第 4 章“脱胎，换骨”

全事件触发

4.1 “脱胎换骨 ”

作为通讯协议的设计者，总是希望能够充分利用有限的总线带宽，传输更多的信息，那么也就要求协议中没有或尽可能少的传输“冗余信息”。

在上一章 3.4.4 中笔者建议只使用“参数读取指令的应答部分（0x1E）”作为“心跳指令”向主机传送心跳信息，省略了主机发送的“数据读取指令（0x0E）”，原因在于周期发送的“心跳信息读取指令（0x0E）”，是总线上可预期的“信息”，从信息学的角度来说它不含有任何新鲜的内容，完全可以去除，如此推而广之，轮询矩阵中周期性的数据读取指令都应该属于此类“冗余信息”，因为这些“读取指令”在总线上都是可根据轮询矩阵的排列而“预期”的，也就是可以被去除的内容。

另外，从机“数据上报指令（0x06）”和对“数据读取指令的应答（0x19）”如果其所携带的数据内容没有发生变化，也就不需要重复的上报或回复了，这就是所说的重复传输冗余。

去除轮询矩阵中的“读取指令（0x09）”，任由各从机以一定频度使用“应答指令格式（0x19）”或由事件触发（主动使用 0x19）将信息传递给主机是可行的，此时的总线是一个由事件触发的完全竞争环境，理论上其总线利用率可以达到最高，至此 TTCANopen 协议完成了主从模型的命令/应答方式至 CAN 的事件触发方式的演绎过程。

4.2 无确认应答指令集

之前，我们介绍的指令集，都是“指令”和“应答”成对出现的，我们现在将“0x09 指令”的应答指令“0x19 指令”和“0x07 指令”的应答指令“0x17 指令”单独提炼出来，成为两条独立的传输信息的指令。

4.2.1 TC0035A000 子协议（无确认应答过程变量基本指令）

无确认应答过程变量指令

本子协议（通讯子协议）定义了 2 个无“确认应答”，“过程变量”的基本指令，分别是“信息单上传指令”、“信息单下传指令”。本子协议指令操作仅适用于“设备信息空间”即：第 0x7000 段的数据交互。

1 信息单上传指令

功能码：0x19

设备（从机）主动发送其指定基地址的数据到“主机”，一次可上传 1～8 个字节的数据，第 2 个字节往后为基地址的顺延。

信息单上传指令格式：

优先级	寄存器基地址	设备地址	0x19	DLC	1～8 个字节数据

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

该指令由设备触发（如告警、计时、数据变化等条件触发），该指令无需“主机”确认应答。

当主机不能处理该寄存器信息，或不接受该指令，则返回错误提示指令如何下，

错误提示指令格式：

原优先级	原寄存器基地址	原设备地址	0x1A	0x2	0x19	1 个字节错误编号

例：0x1 0x7100 0x35 0x1A 0x2 0x19 0x06 （主机不能处理该寄存器信息）

0x1 0x7100 0x35 0x1A 0x2 0x19 0x07 （主机不接受该指令）

2 信息单下传指令

功能码：0x17

“主机”主动向指定设备的基地址填充数据，一次可填充 1～8 个字节的数据，第 2 个字节往后为基地址的顺延。该指令无需设备“从机”确认应答。

信息单下传指令格式：

优先级	寄存器基地址	设备地址	0x17	DLC	1～8 个字节数据

例：0x1 0x7500 0x35 0x17 0x4 0x11 0x22 0x33 0x44

当设备不能处理该寄存器信息，或不接受该指令，则返回错误提示指令如何下，

错误提示指令格式：

原优先级	原寄存器基地址	原设备地址	0x1A	0x2	0x17	1 个字节错误编号

例：0x1 0x7500 0x35 0x1A 0x2 0x17 0x06 （设备不能处理该寄存器信息）

0x1 0x7500 0x35 0x1A 0x2 0x17 0x07 （设备不接受该指令）

3 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。

4.2.2 TC0022A000 子协议（无确认应答配置参数基本指令）

同样，读者很自然的会想到配置参数是不是也有两条类似的指令，回答是肯定的，设备心跳信息和设备故障信息的发送就是其最典型的应用。

无确认应答配置参数指令

本子协议（通讯子协议）定义了 2 个无“确认应答”，“配置参数”的基本指令，分别是“配置信息单上传指令”、“配置信息单下传指令”。本子协议指令操作仅适用于“设备配置参数空间”即：第 0x8000 段至第 0xF000 段的配置参数区。

1 配置信息单上传指令

功能码：0x1E

设备（从机）主动发送其指定基地址的配置参数到“主机”或设备管理器，一次可上传 1～8 个字节的参数，第 2 个字节往后为基地址的顺延。

配置信息单上传指令格式：

优先级	寄存器基地址	设备地址	0x1E	DLC	1～8 个字节参数

例：0x1 0x8110 0x35 0x1E 0x8 0x11 0x22 0x33 0x44 0x00 0x00 0x01 0x33

该指令由设备触发，该指令无需“主机”确认应答。

当主机不能处理该寄存器信息，或不接受该指令，则返回错误提示指令如何下，

错误提示指令格式：

原优先级	原寄存器基地址	原设备地址	0x1F	0x2	0x1E	1 个字节错误编号

例：0x1 0x8110 0x35 0x1F 0x2 0x1E 0x06 （主机不能处理该寄存器信息）

0x1 0x8110 0x35 0x1F 0x2 0x1E 0x07 （主机不接受该指令）

2 配置信息单下传指令

功能码：0x1C

“主机”或设备管理器主动向指定设备的基地址填充配置参数，，一次可填充 1～8 个字节的参数，第 2 个字节往后为基地址的顺延。该指令无需设备“从机”确认应答。

信息单下传指令格式：

优先级	寄存器基地址	设备地址	0x1C	DLC	1～8 个字节数据

例：0x1 0x8100 0x35 0x1C 0x4 0x11 0x22 0x33 0x44

当设备不能处理该寄存器信息，或不接受该指令，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	0x1F	0x2	0x1C	1 个字节错误编号

例：0x1 0x8100 0x35 0x1F 0x2 0x1C 0x06 （设备不能处理该寄存器信息）

0x1 0x8100 0x35 0x1F 0x2 0x1C 0x07 （设备不接受该指令）

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。

4.3 全触发模式

4.3.1 无确认且出错提示事件触发方式

在上位机下位机模式中的事件触发，从机主动向主机发送信息是需要主机进行确认应答的，而在“全触发模式”中，从机向主机发送信息是不需要主机做确认应答的，只有出错时

主机才会发送“出错提示信息”，这大大提高了总线的使用效率。

无确认且出错提示事件触发方式，事件可以是数据变化、越界、超时和周期定时等，通讯由从机发起，在发送的信息指令中包含该从机的地址，主机接收到信息后，不作确认应答，只有指令出错时，进行出错提示，图 4－1 为“信息/出错提示”方式框图。

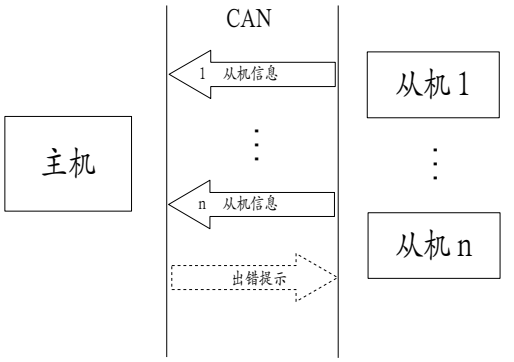

图 4－1 信息/出错提示方式

4.3.2 无确认且出错提示输送方式

同样，对应于主机输送到设备的信息，设备也无需做确认应答，只在出错时给出错误提示即可，我们称之为“输送/出错提示” 方式。通讯由主机发起，在发送的信息指令中包含从机的地址，从机接收到信息后，不作确认应答，只有指令出错时，进行出错提示，图 4－2 为“输送/出错提示”方式框图。

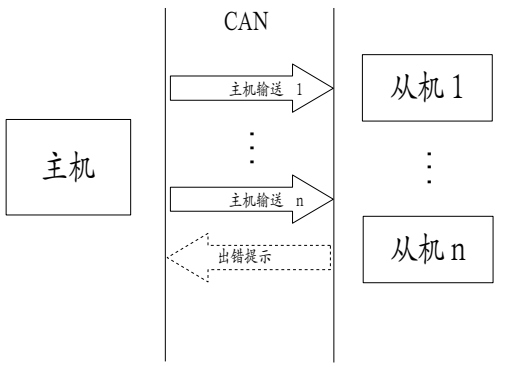

图 4－2 输送/出错提示方式

4.3.3 主机事务处理流程

图 4－3 为参考主机事务处理流程框图。

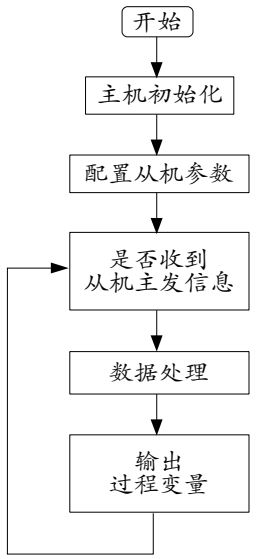

图 4－3 参考主机事务处理流程框图

主机事务流程图较上一章去除了我们认为低效冗余的轮询矩阵。

图 4－4 给出了一个参考主机信息获取和处理指令程序流程（运行态），

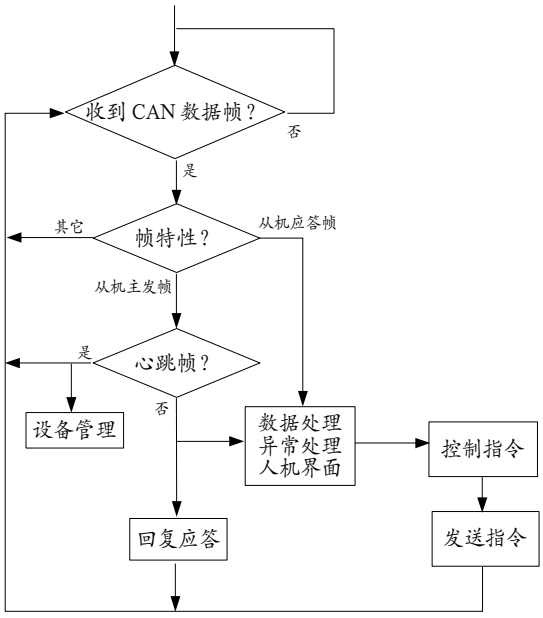

图 4－4 主机信息获取和处理指令程序流程

从流程中可以看出，程序的中心已经完全转换到接收从机信息上面来了，这是一种质的飞跃，使系统更加贴近 CAN 总线的设计初衷。

另外，请注意从机主发的指令，有无需确认应答的指令（0x19），也可以有需要确认应答或回复响应的指令（0x06 和 0x07）。

4.3.4 从机事务处理流程

从机事务流程图与上位机下位机模式中的下位机是基本相同的，只是更加强调图左侧的从机主发业务，而不是右侧的指令应答。图 4－5 为从机参考事务处理流程图。

在从机初始化后，转入“运行态”，应当包括从机“初始信息”的上报过程，在等待指令的过程中，应用的“事件触发”随时可以产生，改变程序流程，完成相关信息的发送。

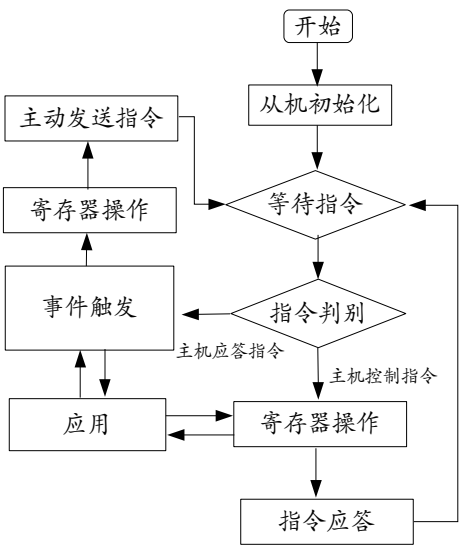

图 4－5 从机事务处理流程图

图 4－6 为从机指令流程图。

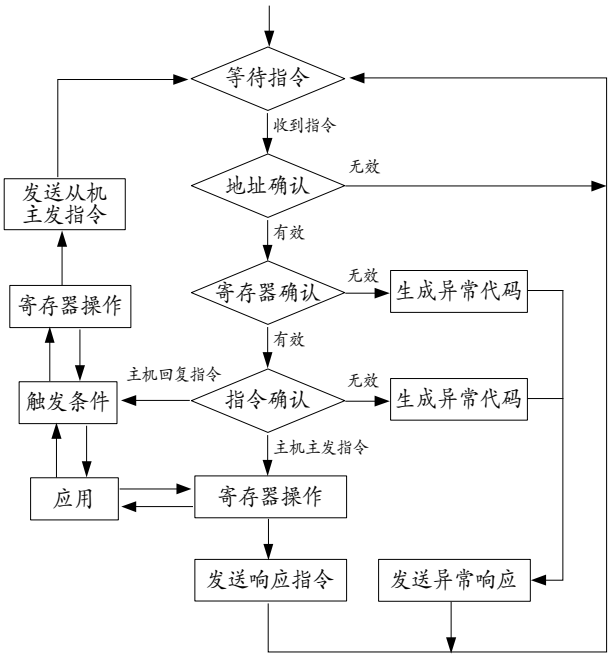

图 4－6 从机程序流程

本章的从机事务处理流程图和指令流程图与上一章看似相同，但其工作重点发生了转移，从机大量的信息是由应用触发主动发送给主机的（0x19），这已经不局限于少量的告警信息（0x06），多个从机主动上报数据在总线上的竞争更加的普遍。

4.4 广播与组广播

4.4.1 广播指令

在本书第二章我们曾提到指令的广播方式，在纯“主从模型”中，广播指令是纯单向的，即从“主机”到“从机”的信息传输，从机不作确认和应答。读者可能已经注意到了，我们在设计指令的时候，凡是“0x0n”的指令，都是要有“0x1n”做应答。在本书的第二章我们学习到的指令还不够丰富，还没有介绍到“0x17”和“0x1C”信息和配置单下传指令，因此广播指令只能拿到本章来讲。

在纯“主从模型”中，主机可以使用“0x17”或“0x1C”作为广播指令，向设备传输信息或配置参数，如：0x01 0x7600 0x00 0x17 0x08 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 本例表

述主机向网络中所有设备的 0x7600～0x7607 写入 8 个数据 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88。设备不作应答，即使某些设备没有定义 0x7600～0x7607 寄存器，也不能发送出错提示信息。而在“全事件触发”模式下，设备是可以发出错误提示信息的，如： 0x1 0x7600 0x35 0x1A 0x2 0x17 0x06。

4.4.2 广播指令在设备上的执行过程

设备在接收到广播指令后，首先将广播指令的广播地址“0x00”置换为本机地址，如：“0x35”，这样就变成了一条主机针对本机的指令，然后再执行该指令。如：广播指令 0x01 0x7600 0x00 0x17 0x08 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 在设备上先用本机地址“0x35”置换广播地址“0x00”，就变为 0x01 0x7600 0x35 0x17 0x08 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88，然后再执行。

4.4.3 高效的广播指令

由于 CAN 总线的“多主”特性，支持总线竞争，因此在非纯“主从模型”下，广播指令得到了极大的扩展，几乎所有的主机针对从机的指令，都可以进行广播传输，而且指令效率非常高，如：读取“设备产品信息”，主机只需要发送一条 0x01 0xFBF0 0x00 0x0E 0x01 0x08 读取配置参数的广播指令，所有设备都会同时使用应答指令 0x01 0xFBF0 0xnn 0x1E 0x08 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 将“设备产品信息”发送给主机，这时应答指令在总线上产生了严重的“竞争”，并依优先级次序（设备地址低的优先级高）传递到主机。此刻总线效率达到了最高值，比主机依序轮询读取“设备产品信息”效率提高了一倍以上。

总线上的设备是多种多样的，有些相同有些不同，寄存器的分布有一致的，也有不一致的，因此统一使用广播指令去读取和配置这些设备并不现实，因此我们引入了组广播的概念。

4.4.4 组广播

TTCANopen 目前并没有强制规定哪些地址用于“组广播”，这是由用户在系统设计时自行定义的，严格的来讲除了广播地址“0x00”之外，其它任何地址都可以作为“组广播”地址使用，这里我们还是推荐用户使用地址域的高端做为组地址，如：0x60～0x6F。一旦某个地址被系统定义为组地址，网络上任何设备都不能再使用该地址作为设备的本机地址。

在 TC0016 子协议（0xF000 段的分配使用）中，我们为设备定义了 16 个寄存器空间，即：

0xF020～ 0xF02F 作为设备存放组地址的空间。设备从属于哪个组，就将组地址依序存放在这 16 个寄存器空间中，也就是说一个设备最多可以同时从属于 16 个组。

设备接收到一条指令后，首先查看指令地址，其是否为广播地址“0x00”，然后继续查看其是否为本设备从属的组地址，如果是则将“广播地址”或“组地址”置换为“本机地址”，然后再执行该指令。

有了“组地址”，我们就能将网络上的设备进行分组，把同类的设备分到一组，有针对性的使用“组广播”指令对其进行配置。分组时，组的总数是可以大于 16 个组的，一个设备可以同时属于不同的组（最多不超过 16 个），这给“组”使用带来了极大的便利。

其实，我们不仅仅在设备配置时可以使用“组广播”，对于“过程变量”也可以使用“组广播”。如：我们使用四个数控千斤顶，顶起一个平台进行调平，我们可以将这四个千斤顶分为一组，另外再将其“两两”分组，可分为 6 组，“三三”分组，又可分为 4 组，一共有 11 个组，这样当我们需要同时驱动多个千斤顶时，就可以使用这些“分组”进行“组广播”调整平台。

4.5 讨论与提示

5.5.1 配置参数或过程变量值域超限问题

在一个系统中，一定会有某些“配置参数”或“过程变量”存在“值域问题”，我们在配置或设置的时候可能会出现“值域越界”的错误。

对于一些简单低成本的系统而言，可能只在系统或设备说明书中阐述了相关“参数”或“变量”的“值域范围”，而没有提供一定的“程序保护措施”，这就要求系统或设备的使用者在配置或设置相关参数或变量时，必须严格按照“说明书”中的要求去做，不能有半点闪失。

对于较高级的系统，一定会要求系统或设备对“值域越界”提供一定的防范措施和必要的“错误提示”，用以更正错误。

通常网络上设备的“配置参数”是由“设备管理器”进行配置的，这要求“设备管理器”的数据库系统非常了解网络上各设备的“配置参数”及其“值域范围”，因此“设备管理器”是有能力辨识超限的配置参数的，并加以提示，也有能力防范通过其人机界面输入的“越限参数”。

对于部分 O 特性过程变量也是有值域范围的，如：D/A 输出等，在主从系统中，通常系

统主机对其控制的设备“过程变量”及其“值域范围”是知晓的，并可防范越界数值的出现。

因此有“专家”建议，在设备端不进行“值域越界检测”，这样可以减少应用层协议中异常错误处理的复杂度和降低设备端的编程成本。的确，许多现场总线应用层协议也是这样做的。在 TTCANopen 协议中，我们将“寄存器”当做一种“数据容器”看待，其通信协议并不关心“容器”中的内容，而只是将其从通信的一端传递到另一端，只有对应设备“应用程序”才会关心容器中的数据内容，以及其是否“超限”，而当相关数据传递到对应设备应用程序中进行“值域甄别”的时候，其与当时的通信过程已经有所脱节，且没有一个有效的措施将“值域超限”的异常信息提示到系统当中去。

而有些“专家”则认为非常有必要在设备端提供“值域甄别”和相关的“错误处理”，这样可以有效的提高整个系统的可靠性，这在理论上无可厚非，但实现上确实有许多需要商榷的内容。首先，在设备端，相关的需要进行“值域甄别”的数据已经通过应用层协议网络指令传递过来了，通讯协议在将“数据”写入到相关寄存器之前，就需要对其“值域”进行甄别，而不能将寄存器当做简单的传递“容器”看待，它需要了解容器中的数值大小，是否在规定的“值域”范围内，如果“超限”则不接受该数据并给出“错误提示”指令，以待该数据被更正重新发送过来。这样通讯协议就不简单的是一个“数据”搬运工，它部分取代了设备应用程序对数据的值域甄别工作，增加了通讯协议的负担和指令系统的复杂性；其二，在应用层提供数据超限错误提示指令，可能是比较复杂的，因为“值域超限”可能存在多种情况，如：“上超”、“下超”，更可怕的是有些数据的值域范围是离散的或部分离散的。另外，当发生值域超限，设备是否需要将正确的值域范围随同“错误提示”指令一同发送出来？这使“错误提示”指令系统更加复杂；第三，可能是更为棘手的问题，我们知道，一条 CAN 指令可以携带 8 个字节的数据，因此它可能一次传递多个参数和变量，而其中，有部分参数或变量发生了“超限”，那么设备将如何处理该指令？

为了应对设备端“值域越界检测”的问题，TTCANopen 提供了一套相对较弱的解决方案，该方案不是强制性的，用户可以选择使用。

首先，TTCANopen 不支持离散和部分离散型值域的检测；

其次，TTCANopen 只针对具有“上、下限”的模拟量进行值域检测；

第三，对于使用“0x08”或“0xD”指令携带的变量或参数，出现“越限”，将“越限”部分内容用该变量或参数的“上限”或“下限”值进行取代并写入对应寄存器中，不提供专门的“错误提示”指令，通常主机或设备管理器，可以通过其对应应答指令“0x18”或“0x1D”所返回的数据对其进行判断；

第四，对于使用“0x17”或“0x1C”指令写入设备的数据，对越限部分，也使用其“上，下限”值进行取代，因为该指令没有应答指令，设备需增加一条“0x19”或“0x1E”指令，将刚写入寄存器的内容播发出来，以做提示。

第五，为了应对用户可能的值域越限“错误提示”的需求，我们在错误提示列表中给出了一个错误提示标识“0xFF”，用户可以根据自身需求选择使用。

4.5.2 事件触发与浪涌现象

支持事件触发是 CAN 总线的特点之一，常见的事件触发包括：定时触发、周期触发、状态变化触发、越界触发等等，过程变量中不同的变量适用不同的触发方式，对于数字开关量一般适用状态变化触发；模拟量适用周期触发或变化增量触发，模拟告警信息适用越界触发。

事件触发方式比主从轮询矩阵采集方式总线利用率要高很多，他不仅仅是省去了“读取指令”，同时可大大减少或避免轮询矩阵采集中常出现的“过采”和“漏采”现象，反应变量变化更加及时准确。但事物总是两方面的，由于事件触发的相互独立和不可以预测性，很可能在某个瞬间时段有大量数据都要通过总线传输，造成总线堵塞，低优先级指令传输延迟加大，即所谓“浪涌”现象。

4.5.3 总线容量

用户在使用和设计 CAN 总线系统时，一定要牢记“CAN 总线是为少量数据传输而设计的”，过于频繁的数据不适用在 CAN 总线上传输，它轻易的就可将总线饱和，在设计系统时用户一定要对通讯流量有一定的预估，并保留一定的通讯余量，一方面用以降低“浪涌”出现的概率，另一方面为出错重传预留空间，确保系统稳定可靠。在一般工业环境下，总线利用率达 60％认为系统是相对安全的，超过 70％风险将快速增加，势必考验系统对风险的耐受度。

4.5.4 优先级

在全触发方式下，指令在总线上的竞争达到了极致，因此合理安排传递数据的优先级就显得格外重要，除了优先级位外，寄存器地址的高低也对指令的竞争能力起了非常重要的作用，因此合理安排数据在寄存器寻址空间的位置是系统设计者的一项重要工作。

4.5.5 0x19 指令和 0x09 指令的应答指令

可能有读者要问，TC0035 通讯子协议中的 0x19 指令与 TC0030 通讯子协议中 0x09 的应答指令，功能码发生了重叠，它们有什么不同?

其实，我们可以这样去理解，0x19 是被事件触发的信息上传指令，这里的“事件触发”可以是设备本地事件，也可以是来自主机的“变量读取”事件（0x09）。

同理，我们也可以类推到 0x17 指令。在引进“多主”和“总线竞争”概念之后，对“过程变量”的操作，0x17 和 0x19 指令将是系统出现率最高的指令。只有在特别强调“可靠性”的场合我们才会启用“命令/应答”方式的指令。

4.5.6 再叙客户－服务

在本书第二章的讨论中，我们曾提到过客户－服务过程，那是在纯“主从模型”下的，当我们拥有了 0x17 和 0x19 指令后，这一过程就变得非常便捷和高效了。首先主机（客户端）用 0x17 向从机（服务端）发出服务请求，从机在完成服务后，清除服务请求标志，并使用 0x19 指令将服务结果数据传送给主机。

同样，客户－服务过程也可以反过来，由从机做为客户端，主机做为服务端。

4.5.7 轮询矩阵所含信息

在本章开始部分，有这样的说法：轮询矩阵中周期性的数据读取指令都应该属于此类“冗余信息”。实际上此说法并不够准确，轮询矩阵虽然不携带“新鲜的信息”，但它决定了信息的传递时序。

5.5.8 统筹安排

在全竞争的环境中，设备是相互独立的，设备间周期触发的信息也没有进行“步调”的协调，对均衡总线流量缺少统筹安排和管理，对“浪涌”的出现没有有效的抑制措施。这正是本书后面章节要解决的一个关键问题。