第 12 章　“雏鹰，展翅”
可编程设备与“类”

12.1 可编程设备

与可编程设备相对应的设备我们称之为“固化设备”，所谓固化设备就是我们上一章结束时提到的定制系统的定制设备；而可编程设备就是那些通用设备，其在系统中的用途，只有在其接入系统后，才被确立。

我们先举一个应用实例进行说明。在 SUV 汽车中有一个温度传感器，其作用是向网络周期播发车厢内的环境温度值，供其他设备使用假定 SUV 汽车标准化组织将该过程变量定义在系统信息空间的 0x4590 地址上，并规定每秒钟播发一次，播发时刻为每整秒后的第 500

ms 处。

同样，在大客车上也有类似的温度传感器，为了更加精准的获取车厢内的环境温度，车厢内布置了 3 个温度传感器，一个在车尾段，一个在车中段，一个在车首段。大客车标准化组织将其定义在 0x4690、0x4694 和 0x4698 系统寄存器上。车尾的温度传感器每秒钟播发两次分别为整秒后的 100ms 处和 600ms 处；中段和首段的温度传感器则每秒播发一次，为整秒后的 610ms 处和 620ms 处。

作为“固化设备”，生产商就需要生产 4 种不同型号的温度传感器，每种传感器用在不同的位置上。

“固化设备”的好处是，用途明确，配置简单，易于设计、生产，调试、更换、维护方便；坏处是，产品种类型号繁多，通用性不强

有部分设备生产商一定会想，只生产一种“通用”的温度传感器，取代上述 4 种型号的传感器，在传感器接入系统时，再根据其接入

位置和用途，按照相关标准化要求配置其过程变量映射地址、播发频率和时刻。

“通用设备”可以减少产品的种类和型号，其用途不明确，设计、生产、调试相对复杂，施工、更换、维护、现场配置繁琐，容易出错对工作人员业务素质要求高。

设计和生产“固化设备”或“通用设备”，设备生产商会根据产品批量进行综合评估，进行选择。

另外，还有一种设备，如：通用 I/O 类产品，从其诞生，就没有指定其工作位置场景和工作运行细节，直到系统设计师将其接入系统方能确定其现场用途，这类产品天生就是要现场配置和编程的。

现场配置和编程，如果不进行规范，任由各设备厂商尽情发挥，生产出来的设备配置方法五花八门，会给系统集成和现场施工调试带来非常大的困难。

本章的宗旨就是要对 TTCANopen 通用类

产品的现场配置和功能编程制定一个相对可接受的规范。

12.2 可编程过程变量寄存器映射配置方案

“固化设备”中，过程变量所对应的寄存器地址映射是固化在应用程序中的，不可随意更改，这使设备的研发相对容易，不需要考虑多种应用场景和配置的可能性，同时设备的应用也被固化和指定。

“通用设备”中，其过程变量所对应的寄存器地址映射是可配置的，因此，它可以适用不同的应用场景。每个设备有不同的种类和不同数量的过程变量，如何配置这些过程变量的映射地址需要进行规范，为此我们启用了 0xE000 段设备参数空间，完成相关的配置规范。

12.2.1 资源映射空间的分配

我们首先涉及的是将设备本地资源映射到过程变量寄存器空间，如：各种数字量、模拟量的输入输出；另外，设备可能还会对某些网络资源敏感，甚至可能会根据已有资源衍生新的资源，这些都需要我们为其分配空间建立映射关系，见表12－1。

表12－1 资源映射空间分配

序号	寄存器空间	用途
1	0xE000～0xE0FF	本地I特性资源
2	0xE100～0xE1FF	衍生I特性资源
3	0xE200～0xE3FF	本地O特性资源
4	0xE400～0xE5FF	衍生O特性资源
5	0xE600～0xE7FF	本地I/O特性资源
6	0xE800～	衍生I/O特性资源

	0xE9FF	
7	0xEA00～0xEAFF	网络敏感资源
8	0xEB00～0xEBFF	保留
9	0xEC00～0xECFF	单字节常数
10	0xED00～0xEDFF	双字节常数
11	0xEE00～0xEEFF	四字节常数
12	0xEF00～0xEFFF	八字节常数

在表中我们还分配了设备所需“常数”的使用空间。

12.2.2 本地 I 特性资源的映射

我们将 0xE000～0xE0FF 空间平均分为 64 组，每组 4 个字节，见表 12－2。

表 12－2 本地 I 特性资源映射寄存器组

组序号	寄存器空间	用途
1	0xE000～0xE003	完成本地 I 特性资源的映射
2	0xE004～0xE007	
……	……	
63	0xE0F8～0xE0FB	
64	0xE0FC～0xE0FF	

使用映射寄存器组必须顺序使用，不可间隔，设备可以根据自身 I 特性资源的多少，确定使用组的个数，但最多为 64 组。

我们用一组四个字节描述一个本地 I 特性资源变量及其到过程变量空间的映射关系，见表 12－3。

表 12－3 本地 I 特性资源变量及映射

寄存器字节数	用途描述
1	本地 I 特性资源变量映射到过程变量空间地址的低 8 位
2	本地 I 特性资源变量映射到过程变量空间地址的高 8 位
3	本地 I 特性资源变量的字节长度（1～4）
4	本地 I 特性资源变量的特性描述 0：为不定特性变量 1：数字量 2：为无符号模拟量 3：为有符号模拟量 4：为浮点数

下面我们使用第一组描述一个 A/D 采集变量的映射关系，见表 12－4。

表 12－4 本地 A/D 采集变量的映射

序号	寄存器	寄存器内容	本例含义
1	0xE000	0x90	映射地址低 8 位
2	0xE001	0x14	映射地址高 8 位
3	0xE002	2	变量字节长度
4	0xE003	2	无符号模拟量

从表中我们可以看到，该定义将这个本地一个 A/D 采集变量映射到了系统寄存器空间的 0x1490 地址上，变量长度为 2 字节，变量为一个无符号模拟量。当然我们也可以将它映射到设备寄存器空间，如：0x7290 上。

12.2.3 本地 O 特性资源的映射

O 特性资源变量的映射要比 I 特性资源变量复杂些，这是因为这些变量可能存在初始值（安全值），以及值域范围，需要我们去设定我们将 0xE200～0xE3FF 空间平均分为 32 组，

每组16个字节，见表12－5。

表12－5 本地O特性资源映射寄存器组

组序号	寄存器空间	用途
1	0xE200～0xE20F	完成本地O特性资源的映射
2	0xE210～0xE21F	
……	……	
31	0xE3E0～0xE3EF	
32	0xE3F0～0xE3FF	

使用映射寄存器组必须顺序使用，不可间隔，设备可以根据自身O特性资源的多少，确定使用组的个数，但最多为32组。

我们用一组十六个字节描述一个本地O特性资源变量及其到过程变量空间的映射关系，

见表 12－6。

表 12－6 本地 O 特性资源变量及映射

寄存器字节数	用途描述
1	本地 O 特性资源变量映射到过程变量空间地址的低 8 位
2	本地 O 特性资源变量映射到过程变量空间地址的高 8 位
3	本地 O 特性资源变量的字节长度（1～4）
4	本地 O 特性资源变量的特性描述 0：为不定特性变量 1：数字量 2：为无符号模拟量 3：为有符号模拟量 4：为浮点数
5	本地 O 特性资源变量
6	
7	

8	初始值或安全值（可选）
9	本地 O 特性资源变量 上限值（可选）
10	
11	
12	
13	本地 O 特性资源变量 下限值（可选）
14	
15	
16	

下面我们使用第一组描述一个 D/A 输出变量的映射关系，见表 12－7。

表 12－7 本地 D/A 输出变量的映射

序号	寄存器	寄存器内容	本例含义
1	0xE200	0x90	映射地址低 8 位
2	0xE201	0x24	映射地址高 8 位
3	0xE202	2	变量字节长度
4	0xE203	2	无符号模拟量
5	0xE204	0x00	D/A 初值
6	0xE205	0x50	
7	0xE206	0x00	

8	0xE207	0x00	（或安全值）为0x5000
9	0xE208	0x00	D/A 上限值为0xA000
10	0xE209	0xA0	
11	0xE20A	0x00	
12	0xE20B	0x00	
13	0xE20C	0x00	D/A 下限值为0x1000
14	0xE20D	0x10	
15	0xE20E	0x00	
16	0xE20F	0x00	

在表中，我们看到 D/A 初值（或安全值）为 0x5000，上限值为 0xA000，下限值为 0x1000，请读者注意，我们不支持离散的值域范围因为这需要更多的资源去描述，通常遇到这种情况需要设备生产商在使用说明书中加以注明当值域越界时，对模拟量则输出其上下限的极限值，如：本例中的 0xA000 或 0x1000，然后再返回应答指令或发送错误提示信息，对于数字量则不存在上下限的问题。

12.2.4 衍生I特性资源的映射

生产者－消费者模型中，主机功能被淡化，许多过去由主机完成的对过程变量的处理过程被移植到设备端来，必然会衍生出新的过程变量，这是系统从集中处理到分布处理演绎过程的必然结果。我们将0xE100～0xE1FF空间平均分为64组，每组4个字节，见表12－8。

表12－8 衍生I特性资源映射寄存器组

组序号	寄存器空间	用途
1	0xE100～0xE103	完成衍生I特性资源的映射
2	0xE104～0xE107	
……	……	
63	0xE1F8～0xE1FB	

64	0xE1FC～ 0xE1FF	

使用映射寄存器组必须顺序使用，不可间隔，设备可以根据自身可能衍生的I特性资源的多少，确定使用组的个数，但最多为64组。

我们用一组四个字节描述一个衍生I特性资源变量及其到过程变量空间的映射关系，见表12－9。

表12－9 衍生I特性资源变量及映射

寄存器字节数	用途描述
1	衍生I特性资源变量映射到过程变量空间地址的低8位
2	衍生I特性资源变量映射到过程变量空间地址的高8位
3	衍生I特性资源变量的字节长度（1～4）
4	衍生I特性资源变量的特性描述

	0：为不定特性变量 1：数字量 2：为无符号模拟量 3：为有符号模拟量 4：为浮点数

从表中我们可以看到其和本地I特性资源变量及映射极为相似，只不过其信息资源来源于对本地资源以及网络资源的整合再生处理，或根据系统需求拟生出的变量。

下面我们使用第一组描述一个衍生I特性资源的映射关系，见表12－10。

表12－10 衍生I特性资源的映射

序号	寄存器	寄存器内容	本例含义
1	0xE100	0x90	映射地址低8位
2	0xE101	0x18	映射地址高8位
3	0xE102	4	变量字节长度
4	0xE103	4	浮点数

从表中我们可以看到，设备将一个 I 特性衍生资源变量映射到 0x1890 系统信息寄存器空间。

12.2.5 衍生 O 特性资源的映射

我们将 0xE400～0xE5FF 空间平均分为 32 组，每组 16 个字节，见表 12－11。

表 12－11 衍生 O 特性资源映射寄存器组

组序号	寄存器空间	用途
1	0xE400～0xE40F	完成衍生 O 特性资源的映射
2	0xE410～0xE41F	
……	……	
31	0xE5E0～0xE5EF	
32	0xE5F0～	

	0xE5FF	

使用映射寄存器组必须顺序使用，不可间隔，设备可以根据其衍生 O 特性资源的多少，确定使用组的个数，但最多为 32 组。

我们用一组十六个字节描述一个衍生 O 特性资源变量及其到过程变量空间的映射关系，见表 12－12。

表 12－12 衍生 O 特性资源变量及映射

寄存器字节数	用途描述
1	衍生 O 特性资源变量映射到过程变量空间地址的低 8 位
2	衍生 O 特性资源变量映射到过程变量空间地址的高 8 位
3	衍生 O 特性资源变量的字节长度（1～4）
4	衍生 O 特性资源变量的特性描述 0：为不定特性变量

	1：数字量 2：为无符号模拟量 3：为有符号模拟量 4：为浮点数
5	衍生 O 特性资源变量 初始值或安全值（可选）
6	
7	
8	
9	衍生 O 特性资源变量 上限值（可选）
10	
11	
12	
13	衍生 O 特性资源变量 下限值（可选）
14	
15	
16	

同本地 O 特性资源变量一样，衍生 O 特性资源变量同样需要设置初始值（安全值）和上下限值。

12.2.6 本地和衍生 I/O 特性资源的映射

同本地和衍生 O 特性资源的映射类同，这里不再累述。

12.2.7 设备对网络资源敏感的配置

除了本地资源和衍生资源，设备可能会对某些网络中播发的资源敏感。我们将 0xEA00～0xEAFF 空间平均分为 64 组，每组 4 个字节，见表 12－13。

表 12－13　网络资源敏感寄存器组

组序号	寄存器空间	用途
1	0xEA00～0xEA03	完成

2	0xEA04～0xEA07	设备对网络资源敏感的配置
……	……	
……	……	
63	0xEAF8～0xEAFB	
64	0xEAFC～0xEAFF	

使用网络资源敏感寄存器组必须顺序使用，不可间隔，设备可以根据自身可能对网络资源的敏感数量，确定使用组的个数，但最多为64组。

我们用一组四个字节描述一个设备敏感的网络过程变量，见表12－14。

表12－14 设备敏感的网络过程变量

寄存器字节数	用途描述
1	设备敏感的网络过程变量地址的低8位

2	设备敏感的网络过程变量地址的高 8 位
3	设备敏感的网络过程变量的长度（1～4）
4	设备敏感的网络过程变量的特性描述 0：为不定特性变量 1：数字量 2：为无符号模拟量 3：为有符号模拟量 4：为浮点数

从表中我们可以看到其和本地 I 特性资源变量及映射极为相似，只不过其信息资源来源于网络。

下面我们使用第一组描述设备对网络上通过系统寄存器空间 0x1890 地址传递的 A/D 过程变量敏感的配置关系，见表 12－15。

表 12－15 对网络信息敏感的配置

序号	寄存器	寄存器内容	用途
1	0xEA00	0x90	敏感过程变量地址低 8 位
2	0xEA01	0x18	敏感过程变量地址高 8 位
3	0xEA02	4	敏感过程变量字节长度
4	0xEA03	4	浮点数

从表中我们可以看到，设备对系统寄存器空间 0x1890 的信息敏感，并将该信息解释为一个 4 字节浮点变量。

12.2.8 TC0015 子协议（0xE000 段的分配使用）

除了资源映射，设备在使用本地资源以及网络敏感资源生成衍生资源的时候，可能会用到一些常数，包括我们下面将要讲到的“类”

也会需要常数的支持，因此在 0xE000 段我们划分出部分空间供设备涉及的“常数”使用，见表 12－1。

关于 0xE000 段的分配使用，详见 TC0015 子协议。

12.3 可编程设备功能方案初探

在上一小节，我们完成了本地资源和衍生资源到过程变量寄存器空间的映射，以及设备对网络资源敏感的配置。至此，我们只是初步完成了通用设备过程变量的可配置工作。而通用设备另外一个特征，是功能可配置，也就是说一个通用设备内设了一些设备可能用到的功能，只有在设备接入系统后，确定了角色，我们才能够启用其中的某些功能，并为这些功能配置必要的参数。

我们曾经引用过类似 C＋＋中运算符重载的方法，针对不同操作空间（设备信息和系统

信息空间）重载了功能码的含义。在这里我们很自然的想到了C＋＋中，另一个重要元素——“类”。类将执行某种功能的代码和其相关的参数、变量集成到一起，类可以有多个对象每个对象执行的类代码是相同的，而其操作的参数和变量集则属于各自的对象。在设备应用程序中，我们如法炮制，将执行各种设备功能的代码看做是“类”代码，集成到设备应用程序中，而将其相关对象的参数和变量集中到一起，指定到某寄存器空间按规定去排列。这个类对象参数和变量的空间我们称之为类对象空间。

我们启用D000段作为类对象空间，并规定一个类对象的最大使用空间为64个字节，这样整个段可以容纳64个类对象，见表12－16。类对象要依序分配，不能间隔。

表12－16 类对象分配空间

序号	寄存器	用途

1	0xD000～0xD03F	第 1 个类对象空间
2	0xD040～0xD07F	第 2 个类对象空间
……	……	……
63	0xDF80～0xDFBF	第 63 个类对象空间
64	0xDFC0～0xDFFF	第 64 个类对象空间

每个类对象空间有一通用的框架，便于进行配置和使用，见表 12－17。

表 12－17 类对象统一框架

寄存器字节编号	内容用途
1	本对象的总长度
2	对象的状态，0x22 为配置态，

	0x33 为在用态， 0x44 为停用态。
3	类的应用行业或分类
4	同行业应用的类编号
5～n	类对象参数和变量的配置空间
n＋1	类对象结束符　0x5A

其中，类对象第一个字节是该类数据总长度（按字节计算，包括结束符 0x5A）。第二个字节表征该类对象的存在状态，0x22 为配置态，设备管理器正在配置该类对象，此时该类对象是不可用的；0x33 为在用态，应用程序中该类对象的功能代码可以使用该类对象中的参数和变量，完成特定任务和功能；0x44 为停用态，应用程序不能使用该类对象中的参数和变量。第三个字节，为类的应用行业标识和分类，详见表 12－18，这个表是在不断的扩展中的，目前我们只标识了少数分类。第四个字节，是在行业中（或分类中）的类编号。

最后一个字节为类对象的结束标识 0x5A。在第四个字节和结束符之间是类对象的参数和变量体。

表 12－18 类的应用行业标识

类行业编号	类行业（或分类）
1	标准通用周期触发类
2	标准通用条件触发类
3	常用 I/O 类
4	滤波器类
5	常用 IF_THEN 类
6	运算类

注：对于差异大的应用行业，“类的应用行业标识”可以被重复表述使用。

12.4 类功能代码

至此，我们规范了过程变量到信息空间的映射方法和类对象参数和变量的使用空间。设备开发和生产厂商则要使用这些规范，定义过程变量和类对象，并在应用程序中生成对应的

功能代码。这个过程可能是一个比较复杂的过程，设备完成的类功能越多，对设备空间和CPU 性能的需求就越高，类程序也就越复杂。

12.5 TC1000 子协议（周期触发类）

TC1000 子协议是由 TTCANopen 组织定义的常用周期触发类，目前在 TC1000A001 中定义了三个基本类，供用户参考。

第一个是周期触发本地闭环模拟接收指令类，该类可在设备本地时间运行到规定的副帧相位和子帧相位时刻，触发 1～3 条模拟接收指令，将其传送到设备接收指令循环缓冲区中然后由设备指令处理单元进行指令处理。最常用的模拟接收指令为针对本机的 0x0E 或 0x09 指令。

第二个是周期触发的发送指令类，该类可在设备本地时间运行到规定的副帧相位和子帧

相位时刻，触发 1～3 条本地发送指令，并将其传送到设备发送指令循环缓冲区中等待发送最常用的发送指令为针对本机的 0x07 指令。

第三个是周期触发的参数、变量上传类，该类可在设备本地时间运行到规定的副帧相位和子帧相位时刻，触发 1～3 条针对配置参数、本地资源、衍生资源的参数或变量上传指令，并将其传送到设备发送指令循环缓冲区中等待发送。最常用的发送指令为针对本机的 0x1E 和 0x19 指令和针对系统信息空间寄存器的 0x17 指令。

下面我们分别对这三个类进行定义。

12.5.1 周期触发本地闭环模拟接收指令类

在第 5 章结尾我们引用了一个“画蛇添足”的故事，我们将周期性的 0x09 指令本地化，使它不通过总线传输，而是在设备本地臆

造一个 0x09 指令，放到接收指令缓冲区，传递给“指令处理单元”。

现在我们就在类对象空间设计和分配相关参数和变量，见表 12－19。

表 12－19 周期触发本地闭环模拟接收指令类对象空间分配

寄存器字节编号	内容用途	数据类型	举例
1	本对象的总长度	U8	21（37，53）
2	对象的状态，0x22 为配置态，0x33 为在用态，0x44 为停用态。	U8	0x22
3	类的应用行业	U8	1（标准通用周期触发类）
4	类种类编号	U8	1（第 1 个

			类定义)
5～8	子帧相位	U32	0x00000080
9～10	副帧相位	U16	0x0012
11	副帧周期选择	U8	0、1、2、3
12	发送方式： 0 为不发送 1 发送	U8	1
13～19	本地闭环模拟周期接收的指令（0x0E、0x09）	模拟接收指令字节串	0x1 0x72 0x00 0x32 0x09 0x01 0x08
20	保留	U8	
（21～36）	第二组	子帧相位～保留	
（37～52）	第三组	子帧相位～保留	
21（37，53）	类对象结束符	U8	0x5A

第 1 个字节为类对象长度 21；第 2 个字节为该对象当前的状态，配置态 0x22；第 3 个字节为类的应用行业 1，是 TTCANopen 定制的标准通用周期触发类；第四个字节为该类中的第 1 个类定义；第 5～8 个字节为 32 位无符号整数，代表子帧相位值；第 9～10 个字节为 16 位无符号整数，代表副帧相位值；第 11 位为选择副帧周期（P0、P1、P2、P3）；第 12 位为触发方式，0 为不发送，1 为发送；当设备本地时间运行到与本类对象的副帧相位值和子帧相位值相同（注：当副帧相位为 0 时，则只比对子帧相位值。设备的子帧周期和副帧周期在 0xF000 段中定义。）时，将第 13～19 字节中所包含的指令，放入设备接收指令缓冲区中等待设备指令处理单元的处理。

（注意：对设备寄存器空间，第 16 个字节必须是本机地址，在变更设备地址时，需特别留意。为了避免类似错误，可将其设置为广播地址 0x00。）

该类是由 TTCANopen 组织定义的标准通用类中的第一个类。它可模拟设备周期接收的指令，提供给设备指令处理单元处理，通常应用最多的是模拟 0x0E 和 0x09 指令，使设备周期上报相关寄存器中的内容。TTCANopen 组织只定义了该类对象空间的参数分布，实现类功能则需由设备开发者或设备生产厂商去完成

在该类对象中最多可以提供 3 个模拟周期接收指令，用户可以根据需求填充第二组和第三组相关参数，并可以使用“发送方式”确定指令是否需要发送。当应用中需要多于 3 个模拟周期接收指令时，可以定义多个类对象。

12.5.2 周期触发发送指令类

与上一个类相似，只不过本类触发的指令放到了发送指令缓冲区中。

相关类对象空间设计和分配，见表 12－20。

表 12－20 周期触发发送指令类对象空间分配

寄存器字节编号	内容用途	数据类型	举例
1	本对象的总长度	U8	21（37，53）
2	对象的状态	U8	0x22
3	类的应用行业	U8	1
4	类种类编号	U8	2
5～8	子帧相位	U32	80
9～10	副帧相位	U16	0
11	副帧周期选择	8	0、1、2、3
12	发送方式： 0 为不发送 1 发送	U8	1
13～19	周期触发的发送指令（0x07、0x09）	发送指令字节串	0x01 0x76 0x00 0x32 0x07 0x01 0x08

20	保留	U8	
（21～36）	第二组	子帧相位～保留	
（37～52）	第三组	子帧相位～保留	
21（37，53）	类对象结束符	U8	0x5A

该类是由 TTCANopen 组织定义的标准通用类中的第二个类。通常应用最多的是发送 0x07 数据索取指令或针对系统信息寄存器空间的 0x09 指令。（注意：第 16 个字节必须是本机地址，在变更设备地址时，需特别留意。通常可以使用设备发送指令“滤波”技术避免此类错误的发生。）

12.5.3 周期触发参数变量上传类

本类触发的指令携数据场数据放入发送指令缓冲区中，该数据场数据来自本地资源，或

衍生资源，也可以是配置参数中的状态信息。

相关类对象空间设计和分配，见表 12－21。

表 12－21 周期触发参数变量上传类对象空间分配

寄存器字节编号	内容用途	数据类型	举例
1	本对象的总长度	U8	21（37，53）
2	对象的状态	U8	0x22
3	类的应用行业	U8	1
4	类种类编号	U8	3
5～8	子帧相位	U32	80
9～10	副帧相位	U16	0
11	副帧周期选择	8	0、1、2、3
12	发送方式： 0 为不发送 1 为简单周期发送	U8	1

	2 为智能周期发送		
13～18	周期触发参数变量上传指令（0x1E、0x19、0x17）	指令字节串	0x01 0x72 0x00 0x32 0x19 0x08
19～20	保留		
（21～36）	第二组	子帧相位～保留	
（37～52）	第三组	子帧相位～保留	
21（37，53）	类对象结束符	U8	0x5A

当“发送方式”为 0 时，取消该周期发送指令；为 1 时，该类在指令送入发送缓冲区时，应用程序会将其基地址对应的参数或变量信息添加到指令发送序列的数据场中，并更新其历史数据；为 2 时，应用程序会比对该指令其基地址对应的参数或变量信息是否与其历史数据

相同，相同则取消本次指令发送，不同则发送并更新其历史数据。

“智能周期发送”可以有效的避免周期发送中的相同数据重复发送的情况，节省有限的总线带宽。

需要提醒的是，在设备现场改变本机地址后，上述类对象中的本机地址也应当进行相应更改。

12.6 TC1001 子协议（条件触发类）

TC1001 子协议是由 TTCAopen 组织定义的常用条件触发类，目前在 TC1001A001 中定义了三个基本类，供用户参考。

12.6.1 简单变化触发类

该类关注的数据发生变化，便将数据发送到网络上。

相关类对象空间设计和分配，见表 12－22。

表 12－22 简单变化触发类对象空间分配

寄存器字节编号	内容用途	数据类型	举例
1	本对象的总长度	U8	17（29，41，53）
2	对象的状态	U8	0x22
3	类的应用行业	U8	2
4	类种类编号	U8	1
5～8	禁发间隔	U32	20
9	执行标识： 0 为不执行 1 为执行	U8	1
10～15	简单变化触发指令（0x1E、0x19、0x17）	指令字节串	0x1 0x70 0x00 0x32 0x19 0x04
16	保留	U8	

（17～28）	第二组	禁发间隔～保留	
（29～40）	第三组	禁发间隔～保留	
（41～52）	第四组	禁发间隔～保留	
17（29，41，53）	类对象结束符	U8	0x5A

“禁发间隔”是指该类对象触发的同一指令两个相邻发送帧的最小时间间隔（单位为TTCANopen 时间的 0.1ms 计数），是为了防止某些变量过于频繁的变化，连续触发指令将通讯堵塞而设定的，该值可以设置为 0。

该类应用代码获取当前该类对象指令基地址所指的当前数据，并与其历史数据（上次发送时）进行比较，如果不同，且超过“禁发间隔”，则将当前数据拷贝到历史数据区，触发

指令发送，并更新该指令的“上次发送时刻”

“上次发送时刻”是本地时钟记录的该指令上次在该类中触发指令的时刻（TTCANopen 时间），它和“禁发间隔”一起，确定该指令下次发送的最近时刻。（注：该信息在该类的隐性结构数据中）

由于在一个类对象空间中我们可能定义了多个触发指令，而应用中不一定全部都用到，这样我们需要用“执行标识”来标识其有效性

12.6.2 增量触发类

该类关注的数据发生变化且增量达到预设的增量值时，便将数据发送到网络上。

相关类对象空间设计和分配，见表 12－23。

表 12－23 增量触发类对象空间分配

寄存器字节编号	内容用途	数据类型	举例

1	本对象的总长度	U8	17（29，41，53）
2	对象的状态	U8	0x22
3	类的应用行业	U8	2
4	类种类编号	U8	2
5～8	预设增量	U8、U16、U32 或 f32	500（数据类型同指令所指，只取正值）
9	执行标识： 0 为不执行 1 为执行	U8	1
10～15	增量变化触发指令（0x19、0x17）	指令字节串	0x1 0x72 0x00 0x32 0x19 0x04
16	保留	U8	
（17～28）	第二组	预设增量～保留	

（29～40）	第三组	预设增量～保留	
（41～52）	第四组	预设增量～保留	
17（29，41，53）	类对象结束符	U8	0x5A

该类应用代码获取当前该类对象指令基地址所指的当前数据，并与其历史数据（上次发送时）进行比较，如果其“变化值”大于“预设增量”，则将当前数据拷贝到历史数据区，触发指令发送。“预设增量”要根据变量变化特性去设置，过小则会频繁触发指令，占用过多的总线带宽，过大则对变量值的观测过于“粗糙”。

12.6.3 越界触发类

该类关注的数据发生越界，便将数据发送

到网络上。

相关类对象空间设计和分配，见表 12－24。

表 12－24 越界触发类对象空间分配

寄存器字节编号	内容用途	数据类型	举例
1	本对象的总长度	U8	21（37，53）
2	对象的状态，	U8	0x22
3	类的应用行业	U8	2
4	类种类编号	U8	3
5～8	上界值	U8、U16、U32、I8、I16、I32 或 f32	9000（类型同指令所指）
9～12	下界值	U8、U16、	100（类型同

		U32、I8、I16、I32 或 f32	指令所指）
13	执行标识： 0 为不执行 1 为上越限 2 为下越限 3 为上下越限	U8	1
14～19	越界触发指令（0x19、0x06、0x17）	指令字节串	0x0 0x72 0x00 0x32 0x19 0x04
20	保留	U8	0
（21～36）	第二组	上界值～保留	
（37～52）	第三组	上界值～保留	
21（37，53）	类对象结束符	U8	0x5A

该类应用代码获取当前该类对象指令基地址所指的当前数据，并与其历史数据进行比较如果发生“正越界”，则使用“0 优先级”发送越界指令；如果发生“反越界”，则使用“1 优先级”发送越界指令，同时更新其历史数据。

系统的设计者，要根据变量的特性和系统具体需求选择不同的触发方式，同一变量一般不要同时选择多种触发方式，因为各触发方式在各自的类对象中实现，它们之间没有建立协调机制，可能会产生相互矛盾的地方。如果有需求，则可以新定义一种类，在类对象中描述其多样的触发方式。

12.7 TC1003 子协议（常用 I/O 类）

在生产者－消费者模型出现后，工业现场

将更多的处理过程放到了设备端，我们常称之为预处理。所谓预处理就是通过对设备已有的参数和变量进行相关处理生成新变量的过程（即衍生变量）。

根据行业和设备用途的不同，预处理涉及的内容非常广泛，它可以是设备数字量的逻辑运算和滤波，也可以是多个模拟量的加权求均甚至可以是某些非线性查表操作等。

TC1003 子协议是由 TTCAopen 组织定义的常用 I/O 类，目前在 TC1003A001 中定义了四个基本的带预处理功能的 I/O 类，供用户参考

12.7.1 数字量输入预处理触发类

使用本地数字量输入与预设数字常量进行位逻辑运算，经滤波后生成新的过程变量（即衍生资源），该衍生资源与其自身历史值进行比较，如满足触发条件，则触发指令发送，并更新历史值。触发条件分为三种，分别是“变

化触发”、“上跳触发”和“下跳触发”，三种触发条件“或”的结果为触发标识。

相关类对象空间设计和分配，见表12－25。

表12－25 数字量输入预处理触发类对象空间分配

寄存器字节编号	内容用途	数据类型	举例
1	本对象的总长度	U8	33（61）
2	对象的状态	U8	0x22
3	类的应用行业	U8	3
4	类种类编号	U8	1
5～6	衍生数字量寄存器基地址	U16	0x1262
7～8	本地数字量输入寄存器基地址	U16	0x70D0
9～12	变化触发淹码	U32	0xFFFFFFFF
13～16	上跳触发淹码	U32	0x00000000

17～20	下跳触发淹码	U32	0x00000000
21～24	预设数字常量	U32	0xFFFFFFFF
25	位操作符（同位操作指令功能码）	U8	0x03（异或）
26	滤波系数（0～16）	U8	4
27	执行标识：0 为不执行 1 为执行	U8	1
28～30	触发指令（优先级＋功能码＋DLC）	指令字节	0x1 0x19 0x04 DLC＝衍生数字量长度
31～32	保留	U8	0
33～60	第二组	变化触发淹码～保留	
33（61）	类对象结束符	U8	0x5A

该类首先使用本地 I 特性数字量输入与“预设数字常数”进行位逻辑操作，通常是“异或”操作，即对某些输入位进行反相，然后对其进行滤波，当某位连续有“滤波系数”个同值，则将该“位值”写入“衍生数字量”的对应位中，滤波系数越大滤波作用越强。数字量的有效长度见“触发指令”中的 DLC，这里的“滤波”是逐位滤波的，不是按字节或按 DLC 长度整体滤波的，因为数字量输入的各位可能是彼此独立的。

“衍生数字量”与其自身“历史值”进行比较，参照三种“触发淹码”，对应“淹码”为 1 的位发生了相关的“跳变”则触发指令将该“衍生数字量”播发到网络上，并更新其历史值。

举例中的含义：本地基地址为 0x70D0 的 4 个字节数字量输入，求反后，进行逐位滤波（滤波系数为 4），将其写入衍生变量系统寄存器空间 0x1262，当与其历史值有不同时，则

触发上传指令 0x1 0x12 0x62 0x32 0x19 0x4 0xnn 0xnn 0xnn 0xnn，并将其历史值更新为 0xnn 0xnn 0xnn 0xnn。

12.7.2 数字量输出预处理类

通常设备可以设计一个衍生 O 特性数字变量来获取主机或其它设备的输出控制（设置）该衍生变量与预设常数进行所需逻辑运算后，再进行端口输出。

相关类对象空间设计和分配，见表 12－26。

表 12－26 数字量输出预处理类对象空间分配

寄存器字节编号	内容用途	数据类型	举例
1	本对象的总长度	U8	17（29，41，53）
2	对象的状态	U8	0x22
3	类的应用行业	U8	2
4	类种类编号	U8	2

5～6	衍生数字量寄存器基地址	U16	0x2228
7～8	本地数字量输出寄存器基地址	U16	0x74E0
9～12	预设常量	U32	0xFFFFFFFF
13	位操作符（同位操作指令功能码）	U8	0x03（异或）
14	执行标识：0 为不执行 1 为执行	U8	1
15～16	空	U8	0
17～28	第二组	衍生变量寄存器基地址～保留	
29～40	第三组	衍生变量寄存器基地址～保留	
41～52	第四组	衍生变量寄存器基地址～保留	
17（29	类对象结束符	U8	0xAA

，41，53)			

举例中的含义：设备通过系统寄存器空间的 0x2228 衍生 O 特性数字量，接收主机（或其它设备）的控制指令，将其求反后，在通过本地数字量输出寄存器 0x74E0 进行端口输出。

12.7.3 模拟量输入预处理类

使用本地资源模拟量输入加权后与预设常量相加生成新的过程变量（即：衍生资源）。该衍生资源与其自身历史值进行比较，如满足触发条件，则触发指令发送，并更新历史值。触发条件分为“增量触发”和“上下限触发”

在这里衍生变量、加权后值、预设常量、增量值和上下增量值都为实数类型。

相关类对象空间设计和分配，见表 12－27。

表 12－27 模拟量输入预处理类对象空间分配

寄存器字节编号	内容用途	数据类型	举例
1	本对象的总长度	U8	33（61）
2	对象的状态	U8	0x22
3	类的应用行业	U8	2
4	类种类编号	U8	3
5～6	衍生变量寄存器基地址	U16	0x1800
7～8	本地输入变量寄存器基地址	U16	0x72E0
9～12	加权值	f32	6.6
13～16	预设常量	f32	－111.8
17～20	预设增量	f32	20
21～24	预设上界值	f32	2000
25～28	预设下界值	f32	100
29	执行标识： 0 为不执行 1 为增量触发 2 为上限触发	U8	1

	3 为下限触发 4 为上下限触发		
30～32	触发指令（优先级＋功能码＋DLC）	指令字节	0x1 0x19 0x04 DLC＝衍生变量长度
33～60	第二组	衍生变量寄存器基地址～触发指令	
33（61）	类对象结束符	U8	0x5A

该类的数学表达为：衍生变量 ＝ （输入变量 * 加权值 ＋ 预设常量）；

增量触发过程同 12.6.2。 越界触发过程同 12.6.3。

12.7.4 模拟量输出预处理类

设备支持模拟量输出时，通常设备会派生

出一个相对应的衍生变量，来获取主机或其它设备的输出设置，该衍生变量加权与预设偏置常数相加后，进行输出。

相关类对象空间设计和分配，见表12－28。

表12－28 模拟量输出预处理类对象空间分配

寄存器字节编号	内容用途	数据类型	举例
1	本对象的总长度	U8	21（37，53）
2	对象的状态	U8	0x22
3	类的应用行业	U8	2
4	类种类编号	U8	3
5～6	衍生变量寄存器基地址	U16	0x2800
7～8	本地输出变量寄存器基地址	U16	0x76E0
9～12	加权值	f32	6.6
13～16	预设偏置常量	f32	－111.8
17	执行标识：		

	0 为不执行 1 为执行	U8	1
18～20	保留	U8	
21～36	第二组	衍生变量寄存器基地址～保留	
37～52	第三组	衍生变量寄存器基地址～保留	
21（37，53）	类对象结束符	U8	0x5A

该类的数学表达为：本地输出变量 ＝（衍生变量 * 加权值 ＋ 预设偏置常量）；

从以上常用 I/O 类中，我们看到其将数据预处理和触发指令合成在一个类对象中实现了其实我们完全可以单独定义数据处理类（或称运算类），再单独定义各种触发类，这样在应用中会更加的灵活和方便。

12.8 TC1004 子协议（IF－

THEN 类）

在生产者－消费者模型中，设备对某系统信息敏感，而执行某动作，一般都需要 IF-THEN 类的帮助。IF-THEN 类是一个智能判别类，如：根据温度值（系统信息）开启空调或风扇等。

TC1004 子协议是由 TTCAopen 组织定义的常用 IF－THEN 类，目前在 TC1004.001 中定义了两个基本类，供用户参考。

12.8.1 数字量敏感 IF-THEN 类

设备对某数字量敏感，从而影响另一变量的数值。

相关类对象空间设计和分配，见表 12－29。

表 12－29 数字量敏感 IF-THEN 类对象空间分配

寄存器字节编	内容用途	数据类型	举例

号			
1	本对象的总长度	U8	33（61）
2	对象的状态	U8	0x22
3	类的应用行业	U8	3（常用 IF－THEN 类）
4	类种类编号	U8	1（第 1 个类定义）
5～6	受影响的变量寄存器基地址	U16	0x2100
7～8	敏感数字量寄存器基地址	U16	0x1100
9～12	变化触发淹码	U32	0xFFFFFFFF
13～16	上跳触发淹码	U32	0x00000000
17～20	下跳触发淹码	U32	0x00000000
21～24	预设常量 1	U32	0x0
25～28	预设常量 2	U32	0x1234
29	执行标识：0 为不执行	U8	1

	1 为执行		
30～32	保留	U8	
33～60	第二组	受影响的变量～保留	
33（61）	类对象结束符	U8	0x5A

敏感数字量与其历史值比较，其结果满足触发条件，则将受影响的变量原值与预设常量1“位与”，再和预设常量2“位或”，其结果更新受影响的变量值。

举例中的含义：设备对系统信息（0x1100）敏感，当其发生变化时，其受影响的输出寄存器（0x2100）被设置为0x1234。

12.8.2 模拟量敏感 IF-THEN 类

设备对某模拟量敏感，从而影响另一变量的数值。

相关类对象空间设计和分配，见表12－30。

表 12－30 模拟量敏感 IF-THEN 类对象空间分配

寄存器字节编号	内容用途	数据类型	举例
1	本对象的总长度	U8	33（61）
2	对象的状态，	U8	0x22
3	类的应用行业	U8	3
4	类种类编号	U8	2
5～6	受影响的变量寄存器基地址	U16	0x2100
7～8	敏感模拟量寄存器基地址	U16	0x1400
9～12	加权值	f32	10
13～16	预设偏置常量	f32	－200
17～20	预设比较常理	f32	88
21～24	预设常量 1	U32	0x0
25～28	预设常量 2	U32	0x1234
29	执行标识：0 为不执行	U8	1

	1 当大于时执行 2 当大于等于时执行 3 小于时执行 4 小于等于时执行 5 等于时执行		
30～32	保留	U8	
33～60	第二组	受影响的变量～保留	
33（61）	类对象结束符	U8	0x5A

敏感模拟量经加权与预设偏置常量相加后的结果与预设比较常量进行比较，满足执行标识中的条件，则将受影响的变量原值与预设常量 1“位与”，再和预设常量 2“位或”，用其结果更新受影响的变量值。

12.9 讨论与提示

12.9.1 诚惶诚恐

在前面各章中，我们信心满满，对 TTCA Nopen 各种通讯规范的设计都非常有成就感，是脚踏实地、顺理成章的，有一种“舍我无他”的感觉。

而本章则大不同，用“盲人扶墙”来比喻，是最贴切不过了。

在这里，我们只不过算是“抛砖引玉”，为读者和用户提供一个似乎可行的思路。在读者和用户中，可能会有更加优秀的解决方案。

尤其，当我们设计 IF－THEN 类时，总有一种力不从心的感觉，要知道在实际应用中，这种智能判断执行过程，可以是一个非常复杂的逻辑过程，而我们设计的 IF－THEN 类则过于单薄，无法应对复杂的应用。因此我们需要在设备中嵌入一种微型的“程序语言”和其配套的解译器，用来提高设备的“智能”水平

这是我们努力的方向。

12.9.2 类为通用设备插上翅膀

有了类对象，这个集中存放设备完成某种特定功能所需配置参数和变量的空间，设备开发者和设备生产厂商就可以为通用设备预添加更多的设备功能，使通用设备能够适用更多的应用现场 。

12.9.3 对象词典

学习和使用过 CANopen 的读者，对“对象词典”一定不陌生，它是 CANopen 系统中描述设备参数的列表，在 CANopen 中使用了一个 16 位索引和一个 8 位的子索引共同对设备参数和功能进行查询。

本章的内容很大程度上是提供了一个类似“对象词典”功能的规范，其使用相对简单灵活。

12.9.4 配置信息的保护

到目前为止，我们已经向用户开放了 0x8000、0x9000、0xD000、0xE000 和 0xF000 段配置参数区，配置参数对于一个设备是非常重要的，它决定了此设备非彼设备，为了更加有效的保护这些配置参数，保证其可靠性，我们为设备管理器配置参数专门提供了一套指令集，而且多为命令/应答方式的。

只有这些还不够，在测试中，尤其是在人工调试过程中，很有可能出现对设备配置参数的误设和误删行为。为了减少这种行为，我们为配置参数区提供了一种写保护机制。

在每个设备的 0xF000 段，第一个字节我们称之为“段保护机制”字节。该字节平时状态为 0x00，当我们需要向设备的某配置参数段写数据时，必须将该字节内容设置为该段的段地址[1]。如：我们要向某设备的 0x9100 写入

数据，首先需将该设备的“段保护机制”字节设置为 0x09，方能对 0x9000 段进行写操作；当我们要向 0xF008 写数据时，需将“段保护机制”字节设置为 0x0F 后，方能对 F000 段进行写操作。该字节的保护作用只针对设备“配置参数空间”，对多个设备同时设置时，支持广播和组广播指令。

该保护机制使得对“配置参数空间”的配置操作过程变的更加繁琐，但可以有效的降低我们在调试过程中的错误操作。这也正如许多软件中对重要参数的修改需要“确认”一样。为了“可靠”失去少量的“便捷”是值得的。

关于配置信息保护的更多内容详见：TC0064 子协议（配置参数段保护机制）。

注：该保护机制只针对网络指令操作有效，对应用程序不起作用，也就是说设备本地应用程序对配置参数的操作不受该保护机制的约束相关操作安全需由应用本身去保证。

12.9.5 输出设备的初始值和安全值

当设备上电后，由初始化程序将设备“初始值”置入到输出端口中，当设备出现故障处于“安全态”时，设备应用程序会及时的将“安全值”置入到设备输出端口中。在 TTCA Nopen 协议中，这两个值被认为是相同的。

12.9.6 类功能代码只在设备处于“运行态”方被启动

是的，类功能代码只有在设备处于“运行态”方可执行。因此，类主要是处理设备过程变量的功能机制，一般情况下请不要企图使用“类”来处理配置参数。

注：［1］后来改为双写段地址，这样显得更加清晰，如：0x88、0x99、0xEE 和 0xFF 等。