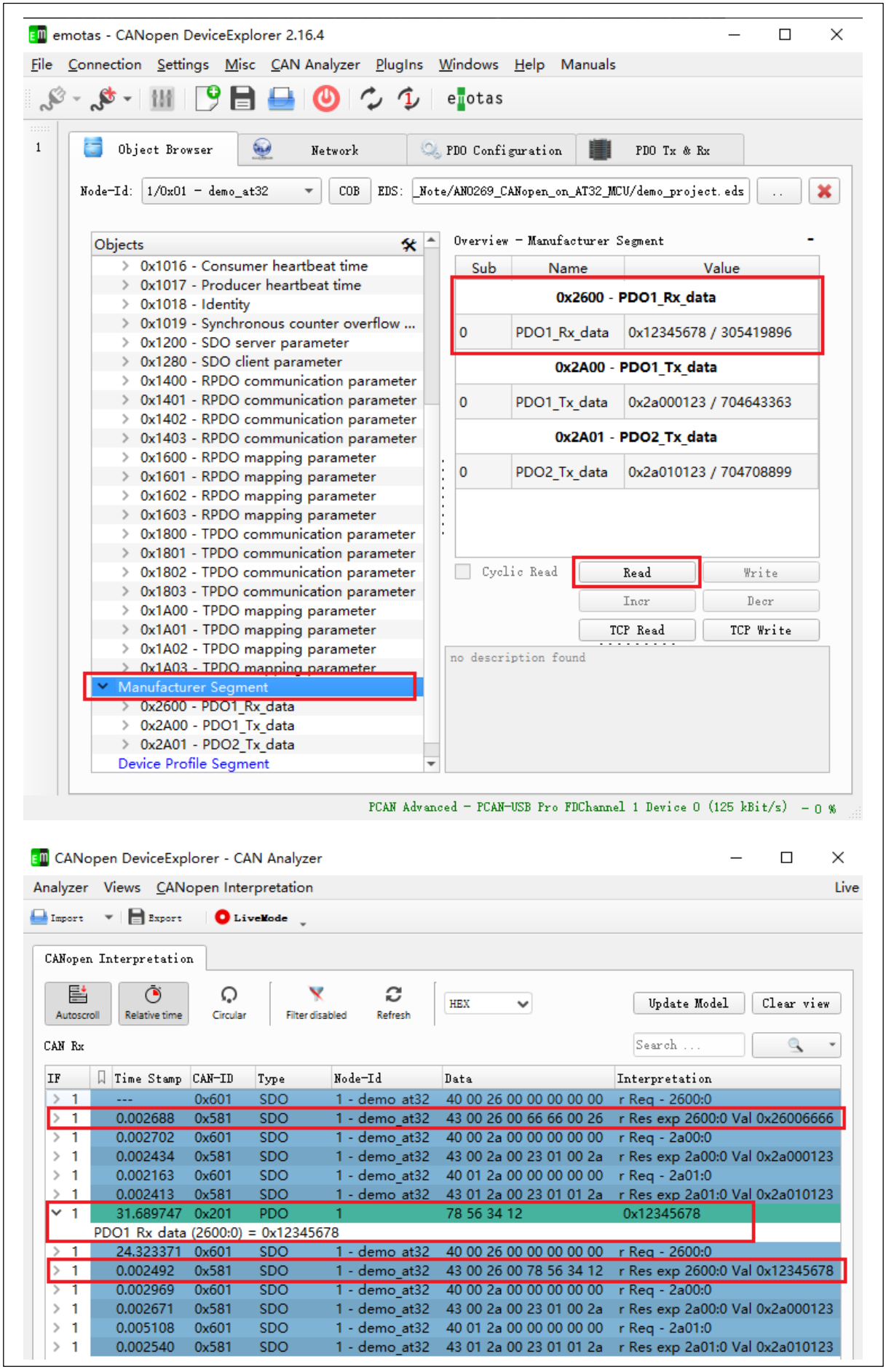


图 31. 读自定义对象

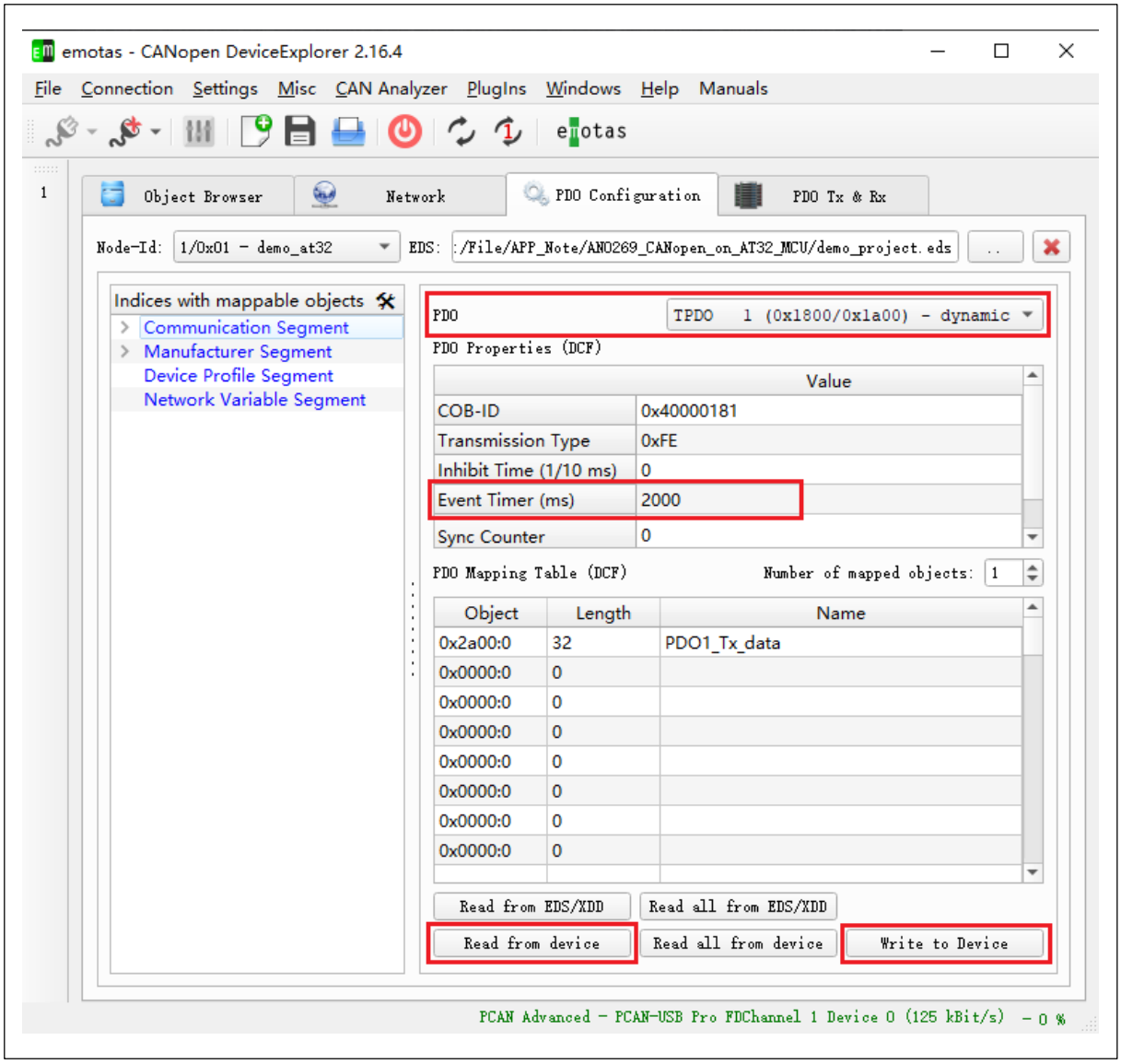

emotas - CANopen DeviceExplorer 2.16.4
File Connection Settings Misc CAN Analyzer PlugIns Windows Help Manuals
Object Browser
Network
PDO Configuration
PDO Tx & Rx
Node-Id: 1/0x01 - demo_at32
COB
EDS: _Note/AN0269_CANopen_on_AT32_MCU/demo_project.eds
Objects
0x1016 - Consumer heartbeat time
0x1017 - Producer heartbeat time
0x1018 - Identity
0x1019 - Synchronous counter overflow ...
0x1200 - SDO server parameter
0x1280 - SDO client parameter
0x1400 - RPDO communication parameter
0x1401 - RPDO communication parameter
0x1402 - RPDO communication parameter
0x1403 - RPDO communication parameter
0x1600 - RPDO mapping parameter
0x1601 - RPDO mapping parameter
0x1602 - RPDO mapping parameter
0x1603 - RPDO mapping parameter
0x1800 - TPDO communication parameter
0x1801 - TPDO communication parameter
0x1802 - TPDO communication parameter
0x1803 - TPDO communication parameter
0x1A00 - TPDO mapping parameter
0x1A01 - TPDO mapping parameter
0x1A02 - TPDO mapping parameter
0x1A03 - TPDO mapping parameter
Manufacturer Segment
0x2600 - PDO1_Rx_data
0x2A00 - PDO1_Tx_data
0x2A01 - PDO2_Tx_data
Device Profile Segment
Overview - Manufacturer Segment
Sub Name Value
0x2600 - PDO1_Rx_data
0 PDO1_Rx_data 0x12345678 / 305419896
0x2A00 - PDO1_Tx_data
0 PDO1_Tx_data 0x2a000123 / 704643363
0x2A01 - PDO2_Tx_data
0 PDO2_Tx_data 0x2a010123 / 704708899
Cyclic Read
Read
Write
Incr
Decr
TCP Read
TCP Write
no description found
PCAN Advanced - PCAN-USB Pro FDChannel 1 Device 0 (125 kBit/s) - 0 %
CANopen DeviceExplorer - CAN Analyzer
Analyzer Views CANopen Interpretation
Live
Import
Export
LiveMode
CANopen Interpretation
Autoscroll
Relative time
Circular
Filter disabled
Refresh
HEX
Update Model
Clear view
CAN Rx
Search ...
IF Time Stamp CAN-ID Type Node-Id Data Interpretation
1 --- 0x601 SDO 1 - demo_at32 40 00 26 00 00 00 00 00 r Req - 2600:0
1 0.002688 0x581 SDO 1 - demo_at32 43 00 26 00 66 66 00 26 r Res exp 2600:0 Val 0x26006666
1 0.002702 0x601 SDO 1 - demo_at32 40 00 2a 00 00 00 00 00 r Req - 2a00:0
1 0.002434 0x581 SDO 1 - demo_at32 43 00 2a 00 23 01 00 2a r Res exp 2a00:0 Val 0x2a000123
1 0.002163 0x601 SDO 1 - demo_at32 40 01 2a 00 00 00 00 00 r Req - 2a01:0
1 0.002413 0x581 SDO 1 - demo_at32 43 01 2a 00 23 01 01 2a r Res exp 2a01:0 Val 0x2a010123
1 31.689747 0x201 PDO 1 78 56 34 12 0x12345678
PDO1 Rx data (2600:0) = 0x12345678
1 24.323371 0x601 SDO 1 - demo_at32 40 00 26 00 00 00 00 00 r Req - 2600:0
1 0.002492 0x581 SDO 1 - demo_at32 43 00 26 00 78 56 34 12 r Res exp 2600:0 Val 0x12345678
1 0.002969 0x601 SDO 1 - demo_at32 40 00 2a 00 00 00 00 00 r Req - 2a00:0
1 0.002671 0x581 SDO 1 - demo_at32 43 00 2a 00 23 01 00 2a r Res exp 2a00:0 Val 0x2a000123
1 0.005108 0x601 SDO 1 - demo_at32 40 01 2a 00 00 00 00 00 r Req - 2a01:0
1 0.002540 0x581 SDO 1 - demo_at32 43 01 2a 00 23 01 01 2a r Res exp 2a01:0 Val 0x2a010123

另外，还可以通过调试软件在线修改配置PDO的参数。

举例：在线配置PDO1发送对象（PDO1（Tx））的通信参数，将该对象的参数“Event timer”由2000修改为0，该节点的PDO1发送对象的报文将停止发送。

在调试软件的“PDO Configuration”窗口的PDO中选择“TPDO 1”项，先点击“Read from device”读取参数，再在该对象参数“Event timer”栏后的值从“2000”改为“0”，并点击“Write to Device”发送写命令给节点，可以看到该对象的报文将停止发送，如下图所示。

图 32. 配置 PDO1 发送对象



此处有关CAN总线上数据报文的信息可自行查看调试软件的“CANopen Interpretation”窗口。

5 版本历史

表 9. 文档版本历史

日期	版本	变更
2025.06.20	2.0.0	最初版本

重要通知 - 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途(及其依据任何司法管辖区的法律的对应情况)，或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：(A) 对安全性有特别要求的应用，如：生命支持、主动植入设备或对产品功能安全有要求的系统；(B) 航空应用；(C) 汽车应用或汽车环境；(D) 航天应用或航天环境，且/或(E) 武器。因雅特力产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险由购买者单独承担，并且独力负责在此类相关使用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。