User Manual

XGate-COP20

嵌入式工业通信协议转换模块

UM01010101 V1.01 Date: 2014/12/12 产品用户手册

类别	内容
关键词	XGate-COP20 Modbus CANOpen
摘 要	介绍 CANopen 基本原理、模块现场总线的参数说明

广州致远电子股份有限公司

修订历史

版本	日期	原因
V1.00	2012/03/27	创建文档
V1.01	2014/12/12	文档标准化

目　录

1. CANOpen 概述 1
1.1 相关术语解释及文档规则 1
1.2 概述 1
1.3 CANopen 对象字典 2
1.4 CANopen 通讯 3
1.5 CANopen 网络配置 4
2. Modbus 概述 5
2.1 概述 5
2.2 Modbus 通信 5
2.3 Modbus 实现模型 7
2.3.1 Modbus RTU 通信模式(UART) 7
2.3.2 Modbus RTU 通信模式(SPI) 8
3. 配置信息 9
3.1 通用参数特别说明 9
3.2 LED 指示 9
3.3 现场总线参数 10
4. 免责声明 21

1. CANOpen 概述

1.1 相关术语解释及文档规则

术语解释：

PDO：Process Data Object，过程数据对象

TPDO：Transmit Process Data Object，发送过程数据对象

RPDO：Receive Process Data Object，接收过程数据对象

SDO：Service Data Object，服务数据对象

NMT：Network Management，网络管理

SYNC：Synchronization Objects，同步报文对象

EMCY：Emergency Objects，紧急对象报文

CAN-ID：Controller Area Network-Identify，控制器局域网标识符

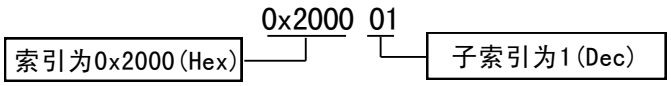
COB-ID：Communication Object-Identify，通信对象标识符

SSDO：Servers Service Data Object，服务数据服务器

DS：Draft Standard，标准草案

文档规则：

本手册中，对象字典索引与子索引的书写遵循如下图所示的规则，其中索引为 16 进制表示，子索引为 10 进制表示，索引与子索引中间用空格符隔开。



本手册的命令和响应的数据实例均为 16 进制（Hex），98 表示 0x98。

1.2 概述

CANopen 协议是在 20 世纪 90 年代末，由 CiA 组织（CAN-in-Automation）在 CAL（CAN Application Layer）的基础上发展而来，一经推出便在欧洲得到了广泛的认可与应用。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。目前 CANopen 协议已经在运动控制、车辆工业、电机驱动、工程机械、船舶海运等行业得到广泛的应用。

图 1.1 所示为 CANopen 设备结构，CANopen 协议通常分为用户应用层、对象字典、以及通讯三个部分。

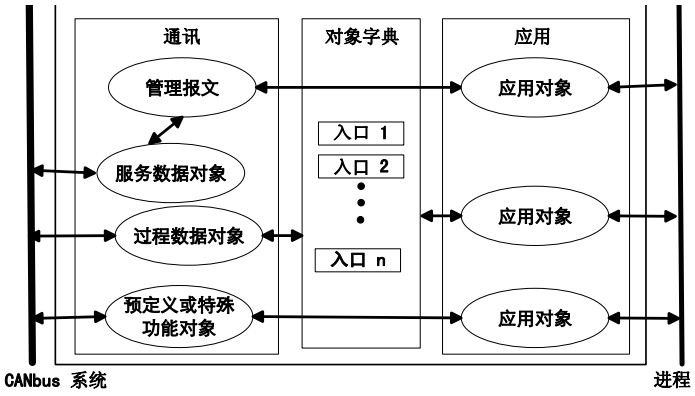


图 1.1　CANopen 设备结构

1.3　CANopen 对象字典

CANopen 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，每个对象采用一个 16 位的索引值来寻址，这个索引值通常被称为**索引**，其有效范围在 0x1000 到 0x9FFF 之间。为了允许访问数据结构中的单个元素，同时也定义了一个 8 位的索引值，这个索引值通常被称为**子索引**。

每个 CANopen 设备都有一个对象字典，包含了描述这个设备和它的网络行为的所有参数。对象字典通常用电子数据文档（EDS：Electronic Data Sheet）来记录这些参数，而非纸制纸张。对于 CANopen 网络中的主节点来说，不需要对 CANopen 从节点的每个对象字典项都访问。

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 协议的核心描述子协议是 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 或 DSP4xx。

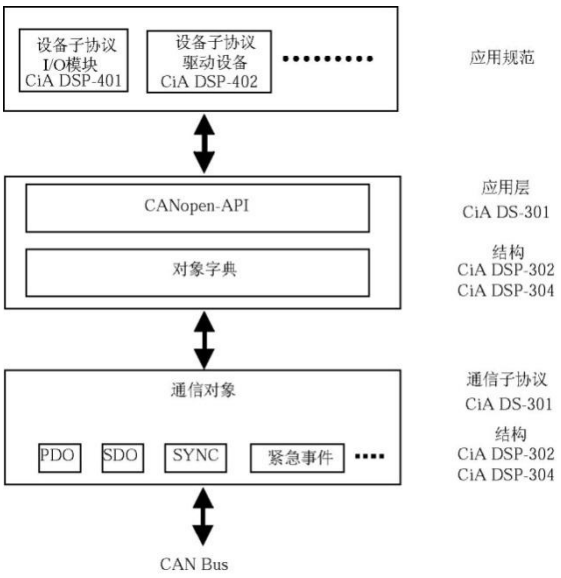


图 1.2　设备对象结构图

CANopen 协议包含了许多的子协议，其主要划分为以下三类。

1. 通讯子协议（Communication Profile）

通讯子协议，描述对象字典的主要形式和对象字典中的通讯对象以及参数。这个子协议适用所有的 CANopen 设备，其索引值范围从 0x1000~0x1FFF。

2. 制造商自定义子协议（Manufacturer-specific Profile）

制造商自定义子协议，对于在设备子协议中未定义的特殊功能，制造商可以在此区域根据需求定义对象字典对象。因此这个区域对于不同的厂商来说，相同的索引的对象字典项定义不一定相同，其索引值范围为 0x2000~0x5FFF。

3. 设备子协议(Device Profile)

设备子协议，为各种不同类型的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，其索引值范围为 0x6000~0x9FFF。

1.4 CANopen 通讯

在 CANopen 协议中主要定义了管理报文对象 NMT（Network Management）、服务数据对象 SDO（Service Data Object）、过程数据对象 PDO（Process Data Object）、预定义报文或特殊功能对象等四种对象。

1. 网络管理 NMT（Network Management）

管理报文负责层管理、网络管理和 ID 分配服务，例如，初始化、配置和网络管理（其中包括节点保护）。网络管理中，同一个网络中只允许有一个主节点、一个或多个从节点，并遵循主从模式。通过 NMT 服务，我们可以对节点进行初始化、运行、监控、复位和停止。所有节点都被认为是 NMT 从站。

2. 服务数据对象 SDO（Service Data Object）

SDO 主要用于主节点对从节点的参数配置。服务确认是 SDO 的最大的特点，为每个消息都生成一个应答，确保数据传输的准确性。在一个 CANopen 系统中，通常 CANopen 从节点作为 SDO 服务器，CANopen 主节点作为客户端。客户端通过索引和子索引，能够访问数据服务器上的对象字典。这样 CANopen 主节点可以访问从节点的任意对象字典项的参数，并且 SDO 也可以传输任何长度的数据（当数据长度超过 4 个字节时就拆分成多个报文来传输）。

3. 过程数据对象 PDO（Process Data Object）

PDO 用来传输实时数据，其传输模型为生产者消费者模型。PDO 通讯没有协议规定，PDO 数据内容由它的 CAN-ID（COB-ID）定义；

- 每个 PDO 在对象字典中用 2 个对象描述：
  - **PDO通讯参数**

    该通讯参数定义了设备所使用的COB-ID、传输类型、定时周期；

  - **PDO映射参数**

    映射参数包含了一个对象字典中的对象列表，这些对象映射到相应的PDO，其中包括数据的长度（单位：位），对于生产者和消费者都必须要知道这个映射参数，才能够正确的解释PDO内容。

- PDO 消息内容是预定义的，如果 PDO 支持可变 PDO 映射，那么该 PDO 是可以通过 SDO 进行配置；
- PDO 可以有多种的传输方式：
  - **同步传输**（通过接收同步对象实现同步），

    同步传输又可分为非周期和周期传输。非周期传输是由远程帧预触发或者由设备子协议中规定的对象特定事件预触发传送。周期传输则是通过接收同步对象（SYNC）来实现，可以设置1 - 240个同步对象触发；
  - **异步传输**（由特定事件触发）

    其触发方式可有两种，第一种是通过发送与PDO的COB-ID相同的远程帧来触发PDO的发送，第二种是由设备子协议中规定的对象特定事件来触发（例如，定时传输，数据状态变化传输等）。

4. 预定义报文或特殊功能对象

预定义报文或特殊功能对象为 CANopen 设备提供特定的功能，方便 CANopen 主站对从站管理。在 CANopen 协议中，已经为特殊的功能预定义了 COB-ID，其主要有以下几种特殊报文：

- 同步（SYNC），该报文对象主要实现整个网络的同步传输，每个节点都以该同步报文作为 PDO 同步触发参数，因此该同步报文的 COB-ID 具有比较高的优先级以及最短的传输时间；
- 时间标记对象（Time Stamp），为各个节点提供公共的时间参考；
- 紧急事件对象（Emergency），当设备内部发生错误触发该对象，即发送设备内部错误代码；
- 节点/寿命保护（Node/Life Guarding），主节点可通过节点保护方式获取从节点的状态。从节点可通过寿命保护方式获取主节点的状态；
- 启动报文对象（Boot-up），从节点初始化完成后向网络中发送该对象，并进入到预操作状态。

1.5 CANopen 网络配置

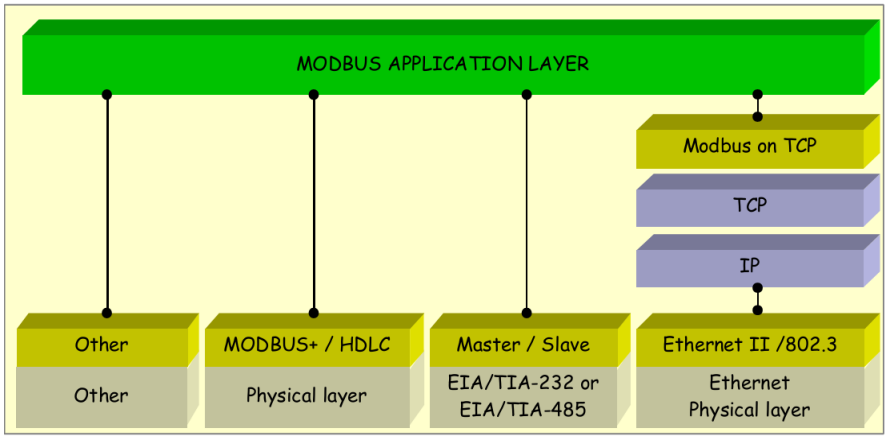
在 CANopen 协议描述文本 DS305 中定义了一种网络配置协议即网络配置服务 LSS (Layer Setting Service)，其通过 CAN 总线，用具有 LSS 主机功能的 CANOpen 模块来查询或修改具有 LSS 从机的 CANOpen 模块的某些参数。

通过使用 LSS ，可以对下面的参数进行查询或修改：

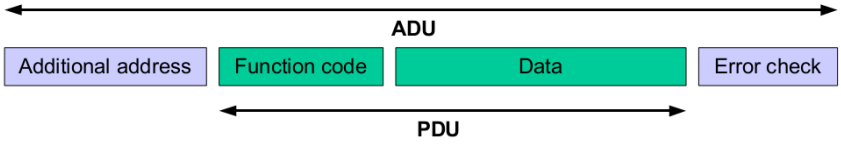
- CANopen 从站的 Node-ID；
- 物理层的位定时参数 (波特率)；
- LSS 地址 (特征对象 1018h)。

2. Modbus 概述

2.1 概述



Modbus 是一个应用层协议，采用主从的通信方式，可应用在不同的总线和网络中。Modbus 定义了一个简单的协议数据帧（PDU：protocol data unit），如果使用到特定的总线或者网络中，用户可增加附加区域，称为应用数据单元（帧）（ADU：application data unit）。

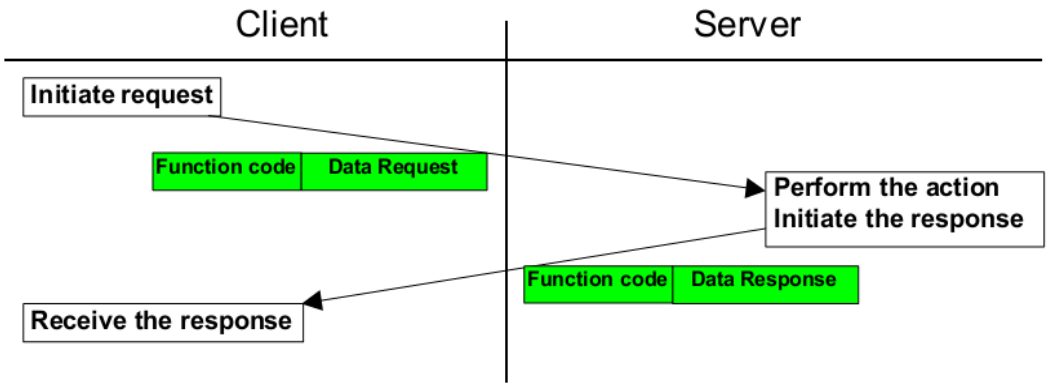


2.2 Modbus 通信

当服务器响应客户端的请求，使用功能码域（Function code）来区分正确应答和错误应答，具体帧格式见下文。

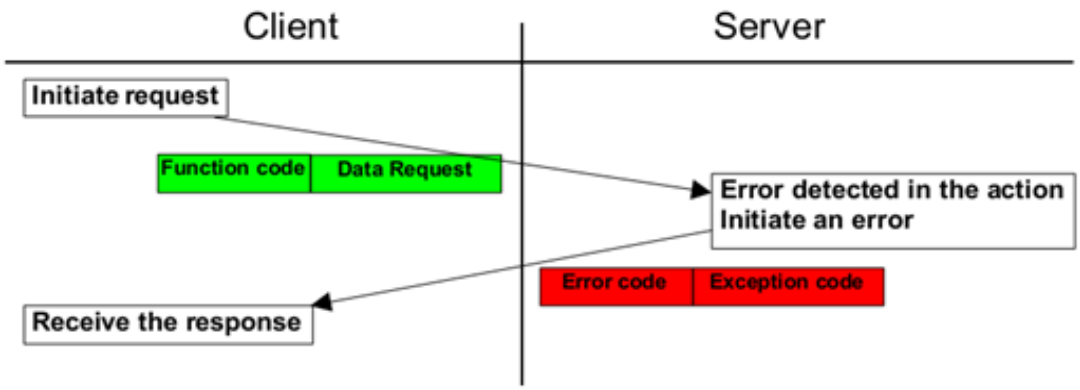
1. 正确响应

如果是正确的应答，只复制原始的功能码。



2. 出错响应

对于出错响应，服务器将功能码的最高位设置为 1，返回。异常码（Exception Code）表示错误类型。



3. 数据帧

Modbus 最初在串行链路上实现，对于串行链路来说 PDU = 256 - Server adress (1 byte) - CRC (2 bytes) = 253 bytes。

RS232 / RS485 ADU = 253 bytes + Server adress (1 byte) + CRC (2 bytes) = 256 bytes.

- 请求 PDU

- 应答 PDU

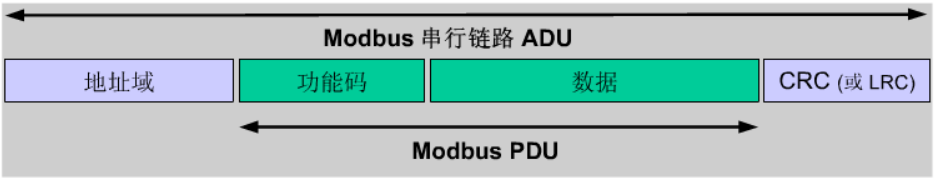
- 异常响应 PDU

4. 编码规则

通信采用大端编码规则，例如 发送寄存器值 0x1234，首先发送 0x12，然后发送 0x34。

2.3 Modbus 实现模型

Modbus 串行链路协议是一个主/从协议，在串口通信上每个从站必须有一个唯一的地址，范围：1~247，地址 0 为广播模式，不需要响应。通信发起方为主站，从站属于被动应答方，不主动发送数据。



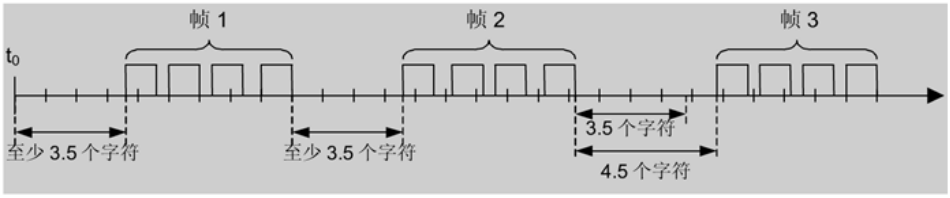
SPI 通信模式为主机先发送命令码，等待帧间隔时间和从机处理命令时间，然后主机发送无效数据（0xFF）获取应答。

2.3.1 Modbus RTU 通信模式(UART)

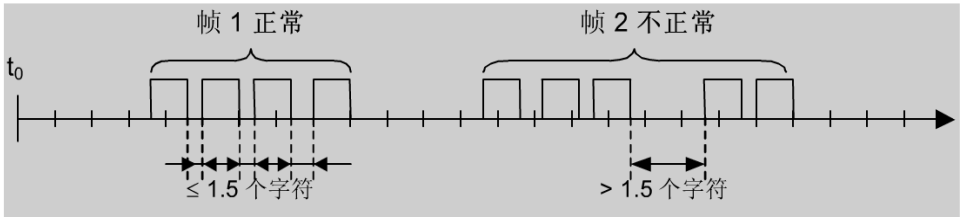
下图为 Modbus RTU 通信模式的报文格式，UART 接口下。

从站地址	功能码	数　据	CRC
1 Byte	1 Byte	0~252 Bytes	2 Bytes 低位 | 高位

报文帧的标识，帧与帧之间至少存在 3.5 个字符的间隔时间，当波特率大于 19200bps，帧间隔时间为 1800us，低于 19200 bps 时间可波特率计算得出。

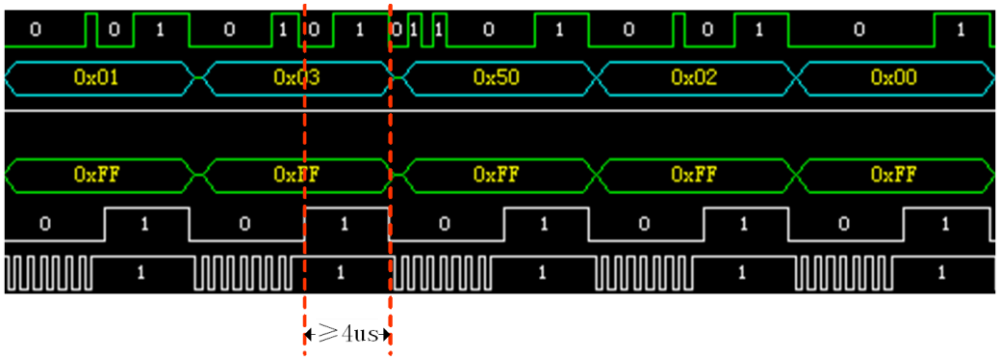


同时字符之间也有明确的时间要求。



2.3.2 Modbus RTU 通信模式(SPI)

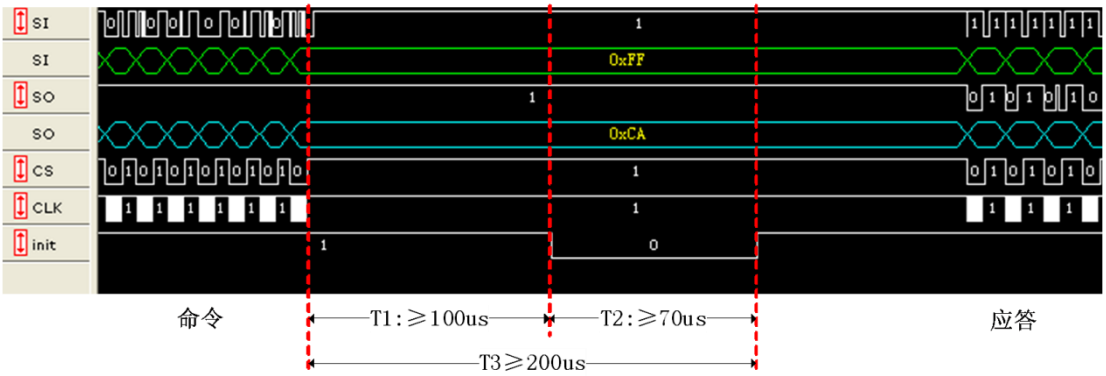
下图为 XGate-HS 模块 SPI 接口下的 Modbus RTU 通信模式的报文格式。每个报文字符间隔时间必须应大于 4us 且小于帧间隔时间。



下图为 SPI 接口的通信流程。主机先发送命令，发送命令结束后，等待大于 T3 的时间，然后发送 0xFF 获取应答。其中 T1 为帧间隔时间，用于从机区分帧的标识，模块目前支持最小 T1 时间为 100us。

T2 为从机处理主机命令和准备应答时间，需要主机等待≥70us。

T3 为主机需要等待的最短时间，为了保证通信的准确率和稳定性，建议 T3＞T1+T2+30us。



例：往寄存 0x5001 发送启动命令 0x0001（Normal Operation Mode）。

UART 接口：写	SPI 接口：写
命令：01 06 50 01 00 01 08 CA	命令：01 06 50 01 00 01 08 CA　FF FF FF FF FF FF FF
响应：01 06 50 01 00 01 08 CA	响应：FF FF FF FF FF FF FF FF　01 06 50 01 00 01 08 CA

在 UART 接口模式下：主机发送 01 06 50 01 00 01 08 CA，然后等待从机发送应答。

在 SPI 接口模式下：主机首先发送命令 01 06 50 01 00 01 08 CA（T1≥字符间隔≥4us），同时从机回复 FF FF FF FF FF FF FF FF，然后主机等待 T3 时间，再发送 FF FF FF FF FF FF FF FF（T1≥字符间隔≥4us）命令，获取从机应答 01 06 50 01 00 01 08 CA。

3. 配置信息

从用户应用程序角度来看，所有的响应设置和数据都通过调用“参数”。用户可以通过配置接口或者应用程序通信接口来读写。

参数根据用途可以分为三类。

- 通用模块参数

用于配置和获取状态信息，适用于所有的 XGate 模块；

- I/O 参数

指定数据长度和 I/O 映射，适用于所有的 XGate 模块；

- 现场总线特定参数

由特定的现场总线定义，更多信息参考附录不同现场总线。

本文档主要阐述现场总线特定参数，通用参数和 I/O 参数见《高速 XGate 系列模块通用用户手册.doc》。

3.1 通用参数特别说明

本模块中，模块处于操作状态为上线，停止状态表示下线，预操作状态则为空闲。

3.2 LED 指示

按照 CANopen 协议规范文档 DS303-3 的定义，在 XGate-COP10 模块中使用两个 LED 指示灯来指示当前模块所处的状态，如表 1 所示。

表 1 LED 状态说明

指示灯状态	现象描述
亮（LED on）	常亮
暗（LED off）	常暗
闪烁（LED flickering）	亮和暗的时间等长，频率大概是 10Hz：亮大约 50ms，暗大约 50ms
闪烁（LED blinking）	亮和暗的时间等长，频率大概是 2.5Hz：亮大约 200ms，暗大约 200ms
闪一下（LED single flash）	一个很短的闪光（大约 200ms）接着是长时间的暗（大约 1000ms）
闪两下（LED double flash）	两个很短的闪光（大约 200ms）中间用一个大约 200ms 的暗来分隔。这个序列用一个长时间的暗（大约 1000ms）来结束
闪三下（LED triple flash）	三个很短的闪光（大约 200ms）中间用大约 200ms 的暗来分隔。这个序列用一个长时间的暗（大约 1000ms）来结束

表 2 错误状态指示灯（ERR）描述

编号	ERROR LED	状态	描述	种类
1	暗	没有错误	器件处于工作状态	强制
2	闪一下	到达警戒值	CAN 控制器的至少一个错误计数器到达或超出了警戒值（错误帧太多）	强制

续上表

编号	ERROR LED	状态	描述	种类
3	闪烁（Flickering）	自动波特率 / LSS	正在进行自动波特率检测或进行 LSS 服务（和 RUN LED 交替闪烁（flickering）	可选
4	闪两下	错误控制事件	发生保护事件（NMT 从机或 NMT 主机）或心跳事件（心跳使用者）	强制
5	闪三下	Sync 错误	SYNC 报文超出配置的通讯循环间隔仍未收到（见对象字典条项 0x1006）	有条件
6	亮	总线关闭	CAN 控制器总线关闭	强制

表 3　运行状态指示灯（RNU）描述

编号	RUN LED	状态	描述	种类
1	闪烁（Flickering）	自动波特率 / LSS	正在进行 LSS 服务	可选
2	闪一下	停止	器件处于停止状态	强制
3	闪烁（Blinking）	预操作	器件处于预操作状态	强制
4	亮	工作	器件处于工作状态	强制
5	暗	故障	请检查模块复位引脚以及电源是否连接正确	

3.3　现场总线参数

本参数表用于配置模块的通用参数。

No.	Modbus Address	Name	Size(B)	Default	Access
101	0x7001	FB Bus Status	2	-	R
102	0x7002	FB Module Status	2	-	R
103	0x7003	FB Password	2	12 34	W
104	0x7004	FB Node Addr Cfg	2	1	R/W
105	0x7005	FB Node Addr Actual	2	-	R
106	0x7006	FB BaudRate Cfg	2	4	R/W
107	0x7007	FB BaudRate Actual	2	-	R
108	0x7008	FB Revision	2	05 26	R/W
109	0x7009	FB Status Bits	2	-	R
110	0x700A	FB Cfg Bits	2	-	R
111	0x700B	FB VendorID	4	00 00 02 B6	R/W
112	0x700D	FB Product Code	4	00 00 01 00	R/W
113	0x700F	FB SerialNumber	4	-	R/W
114	0x7011	FB Product Name	32	“XGate-COP20”	R/W

续上表

No.	Modbus Address	Name	Size(B)	Default	Access
115	0x7021	FB Emcy Code	4	-	R/W
116	0x7023	ErrorMsg	64	{00}	R/W
117	0x7043	FB Time Stamp	6	{00}	R/W

FB Bus Status (#101)

该参数用于检测当前现场总线状态。

参数编码	101
Modbus 地址	0x7001
默认值	0x0000
范围	Bit field
大小	2 Bytes
存储	否
操作	读

- **0x0000 - NoError** 总线正常
- **0x0001 - WarningLimitSet** 警告触发置位
- **0x0002 - WarningLimitReset** 警告触发复位
- **0x0004 - ErrorPassiveSet** 被动错误置位
- **0x0008 - ErrorPassiveReset** 被动错误复位
- **0x0010 - BusOff** 总线关闭
- **0x0020 - Overrun** 缓冲区溢出
- **0x0040 - StuffError** 填充错误
- **0x0080 - FormError** 帧格式错误
- **0x0100 - AckError** 应答错误
- **0x0200 - CrcError** CRC 校验错误
- **0x0400 - RxBuffHighOverrun** 接收缓冲区高区溢出
- **0x1000 - RxBuffLowOverrun** 接收缓冲区低区溢出
- **0x4000 - BusOffReset** Bussoff 复位
- **0x8000 - FirstBusContact** 第一次连接上总线

例：读取总线状态。

UART 接口：读	SPI 接口：读
命令：01 03 70 01 00 01 CF 0A	命令：01 03 70 01 00 01 CF 0A　FF FF FF FF FF FF FF
响应：01 03 02 00 00 B8 44	响应：FF FF FF FF FF FF FF FF　01 03 02 00 00 B8 44

返回

FB Module Status (#102)

本参数指示基于现场总线模块的当前状态信息。

参数编码	102
Modbus 地址	0x7002
默认值	-
范围	-
大小	2 Bytes
存储	NO
操作	R

- **0x00 - Initialisation**

初始化状态。

- **0x04 - Stopped**

停止状态。

- **0x05 - Operational**

操作状态。

- **0x7F - PreOperational**

预操作状态 。

例：读取模块状态。

UART 接口：读	SPI 接口：读
命令：01 03 70 02 00 01 3F 0A	命令：01 03 70 02 00 01 3F 0A　FF FF FF FF FF FF FF
响应：01 03 02 00 00 B8 44	响应：FF FF FF FF FF FF FF FF　01 03 02 00 00 B8 44

返回

FB Password (#103)

本参数用于保护一些特殊的参数不被意外修改。这些参数只有先输入密码，才能修改。被保护的参数有：FB VendorID、FB Product Code、FB SerialNumber、 FB Product Name。

参数编码	103
Modbus 地址	0x7003
默认值	0x1234
范围	0x1234
大小	2 Bytes
存储	YES
操作	W

例：输入密码。

UART 接口：写
命令：01 06 70 03 12 34 6E 7D
响应：01 06 70 03 12 34 6E 7D

SPI 接口：写
命令：01 06 70 03 12 34 6E 7D FF FF FF FF FF FF FF FF
响应：FF FF FF FF FF FF FF FF 01 06 70 03 12 34 6E 7D

返回

FB Node Address Config (#104)

本参数用于保存用户配置 CANopen 节点地址，启动命令（#1 命令）后配置本参数，复位有效，因此会出现配置址可能与实际使用地址（#105）不一致情况。当设置为 0 时，采用 LSS 协议配置本机地址。

参数编码	104
Modbus 地址	0x7004
默认值	0x01
范围	0x00 ~ 0x7F
大小	1 Bytes
存储	YES
操作	R/W

例：设置新节点地址 2。

UART 接口：写
命令：01 06 70 04 00 02 53 0A
响应：01 06 70 04 00 02 53 0A

SPI 接口：写
命令：01 06 70 04 00 02 53 0A FF FF FF FF FF FF FF FF
响应：FF FF FF FF FF FF FF FF 01 06 70 04 00 02 53 0A

返回

FB Node Address Actual (#105)

本参数指示实际网络中 CANopen 节点地址。

参数编码	105
Modbus 地址	0x7005
默认值	-
范围	0x00 ~ 0x7F
大小	1 Bytes
存储	NO
操作	R

例：读取实际节点地址，只有模块启动时才能正确获取，否则获取为 0。

UART 接口：读
命令：01 03 70 05 00 01 8E CB
响应：01 03 02 00 02 39 85

SPI 接口：读
命令：01 03 70 05 00 01 8E CB FF FF FF FF FF FF FF
响应：FF FF FF FF FF FF FF FF 01 03 02 00 02 39 85

返回

FB BaudRate Config (#106)

本参数用于保存用户配置 CANopen 网络波特率，启动命令（#1 命令）后配置本参数，复位有效，因此会出现本参数可能与实际使用参数（#107）不一致情况。

参数编码	106
Modbus 地址	0x7006
默认值	0x04
值范围	0x00 ~ 0x08
大小	1Bytes
存储	YES
操作	R/W

- 0： 1 MBit/sec
- 1： 800 kBit/sec
- 2： 500 kBit/sec
- 3： 250 kBit/sec
- 4： 125 kBit/sec
- 5： 100 kBit/sec
- 6： 50 kBit/sec
- 7： 20 kBit/sec
- 8： 10 kBit/sec

例：读取配置的波特率。

UART 接口：读
命令：01 03 70 06 00 01 7E CB
响应：01 03 02 00 04 B9 87

SPI 接口：读
命令：01 03 70 06 00 01 7E CB FF FF FF FF FF FF FF
响应：FF FF FF FF FF FF FF FF 01 03 02 00 04 B9 87

返回

FB BaudRate Actual （#107）

用于指示 CANopen 网络中实际使用的波特率值。

参数编码	107
Modbus 地址	0x7007
默认值	-
值范围	0x00 ~ 0x08
大小	1Bytes
存储	NO
操作	R

- 0： 1 MBit/sec
- 1： 800 kBit/sec
- 2： 500 kBit/sec
- 3： 250 kBit/sec
- 4： 125 kBit/sec
- 5： 100 kBit/sec
- 6： 50 kBit/sec
- 7： 20 kBit/sec
- 8： 10 kBit/sec

例：读取实际的波特率。

UART 接口：读
命令：01 03 70 07 00 01 2F 0B
响应：01 03 02 00 04 B9 87

SPI 接口：读
命令：01 03 70 07 00 01 2F 0B FF FF FF FF FF FF FF
响应：FF FF FF FF FF FF FF FF 01 03 02 00 04 B9 87

返回

FB Revision (#108)

本参数保存模块的 CANopen 版本信息。

参数编码	108
Modbus 地址	0x7008
默认值	0x05 26
值范围	0x0000 ~ 0xFFFF
大小	2Bytes
存储	YES
操作	R/W

例：读取版本信息。

UART 接口：读
命令：01 03 70 08 00 01 1F 08
响应：01 03 02 26 05 62 27

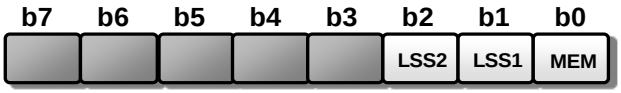
SPI 接口：读
命令：01 03 70 08 00 01 1F 08 FF FF FF FF FF FF FF
响应：FF FF FF FF FF FF FF FF 01 03 02 26 05 62 27

返回

FB Status Bits (#109)

本参数指示当前波特率和节点 ID 配置方式。

参数编码	109
Modbus 地址	0x7009
默认值	0x0001
范围	Bit field
大小	1 Bytes
存储	NO
操作	R



- MEM

  1：当前参数从非易失性存储器中获取；　　0：不是非易失性存储器中获取；

- LSS1

  1：节点地址由 LSS 设置；　　0：节点地址不是通过 LSS 设置；

- LSS2

  1：波特率地址由 LSS 设置；　　0：波特率不是通过 LSS 设置；

例：读取波特率和节点 ID 配置方式。

UART 接口：读
命令：01 03 70 09 00 01 4E C8
响应：01 03 02 00 01 79 84

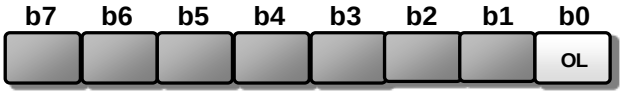
SPI 接口：读
命令：01 03 70 09 00 01 4E C8　　FF FF FF FF FF FF FF
响应：FF FF FF FF FF FF FF FF　　01 03 02 00 01 79 84

返回

FB Config Bits (#110)

本参数设置现场总线下线方式，本版本中此参数配置无效。

参数编码	110
Modbus 地址	0x700A
默认值	-
范围	Bit field
大小	1 Bytes
存储	YES
操作	R



返回

FB VendorID (#111)

本参数用于存储模块的厂商 ID，受密码保护，须先输入密码才能更改，见（命令#103）。

参数编码	111
Modbus 地址	0x700B
默认值	0x00 00 02 B6
范围	0x0000 0000 ~ 0xFFFF FFFF
大小	4 Bytes
存储	YES
操作	R/W

例：读取厂商 ID。

UART 接口：读
命令：01 03 70 0B 00 02 AF 09
响应：01 03 04 00 00 02 B6 7A E5

SPI 接口：读
命令：01 03 70 0B 00 02 AF 09　FF FF FF FF FF FF FF FF FF
响应：FF FF FF FF FF FF FF FF　01 03 04 00 00 02 B6 7A E5

返回

FB Product Code (#112)

本参数用于存储产品代码，受密码保护，须先输入密码才能更改，见（命令#103）。

参数编码	112
Modbus 地址	0x700D
默认值	0x00 00 01 00
范围	0x0000 0000 ~ 0xFFFF FFFF
大小	4 Bytes
存储	YES
操作	R/W

例：读取产品代码。

UART 接口：读
命令：01 03 70 0D 00 02 4F 08
响应：01 03 04 00 00 01 00 FB A3

SPI 接口：读
命令：01 03 70 0D 00 02 4F 08　FF FF FF FF FF FF FF FF FF
响应：FF FF FF FF FF FF FF FF 01 03 04 00 00 01 00 FB A3

返回

FB Serial Number (#113)

本参数用于存储产品序列号，受密码保护，须先输入密码才能更改，见（命令#103）。

参数编码	113
Modbus 地址	0x500F
默认值	0x00 00 00 01
范围	0x0000 0000 ~ 0xFFFF FFFF
大小	4Bytes
存储	YES
操作	R/W

例：读取产品序列号。

UART 接口：读
命令：01 03 70 0F 00 02 EE C8
响应：01 03 04 00 00 00 01 3B F3

SPI 接口：读
命令：01 03 70 0F 00 02 EE C8　FF FF FF FF FF FF FF FF FF
响应：FF FF FF FF FF FF FF FF 01 03 04 00 00 00 01 3B F3

返回

FB Product Name (#114)

本参数用于存储产品名称，受密码保护，须先输入密码才能更改，见（命令#103）。

参数编码	114
Modbus 地址	0x5011
默认值	“XGate-COP20”
范围	-
大小	32 Bytes
存储	YES
操作	R/W

例：读取产品名称。

UART 接口：读
命令：01 03 70 11 00 06 8F 0D
响应：01 03 0C 47 58 74 61 2D 65 4F 43 32 50 00 30 19 43

返回

FB EmcyCode (#115)

本参数用于用户发送紧急代码，当用户设备出现某种错误之后，可通过 CANopen 发送到 CAN 总线上，通知 CANopen 主站设备当前设备发生了错误。本参数包括四个字节，格式为：标准错误码（2Bytes）+用户自定义错误码（2Bytes），紧急错误代码参考 DS301 V4.02。

参数编码	115
Modbus 地址	0x5021
默认值	0x00 00 00 00
范围	0x0000 0000 ~ 0xFFFF FFFF
大小	4 Bytes
存储	NO
操作	W

例：用户发送紧急代码。

UART 接口：写

命令：01 10 70 21 00 02 04 80 00 00 01 BD B9

响应：01 10 70 21 00 02 0B 02

SPI 接口：写

命令：01 10 70 21 00 02 04 80 00 00 01 BD B9 FF FF FF FF FF FF FF FF

响应：FF FF FF FF FF FF FF FF FF FF FF FF FF 01 10 70 21 00 02 0B 02

返回

FB Error Message (#116)

本参数用于存储错误消息（紧急代码）和模块的异常事件。本模块只使用 24 字节(保存 4 个错误记录)，数据格式为：索引（2 个字节） + 错误码（4 个字节）。

参数编码	116
Modbus 地址	0x5023
默认值	-
范围	-
大小	64 Bytes
存储	YES
操作	R

例：略。

返回

FB Time Stamp (#117)

本参数用获取网络时间标识。该时间格式遵循 DS301 V4.02 的定义。

在 CANopen 网络中为了同步所有节点的时间，CANopen 主机会向网络中发送同步时间标识对象。

参数编码	117
Modbus 地址	0x5043
默认值	-
范围	0x 00 00 00 00 00 00 ~ 0xFF FF FF FF FF FF
大小	6Bytes
存储	NO
操作	R

例：当主机发送报文（ID：0x100　Data：0x7C 0x77 0x37 0x02 0x52 0x25）更新网络时间，该时间为以 1984 年 1 月 1 日为起点的相对时间。数据区意思为相对于 1984 年 1 月 1 日为 0x2552 天，0x0237777C 毫秒，可算出当前时间为 2010 年 2 月 27 日，10 时 19 分 49 秒。

命令：0x01 0x03 0x70 0x43 0x00 0x03 0xEE 0xDF

响应：0x01 0x03 0x06 0x77 0x7C 0x02 0x37 0x25 0x52 0x51 0x23

当未接收到网络时间或没有更新网络时间时，其响应命令码如下所示。

错误响应：0x01 0x83 0x04 0x40 0xF3

返回

4. 免责声明

本文档提供有关致远电子产品的信息。本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除致远电子在其产品的销售条款和条件中声明的责任之外，致远电子概不承担任何其它责任。并且，致远电子对致远电子产品的销售和／或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。致远电子产品并非设计用于医疗、救生或维生等用途。致远电子可能随时对产品规格及产品描述做出修改，恕不另行通知。

该系列产品可能包含某些设计缺陷或错误，一经发现将收入勘误表，并因此可能导致产品与已出版的规格有所差异。如客户索取，可提供最新的勘误表。

在订购产品之前，请您与当地的致远电子销售处或分销商联系，以获取最新的规格说明。本文档中提及的含有订购号的文档以及其它致远电子文献可通过访问广州致远电子股份有限公司的万维网站点获得，网址是：

http://www.embedcontrol.com/ 或致电+86-20-22644249 查询。

附录A CRC 校验过程

在 RTU 模式包含一个对全部报文内容执行的，基于循环冗余校验 (CRC - Cyclical Redundancy Checking) 算法的错误检验域。CRC 域检验整个报文的内容。不管报文有无奇偶校验，均执行此检验。

CRC 包含由两个 8 位字节组成的一个 16 位值。CRC 域作为报文的最后的域附加在报文之后。计算后，首先附加低字节，然后是高字节。CRC 高字节为报文发送的最后一个子节。

附加在报文后面的 CRC 的值由发送设备计算。接收设备在接收报文时重新计算 CRC 的值，并将计算结果于实际接收到的 CRC 值相比较。如果两个值不相等，则为错误。

CRC 的计算，开始对一个 16 位寄存器预装全 1，然后将报文中的连续的 8 位字节对其进行后续的计算。只有字符中的 8 个数据位参与 CRC 的运算，起始位，停止位和校验位不参与 CRC 计算。

CRC 的生成过程中，每个字节（8 位）与寄存器中的值异或，然后结果向最低有效位(LSB)方向移动(Shift) 1 位，而最高有效位(MSB)位置充零。然后提取并检查 LSB；如果 LSB 为 1，则寄存器中的值与一个固定的预置值异或；如果 LSB 为 0，则不进行异或操作。

这个过程将重复直到执行完 8 次移位。完成最后一次（第 8 次）移位及相关操作后，下一个 8 位字节与寄存器的当前值异或，然后又同上面描述过的一样重复 8 次。当所有报文中字节都运算之后得到的寄存器中的最终值，就是 CRC。

当 CRC 附加在报文之后时，首先附加低字节，然后是高字节。

实现代码参照提供的例