初级篇

第 1 章“提纲，挈领”

TTCANopen 应用层协议框架

1.1 TTCANopen 协议总则

TTCANopen 开篇从六个方面对协议进行规范。

第一，郑重声明：由于 TTCANopen 网络系统时间的脆弱和不抗攻击性，TTCANopen 不适用于矿山机械、银行、国防军工以及一切涉及到人身安全、重大财产安全和国家军事安全的领域。

第二， TTCANopen 是一个开放的 CAN 应用层协议框架，用户可以根据自己的应用领域制定出适用本专业环境的应用子协议。这些协议需遵守必要的协议编写规范，并通过 TTCANopen 组织测试验证后，方可成为 TTCANopen 协议的组成部分进行发布。为了保持协议的唯一性，用户或其他组织不能够独立发布 TTCANopen 协议和协议的再生、衍生品，TTCANopen 组织是唯一的 TTCANopen 协议发布者。对于那些基于 TTCANopen 的专利、收费子协议和衍生品，TTCANopen 组织有权对其征收基础协议使用费。

第三， TTCANopen 组织发布的 TTCANopen 协议，任何个人和组织均可免费获得，用户可以从 www.ttcanopen.com 网站下载最新的 TTCANopen 协议，并进行传播。为了使 TTCANopen 协议能够持续健康的发展，TTCANopen 对批量商业化生产收取一定的基础协议使用费，对年产一万件（端口）以下的应用和开发免征基础协议使用费。

第四，各子协议的设计要求为：直读性强、易于理解、易于实现。这是 TTCANopen 协议设计的初衷，也是协议的生命力所在，过于复杂的协议过程将被拒绝。

第五，免责声明：TTCANopen 组织只负责维护、验证和发布协议，对用户使用中产生的任何损失均不负有责任。

产生错误，出现损失是任何人都不愿看到的，但在协议的使用过程中，难免会出现这样或那样的问题，有些是协议本身出现了漏洞，有些是对协议的理解出现了偏差，有些是协议

适用的场合不对应，又或者是某些硬件出现问题等等，因此强烈建议用户对系统进行全面的安全认证和必要的冗余设计，在涉及到人身安全、重大财产安全、国防安全领域必须慎之又慎或者禁止使用，毕竟 CAN 只是一个轻量级的低成本现场总线，安全可靠等级上并非无懈可击。

TTCANopen 组织会努力修复协议中的漏洞，对新加入的子协议进行认真测试，同时也热切的盼望广大读者、用户和组织参与进来，TTCANopen 组织将不胜荣幸和感激您的光临，为此我们在 www.ttcanopen.com 网站设立了读者论坛。

第六，TTCANopen 组织实行会员制，TTCANopen 组织首先将新的子协议在会员中进行测试，然后再进行公测。会员在享有会章规定的权利的同时，也应当尽到会章规定的各项义务，TTCANopen 组织为每个会员分配一个唯一的专用 CVID 编号和一个 16 比特的 CPID 供会员使用（参考 USB [1] 中的 VID 和 PID 的相关规范），并将其通过测试的 TTCANopen 产品纳入数据库管理，为 TTCANopen 智能检索系统提供数据支撑。

1.2 TTCANopen 子协议命名

TTCANopen 协议是由众多的“基本子协议”（TC）和“应用子协议”（YTC）组成的，用户可根据其应用环境选择已有的“应用子协议”或利用“基本子协议”重新组织生成适用的“应用子协议”，新生成的应用子协议需经 TTCANopen 组织认证，并由 TTCANopen 组织统一发布。

各“基本子协议”用“TC”标记开头，后跟一个 4 位数字（取值 0000～9999），表示子协议编号，然后跟一个大写字母和一个三位数字（取值 000～999），大写字母标识协议的进展阶段，A 为内部测试阶段，B 为会员测试阶段，C 为公测阶段，D 为稳定阶段，三位数字表示子协议扩展或修订编号，最后是子协议标题。

一个完整的“基本子协议”命名为：**TTCANopen 之 TC0016A000 0xF000 段的分配使用**表述这是一个编号为 0016 的 TTCANopen 系列“基本子协议”，处于内部测试阶段，扩展或修订编号为 000，子协议标题为：0xF000 段的分配使用。

编号为 0000 的“基本子协议”是 TTCANopen 系列各子协议的根协议，是其他子协议的前导和依据文件。

各子协议如存在“继承”关系的应当明确注明，在协议中可不必重复描述“继承内容”，

只需描述其“扩展”内容即可。

编号为 0001～0999 的协议是 TC 系列子协议中的“简单”功能协议族。

编号为 1000～1999 的协议是 TC 系列子协议中的“基本类”功能协议族。

编号为 2000～2999 的协议是 TC 系列子协议中的“智能”功能协议族。

编号为 3000～4999 的协议是 TC 系列子协议中的“专业领域”功能协议族。

编号为 5000～6999 的协议是 TC 系列子协议中的“应用厂家”功能协议族。

编号为 7000～9999 的协议是 TC 系列子协议备用序列。

各“基本子协议”是应用协议的组成元素，每个“基本子协议”只描述简单的基本协议内容和单一的行为功能。

各“应用子协议”用 YTC 标记开头，编号规则与“基本子协议”类似，后跟一个 4 位数字（取值 0000～9999），表示子协议编号，然后跟一个大写字母和三位数字（取值 000～999），大写字母标识协议的进展阶段，A 为内部测试阶段，B 为会员测试阶段，C 为公测阶段，D 为稳定阶段，三位数字表示应用子协议扩展或修订编号，最后是应用子协议标题。

一个完整的“应用子协议”命名为：**TTCANopen 之 YTC0001A000 简单的配置应用之一**

表述这是一个编号为 0001 的 TTCANopen 系列“应用子协议”，处于内部测试阶段，扩展或修订编号为 000，应用子协议的标题为：简单的配置应用之一。

“应用子协议”是根据应用环境，将多个“基本子协议”集合在一起，并对其进行适度的扩充和裁剪，生成一个与应用环境相匹配的终极使用协议。

编号为 0001～0999 的协议是 YTC 系列子协议中的 “简单 应用子协议”族。

编号为 1000～1999 的协议是 YTC 系列子协议中的 “基本类应用子协议”族。

编号为 2000～2999 的协议是 YTC 系列子协议中的 “智能应用子协议”族。

编号为 3000～4999 的协议是 YTC 系列子协议中的“专业领域应用子协议”族。

编号为 5000～6999 的协议是 YTC 系列子协议中的“应用厂家子协议”族。

编号为 7000～9999 的协议是 YTC 系列子协议备用序列。

本书针对典型常用的“子协议”，由简入繁进行讲述。

1.3 TTCANopen 协议基础

CAN 应用层协议其中最主要的工作之一就是定义和规划 CAN 标识符的使用。

CANopen 和 DeviceNet 两个协议都使用了 11 位 CAN 标识符，其 ID 资源受限，CANopen 采用了对象词典的方法，DeviceNet 采用了动态“对象”分配的方式来弥补 ID 资源的匮乏，其协议指令不具备直读性，并将协议内容伸展到了“数据区”，使解析和维护协议的复杂度提高了许多，对于普通的工程师来讲，解析协议的过程就如同经历了一场编程噩梦。

为此，TTCANopen 使用了具有 29 位 ID 资源的 CAN2.0b 协议扩展帧格式规范，并将其 29 位 ID 资源固定分割为 4 个有实际含义的子段“优先级段”、“寄存器基地址段”、“ 设备编号段”和“功能码段”，使得 TTCANopen 应用层协议具有很强的直读性，为协议的解析和维护铺平了道路，使设计者从繁琐的协议解析中解脱出来，将更多的精力放在系统集成和测试中来。

1.3.1 TC0000A000 根协议（ID 划分）

在 TTCANopen 之 TC0000A000 中定义了协议的基本根内容，主要包括 ID 的划分、寄存器空间和设备编号的分配等。

1 TTCANopen 指令格式定义

TTCANopen 采用 CAN2.0b 协议扩展帧格式，其中 29 位 ID 划分见图 1－1，

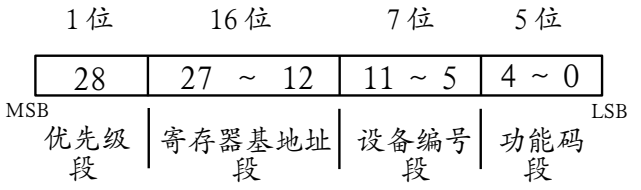

图 1－1： TTCANopen ID 划分方案

优先级段：最高位 28 位，为 1 位指令“优先级”，取值 0x0 或 0x1，值越小优先级越高，协议将 CAN 指令按紧急程度分为两类，一类是普通指令；一类是紧急指令。应用层协议使用优先级为“0x1”发送普通指令，使用优先级为“0x0”发送紧急指令。

寄存器基地址段： 27～12 位，是一个 16 位寄存器寻址参数，取值范围 0x0000 至 0xFFFF，值越小其优先级越高，每个寄存器地址对应一个字节寄存器，应用层协议将数据存放在对应寄存器中；一条 CAN 指令其数据场可携带不多于 8 个字节的数据（对于 CAN FD 协议为不多于 64 个字节的数据)，寄存器基地址标识的是 CAN 指令数据场中第 1 个数据字节的地址，其它字节的地址依序递增。

设备编号段：11～5 位，为 7 位设备在总线上的编号（或称设备地址），取值范围 0x00~0x7F，可提供 128 个编号，值越小其优先级越高 ，其中：0x00 为广播地址。

功能码段：4～0 位，为 5 位“功能编码”，取值范围 0x00~0x1F，可提供 32 条指令，值越小其优先级越高。

指令的优先级最终由 29 位 ID 值决定，值越小优先级越高（显性电平为 0）。

基本指令格式表述：使用 CAN2.0b 协议扩展帧格式的数据帧

优先级	寄存器基地址	设备地址	功能码	DLC	0～8 个字节数据

指令格式中的优先级对应优先级段 (28 位)，寄存器基地址对应寄存器基地址段(27~12 位)，设备地址对应设备编号段(11 ~5 位)，功能码对应功能码段(4~0 位)，DLC 表示指令数据场携带的数据字节数，最后是 0～8 字节的数据场。

注：对于 CAN FD 标准 ，其对数据长度编码进行了扩展，当数据长度大于 8 时，使用了非线性编码，见表 1－1 阴影部分。其指令传输数据场字节数也相应非线性增加。

表 1－1 CAN FD 中 DLC 编码对应的数据长度

DLC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
数据长度值	0	1	2	3	4	5	6	7	8	12	16	20	24	32	48	64

2 寄存器空间的划分

协议为系统和设备定义了一个 16 位寄存器寻址空间，每一个地址对应一个 8 位寄存器，凡是需要参与或可能参与 CAN 总线通讯的参数和变量，都要使用或映射到该寄存器空间，寄存器空间分配见图 1－2， 其中：0x0000~0x6FFF 定义为“系统信息空间”；0x7000~0x7FFF

定义为“设备信息空间”；0x8000~0xFFFF 定义为“配置参数空间”。

“系统信息空间”是 TTCANopen 网络系统的全局信息映射的寄存器空间；“设备信息空间”是各设备自身信息存放的寄存器空间；“配置参数空间”是各设备配置参数存放的寄存器空间。

TTCANopen 在“系统信息空间”使用面向报文的通讯协议；在“设备信息空间”和“配置参数空间”使用面向节点的通讯协议。

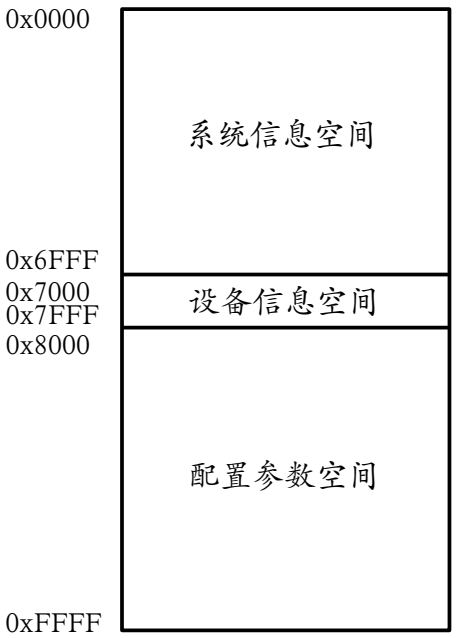

图 1－2 寄存器空间分配

为了便于对寄存器空间的使用和管理，TTCANopen 将寄存器空间划分为 16 个区段，即：

第 0x0000 段：0x0000～0x0FFF TTCANopen 组织规划，分配给共性系统过程变量。

第 0x1000 段：0x1000～0x1FFF 保留

第 0x2000 段：0x2000～0x2FFF 保留

第 0x3000 段：0x3000～0x3FFF 保留

第 0x4000 段：0x4000～0x4FFF 保留

第 0x5000 段：0x5000～0x5FFF 保留

第 0x6000 段：0x6000～0x6FFF 保留

第 0x7000 段：0x7100～0x7FFF 分配给设备信息过程变量。

第 0x8000 段：0x8000～0x8FFF 分配给设备特征配置参数。

第 0x9000 段：0x9000～0x9FFF 保留

第 0xA000 段：0xA000～0xAFFF 保留

第 0xB000 段：0xB000～0xBFFF 保留

第 0xC000 段：0xC000～0xCFFF 保留

第 0xD000 段 0xD000～0xDFFF 保留

第 0xE000 段：0xE000～0xEFFF 保留

第 0xF000 段：0xF000～0xFFFF TTCANopen 组织规划，分配给设备共性配置参数。

其中：第 0x0000 段至第 0x7000 段为“过程变量”所使用；第 0x8000 段至第 0xF000 段为“配置参数”所使用。各段具体定义见相关“基本子协议”。

3 字节传送顺序

在指令中寄存器基地址为一个 16 位数据，用两个字节表述，如：0x1234 表示为两个字节 0x12 0x34，高字节在前 0x12，低字节在后 0x34。

指令携带的数据按寄存器地址先后顺序传送，由低到高，无论数据表述的是单字节数据还是多字节数据。

4 兼容性要求

为了兼容早期的协议，指令的前 7 位不能出现连续隐性电平（1）。

5 对 CAN FD 的支持

支持 CAN FD 协议中对数据场的扩展。

1.3.2 0xF000 段的分配使用（TC0016）

设备 0xF000 段寄存器分配给各设备共有的通用配置参数使用，如：设备厂家参数、设备诊断参数、设备心跳参数、设备协议参数、设备状态参数、设备地址和通讯速率等，每一个设备都拥有一个属于其自身的 0xF000 段寄存器空间，该段参数空间的分配由 TTCANopen 组织统一制定，参见下表。

寄存器地址	参数类型	读写特性	参数描述	值域
0xF000	U8	R/W	段保护机制 有效值：0x88，0x99～0xFF	
0xF001～ 0xF007	7 * U8	R/W	段保护机制加密字节（可选）	
0xF008	U8	R/W	设备工作状态 0x11：复位态 0x22：配置态 0x33：运行态 0x44：停止态 0x55：静默态	0x11，0x22，0x33，0x44，0x55； 0x1E，0x2E，0x3E，0x4E，0x5E； （E 为对应安全态）
0xF009	U8	R/（W）	预状态	0x22，0x33，0x44
0xF00A	U8	R/W	设备心跳使能	0x00 或 0x11
0xF020～ 0xF02F	U8	R/W	16 个组地址存放	0x60～0x6F
0xF030～ 0xF033	U32	R/W	系统天计数	
0xF034～ 0xF037	U32	R/W	子帧周期（M）	被 864000000 整除
0xF038～ 0xF039	U16	R/W	副帧周期（P0）	用于心跳
0xF03A～ 0xF03B	U16	R/W	副帧周期（P1）	
0xF03C～ 0xF03D	U16	R/W	副帧周期（P2）	
0xF03E～ 0xF03F	U16	R/W	副帧周期（P3）	
0xF040～ 0xF043	U32	R/W	心跳子帧相位（N）	1～M
0xF044～ 0xF045	U16	R/W	心跳副帧相位（Q）	1～P0
0xF046	U8	R/（W）	设备地址	0x10～0x6F
0xF047	U8	R/（W）	通讯速率 1：10k；2：20k；3：50k 4：100k；5：125k；6：250k 7：500k；8：800k；9：1M	1～9
0xF050～ 0xF05F	U8	R/W	系统信息过程变量 写指令的地址筛选	设备地址值
0xFA10～ 0xFA13	U32	R	设备心跳时间	0～863999999
0xFA14	U8	R	接收错误计数	0～128
0xFA15	U8	R	发送错误计数	0～128
0xFA16	U8	R	心跳失步计数	0～0xFF

0xFA17	U8	R	设备工作状态同期映射	
0xFA18～ 0xFA1F	U8	R	通用设备错误标识 前 4 个字节（T） 后 4 个字节（厂）	
0xFA20～ 0xFA2F	U8	R	特殊产品错误标识（可选）	
0xFBC0～ 0xFBC1	U16	R	产品领域	由各行业标准化组织向 TTCANopen 组织申请针对 0x9000 段的标准化定义
0xFBC2～ 0xFBC3	U16	R	产品分类	
0xFBC4～ 0xFBC5	U16	R	产品编序	
0xFBC6～ 0xFBC7	U16	R	产品升级标号	
0xFBC8～ 0xFBC9	U16	R	产品领域	由设备厂商针对 0x9000 段的企业标准化定义（注：不能与行业标准化同时存在）
0xFBCA～ 0xFBCB	U16	R	产品分类	
0xFBCC～ 0xFBCD	U16	R	产品编序	
0xFBCE～ 0xFBCF	U16	R	产品升级标号	
0xFBE0～ 0xFBE1	U16	R	产品适用 YTC 编号	0001～ 9999
0xFBE2～ 0xFBE3	U16	R	产品适用 YTC 修订编号	000～ 999
0xFBF0～ 0xFBF1	U16	R	产品编号	0～65535
0xFBF2	U8	R	产品硬件版本号	0～255
0xFBF3	U8	R	产品软件版本号	0～255
0xFBF4～ 0xFBF5	U16	R	厂商 CVID	0～65535
0xFBF6～ 0xFBF7	U16	R	产品 CPID	0～65535

1.3.3 0x8000 段的分配使用 （TC0009）

设备 0x8000 段寄存器分配给各设备个性特征配置参数使用，这些参数与具体设备相关，包括配置本地资源、过程变量的初始值、设备工作参数和部分状态参数等，该段参数空间分配和定义由各设备生产厂商根据具体设备进行定制， 每一个设备都拥有一个属于其自身的 0x8000 段寄存器空间。

TTCANopen 推荐将设备状态参数分配到 0x8A00 以后的寄存器地址空间当中。

1.3.4 0x7000 段的分配使用（TC0008）

设备 0x7000 段寄存器分配给各设备过程变量使用，这些变量与具体设备相关，包括设备

的各种 I/O 变量等，该段变量空间分配和定义由各设备生产厂商根据具体设备进行定制，每一个设备都拥有一个属于其自身的 0x7000 段寄存器空间。

1.3.5 TC0020A000 子协议（配置参数基本指令）

配置参数基本指令集

本子协议（通讯子协议）定义了 2 个配置参数基本指令，分别是“参数设置指令”、“参数读取指令”和每个指令对应的“应答指令”以及错误诊断应答指令，本子协议指令操作适用于“配置参数空间”即：第 0x8000 段至第 0xF000 段的参数配置。

1 参数设置指令

功能码：0x0D

“主机或设备管理器”向指定设备的配置参数寄存器基地址写入配置参数，一次可写入 1～8 个字节的参数，第 2 个字节往后为基地址的顺延。

参数设置指令格式：

优先级	寄存器基地址	设备地址	0x0D	DLC	1～8 个字节参数

例：0x1 0x8000 0x35 0x0D 0x4 0x11 0x22 0x33 0x44

由于用二进制表述比较麻烦，这里将指令用十六进制分段表述，参见 29 位 ID 划分。

第一段为“1”位优先级，取值范围：0x0～0x1，本例中的 0x1；

第二段为“16”位寄存器基地址，取值范围：0x0000～0xFFFF，本例中的 0x8000；

第三段为“7”位设备地址，取值范围：0x00～0x7F，本例中的 0x35；

第四段为“5”位功能码，取值范围：0x00～0x1F，本例中的 0x0D；

第五段为“4”位 DLC 数据长度，取值范围：0x0～0x8，本例中的 0x4；

后面为 0～8 个字节配置参数或数据，本例中的 0x11 0x22 0x33 0x44；

本例表述，主机向地址为 0x35 的设备其基地址为 0x8000 的连续 4 个寄存器写入配置参数，其为 0x11 0x22 0x33 0x44。

设备在其原“指令功能码”上加 0x10，并读取刚设置好的寄存器内容进行回复。

参数设置指令应答格式：

原优先级	原寄存器基地址	原设备地址	0x1D	原 DLC	1～8 个字节参数

例：0x1 0x8000 0x35 0x1D 0x4 0x11 0x22 0x33 0x44

当配置参数基地址不存在，或为只读时，访问寄存器范围越界，则返回错误诊断指令如何下，

错误诊断指令格式：

原优先级	原寄存器基地址	原设备地址	0x1F	0x2	0x0D	1 个字节错误编号

例：0x1 0x8000 0x35 0x1F 0x2 0x0D 0x03 （无此寄存器地址，或为只读）

0x1 0x8000 0x35 0x1F 0x2 0x0D 0x04 （寄存器地址越界）* 错误编号见附录 1。

注：当指令被诊断出错误（包括局部错误），整条指令被判无效。

2 配置参数读取指令

功能码：0x0E

“主机或设备管理器”向指定设备发出读取其指定配置参数寄存器的内容，指令带一个字节的数据，该数据表述将要连续读取配置参数的寄存器个数 n，n＝1～64，当 n 大于 8 时，设备需要使用多条指令响应。

配置参数读取指令格式：

优先级	寄存器基地址	设备地址	0x0E	0x1	请求读取的寄存器个数

例：0x1 0x8000 0x35 0x0E 0x1 0x04

本例表述，主机要读取地址为 0x35 的设备其基地址为 0x8000 的连续 4 个字节的配置参数。

设备在其原“指令功能码”上加 0x10，并在数据场添加相关参数进行回复。

配置参数读取指令应答格式：

原优先级	原寄存器基地址	原设备地址	0x1E	DLC	1～8 个字节参数

例：0x1 0x8000 0x35 0x1E 0x4 0x11 0x22 0x33 0x44

上例表述地址为 0x35 的设备回复主机的读取指令，将其基地址为 0x8000 的连续 4 个配置参数寄存器的内容 0x11 0x22 0x33 0x44 返回到主机。

又例：0x1 0x8000 0x35 0x0E 0x1 0x0E

响应 1：0x1 0x8000 0x35 0x1E 0x8 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88

响应 2：0x1 0x8008 0x35 0x1E 0x6 0x99 0xAA 0xBB 0xCC 0xDD 0xEE

当读取配置寄存器不存在或越界，指令 DLC“请求读取的寄存器个数”出错时，则返回错误诊断指令

错误诊断指令格式：

原优先级	原寄存器基地址	原设备地址	0x1F	0x2	0x0E	1 个字节错误编号

例：0x1 0x8000 0x35 0x1F 0x2 0x0E 0x3 （无此寄存器地址）

0x1 0x8000 0x35 0x1F 0x2 0x0E 0x4 （寄存器地址越界）

0x1 0x8000 0x35 0x1F 0x2 0x0E 0x2 （指令参数出错，DLC 错，或 n 不在 1～64 范围内）

配置参数读取指令的特殊用法，为了提高对配置寄存器的读取效率，我们提供了以下读取整段配置寄存器的指令。

整段配置参数读取指令格式：

优先级	寄存器段地址	设备地址	0x0E	0x1	0x0

例：0x1 0x8000 0x35 0x0E 0x1 0x0

0x1 0xF000 0x35 0x0E 0x1 0x0

编号为 0x35 的设备，将会用多条配置参数读取指令的应答格式将该段已分配的配置参数全部返回主机。

3 其它的错误诊断指令

功能码：0x1F

某些设备对某些指令是不支持的，一旦发生这种情况设备会给出 0x5 错误标识（错误标识见附录 1）。

如：我们对配置参数寄存器使用了过程变量的操作指令，

0x1 0x8000 0x35 0x09 0x1 0x04 （0x09 指令是读过程变量寄存器的指令）

设备返回： 0x1 0x8000 0x35 0x1F 0x2 0x09 0x05 （设备不支持该指令）

同样我们对过程变量也不能使用配置参数的指令。

4 对 CAN FD 的支持

当系统和设备需要支持 CAN FD 时，本子协议支持 CAN FD 对传输数据场的扩展。

举例：

例：0x1 0x8000 0x35 0x0D 0x9 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x99 0xAA 0xBB 0xCC

响应：0x1 0x8000 0x35 0x1D 0x9 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x99 0xAA 0xBB 0xCC

例：0x1 0x8000 0x35 0x0E 0x1 0x09

响应：0x1 0x8000 0x35 0x1E 0x9 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x99 0xAA 0xBB 0xCC

1.3.6 TC0030A000 子协议（设备过程变量基本指令）

设备过程变量基本指令

本子协议（通讯子协议）定义了 2 个由“主机”发起的操作“设备过程变量”的基本指令，分别是“变量写指令”、“变量读指令”和每个指令对应的“应答指令”以及错误诊断应答指令，本子协议指令操作仅适用于“设备信息空间”即：第 0x7000 段的过程变量读写。

1 变量写指令

功能码：0x08

“主机”向指定设备的过程变量寄存器基地址写入数据，一次可写入 1～8 个字节的数据，第 2 个字节往后为基地址的顺延。

变量写指令格式：

优先级	寄存器基地址	设备地址	0x08	DLC	1～8 个字节数据

例：0x1 0x7600 0x35 0x08 0x4 0x11 0x22 0x33 0x44

本例表述，主机向地址为 0x35 的设备其基地址为 0x7600 的连续 4 个寄存器写入数据，其值为 0x11 0x22 0x33 0x44。

设备在其原“指令功能码”上加 0x10，并读取刚写好的寄存器内容进行回复。

变量写指令应答格式：

原优先级	原寄存器基地址	原设备地址	0x18	原 DLC	1～8 个字节数据

例：0x1 0x7600 0x35 0x18 0x4 0x11 0x22 0x33 0x44

当变量基地址不存在，或为只读；寄存器范围越界，则返回错误诊断指令如何下，

错误诊断指令格式：

原优先级	原寄存器基地址	原设备地址	0x1A	0x2	0x08	1 个字节错误编号

例：0x1 0x7600 0x35 0x1A 0x2 0x08 0x03 （无此寄存器地址，或为只读）

0x1 0x7600 0x35 0x1A 0x2 0x08 0x04 （寄存器地址越界）* 错误编号见附录 1。

注：当指令被诊断出错误（包括局部错误），整条指令被判无效。

2 变量读指令

功能码：0x09

“主机”向指定设备发出读取其指定过程变量寄存器的内容，指令带一个字节的数据，该数据表述将要连续读取设备过程变量的寄存器个数 n，n＝1～64，当 n 大于 8 时，设备需要使用多条指令响应。

变量读取指令格式：

优先级	寄存器基地址	设备地址	0x09	0x1	请求读取的寄存器个数

例：0x1 0x7100 0x35 0x09 0x1 0x04

本例表述，主机要读取地址为 0x35 的设备其基地址为 0x7100 的连续 4 个字节的过程变量

设备在其原“指令功能码”上加 0x10，并在数据场添加相关数据进行回复。

变量读取指令应答格式：

原优先级	原寄存器基地址	原设备地址	0x19	DLC	1～8 个字节数据

例：0x1 0x7100 0x35 0x19 0x4 0x11 0x22 0x33 0x44

上例表述地址为 0x35 的 设备回复主机的读取指令，将其基地址为 0x7100 的连续 4 个过程变量寄存器的内容 0x11 0x22 0x33 0x44 返回到主机。

又例：0x1 0x7100 0x35 0x09 0x1 0x0E

响应 1：0x1 0x7100 0x35 0x19 0x8 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88

响应 2：0x1 0x7108 0x35 0x19 0x6 0x99 0xAA 0xBB 0xCC 0xDD 0xEE

当读取过程变量寄存器不存在或越界，指令 DLC“请求读取的寄存器个数”出错时，则返回错误诊断指令

错误诊断指令格式：

原优先级	原寄存器基地址	原设备地址	0x1A	0x2	0x09	1 个字节错误编号

例：0x1 0x7100 0x35 0x1A 0x2 0x09 0x03 （无此寄存器地址）

0x1 0x7100 0x35 0x1A 0x2 0x09 0x04 （寄存器地址越界）

0x1 0x7100 0x35 0x1A 0x2 0x09 0x2 （指令参数出错，DLC 错，或 n 不在 1～64 范围内）

过程变量读取指令的特殊用法，为了提高对过程变量的读取效率，我们提供了以下读取整段过程变量寄存器的指令。

整段过程变量读取指令格式：

优先级	0x7000	设备地址	0x09	0x1	0x0

例：0x1 0x7000 0x35 0x09 0x1 0x0

编号为 0x35 的设备，将会用多条过程变量读取指令的应答格式将该段过程变量全部返回主机。

3 其它的错误诊断指令

功能码：0x1A

某些设备对某些指令是不支持的，一旦发生这种情况设备会给出 0x5 错误标识（错误标识见附录 1）。

如：我们对过程变量寄存器使用了配置参数寄存器的操作指令，

0x1 0x7100 0x35 0x0E 0x1 0x04 （0x0E 指令是读配置参数寄存器的指令）

设备返回： 0x1 0x7100 0x35 0x1A 0x2 0x0E 0x05 （设备不支持该指令）

同样我们对配置参数也不能使用过程变量的指令。

4 对 CAN FD 的支持

当系统和设备需要支持 CAN FD 时，本子协议支持 CAN FD 对传输数据场的扩展。

1.3.7 TC0031A000 子协议（位操作指令）

我们最初并没有将“位操作”纳入 TTCANopen 的指令集，但北京中铁航机电总经理王剑宇先生认为“位操作”在“工控”领域非常普遍，而且非常重要，并亲自编写了“位操作指令集”。这很是令我们感动。

位操作指令集

本子协议（通讯子协议）定义了 2 个对“过程变量”进行“位操作”的基本指令，分别是“位与指令”、“位或指令”和每个指令对应的“应答指令”，本子协议指令操作仅适用第 0x7000 段的过程变量，通常只针对数字量。

1 位与指令

功能码：0x01

“主机”用指令自带的数据与指定设备基地址中的数据进行“位与”操作，结果仍放在设备基地址中，一次可进行 1～8 个字节的数据的操作，并将“位与”后的数据结果返回“主机”，第 2 个字节往后为基地址的顺延。

位与指令格式：

优先级	寄存器基地址	设备地址	0x01	DLC	1～8 个字节数据

例：0x1 0x7400 0x35 0x01 0x4 0x11 0x22 0x33 0x44

本例表述，“主机”用 0x11 0x22 0x33 0x44 与地址为 0x35 的设备其基地址为 0x7400 的连续 4 个寄存器中的数据进行“位与”操作，结果仍放在设备原寄存器中。

假设：0x7400 0x7401 0x7402 0x7403 寄存器中分别存储的数据为 0x44 0x55 0x66 0x77，“位与”操作后的结果为 0x00 0x00 0x22 0x44。

位与指令应答格式：

原优先级	原寄存器基地址	原设备地址	0x11	原 DLC	位与操作后结果数据

返回响应为：0x1 0x7400 0x35 0x11 0x4 0x00 0x00 0x22 0x44

2 位或指令

功能码：0x02

“主机”用指令自带的数据与指定设备基地址中的数据进行“位或”操作，结果仍放在设备基地址中，一次可进行 1～8 个字节的数据的操作，并将“位或”后的数据结果返回“主机”，第 2 个字节往后为基地址的顺延。

位或指令格式：

优先级	寄存器基地址	设备地址	0x02	DLC	1～8 个字节数据

例：0x1 0x7400 0x35 0x02 0x4 0x11 0x22 0x33 0x44

本例表述，“主机”用 0x11 0x22 0x33 0x44 与地址为 0x35 的设备其基地址为 0x7400 的连续 4 个寄存器中的数据进行“位或”操作，结果仍放在设备原寄存器中。

假设：0x7400 0x7401 0x7402 0x7403 寄存器中分别存储的数据为 0x44 0x55 0x66 0x77，“位或”操作后的结果为 0x55 0x77 0x77 0x77。

位或指令应答格式：

原优先级	原寄存器基地址	原设备地址	0x12	原 DLC	位或操作后结果数据

返回响应为：0x1 0x7400 0x35 0x12 0x4 0x55 0x77 0x77 0x77

3 错误诊断

功能码：0x1A

当上述位操作基地址不存在、或为只读；寄存器范围越界；不支持该指令，则返回错误诊断指令如何下。

错误诊断指令格式：

原优先级	原寄存器基地址	原设备地址	0x1A	0x2	原功能码	1 个字节错误编号

例：0x1 0x7400 0x35 0x1A 0x2 0x01 0x03 （无此寄存器地址，或为只读）

0x1 0x7400 0x35 0x1A 0x2 0x02 0x04 （寄存器地址越界） * 错误编号见附录 1。

0x1 0x7400 0x35 0x1A 0x2 0x02 0x05 （不支持该指令） * 错误编号见附录 1。

注：当指令被诊断出错误（包括局部错误），整条指令被判无效。

4 对 CAN FD 的支持

当系统和设备需要支持 CAN FD 时，本子协议支持 CAN FD 对传输数据场的扩展。（注：大于 8 个字节的“位”操作是很少出现的。）

1.3.8 TC0032A000 子协议（不推荐的位异或操作指令）

本子协议中的位操作指令可能会给系统带来风险，本协议需同 TC0060 子协议一同使用

以降低风险。

不推荐位异或操作指令集

本子协议（通讯子协议）定义了 1 个对“过程变量”进行“位操作”的基本指令，即“位异或指令”和其“应答指令”，本子协议指令操作仅适用第 0x7000 段的过程变量，通常只适用于数字量。

1 位异或指令

功能码：0x03

“主机”用指令自带的数据与指定设备基地址中的数据进行“位异或”操作，结果仍放在设备基地址中，一次可进行 1～8 个字节的数据的操作，并将“位异或”后的数据结果返回“主机”，第 2 个字节往后为基地址的顺延。

位异或指令格式：

优先级	寄存器基地址	设备地址	0x03	DLC	1～8 个字节数据

例：0x1 0x7400 0x35 0x03 0x4 0x11 0x22 0x33 0x44

本例表述，“主机”用 0x11 0x22 0x33 0x44 与地址为 0x35 的设备其基地址为 0x7400 的连续 4 个寄存器中的数据进行“位异或”操作，结果仍放在设备原寄存器中。

假设：0x7400 0x7401 0x7102 0x7103 寄存器中分别存储的数据为 0x44 0x55 0x66 0x77，“位异或”操作后的结果为 0x55 0x77 0x55 0x33。

位异或指令应答格式：

原优先级	原寄存器基地址	原设备地址	0x13	原 DLC	位异或操作后结果数据

返回响应为：0x1 0x7400 0x35 0x13 0x4 0x55 0x77 0x55 0x33

（注：错误诊断和对 CAN FD 的支持同 TC0031）

1.3.9 TC0033A000 子协议（不推荐的位移操作指令）

本子协议中的位操作指令可能会给系统带来风险，本协议需同 TC0060 子协议一同使用以降低风险。

不推荐位移操作指令集

本子协议（通讯子协议）定义了 1 个对“过程变量”进行“位移”的基本指令，即“位移指令”和其“应答指令”，本子协议指令操作仅适用第 0x7000 段的过程变量，通常只适用于数字量。

1 位移指令

功能码：0x04

“主机”指令携带 3 个字节的数据，第一个数据表述操作目标基地址寄存器字节个数，只能选取 1、2、4、8 中之一；第二个数据表述位移方式，0 为左移，1 为右移，2 为循环左移，3 为循环右移；第三个数据表述位移的位数（0～64）。指令将指定设备基地址寄存器中的变量，按这 3 个字节数据指定的位移要求，进行位移操作，结果仍放在设备基地址中，并将“位移”后的数据结果返回“主机”。

位移指令格式：

优先级	寄存器基地址	设备地址	0x04	0x3	字节数	位移方式	移动位数

例：0x1 0x7400 0x35 0x04 0x3 0x02 0x01 0x01

本例表述，“主机”指示地址为 0x35 的设备其基地址为 0x7400 的连续 2 个寄存器中的变量进行“右移 1 位”操作，结果仍放在设备原寄存器中。

假设：0x7400 0x7401 寄存器中分别存储的数据为 0x12 0x31，“右移 1 位”操作后的结果为 0x89 0x18 。（请注意：寄存器的排列顺序，通常我们会将 2 字节看成 U16，4 字节看称 U32，8 字节看成 U64 处理。本例中，0x12 和 0x31，U16 的表述为 0x3112，右移一位的结果位 0x1889。）

“设备”在原“指令功能码”上加 0x10，用“位移”操作后的结果信息进行回复。

位移指令应答格式：

原优先级	原寄存器基地址	原设备地址	0x14	DLC	位移操作后结果数据

返回响应为：0x1 0x7400 0x35 0x14 0x2 0x89 0x18

（注：错误诊断和对 CAN FD 的支持同 TC0031，当指令 DLC 或携带的 3 个字节“位移参数”出错，则返回 0x02“参数错误”指令。）

1.4 讨论与提示

1.4.1 提高单条指令效率

一条 CAN 指令一次可以携带不多于 8 个字节的数据，TTCANopen 协议中 CAN 指令携带的数据是基于其 ID 字中寄存器基地址指向的连续寄存器字节空间，因此我们应当尽量将设备相关联的配置参数、过程变量放在一起，并将其定义在连续的寄存器空间上，以提高指令携带数据的效率。（注：CAN FD 一次最多可携带 64 个字节数据）

1.4.2 多字节数据的传输

在传输大于 8 个字节数据时，CANopen 和 DeviceNet 两个协议都为此专门定制了新的通讯格式，并将指令内容伸向了 CAN 的数据场中的数据字节，从而压缩了每条指令可有效传输的数据量，协议不但要完成分包和组包，还要在传输过程中处理错误和重传机制，使多字节传输协议解析较为复杂。TTCANopen 协议对多字节数据传输的处理则非常简单清晰自然，甚至不需要定制新的通讯格式，利用寄存器地址的连续性，使用多条“普通指令”就可巧妙的完成多字节数据的传输，并且当发生错误或缺失时，不需立即重传全部数据，只需接收者事后向发送者提交“缺失”部分数据请求指令即可。（注：CAN FD 对多字节数据的传输效率更高，但要注意大于 8 字节后的非线性编码。）

1.4.3 配置参数与过程变量

顾名思义，配置参数是设备正常运行所需的配置信息和标识，主要包括通用配置参数（0xF000 段）和特征配置参数（0x8000 段）。设备配置参数可由主机在设备上电后通过网络进行配置；而有些设备则是在设备上电后，由其初始化程序从非易失存储器中读取信息完成自我配置的，我们称为设备“自举”。一般配置参数属于稳态信息，主机或设备管理器对配置参数的设置和读取往往只是一次性的。（注意：我们将设备部分状态信息也放到了配置参数段中。）

过程变量是设备完成配置后，进入到运行状态，需要通过网络进行实时或准实时交换的变量信息，如：数字的或模拟的输入输出变量等。过程变量属于非稳态信息，CAN 网络系统运行的主要目的就是监视和控制这些变量。

1.4.4 两套指令集

读者可能发现我们在设计指令集时，设计了两套指令集，一套用于“配置参数”；一套用于“过程变量”，这主要是为了便于管理和减少差错而设计的，在本书高级篇中当“设备管理器”独立出来之后，分别设计指令集就显得更加必要了。

1.4.5 寄存器地址映射

TTCANopen 协议定义的 16 位寄存器地址寻找空间是一个虚拟空间，真实的设备一般不会也没必要按照此空间分配设备实际的物理寄存器地址空间或应用程序“变量”存储地址空间，这样设备应用程序就需要完成虚实两个地址空间的映射，这对于一个程序员来说并不困难，这里需要指出的是 TTCANopen 协议的 16 位的寄存器地址寻找空间是以“字节”为单位的虚拟空间，从低到高有序排列的，各通讯子协议将这些字节寄存器当做一个个虚拟容器看待，并不关心里面放了什么，容器中的内容主要是由应用程序来解析的，程序员一定要注意应用程序中“多字节变量”的字节排序与 TTCANopen 虚拟空间字节排布的顺序差异。

思考：一个 4 字节的长整形变量数组 long［4］，一共 16 个字节，如何完成虚实映射?

至此，完成了“实空间”与“虚空间”的转换问题，本书后续内容都会以“虚空间”为基点进行描述，图 1－3 为设备寄存器转换关系示意图。

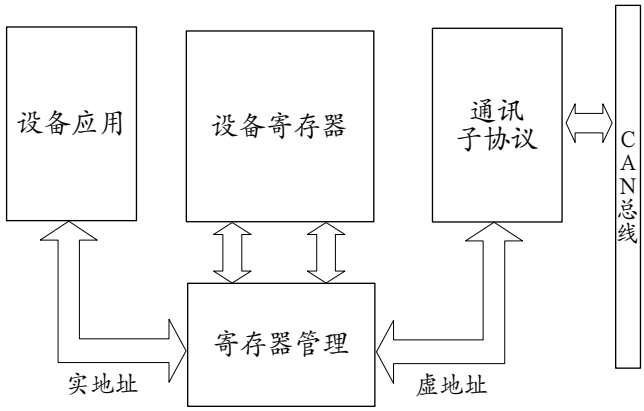

图 1－3 为设备寄存器转换关系示意图。

1.4.6 为 CAN 底层协议无法检测到的重复发送小概率错误提供应用层解决方案

虽然 CAN 网络可以检测到几乎所有全局和局部错误，但依旧存有漏洞，如：当发送端发

送到最后一位时，出现局部错误，从而启动了重发机制，而接收端在不知晓的情况下接收到了两个完全相同正确的完整帧。因此，一般 CAN 网络建议用户禁止传输增量值和切换消息，这给应用带来了诸多不便。

为了应对上述小概率事件，TTCANopen 在应用层提供了相应的检测手段和处理机制（TC0060），即：在发送具有重发风险的指令帧（如：TC0032）时，在其前面插入一个“特征前导帧”作为标识。“特征前导帧”是数据长度为零（DLC＝0）的发送帧。

这样在接收端就可据此甄别出重复发送帧，并将其剔除，接收端不对“特征前导帧”做出应答响应。

（注：在 TTCANopen 中传输增量值和切换消息只能是“过程变量”，不能在“配置参数”中出现。）

1.4.7 通讯模型

从指令格式上看，TTCANopen 指令中没有能够将“信源”和“信宿”地址全都包容在内，究其原因是为了保证协议的“直读性”和“易用性”，29 位 ID 资源仍然“不够用”，只能容纳“信源”或“信宿”一个地址，我们现在学习到的通讯子协议中，采用了“主从模型”指令规范，指令中仅仅指明了设备（从机）地址，而通讯的另一方则是“默认主机”，这对于“中心监控”系统是非常适用的。

由此可以看出 TTCANopen 协议 ID 划分方式，比较容易的完成了主机与从机的通讯，而要完成设备间的直接通讯则有一定的困难。

现场总线常用的三种通讯模型中，除了“主从模型”还有“客户－服务模型”和“生产者－消费者模型”。

其中，支持“生产者－消费者模型”则是 CAN 现场总线的一大特点，也是对应用层协议制定者的考验。

另外，需要指出的是系统的最终通讯模型是在“应用子协议”中制定的，可以是单一通讯模式，更多的是复合通讯模式。

1.4.8 协议的直读性

TTCANopen 使用了具有 29 位 ID 资源的 CAN2.0b 协议扩展帧格式规范，并将其 29 位 ID 资源固定分割为 4 个有实际含义的子段，即：“优先级段”、“寄存器基地址段”、“ 设备

编号段”和“功能码段”，使得 TTCANopen 应用层协议具有很强的直读性，大大降低了协议的解析难度，有效的提高了设计人员的研发效率。

TTCANopen 协议的直读性同样给系统调试和维护带来了莫大的方便，调试人员可以使用最简单的总线指令监视设备就可了解掌握系统运行情况，而使用其他协议则需要使用复杂昂贵的总线分析设备或仪器。

1.4.9 组态软件

支持标准串口和 Modbus [2] 协议是各种组态软件与生俱来的特性，而对多数 CAN 协议的支持则需要编制复杂的 OPC [3] 程序来完成，细心的读者可能已经发现 TTCANopen 协议的直读性与 Modbus 协议有着很大的相似性，我们只要将 Modbus 协议驱动程序稍作改动就能够支持 TTCANopen 协议了。同时利用 CAN/USB 转换模块将 CAN 总线通过 USB 接口虚拟仿真成标准串口连接入上位机（PC 机），即可完成与组态软件的完美对接，TTCANopen 协议可以方便的利用现有组态软件资源，为系统的设计者提供了极大的便利。

另外，电子工程师常用的虚拟仪器 LabVIEW [4] 软件对串口的支持也是非常完备的，这使得用户非常容易的将 TTCANopen 设备融入到其虚拟仪器设计中。

1.4.10 与生俱来，对 CAN FD 的支持

TTCANopen 协议是以连续寄存器空间为信息载体，应用层协议指令是对该连续寄存器空间的读写访问，由此使其能够顺延的兼容 CAN FD 协议对数据场长度的扩充。

注：［1］USB 是在 1994 年底由英特尔、康柏、IBM、Microsoft 等多家公司联合提出的。
［2］Modbus 由 Modicon 开发。
［3］OLE for Process Control 由 OPC 基金会管理。
［4］LabVIEW 由美国国家仪器公司开发。