第 2 章“承前，启后”

主从模型

2.1 “承前启后”

“主从模型”是集中监控系统最常用的通讯模型，有 70％以上的系统可以用它来完成，485 总线＋MODBUS 协议则是应用“主从模型”的经典组合，大多数工程师对它都不陌生，市面上几乎所有的组态软件对它都有完美的支持。

“主从模型”系统由一个主机和多个从机组成，其最大特点是：有着严格的“收发”通讯时序，即由“主机”发起通讯，“从机”进行应答，总线不存在着“竞争”，时序简单，可靠性高，且编程难度低，因此它也成为应用最多、最广、最易掌握的通讯模型之一，很多工程师都是从这里开始起步的，走好第一步，才能为后续的发展奠定坚实的基础。

TTCANopen 应用层协议也是从这里开始，从一粒种子，不断的吸收营养，慢慢的成长为一棵枝繁叶茂的大树。

2.2 YTC0012 主从模式之二

TTCANopen 之 YTC0012 协议是 TTCANopen“应用子协议”中比较典型的主从模型应用环境，其具体内容参见下表：

段寄存器	支持协议		备注
0x7000 段	TC0030A000 TC0031A000		TC0008（由用户设备自定义）
0x8000 段	TC0020.A000		TC0009（由用户设备自定义）
0xF000 段	TC0016A001 TC0020A000		

从表中，我们可以看到 YTC0012 只支持 3 个段地址的访问，其它段是不可用的，这 3 个段都是规定的面向节点的通讯协议空间，与其支持的相关指令集相匹配，是一个非常典型的

主从通讯模型应用环境。

2.3 主从模型

1 命令 / 应答方式

所谓“主从通讯”模型，包括主机对从机的“命令/应答”方式和主机对从机的广播方式。（注：在后面的章节我们还会继续“广播方式”的探讨。）

主机对从机的“命令/应答”方式，通讯由主机发起，在主机发起的命令中包含指定的应答从机地址，从机接收到命令后，对比命令中的从机地址，如果是“自己”，则立即执行指令并进行回复“应答”，如果不是“自己”，则忽略此命令，在该方式下，主机采用“轮询”的方式，向各从机发布命令获取数据和设置数据，命令与应答是成对出现的，图 2－1 为“命令/应答”方式框图。

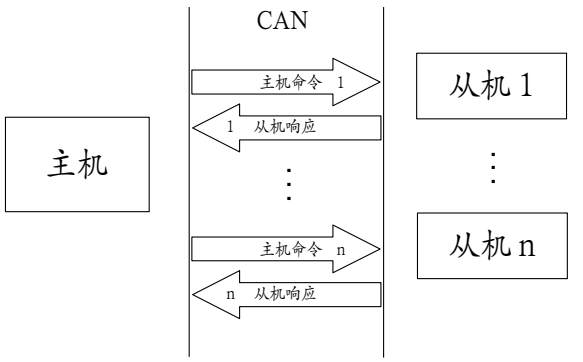

图 2－1 “命令/应答”方式框图

主机对从机的广播方式，是主机对全部从机发送指令，从机无需进行应答，图 2－2 为广播方式框图。

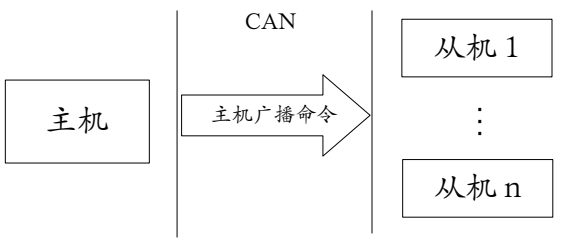

图 2－2 广播方式框图

对于 TTCANopen 协议来说，各种指令与应答都是针对主机或从机的“寄存器”进行的“读写”操作，我们“认定”读取“寄存器”中的数据是“已经”准备好的，而写“寄存器”是“已经”就绪的（这一点很重要，对本书的后续章节也同样适用），因此，TTCANopen 协议指令在设备上执行的时间只是“读写”寄存器的时间，这样一次“命令/应答”回合的用时是快捷的和可预测的，这对提高总线通讯效率是非常有好处的，而 CAN 底层协议在通讯发生错误时，会自动进行重发，并屏蔽通讯严重故障设备，因此在应用层我们不再提供重发机制，当主机在规定的时间内没有收到从机应答，主机会启动发下一条指令。

对于一个 CAN 系统来说，一般有三种数据要参与通讯，“配置参数”、“过程变量”和“错误信息”，其中“配置参数”是指通讯参数、设备管理配置、设备特征参数以及过程变量的初始值等，这些参数一般只需在设备“运行”前进行一次性配置即可；“过程变量”是指设备“运行”产生的数据变量（如：各种 I/O 变量）和“运行”所需的数据变量，这些变量是在设备“运行”过程中产生的并要和其他设备或主机进行交互，完成某种系统任务功能；“错误信息”是指设备运行和执行指令过程中产生的错误诊断信息，其中部分通讯错误可以通过通讯子协议获取，而设备应用和硬件及厂家定义的错误等，则需要主机轮询“设备错误标识寄存器”来获得。

通常在“主从通讯”模型中，主机承担着三种角色，1：网络管理和配置管理；2：数据处理；3：错误处理。他们的通讯内容分别对应着“配置参数”、“过程变量”和“错误信息”。

2 主机事务处理流程

对于具体的用户应用，主机事务处理流程可能有所不同，下面给出一个简易主机参考事务处理流程图，见图 2－3。

① 主机初始化：包括对主机通讯参数的设置和主机特性参数的配置等；

② 配置从机参数：通过 CAN 网络配置从机特性参数和过程变量的初始值；

③ 轮询矩阵：进入运行阶段，按照轮询矩阵的排布顺序从 CAN 网络采集数据；

④ 数据处理：结合应用对采集到的数据进行处理，包括异常处理，生成输出过程变量（包括转发过程变量，注：主从通讯模式是不支持设备间直接通讯的，如有需求则要通过主机转发。）；

⑤ 输出过程变量：向 CAN 网络发送输出过程变量（包括对从机的控制，这些控制

有些是系统应用程序生成的，有些是用户通过应用界面交换产生的）；

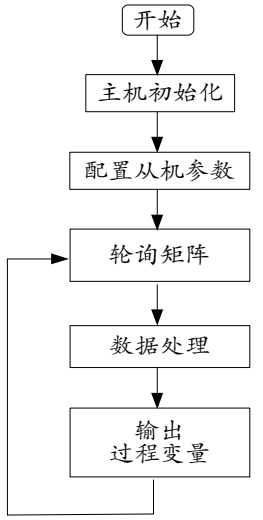

图 2－3　简易主机事务处理流程

除了上面宏观的主机事务处理流程，针对“命令/应答”过程也会有一个微观的“命令/应答”主机程序流程，见图 2－4。

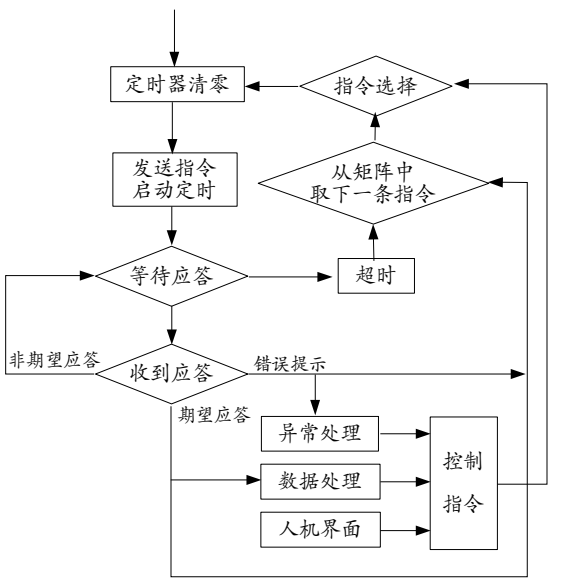

图 2－4“命令/应答”主机程序流程

① 发送指令：无论主机是对从机进行配置还是获取输入过程变量和发送输出过程变量，都要获取相关联从机的应答，而这种应答是有限时间的，所以主机在成功发令后都要启动应答等待定时器；

② 获得应答：当主机收到应答后，判别其为完整的“期望应答”或“错误提示”并进行相应的“数据处理”和“异常处理”，同时可进入下一条指令的发送循环；如果为“非期望应答”则继续等待；

（注：对于多“应答”指令，每收到一个应答，都应在接收中断服务程序中重置等待定时器，给其提供一个较小的等待增量。）

③ 等待超时：当等待超时，主机将从轮询矩阵中选择下一条指令进行发送；

④ 数据处理和异常处理：这两个模块与“取下一条指令”是并发的，而不是等这两个模块处理完再去执行“取下一条指令”，这样的设计是为了提高 CAN 总线的利用率，最大限度的减少总线的空闲时间，实际上包括这两个模块和更多的系统模块程序线程，我们努力将他们放在总线发送数据和接收数据时刻并行工作；

⑤ 控制指令：这些指令是由主机应用程序产生的即时控制指令，他们可以来源于数据处理、异常处理和人机界面等，这些指令一旦产生，原则上一般应当优先于“轮询指令”发送，具体应当由应用的控制逻辑在指令选择器上完成发送次序的排列。

（注：如果发送的是广播指令，则不用启动定时器等待应答，可直接发送下一条指令。）

对于以 PC＋Windows 作为主机构成的中心监控系统而言，由于 Windows 操作系统的非实时性，再加上 USB 转 CAN 模块的耗时，会使系统实时性打折，等待应答的时间会增长。

3 从机事务处理流程

从机的事务处理流程比主机要相对简单许多，下面给出一个简易从机参考事务处理流程图，见图 2－5。

① 从机初始化：包括对从机通讯参数的设置、从机特性参数和从机过程变量的初始值的配置等，其中从机特性参数和从机过程变量初值可由从机自我配置（自举），也可由主机对其进行远程配置；

② 等待指令：从机不能主动发送指令，只能在收到针对自己的指令后进行应答，在从机等待的过程中，从机的“应用程序”一直在运行着，并和寄存器发生交互操作，如不

断的刷新 A/D 采集值，预备主机读取到最“新鲜”的数据；

③　寄存器操作：TTCANopen 协议是针对寄存器的协议，几乎所有的指令都是对相关寄存器进行“读”“写”操作；

④　指令应答：寄存器操作后应当立即生成应答指令，并进行回复；

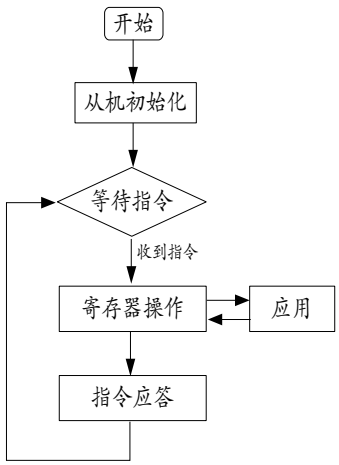

图 2－5　简易从机参考事务处理流程图

下面是“命令/应答”从机程序流程，见图 2－6，

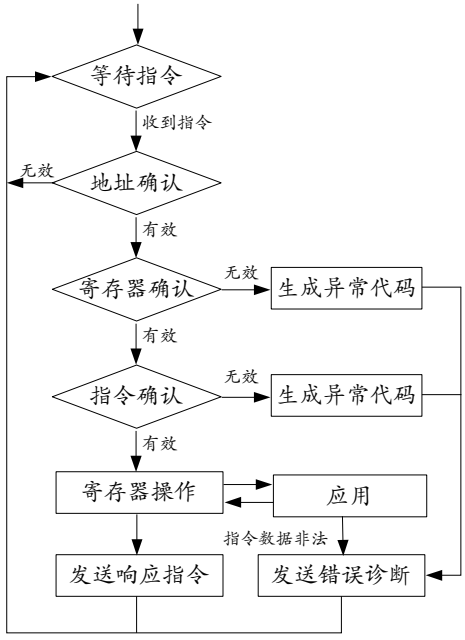

图 2－6　命令/应答”从机程序流程

① 等待指令：在“主从模型”下从机不会主动发送指令，而是处于接收指令状态；

② 地址确认：从机收到指令后，对指令中的地址进行判别，是“自己”则进行下一步的判别；否则放弃该指令，继续等待接收新的指令；

③ 寄存器确认：对指令中的“寄存器地址”进行确认，“支持”则进行下一步判别；否则生成相关异常代码，向主机回复“错误诊断指令”，

④ 指令确认：对指令中的“功能码”进行确认，“支持”则完成相关操作（访问相关寄存器）；否则生成相关异常代码，向主机回复“错误提示指令”；

⑤ 寄存器操作：在这里设备的应用程序应对指令数据的合法性进行判定，合法即完成指令所要求的寄存器操作，否则即生成错误诊断指令返回主机；

⑥ 发送响应指令：按指令要求生成应答指令，并发送；

（注：如果接收到的是广播指令，则无需应答，完成指令操作后直接进入等待状态）

2.4 讨论与提示

2.4.1 “应答块”

主机在采集从机连续多个（大于 8）寄存器数据时，从机需要连续使用多条指令进行回应，我们可以把这些应答指令看为一体，称其为“应答块”，用户在设计程序流程的过程中，应当注意对“应答块”的处理，在配置信息时尽量将相关信息放在地址连续的寄存器中，以提高信息读取效率。

另外，在设计主机定时器时长时，应当考虑对“应答块”的接收时间，一般我们会在接收中断服务中为定时器提供一个增量。

（注：如果系统或设备支持 CAN FD，对多字节的传输更加有利，但要注意大于 8 字节后的非线性编码。）

2.4.2 局部访问

一条 CAN 指令一次最多可以携带八个字节的数据，我们通常把特性相同内容相关的变量或参数安排在一个连续寄存器空间，用以提高 CAN 指令的效率，例如：我们把设备厂商参数集中放置在 0xFBF0 ～ 0xFBF7 连续八个寄存器中（参看 TC0016），这样主机只要一条指令 0x01 0xFBF0 0x35 0x0E 0x1 0x08 便可获取地址为 0x35 的设备的“产品编号”“产品硬件版本号”“产品软件版本号”“厂商 CVID”和“产品 CPID”五项内容。当然主机也可以只读取

该寄存器空间的局部，如：0x01 0xFBF3 0x35 0x0E 0x1 0x01 只读取“产品软件版本号”，甚至只读取“厂商 CVID”的高字节，如： 0x01 0xFBF5 0x35 0x0E 0x1 0x01，对于 TTCANopen 协议这都是准许的，因为 TTCANopen 通讯协议将寄存器空间看做是一个可传递的“字节容器”。但有些设备厂商生产的设备，为了简化寄存器空间访问的复杂性，其只支持“整体访问”如：0x01 0xFBF0 0x35 0x0E 0x1 0x08，而不支持“局部访问”如： 0x01 0xFBF0 0x35 0x0E 0x1 0x07 ，和 0x01 0xFBF5 0x35 0x0E 0x1 0x01，也是可以理解和得到系统支持的。

另外，对于一个多字节变量如：浮点数的局部读写是没有什么意义的，但其在 TTCANopen 中却是可实现的，而有些用户则会认为对一个完整变量的局部访问当属非法。

为了方便设备厂商和用户对“局部访问”操作的限制，我们特别提供了一个错误提示标识符“0x09”，表示设备不支持该局部访问。如：当主机发送 0x01 0xFBF0 0x35 0x0E 0x1 0x07 或 0x01 0xFBF5 0x35 0x0E 0x1 0x01 等对 0xFBF0 ～ 0xFBF7 的局部访问指令时，设备可以返回 0x01 0xFBF0 0x35 0x1F 0x2 0x0E 0x09 或 0x01 0xFBF5 0x35 0x1F 0x2 0x0E 0x09 错误提示指令。

2.4.3 CAN 指令寄存器读写与其对应的 I／O 操作

在本章正文中，我们曾经提到过：“ 对于 TTCANopen 协议来说，各种指令与应答都是针对主机或从机的“寄存器”进行的“读写”操作，我们“认定”读取“寄存器”中的数据是“已经”准备好的，而写“寄存器”是“已经”就绪的。”

在这里 CAN 指令对寄存器的读写操作，与其对应的 I/O 操作并不是完全等价的，对于 I 特性过程变量，如 A/D 采集，设备本地应用程序不断的以一定的频率采集数据并刷新其对应的过程变量寄存器，当设备接收的主机指令读取该过程变量寄存器时，设备直接将当前该寄存器中的数值返回到主机，而不是现去采集填充；对于 O 特性过程变量，如 D/A 输出，当设备接收的主机指令设置该过程变量寄存器时，设备按主机指令数据更新该 D/A 过程变量寄存器值，并立即返回应答指令，设备本地应用程序检测到该 D/A 过程变量寄存器值发生了更新，然后将其值输出到 D/A 端口。从以上我们看出对于 TTCANopen 协议，其 I 特性过程变量寄存器和 O 特性过程变量寄存器并不直接对应设备的 I/O 端口，它们只是设备端 I/O 信息传递的中间载体。

另外，还有一种过程变量，它即可被主机通过 CAN 指令设置改写，体现为 O 特性；也可被设备本地应用程序改写，体现为 I 特性。我们称之为 I/O 特性变量。

2.4.4 CAN 接收滤波器

不同的 CAN 芯片生产厂家都为其 CAN 接收端口设计了不同的硬件接收滤波器，用于屏蔽和有选择的接收总线上的 CAN 指令，合理的使用这些接收滤波器可有效的提高 CAN 端口应用程序的工作效率。

TTCANopen 协议将 CAN 指令 ID 字划分为 4 个固定字段，即“优先级”、“寄存器地址”、“设备地址”和“功能码”，较其它协议更加适合使用接收滤波器进行选择性验收，即便如此，本书本着一般性原则，在系统流程描述中仍旧使用软件过滤的方法，使用 CAN 接收滤波器留作用户在系统实践中去完成，本书以后将不再涉及相关内容。

2.4.5 从机设备信息表

用户应当为每一个从机设备建立一个设备信息表，表述从机设备功能、支持协议、资源配置、设备地址、寄存器分配和必要的原理框图等信息。

2.4.6 轮询矩阵

主机对过程变量的采集是通过轮询矩阵完成的，建立一个合理的轮询矩阵是主机应用的重要环节，由于各种过程变量的特性、快变慢变和重要程度不同决定了对各过程变量的采集周期要求不同，因此在一个轮询矩阵过程中，不同的过程变量被采集的次数是不同的，下面给出一个轮询矩阵样例：

A1	B	A2	D1	A3	B	A4	C1	A5	B	A6	A7
A1	B	A2	D2	A3	B	A4	C2	A5	B	A6	A7
A1	B	A2	E	A3	B	A4	C3	A5	B	A6	A7
A1	B	A2	D3	A3	B	A4	C4	A5	B	A6	A7
A1	B	A2	D4	A3	B	A4	C5	A5	B	A6	A7
A1	B	A2	D5	A3	B	A4	C6	A5	B	A6	A7
A1	B	A2	D6	A3	B	A4	C7	A5	B	A6	A7
A1	B	A2	E	A3	B	A4	C8	A5	B	A6	A7
A1	B	A2	D7	A3	B	A4	C9	A5	B	A6	A7
A1	B	A2	D8	A3	B	A4	C10	A5	B	A6	A7
A1	B	A2	D9	A3	B	A4	C11	A5	B	A6	A7
A1	B	A2	D10	A3	B	A4	C12	A5	B	A6	A7
A1	B	A2	E	A3	B	A4	C13	A5	B	A6	A7
A1	B	A2	D11	A3	B	A4	C14	A5	B	A6	A7
A1	B	A2	D12	A3	B	A4	C15	A5	B	A6	A7

在这个轮询矩阵中，A 类变量被采集了 18 次，B 变量被采集了 45 次，C 类和 D 类变量被采集了 1 次，E 变量被采集了 3 次。

2.4.7 主机转发信息

通常原始意义上的集中监控系统，总是把主机当做系统的智能中心，所有的执行判断逻辑都集中于此，从机只是简单的数据源和执行机，不具备智能逻辑判断能力。

随着时代的发展，分布式系统更愿意把专有的智能内容独立出来，将其放置在分布的单机或从机上，以降低主机系统的繁杂度。

例如：一个简单的主从模型系统中，有两个从设备，一个是温度传感器，一个是空调启动开关。通常传统的中心监控系统，是由主机不断的采集温度值，并对其进行逻辑判断，然后发出指令开启或关闭空调。而在现代分布式系统中，逻辑判断能力赋予了空调启动开关，主机的任务只是将采集来的温度值“写”入到空调启动器相关的寄存器即可。

当系统演绎到“生产者消费者”模型后，实现了设备间的直接通讯，温度传感器的温度值可直接通过系统寄存器空间播发到网络上，空调控制器对该信息敏感，从而控制空调的启停，这一过程免去了主机转发的环节，缩短了信息闭环路径，提高了系统的时效，这也是 TTCANopen 后续章节需要实现的目标。

2.4.8 客户－服务

TTCANopen 主从通讯模型中，主机读取从机寄存器内容可以看作是一个简单快捷的客户/服务关系。然而，我们讲的普通意义上的客户/服务关系往往比这更复杂，它更像 internet 网应用中的客户/服务关系，需要服务器返回的信息可能会更多，用时更长。这在纯主从模型中实现起来则需要多个步骤才能完成。首先，客户端（主机）发出服务请求标识，如：写从机的 0x7900 寄存器为 1（请求服务标识），此时从机的应答只是表明从机接受了该服务请求，并没有把服务结果返回给主机，服务器（从机）执行该服务任务操作后，将任务结果放置在 0x7300～0x7320 的寄存器中，并将 0x7900 置 0，表示服务完成。主机在发出服务请求标识后，通过轮询矩阵不断的读取 0x7900 中的内容，当其被置 0 后，主机发出读取 0x7300～0x7320 寄存器指令，以获得服务结果。

上述服务，主机和从机发生了多轮“命令/应答”过程。如果从机能够主动发送信息给主机，这一服务通讯过程的效率就会有所提高。

2.4.9 告警信息

在一个 CAN 应用系统中，当发生告警时，告警信息应尽快的能够被主机采集到，那么就要求告警信息在轮询矩阵中有较高的采集频率；而在未发生告警时，对告警信息的高频采集显然浪费了总线带宽，在纯主从通讯模型中这对矛盾是很难协调的，如果从机能够在发生告警时及时主动上报告警信息，问题就迎刃而解了，这也就是本书下一章将要介绍的上位机和下位机的通讯关系，上位机可以轮询下位机的信息，同时下位机也可以主动上报信息，因此总线也就可能产生“竞争”，由此引入了 CAN 总线的无损竞争特性。