第 3 章“蜕变，化蝶”
上位机下位机模式

3.1 “蜕变化蝶”

本书上一章 2.4.9 中提到“告警信息”的采集是“主从模型”集中监控系统难于处理的顽疾，系统要在“实时性”和“总线利用率”上进行权衡，很难取得“双赢”的结果，但是，如果发生告警时，“从机”能够以较高的优先级主动发送告警信息到主机，情况就完全不同了，由此系统取得了双赢的结果，此时“主机与从机”的关系，已经蜕变为“上位机与下位机”的关系，上位机可以查询下位机的信息，下位机也可主动上报信息，原来“主从模型”中的“命令/应答”时序被打破，总线产生了竞争的可能，由此引入了 CAN 总线的无损竞

争特性，蛹儿化成了蝴蝶。

由于书写和称谓习惯上的原因，这里仍然称上位机为“主机”，下位机为“从机”。

3.2 从机主动发送的指令集

在开始介绍上位机下位机模式之前，先让我们了解一下几个“从机”主动发起的指令。

3.2.1 TC0034A000子协议（从机主动发送的过程变量基本指令）

从机主动发送的过程变量基本指令

本子协议（通讯子协议）定义了2个由“从机”发起的操作“过程变量”的基本指令分别是“数据上报指令”、“数据请求指令”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 段的数据交互。

1 数据上报指令

功能码：0x06

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

数据上报指令格式：

优先级	寄存器基地址	设备地址	0x06	DLC	1～8 个字节数据

例：0x0 0x7100 0x35 0x06 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），“主机”在其原“指令功能码”上加 0x10，并读取原指令信息内容进行回复。

数据上报指令应答格式：

原优先级	原寄存器基地址	原设备地址	0x16	原DLC	1～8个字节数据

例：0x0 0x7100 0x35 0x16 0x4 0x11 0x22 0x33 0x44

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	0x06	1个字节错误编号

例：0x0 0x7100 0x35 0x1A 0x2 0x06 0x06

（主机不能处理该寄存器信息）

0x0 0x7100 0x35 0x1A 0x2 0x06 0x07

（主机不接受上报指令）

2 数据请求指令

功能码：0x07

设备（从机）请求“主机”向其指定基地址填充数据，指令带一个字节的数据，该数据表述将要连续请求数据的寄存器个数 n，n＝1～64，当 n 大于 8 时，“主机”需要使用多条指令响应。（注：请求填充的寄存器空间必须连续有效）

数据请求指令格式：

优先级	寄存器基地址	设备地址	0x07	0x1	请求填充的寄存器个数

例：0x1 0x7600 0x35 0x07 0x1 0x04

本例表述，地址为 0x35 的设备请求“主机”向其基地址为 0x7600 连续 4 个寄存器填充数据。

“主机”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。

数据请求指令应答格式：

原优先级	原寄存器基地址	原设备地址	0x17	DLC	1～8 个字节数据

例：0x1 0x7600 0x35 0x17 0x4 0x11 0x22 0x33 0x44

上例表述“主机”响应，向地址为 0x35 的设备其基地址为 0x7600 的连续 4 个寄存器写入 0x11 0x22 0x33 0x44。

又例：0x1 0x7600 0x35 0x07 0x1 0x0E

响应 1：0x1 0x7600 0x35 0x17 0x8 0x11

0x22 0x33 0x44 0x55 0x66 0x77 0x88

响应 2：0x1 0x7608 0x35 0x17 0x6 0x99 0xAA 0xBB 0xCC 0xDD 0xEE

当主机不具备填充该寄存器信息的能力，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	0x1A	0x2	0x07	1 个字节错误编号

例：0x1 0x7600 0x35 0x1A 0x2 0x07 0x06（主机不能填充该寄存器信息）

0x1 0x7600 0x35 0x1A 0x2 0x07 0x07（主机不接受该指令）

0x1 0x7600 0x35 0x1A 0x2 0x07 0x2（指令参数出错，DLC 错，或 n 不在 1～64 范围内）

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。

3.2.2 TC0021A000 子协议（从机主动发送的配置参数基本指令）

读者很容易想到“配置参数”是不是也会有从机主动上报的情况呢？这需要我们假想一下它的应用场景。如：当一个“自举”设备，自我配置完成后，想把自己的配置状况主动报送给主机或设备管理器；当某设备检出其配置参数失效或不正确，可以主动发送配置参数请求指令，让主机或设备管理器为其重新配置该参数。总之，我们可以想象到一些应用场景，而且我们也确实定义了这些指令的存在，但我们还是强烈建议用户不要轻易使用这些指令。

因为当多个从设备主动参与设备配置活动的情况下，会扰乱主机或设备管理器对设备配置管理的节奏，从而给系统带来风险。

从机主动发送的配置参数基本指令

本子协议（通讯子协议）定义了 2 个由“从机”发起的操作“配置参数”的基本指令分别是“配置参数上报指令”、“配置参数请求指令”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 配置参数上报指令

功能码：0x0B

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

配置参数上报指令格式：

优先级	寄存器基地址	设备地址	0x0B	DLC	1～8个字节配置参数

例：0x1 0x8100 0x35 0x0B 0x4 0x11 0x22 0x33 0x44

该指令由设备触发，“主机”在其原“指令功能码”上加0x10，并读取原指令参数内容进行回复。

配置参数上报指令应答格式：

原优先级	原寄存器基地址	原设备地址	0x1B	原DLC	1～8个字节配置参数

例：0x1 0x8100 0x35 0x1B 0x4 0x11 0x22

0x33 0x44

当主机或设备管理器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	0x0B	1 个字节错误编号

例：0x1 0x8100 0x35 0x1F 0x2 0x0B 0x06（主机不能处理该寄存器信息）

0x1 0x8100 0x35 0x1F 0x2 0x0B 0x07（主机不接受该指令）

2 配置参数请求指令

功能码：0x0C

设备（从机）请求“主机”或设备管理器

向其指定基地址填充配置参数，指令带一个字节的数据，该数据表述将要连续请求配置参数的寄存器个数 n，n＝1～64，当 n 大于 8 时，“主机”需要使用多条指令响应。

配置参数请求指令格式：

优先级	寄存器基地址	设备地址	0x0C	0x1	请求填充的寄存器个数

例：0x1 0x8100 0x35 0x0C 0x1 0x04

本例表述，地址为 0x35 的设备请求“主机”或设备管理器向其基地址为 0x8100 连续 4 个寄存器填充配置参数。

“主机”或设备管理器在其原“指令功能码”上加 0x10，并在数据场添加相关配置参数进行回复。

配置参数请求指令应答格式：

原优先级	原寄存器基地址	原设备地址	0x1C	DLC	1～8 个字节配置参数

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

上例表述“主机”或设备管理器响应，向地址为 0x35 的设备其基地址为 0x8100 的连续 4 个寄存器写入 0x11 0x22 0x33 0x44。

又例：0x1 0x8100 0x35 0x0C 0x1 0x0E

响应 1：0x1 0x8100 0x35 0x1C 0x8 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88

响应 2：0x1 0x8108 0x35 0x1C 0x6 0x99 0xAA 0xBB 0xCC 0xDD 0xEE

当主机不能填充该寄存器信息，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	0x0C	1 个字节错误编号

例：0x1 0x7100 0x35 0x1F 0x2 0x0C 0x06（主机不能填充该寄存器信息）

0x1 0x7100 0x35 0x1F 0x2 0x0C 0x07（主机不接受该指令）

0x1 0x7600 0x35 0x1A 0x2 0x07 0x2（指令参数出错，或 n 不在 1～64 范围内）

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。

3.3 上位机下位机模式

3.3.1 事件触发方式

事件触发方式是从机向主机主动发送信息的一种形式，或称为“信息/确认”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，返回确认指令，图 3－1 为“信息/确认”方式框图。

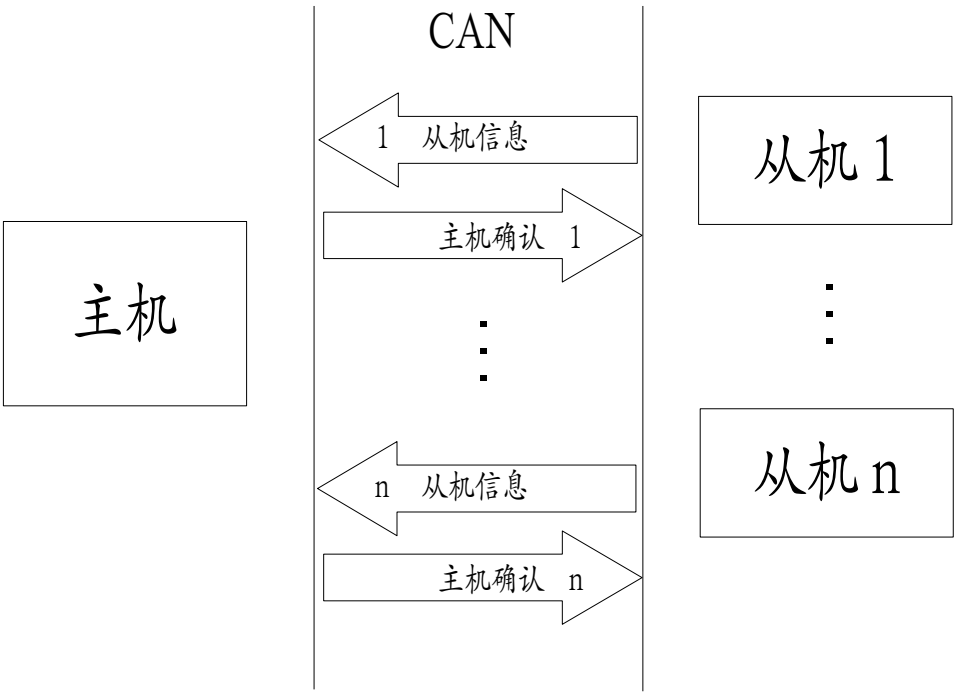

图 3－1 “信息/确认”方式框图

3.3.2 索取数据方式

在应用中会遇到这样的情况，从机是某些

数据的主动索取和处理者，而主机是数据的提供方，主机不能预知从机何时需要何种数据，直到从机发出索取指令后，方能按从机的要求提供数据，称之为索取数据方式或“索取/输送”方式，通讯由从机发起，在发送的信息指令中包含该从机的地址，和索取信息的寄存器地址，主机接收到该索取指令后，按要求填充相应寄存器中的数据发送给该从机，图 3－2 为“索取/输送”方式框图。

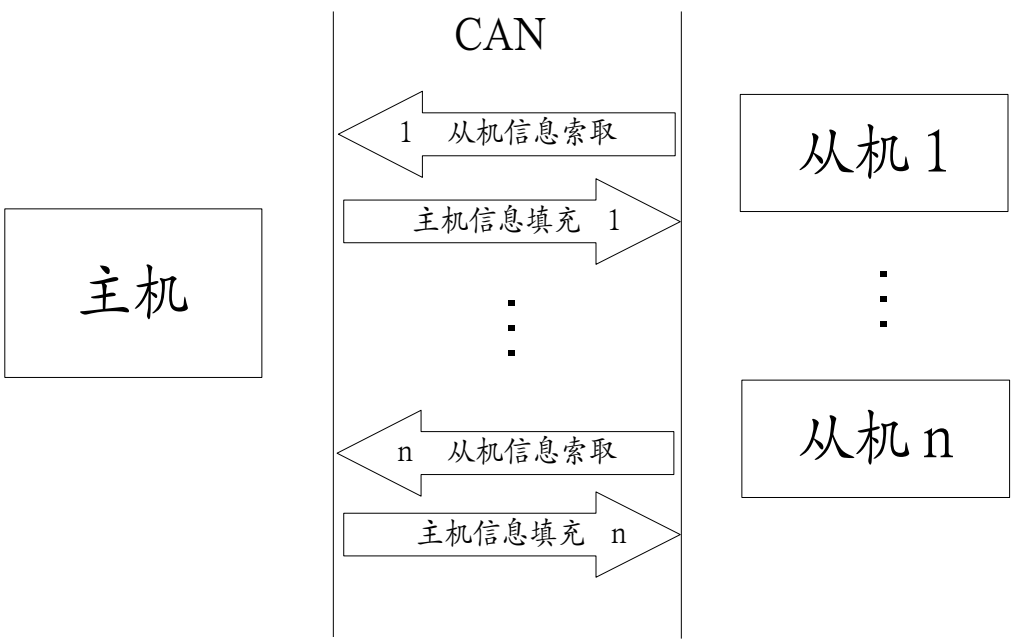

图 3－2“索取/输送”方式框图

3.3.3 主机事务处理流程

主机事务处理流程较“主从模型”增加了从机主动发送指令信息的处理，图 3—3 为参考主机事务处理流程框图。

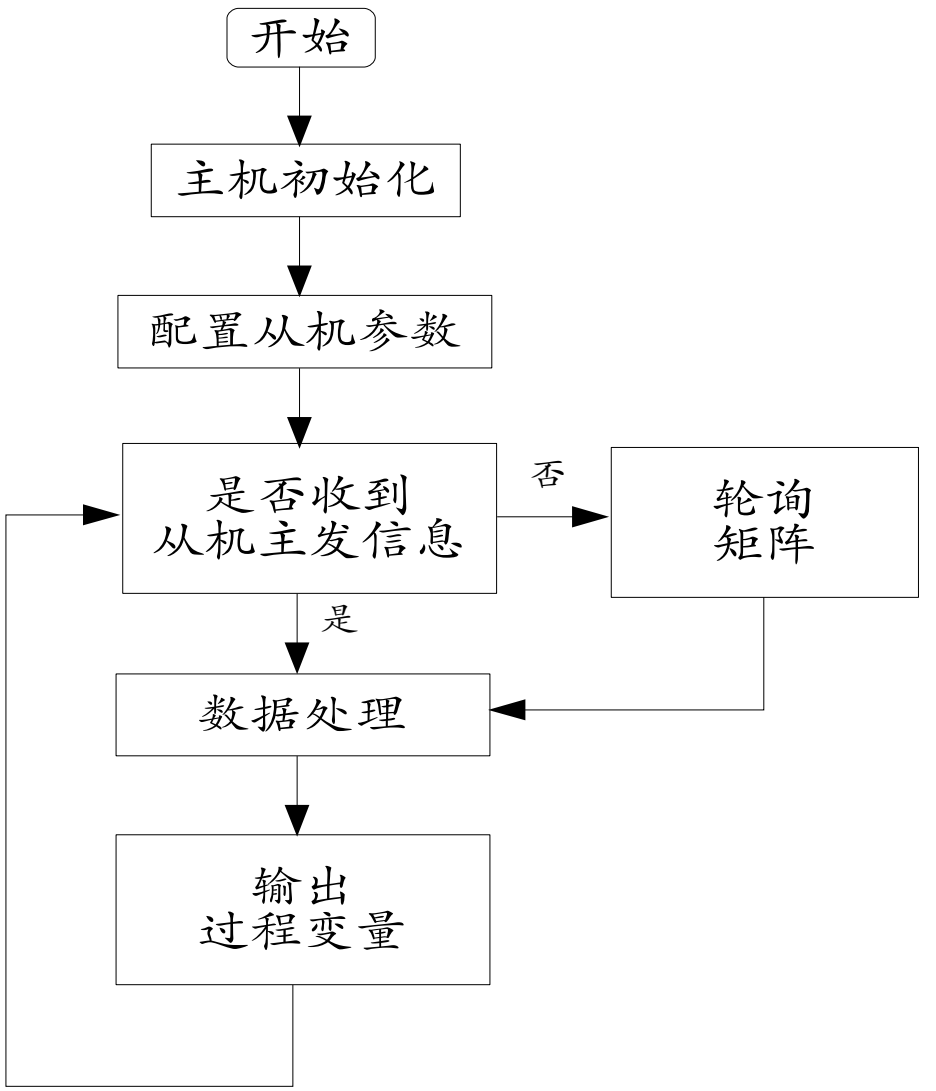

图 3—3 参考主机事务处理流程框图

从图中可以看出，从机主发信息的处理，优先于主机轮询采集数据的处理，这对于从机告警信息的及时处理是非常必要的，然而事务总是两方面的，由此也打破了主机轮询时序的稳定性。

图 3－4 给出了一个参考主机运行时指令程序流程，

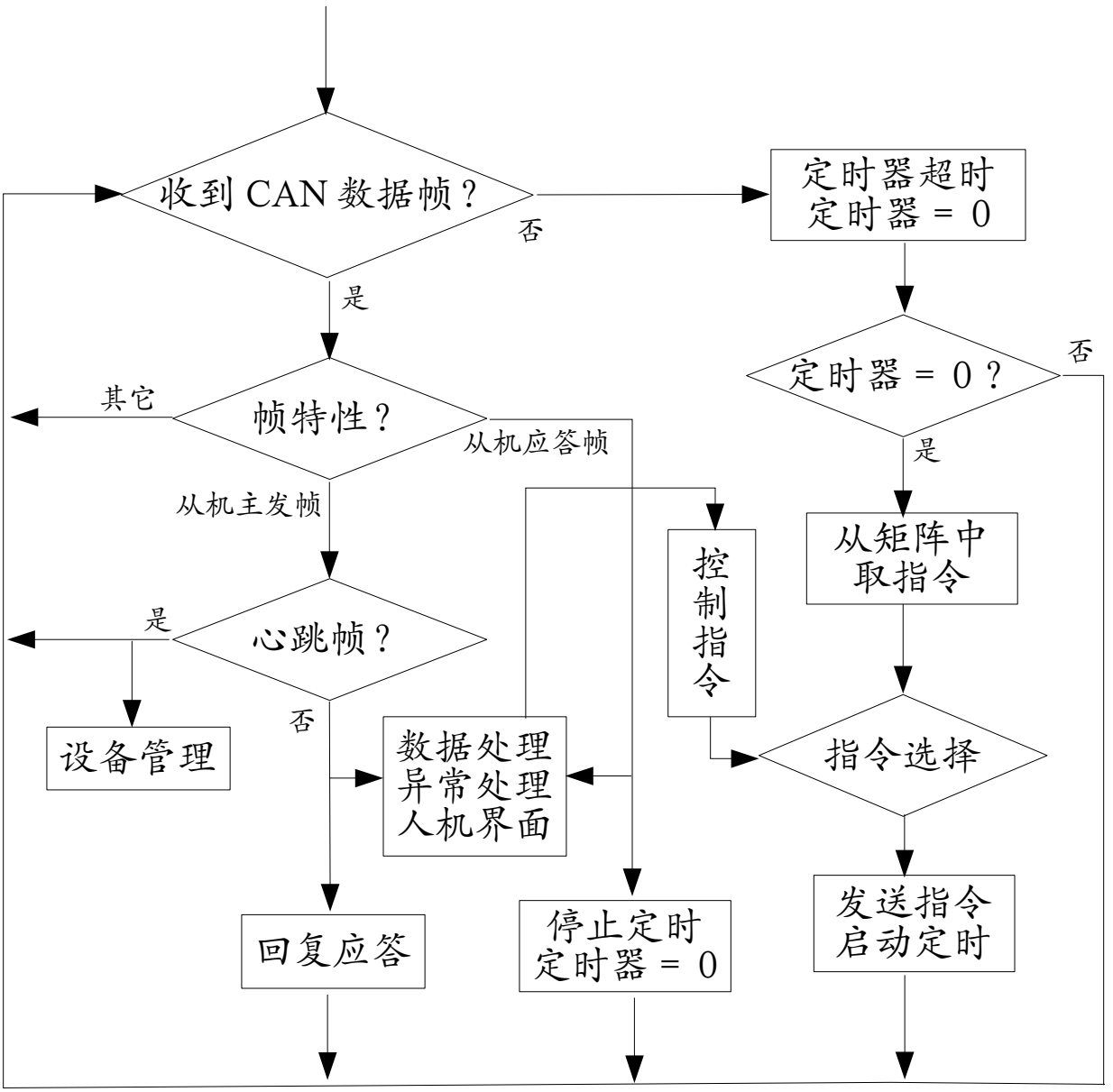

图 3－4 主机运行时指令程序流程

从流程中可以看出，程序的核心已经由“主从模型”的“发令”转换为指令的“接收”上面来，这是一个质的转变；

由于从机可以主动发令，系统的设计者可以为从机编制“心跳指令”，周期的上报设备状态，便于主机对设备进行管理；

① 检测是否收到 CAN 数据帧：由于从机可以主动发送指令，总线上的指令信息流已经不象“主从模型”那样完全由主机掌控，从机的“心跳信息”、“事件触发信息”和“索取信息要求”随时都有可能发送到主机，需要主机即时对其进行处理，因此主机起始的指令程序流程便是检测是否接收到从机发来的指令；

② 如果主机没有接收的从机发送来的“指令”：判断定时器是否归零，如果归零，则通过轮询矩阵获取指令启动发送程序，这与上一章的主从模型相近；

③ 控制指令：主机应用程序如“数据处理”、“异常处理”和“人机界面”等产生的控制指令，通过指令选择器优先于轮询指令发送；

④ 主机接收到 CAN 数据帧：对帧特性进行判别，是否“从机主发帧”、“从机应答帧”或其他，其他则返回继续接收或进入轮询发令；

⑤ 从机应答帧：如若主机收到的是从机应答帧，则其处理过程同“主从模型”；

⑥ 从机主发帧：当主机接收到从机主发帧，在当前协议模式下，主要有三种帧，即“从机心跳”、“从机事件触发”和“从机索取数据”；

⑦ “从机心跳”：是具备主发功能的从机设备，按照规定周期定时向主机发送其工作状态参数的 CAN 指令，需要指出的是在纯“主从模型”的从机设备上，这些参数是通过主机发送轮询指令获取的，显然“心跳”机制

在总线效率上大大优于“轮询”方式；设备工作状态参数和异常标识主要由主机的“设备管理”和“异常处理”程序进行处理；

⑧ “从机事件触发”：从机由紧急事件触发的 CAN 指令，在当前协议模式下，主要是“告警信息”，使主机能够及时的对其进行处理，避免了在轮询模式下对该信息的高频度的轮询访问，提高了总线效率，实现了本章开始所说的“双赢”；

⑨ “从机索取数据”：从机向主机索取数据，主机收到该信息后，应当及时的向从机相关的寄存器填充数据供从机使用，本指令大大拓展了 TTCANopen 协议的应用范畴。

3.3.4 从机事务处理流程

从机的事务处理流程比主从模型多了一个分支，即由应用触发的从机主动发送指令的过程下面给出一个简易从机参考事务处理流程图，

见图 3－5。

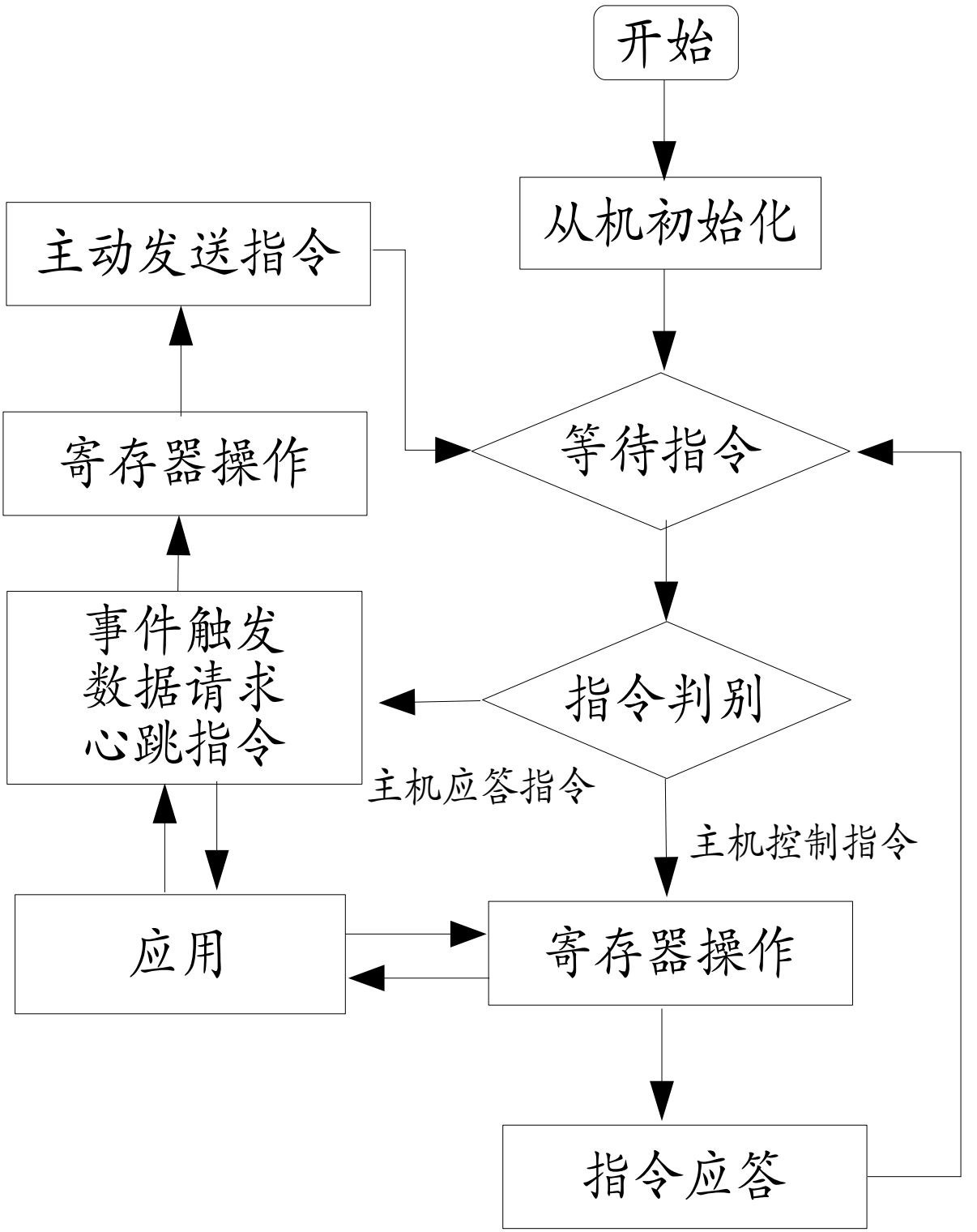

图 3－5　简易从机参考事务处理流程图

图中，右边是典型的“主从模型”流程，左边表述的是一个由“应用”触发的从机主发指令的过程。

① 从机在完成本机初始化后，就进入到等待接收指令状态，从机能够接收到两种指令一种是来自主机主发的指令，设备的运行过程同“主从模型”是一样的；另一种是来自主机的应答指令，该指令是对该从机主发指令的应答闭环。

② 设备主发指令，是由设备事件触发的由设备主动发送给主机的指令，在上位机下位机模型下，主要包括：设备心跳、报警信息等这些指令发出后是需要主机响应回复的。

图 3－6 给出了一个参考从机指令程序流程，

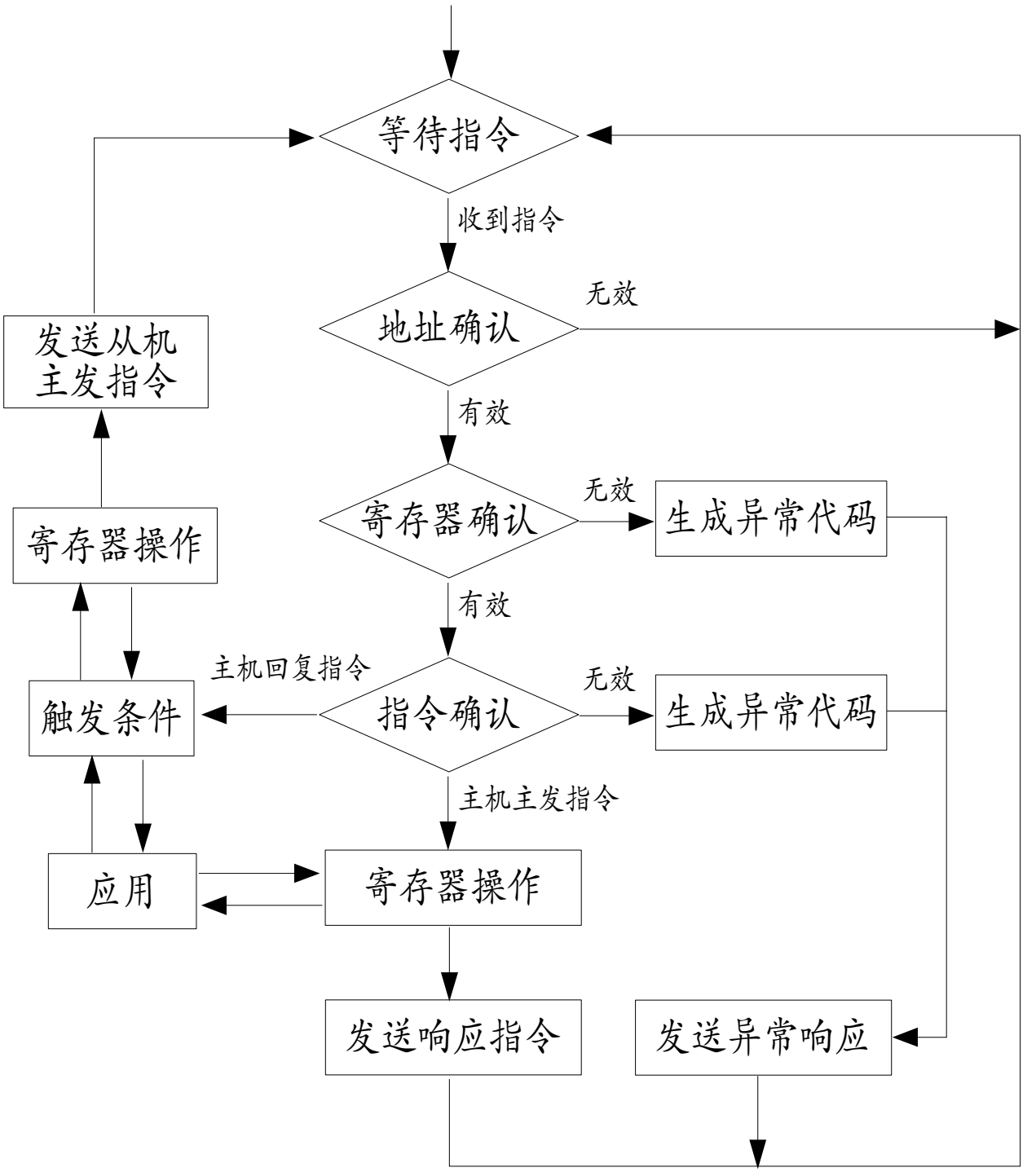
等待指令
收到指令
地址确认
无效
发送从机
主发指令
有效
寄存器确认
无效
生成异常代码
寄存器操作
有效
主机回复指令
触发条件
指令确认
无效
生成异常代码
主机主发指令
应用
寄存器操作
发送响应指令
发送异常响应

图 3－6　从机程序流程

3.4 讨论与提示

3.4.1 关于本书的流程图

需要提醒读者的是本书提供的流程图都是示意性的，旨在说明功能原理，用户实际设计时，应当根据具体情况灵活应用。

3.4.2 等待定时

由于引进了 CAN 总线的竞争机制，打破了原来“主从模式”中严格的收发时序，等待定时的合理设置是非常必要的。

主机发令后，一定要为从机应答保留一定的响应时间，再发下一条指令，如果收到“非预期”指令，还要继续为“预期”应答延长一定的等待时间。

对于从机事件触发指令，要设定它的最高触发频度，为低优先级的指令通讯提供机会。要时刻记住，**CAN 总线是为少量现场数据通讯而设计的**，对于大容量的数据传输应当使用

其他通讯方式，如点对点的通讯等。

3.4.3 优先级

由于在上位机下位机模式中，引进了从机主动发令的机制，“主从模型”中的指令时序被打破，总线产生了竞争，因此指令中 ID 字的优先级机制将会发挥作用，ID 值越小优先级就越高，我们将 ID 字划分了 4 个段，其中，第一段“1 位优先级”子段对指令的优先级影响最大，对于普通指令该位“置 1”，对于告警信息等少量紧急指令才会启用最高“0”优先级；当第一段优先级相同时，第二段子段“16 位寄存器地址”将起作用，地址越低优先级越高，因此我们在为过程变量分配地址时应本着优先级分配原则，将需要高优先级的过程变量分配在较低的寄存器地址中；当寄存器地址相同时，第三段子段“7 位设备地址”将发挥作用，地址越小的优先级越高；当地址也

相同时“5 位指令功能码”将发挥作用，编码越小的优先级越高。在 CAN 总线传输过程中是禁止两个 ID 字相同的指令同时竞争传输的。

3.4.4 “心跳指令”

心跳机制是从机以一定的周期主动向主机发送其设备状态参数的一种行为，主机通过收集从机的心跳信息，了解从机的工作状态和管理整个网络系统运行，如果主机多次在规定的周期内没有收到某从机的“心跳信息”，主机可以判定该从机出现故障或退出运行。

从机发送心跳指令可以有两种指令方式，一，使用本章新引进的“配置参数上报指令（0x0B）”，这是许多设计者很自然的想法，但笔者不以为然，“配置参数上报指令”的设计是需要主机进行确认应答的，由于“心跳信息”是周期性的，主机对其有一定的“预期”并且主机不需要对从机的“心跳指令”进行应

答，笔者更加赞同使用第二种指令方式；二，使用“配置参数读取指令的应答部分（0x1E）”作为“心跳指令”，该指令在形式上本身就是一个从机对主机的应答，其更加适合作“心跳指令”使用，用户可以这样认为“心跳信息”是主机使用“配置参数读取指令（0x0E）”周期读取的，各从机使用“配置参数读取指令的应答格式（0x1E）”进行应答，由于主机对从机“心跳信息”的周期读取是可“预期”的行为，它不携带任何有用的“信息”，因此可以不用发送，只余下从机应答指令（0x1E）携带“心跳信息”周期的回送到主机，这一原理在本书的下一章将有更加广泛的应用。

心跳指令指向的寄存器地址空间被规定为0xFA10~0xFA17，其中，

0xFA10～0xFA13－－设备心跳时刻；

0xFA14－－－－－－发送错误计数；

0xFA15－－－－－－接收错误计数；

0xFA16－－－－－－设备失步计数；

（目前没有用到）

0xFA17－－－－－－－设备工作状态；

通常我们会使用普通优先级发送设备（从机）心跳。

3.4.5 设备错误告警

在“主从模型”中，设备错误告警是靠主机不断的轮询设备错误标识寄存器（0xFA18～0xFA1F）获取的。在“上位机下位机模型”中，当错误标识寄存器发生变化时，我们可以用零优先级的“0x0B”指令及时将“出错告警”发送到主机。我们同样也可以使用零优先级的“0x1E”指令将其发送至主机。

3.4.6 配置设备状态

在“主从模型”中，设备（从机）发令，完全受主机控制，主机“不问”，从机“不答”。而在“上位机下位机模型”中，情况发

生了变化，设备（从机）可以主动发令了，而我们有时不希望它这么做，尤其是在主机正在配置整个网络期间，因此我们需要为设备（从机）设置设备状态，即：“工作状态”、“配置状态”和“停止状态”，可能还有一个非稳态的“复位状态”。使设备（从机）在不同的状态，有着不同的“行为规范”。

3.4.7 “漏采”与“超采”

在上位机下位机模式中，上位机对下位机的信息采集，仍然使用的是轮询矩阵的方法，虽然我们在编制轮询矩阵时，为变化较快的变量提供了较高的采集频率，为变化较慢的变量提供较低的采集频率，但我们仍然不能避免“漏采”和“超采”现象的发生。所谓“漏采”是当变量发生了一次甚至多次变化而未能被主机采集到。所谓“超采”是变量未发生任何变化而被多次采集。上述两种现象是轮询矩

阵自身无法克服的，而解决之道则是“设备事件触发”将变化的变量及时的通知到主机，而本章讲述的“0x06”指令并不是最好的选择，接下来下一章我们将介绍全触发方式。