[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
H04L 12/28

[12] 发明专利说明书

[21] ZL 专利号 97180666.7

[43] 授权公告日 2003 年 7 月 23 日

[11] 授权公告号 CN 1115822C

[22] 申请日 1997.10.16 [21] 申请号 97180666.7

[30] 优先权

[32] 1996.10.16 [33] US [31] 60/028,651

[32] 1996.11.12 [33] US [31] 60/031,249

[86] 国际申请 PCT/US97/18750 1997.10.16

[87] 国际公布 WO98/17033 英 1998.4.23

[85] 进入国家阶段日期 1999.6.16

[71] 专利权人 汤姆森消费电子有限公司

地址 美国印第安纳州

[72] 发明人 K·F·霍兰德

审查员 焦景梅

[74] 专利代理机构 中国专利代理(香港)有限公司

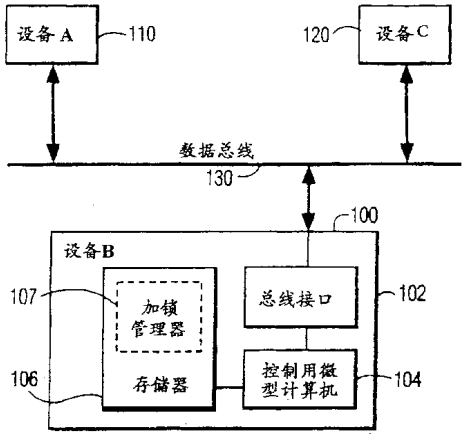
代理人 程天正 陈景峻

权利要求书 2 页 说明书 23 页 附图 5 页

[54] 发明名称 利用持续时间数据控制电子设备之间的通信的方法和装置

[57] 摘要

一种带有控制设备操作模式的装置的设备，它处理接收自第二和第三电子设备的控制和查询数据，以便对这些设备间的通信关系进行管理。本发明涉及到一种系统，它用于通过诸如数字数据总线之类的互连装置在诸如消费者电子设备或类似设备之类的多个电子设备之间进行通信。



ISSN1008-4274

1. 一种设备（B），该设备包括：

（a）用来和至少一个由网络总线（130）互连起来的第二设备（A）进行通信从而在上述第一设备与第二设备之间建立起关系的装置（102），所述通信装置能接收来自第三电子设备的查询数据；

（b）第一装置（106），它用于存储与上述关系的持续时间有关的信息；

（c）第二装置（106），它用于存储与用来终止上述关系的条件有关的信息；

（d）用于响应来自上述第二设备的控制数据而对所述设备的操作模式进行控制的装置（104），所述控制装置能响应所述查询数据来处理上述持续时间的信息并且能响应所述终止条件和所处理的持续时间的信息这两者之一而终止上述关系。

2. 如权利要求 1 的设备，其特征在于，所述用于控制的装置产生经过处理的持续时间的信息，该信息包括第一种状态和第二种状态中的一种，在上述第一种状态中，响应所述查询数据而终止所说的关系，在上述第二种状态中，不论所说的查询数据如何都仍维持上述关系。

3. 如权利要求 2 的设备，其特征在于，所述用于控制的装置响应所述查询数据，通过在该设备与第三设备之间形成一种新关系而终止所说的关系。

4. 如权利要求 3 的设备，其特征在于，所述控制装置能响应对所述查询数据的接收而询问上述第二设备以获得对终止所述关系的许可。

5. 如权利要求 4 的设备，其特征在于，所述控制装置还能监视前述第二设备的状态。

6. 如权利要求 5 的设备，其特征在于，所述控制装置还能使上述第二设备改变包含在所述设备中的信息。

7. 如权利要求 1 的设备，其特征在于，所述预定条件涉及到前述设备与第二设备之间的关系以及所述设备的内部条件这两者中的一个。

8. 一种用于对连在网络总线上的用户电子设备进行控制的方法，该方法包括：

（a）和至少一个由网络总线互连起来的第二设备进行通信，从而在上述用户电子设备与第二设备之间建立起关系；

（b）对所述用户电子设备的操作模式进行控制，至少有一种操作模式可由上述网络总线上的第二设备来控制，从而在上述用户电子设备与第二设备之间建立起关系；

（c）存储与上述关系的持续时间有关的信息并存储与用来终止上述关系的条件有关的预定条件；

（d）接收来自第三设备的查询数据；

（e）处理与上述关系有效性有关的信息以及查询数据；

（f）响应上述终止条件和所处理的信息中的一种而终止所述关系。

9. 如权利要求 8 的方法，其特征在于，所述终止步骤还包括在上述用户电子设备与第三设备之间形成一种新关系。

10. 如权利要求 8 的方法，其特征在于，所述的终止上述关系的步骤是响应查询上述第二设备以获得终止上述关系的许可这样一个步骤而实施的。

11. 如权利要求 10 的方法，其特征在于，该方法还包括这样的步骤即监视上述第二设备的状态。

12. 如权利要求 11 的方法，其特征在于，该方法还包括能使上述第二设备改变包含在所述用户电子设备中的信息的步骤。

利用持续时间数据控制电子设备之间的通信的方法和装置

发明的领域

本发明涉及用于通过诸如数字数据总线之类的互连装置在诸如用户电子设备或类似设备的多个电子设备之间进行通信的系统。具体地说，本发明涉及到管理这种设备的互操作性的装置。

本发明的背景

数据总线可用于把诸如电视接收机、显示设备、盒式录像机（VCR）、直接广播卫星（DBS）接收机以及家用控制设备（例如保安系统或温度控制设备）之类的电子设备相互连接起来。使用数据总线的通信遵循总线协议。总线协议的实例包括用户电子总线（Consumer Electronics Bus）或 CEBus 以及 IEEE 1394 高性能串行总线（IEEE 1394 High Performance Serial Bus）。

总线协议一般同时用于传递控制信息和数据。例如，在“控制通道”上传递 CEBus 控制信息，所述“控制通道”具有电子工业协会(EIA)规范 IS—60 中定义的协议。可用周知为 CAL（通用应用语言）的程序语言的形式来规定用于特定应用的控制信息。

用户电子设备正变得愈加复杂并且有更多的性能。尽管用数据总线将这些复杂的设备连接起来对提供一个完整的音频／视频（A／V）系统来说是必需的，但这样做会产生多种问题。例如，一个设备的某些性能可能需要与连接于总线的一个或多个设备相互作用。为在另一个设备中完成特定的操作，可能需要某一设备的一种功能。多个设备的需求之间就可能会出现矛盾。欧洲专利申请书 0604166 说明了这样一种通信系统，它能在没有间断的情况下允许传递比传输信号的数据域中所提供的更多的数据。

目前，当一个以上的设备竞争使用共用资源例如一连在网络总线上的第三设备时，第一个设备会及时地对该设备进行控制。另一设备不能使用上述共用资源直至第一设备释放了对该资源的占有。这在简单网络中例如包括电视机、分线盒和 VCR 的网络中不会有问题。在这种简单的网络中，VCR“受控”于分线盒（cable box），也就是说，VCR 只记录分线盒所提供的内容。但是，在更复杂的网络中，两个设备

例如分线盒和数字广播接收机会争夺对共享资源即 VCR 的控制权。

发明概要

在多种控制应用中，应排斥对一个设备的总体使用。这种排斥性通常有益于单个的设备。在这种情况下，会在两个设备间短时间或长时间地形成称为“加锁”的特定关系。这种加锁关系允许一个设备控制对第二个设备（即“被锁住的设备”）的接入。

一般地说，有两种加锁方法。在第一种加锁方法中，被锁住的设备判断它是否可被解锁和／或相对于另一设备而被锁住。锁住该设备的请求可直接传给被锁住的设备。如果被锁住的设备不能满足这一新的请求，则被锁住的设备会询问对其加锁的设备（即加锁设备）以确认是否可终止当前的锁定并实现一新的锁定。如果加锁设备没有响应，例如在故障的情况下，则被锁住的设备会终止所说的锁定并形成新的锁定。在第二种方法中，要锁住另一设备的设备在网络上广播这样的请求即：请求锁住了该所需设备的任何设备都终止自己的锁定。

一般地说，依照本发明，第一电子设备包括用于和数据总线相连的装置以及控制装置，该控制装置用于对第一电子设备与连在所述数据总线上的第二和第三电子设备之间的通过该数据总线的通信进行控制。所述控制装置处理自第二和第三电子设备收到的控制和查询数据，以便对这些设备之间的通信关系进行管理。

依照本发明的一个方面，提供了这样一种设备，它具有用来和至少一个由网络总线互连起来的第二设备进行通信从而建立起上述第一设备与第二设备之间关系的装置，所述通信装置能接收来自第三设备的查询数据。此外，上述设备还具有：一第一装置，它用于存储与上述关系的持续时间有关的信息；一第二装置，它用于存储与用来终止上述关系的条件有关的信息；以及，用于响应来自上述第二设备的控制数据而对所述设备的操作模式进行控制的装置。所述控制装置能处理上述关系信息和查询数据并且能响应所述终止条件和查询数据之一而终止上述关系。

依照本发明的另一个方面，所述用于控制的装置产生经过处理的持续时间的信息，该信息包括第一种状态和第二种状态中的一种，在上述第一种状态中，响应所述查询数据而终止所说的关系，在上述第二种状态中，不论所说的查询数据如何都仍维持上述关系。

依照本发明的再一个方面，所述用于控制的装置响应所述查询数据，通过在该设备与第三设备之间形成一种新关系而终止所说的关系。

依照本发明的还一个方面，所述控制装置能响应对所述查询数据的接收而查询上述第二设备以获得对终止所述关系的许可。

依照本发明的又一个方面，提供了这样一种系统，它带有通过网络总线而彼此互连起来的多个设备。每个设备均如前所述那样具有：用于通信的装置；用于存储信息的第一和第二装置；以及，用于对所述设备的操作模式进行控制的装置。

依照本发明的连接在网络总线上的用户电子设备的方法方面，该方法包括：和至少一个由网络总线互连起来的第二设备进行通信；对所述用户电子设备的操作模式进行控制；存储与上述关系持续时间有关的信息并存储与用来终止上述关系的条件有关的预定条件；接收来自第三设备的查询数据；处理与上述关系有效性有关的信息以及查询数据，以及，响应上述终止条件和查询数据之一而终止所述关系。

对附图的简要说明

参照附图可以更好地理解本发明，附图中：

图 1A 以框图的形式示出了用由数据总线连到一起的电子设备所构成的系统；

图 1B 以框图的形式示出了图 1A 所示系统的一部分的进一步的细节；

图 2 是表示加锁机制系统的总体操作的流程图；

图 3 是表示第一事务处理类型的流程图；

图 4 是表示第二事务处理类型的流程图；

图 5 是表示第三事务处理类型的流程图；

图 6 是表示第四事务处理类型的流程图；

图 7 是表示第五事务处理类型的流程图。

在附图中，不同图中相同的标号表示相同或相似的装置。

对附图的详细说明

图 1A 示出了包括分别用标号 110、100 和 120 表示的三个电子设备 A、B 和 C 的系统，这三个设备通过数据总线 130 连在一起。设备 A、B 和 C 可以是各种类型的电子设备，包括诸如音频和／或视频信号处理

设备及家用控制设备之类的用户电子设备。用户电子设备的特定实例包括电视（TV）接收机、显示设备、分线盒、磁带录像机（VTR 或 VCR）直接广播卫星接收机（如 DSS 卫星接收机）、光盘（CD）设备、数字视频光盘（DVD）设备。家用控制设备包括保安设备（例如摄像机、锁）、遥控开关设备以及环境控制设备（例如加热器、制冷器）。包含本发明原理的系统可包括多于图 1A 所示的三个设备。而且，每个设备均可以是不同类型的

设备。例如，本发明的系统可以包括家中的多个设备，包括ＴＶ、ＶＣＲ、分线盒、卫星接收机、音频系统以及家用控制设备。

在图１Ａ所示的实例中，数据总线１３０包括多个信号通路，以便在设备之间传递数据和控制信息。总线的结构可以是串行或者是并行的。可按诸如上述 CEBus 或 IEEE1394 协议之类的多种总线协议中的任何一种在总线上传递数据。经由总线１３０传递的数据包括视频和音频数据。例如，接收数字视频信号的卫星接收机可经由总线１３０将数字视频和音频数据传至数字磁带录像机（ＤＴＲ）以便加以录制。

设备１００包括总线接口装置１０２，它用于将设备１００连接于总线１３０。总线接口１０２包括适用于接收来自总线１３０的数据并在总线１３０上传输数据的硬件。例如，接口１０２可包括寄存器或缓冲存储器以便接收和暂时存储来自总线１３０的数据直至设备１００准备处理该数据并对该数据进行存储直至准备在总线１３０上发送。

接口１０２受控于控制用微计算机（μＣ）１０４。术语“微计算机” 表示包括但并不限于诸如微处理器、微计算机、控制器以及用一个或多个集成电路和／或分立元件实现的控制用计算机之类的各种设备。微计算机１０４控制着数据在接口１０２与设备１００内的诸如信号处理器（图１Ａ中未示出）之类的其它装置之间的传递。例如，如果设备１００是ＤＴＲ并且总线接口装置１０２正在接收数字视频数据以便加以录制，则控制用微计算机μＣ１０４就会确保接口１０２将视频和音频数据传给设备１００内的信号处理电路，因此，能按预期进行录制。另外，控制用微计算机μＣ１０４控制着接口１０２，从而，接口１０２会根据适当的总线协议将数据传至和传自总线１０３。

微计算机１０４在存储在存储器１０６内的控制程序和数据的控制下实现预定总线协议。存储器１０６可包括多种存储器，包括ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ以及硬盘驱动器或其它类型的大容量存储器设备。所述控制程序可包括各种例程，这些例程用于对诸如使通信有效或无效、使总线操作与设备１００内的其它操作相协调之类的总线接口功能进行控制并对用于总线通信的数据进行格式化，。

微计算机μＣ１０４所提供的对总线接口１０２进行控制的一个方

面是确定如何解决对设备１００内的资源使用的相互冲突的请求。例如，考虑这样的情况，其中，设备１１０是分线盒，设备１２０是ＤＳＳ卫星接收机，设备１００是视频磁带录像机（ＶＴＲ）。用户可对ＶＴＲ１００进行编程以便在特定的时间录制经由分线盒１１０的特定的节目。一个有冲突的命令会使ＤＳＳ设备１２０试图在ＶＴＲ１００正录制经由分线盒１１０的节目时使用该ＶＴＲ１００。通常的系统通过忽略ＤＳＳ设备１２０试图进行的使用或者通过通知用户不能使用ＶＴＲ１００来解决上述问题。

依照本发明的一个方面，图１Ａ所示的系统实现了一加锁管理器，以便按以下详细加以说明的方式解决资源的冲突。在图１Ａ所示的示例性实施例中，微计算机μＣ１０４在“加锁管理器”例程的控制下进行操作，以确保设备１００能按以下详细说明的方式解决冲突，所述“加锁管理器”包括在总线控制程序内。一“加锁进程”１０７０例程也包括在总线控制程序内；在以下对图２的详细说明中说明了加锁进程１０７０的操作。所述加锁管理器用于建立连接于总线１３０的设备之间的通信关系或加锁条件，并且用于确定在什么样的条件下修改前述关系。加锁管理器的操作涉及到某些参数，这些参数存储在存储器１０６的一部分１０７内，以下将详细说明这些参数。与加锁关系有关的涉及解决冲突的参数存储在图１Ａ的存储器１０６的加锁管理器区域１０７内。三种这样的参数是持续时间、被锁住的地址以及解锁条件。图１Ｂ说明了这三个参数，它们作为三个独立的项目１０７１、１０７２和１０７３存储在存储区１０７内。应该认识到，尽管在图１Ａ中未作显示，设备１１０和１２０各包括这样的性能，它们能提供与本文所述的如设备１００内的设备１０２、１０４、１０６和１０７所提供那样的功能相类似的功能。设备１１０和１２０内的结构可与设备１００中所示的结构相类似，但也可以用其它结构来产生相似的功能。

作为加锁管理器的操作实例，考虑上述的冲突实例。利用所述加锁管理器，用ＶＴＲ１００录制来自分线盒１１０的节目涉及到分线盒１１０将加锁请求传给ＶＴＲ１００的总线控制能力。加锁请求要求ＶＴＲ１００形成一种与分线盒１１０的关系即形成被锁住的条件。微计算机１０４通过总线接口装置１０２接收上述加锁请求并确定Ｖ

ＴＲ１００是否已被另一装置锁住。如果该ＶＴＲ未被锁住，即该ＶＴＲ可用于加锁，则ＶＴＲ１００就向分线盒１１０发送应答，以表明在加锁进程１０７０中已经建立了加锁。对来自分线盒１１０的预定节目的录制就会开始进行，并且，除了在某些条件之外上述加锁关系会防止录制中断，所说的某些条件是由诸如持续时间和解锁条件之类的参数限定的，并且，正如以下详细说明的那样，所说的某些条件是与上述加锁关系有关联的。上述参数的值是由所述加锁设备即发起加锁条件的设备（本例中是分线盒１１０）形成的并被传送至且存储在被锁住的设备（本例中是ＶＴＲ）的存储器１０６的区域１０７的相应部分内。当ＤＳＳ设备１２０需要使用ＶＴＲ１００时，ＤＳＳ设备内的加锁管理器的功能会向ＶＴＲ１００发送一加锁请求。由于在ＶＴＲ１００与分线盒１１０之间有预先存在的加锁关系，故ＶＴＲ１００内的加锁管理器会根据存储在ＶＴＲ１００的现有加锁条件的参数来处理来自ＤＳＳ１２０的加锁请求并按下述方式解决冲突请求。

图２说明了与加锁管理器１０７相配合的加锁进程１０７０的操作。加锁管理器１０７对一“主设备”所发起的关系或锁定进行管理。加锁进程１０７０同时锁住两个设备。当主设备要发起加锁时，该设备就向从设备发送一命令。一个设备在涉及到加锁控制的多种状态时可处于其中的任何一种状态。图２中的椭圆２１０、２２０、２３０和２４０说明了上述四种上述状态即：未被锁住、被锁住、解锁请求挂起以及改变锁指针。连接椭圆的线说明了上述状态之间的转换。如图２顶部所示，加锁管理器可接收包括加锁和解锁请求在内的请求。如图２底部所示，加锁进程可响应请求而产生诸如“grant_lock”（允许加锁）或“lock_in_use”（正在使用锁）之类的应答。而且，加锁进程可产生一解锁请求和一设备出错／重置指示。

如图２所示，所述系统响应一加锁请求而从未被锁住的状态２１０转换成被锁住的状态２２０并发出一 grant_lock 应答。在响应被释放的锁时则发生从被锁住的状态２２０返回至未被锁住的状态２１０。经由状态２３０和２４０的转换则发生锁定释放（即开锁）。所述系统会响应一解锁请求而从状态２２０到解锁请求挂起状态２３０。在状态２３０下，所述系统会处理诸如持续时间参数和解锁条件之类的

参数，以确定是否应将锁定释放。如果不解除该锁，就发送一Lock_in_use 消息，所述系统会返回至被锁住的状态２２０。如果可以将锁释放，就发送一解锁消息，所述系统会转换成状态２４０，在这种状态中，被锁住的地址或加锁指针均会变成当前请求加锁的设备的地址或加锁指针。然后，所述系统转换至被锁住状态２２０并发送出一 grant_lock 消息。

图 3 至 7 说明了加锁管理器１０７和加锁进程１０７０在各种条件下即在设备间各种事务处理过程中的操作。在图 3 中，设备 B 未被锁住，并且，设备 B 响应一来自设备 A 的加锁请求而建立起对设备 A 的锁定。

在图 4 中，设备 B 响应先前的加锁请求而与设备 C 锁定，并且，收到一来自设备 A 的有冲突的加锁请求。持续时间参数设置为 0，这意味着设备 B 在解锁并与设备 A 再锁定之前不必请求同意解除源于设备 C 的锁。设备 B 只需判断解锁条件参数以确定是否解除与 C 的锁定。

在图 5 中，设备 B 响应来自设备 C 的先前加锁请求而与设备 C 锁住，并且，将持续时间参数置为 1。设备 B 向设备 C 发送一也称为“解除继承锁定”的解锁请求。设备 C 中的加锁管理器确定是否将锁解除。在图 5 中，设备 C 确定保持所说的锁并发送一“资源正在使用”的消息，该消息由设备 B 转发给设备 A。

图 6 示出了除了设备 C 通过解除设备 C 对设备 B 的锁定而对设备 A 的加锁请求加以响应以外与图 5 中的事务处理（即持续时间 = 1）相类似的事务处理。因此，设备 C 会向设备 B 发送一解锁消息，随后，设备 B 对设备 A 的锁定进行授权。

图 7 示出了这样一种事务，在该事务中，在持续时间等于 0 的情况下设备 B 与设备 C 锁住，并且，设备 A 招呼所有的加锁管理器以表明设备 A 要对设备 B 进行加锁。设备 C 会把设备 C 将释放对设备 B 的锁定通知给设备 A。然后，设备 A 向设备 B 发送一加锁请求，并且，由于持续时间参数被置为 0，故设备 B 会与设备 A 锁定并向设备 C 发送一消息以指示设备 C 去解除与设备 B 的锁。

此外，只有加锁设备或主设备需要加锁管理器１０７而从设备或可被锁住的设备仅使用加锁进程１０７０这种情况也属于本发明的范围。这种系统的的操作与本文所述系统相类似。

以下将说明有关本发明加锁管理器方面进一步的细节，包括该加锁管理器的除以上已作过说明的以外的那些特征。仅为方便起见，以下详细说明包括基于使用CAL程序设计语言的情况，但是，应该认识到，也可用任何适当的其它程序设计语言来代替CAL程序设计语言。

使用加锁的应用实例

以下将说明除业已说明了的那些应用以外的需要有加锁关系的情况。在第一个实例中，一设备（“设备A”）要将若干数据包下载到远程设备（“设备B”）中。设备A不希望覆盖掉写到设备B中的数据，直至完成下载。在正常情况下，这一功能需要在两个设备中均有分段网络服务（Segmented Network Service）。但是，依照本发明，设备A会形成一种临时的加锁关系，这种关系会确保设备A是能对设备B进行写操作的唯一设备。在传递了数据之后，设备A会终止上述加锁关系并对设备B“解锁”。

在第二个实例中，诸如分线盒之类的外部设备执行一节目定时器事件。分线盒会“锁住”VCR的送带机构。这就会使VCR处于录制模式，直至预约时间结束或者出现更优先条件。所述分线盒向VCR发送一Stop（停止）命令并在所述事件结束时对VCR解锁。

在第三个实例中，一个设备要显示一OSD（屏上显示，即ON SCREEN DISPLAY）消息一段时间。该设备为自己锁住显示器并发送相关的显示信息。所述显示设备显示上述信息一段时间。所述被锁设备由加锁设备本身或有条件的事件例如超时而被解锁。

还实现了加锁的一种特殊形式即“监听”。控制与多种设备的互操作需要能够知道这些设备的状态。包括监听能力的设备可对一组特定设备例如位于房屋内的设备所发送出的广播报告进行鉴别。

例如，可将状态变化信息自动地从一个设备发送至另一个设备。一VCR将其运转模式状态更新给一显示设备，或者，一温度传感器将最新信息传送给环境管理器。这一过程有两个问题。第一个问题是确定报告应送到何处。另一个相应问题是所述接收设备要确定哪些报告的信息对其功能是重要的。这种情况会在同一类型的设备诸如恒温器或保安传感器之类向同一房屋内的多个子系统发出多个实例的报告时出现。需要对信息源进行鉴别。所述监听功能可以实现这一功能。

存在有多种监听应用。某些比另一些更具动态性。环境控制是稳定

（静态）监听应用的一个实例。在这种系统中，监听功能在初始调整功能之后是相对静态的。这种系统需要用于监听的可靠且廉价的装置。诸如安全和特殊使用之类的问题是重要的。所述系统的配置在安装时进行并且很少非常频繁地改变。

这一点与A／V系统完全相反。A／V系统是真正动态的。设备会频繁地进行安装和拆卸。视频源和请求功能会使得系统配置迅速且经常地改变。播放、录制、显示信息以及计时器事件均需要在对设备的监听和加锁方面进行改变。需要将来自VCR的监听源设备立刻变成DBS设备。还要有大量的设备间的相互监听。TV可以正在监听一个设备即环境系统，以便在分线盒正在监听VCR的状态时进行显示。

所需的加锁和监听有多种级别。在多种情况下，应用仅需要一种设备级的加锁。但是，也存在要在对象级上进行加锁的例子。（也就是说，一个设备可包括一个以上的对象，例如，一个设备可在该设备内有多种独立的可控功能。）这种情况之一是VCR应用，在这种应用中，要锁住传动机构而同时允许其它设备编辑或增加计时器事件。与此相似，尽管要锁住TV中的显示对象以保证进行适当的显示，但是，不应锁住设备对其它通信的响应能力。

上述加锁属性有相关的、有条件的且继承的特征。有条件加锁特征的一个实例是当一个设备业已锁住了一个对象的时候。尚未被锁住的对象可以被锁住；但是，该设备却可能是不能加锁的。在这种情况下，仅能由已锁住对象之一的同一设备来锁住该设备。继承加锁是指一个设备中的所有对象均继承了所述对象的被锁住状态。

加锁管理器

要锁住其它设备的所有设备都必须有一“加锁管理器”（Lock Manager），它跟踪所有由上述设备所发起的活动锁的情况并负责对远程设备加锁和解锁。以下说明了这种由CAL实现的加锁管理器的结构。

加锁管理器				(16)数据表 存储器
为设备所形成的锁提供中心交换站				
IV	R / V	类型	名称	描述表功能
a61)	R	n	number_of_locks (锁数)	size_of_memory (存储器大小)
b(62)	R	n	Length_of_lock_ pointer (锁指针长度)	length_of_record (纪录长度)
C*(43)	R/W	n	current_lock (当前锁)	current_index (当前指针)
l*(6c)	R/W	d	Lock_pointer (锁指针)	Memory_block (存储块)

以下对上述参数进行说明.

1. Lock_pointer (锁指针) 的定义

每个数据存储器 memory_block (存储块) 的实例变量 (IV) 的记录均是一个指针, 它指向以远程方式驻留在网络上的被锁住的设备. 所述 Lock_pointer 数据包括三个域: (1)<Device_Network_Address> (设备网络地址), (2)<Locked_Context_ID> (被锁住的描述表标识), (3)<locked_Object_ID> (被锁住的对象标识). 那些以空白表示的域被认为是 null (空).

上述设备网络地址域是被锁住的设备的网络地址. 在 CEBus 应用中, 将 MAC 地址用作该网络地址.

Locked_Context_ID 域包括两个字节. 第一个字节是描述表号或描

述表通配符“DE” 。表通配符 DE 是指所说的锁对应于设备中同一描述表的所有实例。第二个字节是 Context_Class_ID（描述表类标识）。其余的字节均为空（null）。

locked_Object_ID 域分成两个字节。第一个字节是 Object_Number（对象号）或对象通配符“00” 。当使用对象通配符时，第二字节就是 Object_Class（对象类）。在未使用上述通配符时，第二个字节就为 null。这就可锁住在同一对象类（Object Class）中的所有对象。

2．解锁消息定义

上述被锁住的设备向加锁管理器发送一解锁消息。有三种定义过的解锁消息：（1）Device_Unlock（设备解锁），（2）Context_Unlock（描述表解锁），以及（3）Object Unlock（对象解锁）。

可用 Macro（宏）U 4 来发送解锁消息。U 4 宏包含两个变元域。第一个变元包含 Device_Network_Address。第二个变元包括描述表和对象信息。所述宏的原型是：

“<context_ID> < object_ID > U4 < data token （数据令牌）> < size of data（数据大小）> < escape token >

< Device_Network_Address > F5 > <data token> <size of data> < escape token> <Context_Number> <Context_class_ID> <Object_ID> <Object_Class>”

在锁住一描述表时，Context_Number 和 Context_class_ID 均放置在上述第二个变元域内，（不提供 Object_ID 和 Object_Class）。所述对象信息为 null。

在锁住一对象时，Context_Number、Context_class_ID、Object_ID 和 Object_Class 均在第二个变元内提供。

a） 设备解锁消息

Device_Unlock 消息必须包含被锁住的设备的< Device_Network_Address >。该消息中不包含上述第二个变元。

“00 04 55 34 F5 F5 F4 34 F6 < Device_Network_Address > F5”

放置在加锁管理器对象中的加锁指针矢量的实例

	设备网络地址	Locked_Context_ID		Locked_Object_ID	
	第一至第四字节	第五字节	第六字节	第七字节	第八字节
锁的类型	Locked_Address	Context_Number或通配符	Context_Class_ID	Object_ID或通配符	null或Object_Class
设备锁	Locked_Address				
描述表锁	Locked_Address	A0	(9F)*0…9E	01	
特定描述表	Locked_Address	(DF)*A0..DE	(9F)*0…9E	01	
同一描述表类的所有描述表	Locked_Address	DE	(DF)*A0..DE	01	
设备中的所有描述表	Locked_Address	DE	00	01	

对象锁	Locked_Address	(DF)*A0..DE	(DF)*A0..DE	(3F)*01..3E	<任何值>
描述表类中的所有对象	Locked_Address	(DF)*A0..DE	(DF)*A0..DE	00	<任何值>

加锁的设备

将一个设备锁住的能力取决于先前设置的锁。设备加锁要求复盖掉所有其它的锁。

加锁的过程使用了下表中的变量（IV）。在所有的过程中必须有locking_address IV。持续时间的缺省级别是“0”。 下表包含了对IV的详细说明。

IV	R / W	类型	名称
B (24)	R / W	N	persistence（持续时间）
E (45)	R / W	D	Key_Entry（密钥项）
G* (47)	R / W	D	Locking_Address（加锁地址）
0(4F)	R / W	D	Unlock_Condition（开锁条件）

a）持续时间 IV

持续时间 IV 会影响设备的解锁行为。目前有三种定义过的持续时间级：（0）Loose（不严格或松散）；（1）Tight（严格）以及（2）listening（监听）。持续时间 IV 的缺省值为 Loose。当一个设备变成未被加锁的状态时，它就应将持续时间设置为缺省值。

当持续时间值为“0”， 就会出现 loose 加锁级。loose 加锁会导致一个设备对试图对其进行控制的各设备有所区别。但是，这种结

合是 loose（松散）的。这意味着被锁住的设备可以允许对新的锁定的请求并确信以前的加锁设备一定会解除已存在的锁定。当加锁级别是 loose 时，加锁的设备应根据请求自动地释放所说的锁。这会导致快速地解开先前的锁并允许进行新的锁定。

当以 loose 方式锁住的设备接收到改变 Locking_address 的请求时，该设备就会返回一已完成的消息并通知先前加锁的设备解开所说的锁。loose 加锁意味着任何设备都可以改变 Locking_addressIV。

当持续时间域设置为“1”时，就会出现 tight（严格）加锁。这使得只有加锁设备才能改变或允许改变 Locking_addressIV。这一点在第三设备要在被锁住的设备内建立锁定时是有用的。在这种情况下，该第三设备查询被锁住的设备。被锁住的设备向加锁设备的加锁管理器对象（Lock Manager Object）发送一解锁消息。加锁设备可以生成锁也可以不生成锁，也就是说，加锁设备在解除先前的锁时会返回一完成令牌（Completed Token）FE。否则，加锁管理器会用（8）码返回一带有出错_资源在使用中的FD令牌。

将持续时间 IV 设置为“2”会导致称为监听（如前所述）的加锁形式。在监听的过程中，“被锁住的”设备会将其自身与另一设备相结合。一个设备可根据其应用而实现多种持续时间值。

b) Key_EntryIV（密钥项 IV）

Key_EntryIV 可用作进入被锁住设备的一个简单的密钥。这一 IV 能接触受保护的变量。这一选择可以保护诸如特殊诊断和安装操作之类的私有功能不被普遍地使用。它还允许一个设备在监听应用中鉴别不同的来源。

Key_EntryIV 有两种用法。第一种是在形成 tight 加锁（持续时间 = 1）时的用法。加锁设备提供一数据类型变量 Key_Entry 的值，该值授权使用所述设备的私有区。该值为说明的目的定义为“lock_key”。

第二种是在监听应用（持续时间 = 2）中的用法。监听功能将 Key_EntryIV 用作一附加的鉴别器。本文的后面会说明 Key_Entry 功能。

除了私有的第一个级别以外，生产商可以选择将 Key_EntryIV 制成一需要确认的受保护变量。这就会将特定的使用的情况与确认区分开来。当一个设备未被锁住或持续时间未设置成“loose”时，

Key_EntryIV 是不活动的。缺省的 Key_Entry 值是“null”。每当一个设备未被加锁或持续时间级别发生变化时，Key_EntryIV 都会恢复成其缺省值。这就会导致只有加锁的设备能够提供或读取 Key_EntryIV 的值。这允许只有一个设备接触私有数据或功能。

“lock_key” 可根据应用而是一设置在ＲＯＭ中的固定密钥或者是诸如“委托密钥”之类的可写密钥。正常情况下只有一个有效的 Key_Entry 值；但是，也可以有多个有效的 Key_EntryIV。生产商可以选择将 lock_keyIV 制成是可写的。通过解除所说的锁、或将锁持续时间变成〈〉1，或将 Key_EntryIV 设置成无效的项（例如逻辑长度是 0）或一 false（假）值，受保护设备会再次变为私有的。

c）Locking_AddressIV（加锁地址 IV）

Locking_addressIV 是一数据类型变量并且是锁住上述设备的网络地址。在 CEBus 应用中，将ＭＡＣ地址用作上述网络地址。

一个加锁设备在将其网络地址写进适当的 Locking_addressIV 内时会发起对一个设备的锁定。一旦一个锁就位，将 Locking_AddressIV 设置成一无效的地址（例如逻辑长度为 0）就会使该锁失效。对一个不支持加锁的设备的加锁请求会导致一返回值 Error 无效的 IV（出错_无效的 IV）。

d）Unlock_ConditionIV(开锁条件 IV)

第三个变量是 Unlock_ConditionIV，它允许用户指定自动将锁解除的条件。解锁条件可以使用在设备中能找到的 IV，（这一点与事件管理器对象相类似）。

Unlock_ConditionIV 包含有一布尔表达式，该表达式规定了对该设备的解锁条件。所述布尔表达式是一按ＣＡＬ语法所限定的ＣＡＬ表达式。该程序等待在设备中出现诸如计时器到时或超出阀值之类的某些条件。所述设备根据指定的解锁条件来解除所说的锁。请注意，解锁条件位于被锁住的设备内。持续时间值可以是 loose 或 tight。

用 setArray 方法来将一个条件加载到 Unlock_Condition 之内。在这种情况下，要加载的数据是实际的ＣＡＬ条件程序。该程序包括要等待的会导致解锁活动的条件。

正如在事件管理器对象中那样，在ＣＡＬ语法中用于 Unlock_ConditionIV 正确操作方面有两点稍微的不同。第一点不同是

识别要等待的实例变量．必须要提供 IV 的描述表和对象位置，而不是一个简单的 IV 标识符．例如，Unlock_Condition 总是检测（Audio control，Volume Control，current_value>Audio Context，Volume Control，max_value），而不是检测（current_value> max_value）．可以通过要求所有的字节来正确地分析语句。

包含在 Unlock_Condition 中的布尔表达式至多允许有两种要评估的关系．例如，当小时等于 1 2 或月份等于 6 时，时钟对象就可以解锁。上述布尔表达式中的第一个 IV 是所报告的值。除标准的关系操作符以外，还规定了一额外的操作符即 DELTA 操作符。请注意，在（用 setArray 方法）写 Unlock_ConditionIV 时，所述对象必须检查放置在 IV 中的条件的有效性。如果该条件是无效的，就返回一 Error_Bad Argument type（Error_错误变元类型）（1 2），并且，不会出现 Unlock_Condition。

加锁的实现

通过用 setArray 方法将 Unlock_ConditionIV 设置成一有效的目的地地址，就可以锁住一个设备。在 CEBus 应用的情况下，网络地址包括按 MSB、LSB 顺序出现的单元地址和房屋代码。一个用单元代码 0 0 3 4、房屋代码 0 0 1 A 的加锁消息的实例会导致下列消息:

<context> <object>46 471 Ff5 Ff5 Ff4 34 Ff6 00 34 00 1A (<context> <object>setArray 'GA'<DEL><DEL> <DATA> 34<ESCAPE> 0034 001A)

将对象 Locking_AddressIV 设置成一无效的地址（例如逻辑长度为 0）会将对象解锁。

(<context><object>setArray 'GA"<DEL><DEL><DATA>30<ESCAPE>

在不支持这一功能时就返回一 Error-InvalidIV(错误－无效 IV)。

1 对象加锁的实现

通过在各个对象中设置加锁进程可实现对象层次上的加锁。

2 描述表加锁的实现

上述对象层次的加锁可扩展至描述表。通过在描述表控制对象中设置加锁进程可实现描述表层次上的加锁.

将 Locking_AddressIV 设置成一有效值可以锁住描述表。描述表中例举的对象继承了相同的 Locking_Address。（这仅适用于持续时间

〈〉2 时。）如果已锁住了描述表中的对象，则不可能锁住该描述表，除非对象的 Locking_AddressIV 与描述表的 Locking_Address 值相同。

通过将加锁设备的网终地址写进置于描述表控制对象中的 Context_Locking_AddressIV 内，可以发起对描述表的加锁。

<context_ID>01 01 46 47 54 F5 F5 F4 34 F6 <Locking_Address>

Context_Persistence（描述表持续时间），context_Key_Entry（描述表加锁项），Context_Locking_Address（描述表加锁地址）以及 Context_Unlock_ Condition IV（描述表开锁条件）分别从对象 IV Persistence，Key_Entry，Locking_Address 以及 Unlock_Condition 中继承它们的行为。

3 设备加锁的实现

通过在节点控制对象（Node Control Object）中设置加锁进程可实现设备层次上的加锁。所述节点控制对象可以有自己的专用对象加锁进程。所述设备层次上的加锁变量是:

	Node Control Object（节点控制对象）			(01) Node Control（节点控制）
	节点控制对象中允许设备加锁的变量			
IV	R / W	类型	名称	描述表功能
o(6Fh)	R	d	object_list（对象表）	描述表中所使用的对象的列表
z(6a)	R / W	N	Device_Persistence（设备持续时间）	持续时间
v(76)	R / W	D	Device_Key_Entry（设备密钥项）	Key_Entry（密钥项）

T*(54)	R / W	D	Device_Locking_ Address (设备加锁地址)	Locking_Address (加锁地址)
V(56)	R / W	D	Device_Unlock_ Condition (设备开锁条件)	Unlock_Condition (开锁条件)

Device_Persistence , Device _Key_Entry , Device _Locking_Address 以及 Device_Unlock_ Condition IV 分别从对象 IV Persistence, Key_Entry, Locking_Address 以及 Unlock_Condition 中继承它们的行为。

通过将加锁设备的网络地址写进 Device_Locking_Address IV, 可以发起对设备加锁。

00 01 46 54 F5 F5 F4 34 F6 <Locking_Address >

加锁方案

为了说明起见, 加锁设备是要锁住另一设备的设备。被锁设备是要被锁住的设备。加锁请求可发往要被锁住的设备或者发往占有锁的加锁管理器。在正常条件下,可通过试图将一新地址写进 Locking_Address IV 而形成一加锁请求。该请求可以是直接的也可以是有条件的。

a) 直接加锁请求

用单元代码(Unit Code) 0 0 3 4、房屋代码(House Code) 0 0 1 A 的加锁消息的实例会导致下列消息:

<context> <object> 46 4741 Ff5 Ff5 Ff4 34 Ff6 00 34 00 1A(<context> <object> setArray 'G' <DEL> <DEL <DATA >'4' <ESCAPE> 0034 001A)

设备必须确定它是否是可锁住的。如果未锁住上述设备、对象、描述表是没有锁住的, 就返回一已完成的令牌。加锁设备将加锁指针放进加锁管理器对象。

b) 有条件的对象加锁请求

用单元码 0 0 3 4、房屋码 0 0 1 A 的加锁消息的实例会导致下列

消息:

<context><object> 5647 F7<context><object> 46 471 F5f5 Ff5 Ff4 34 Ff6 00 34 00 1A EB 44 47 F8

<context> <object>I (IF "G" <BEGIN> <context> <object>setArray "G"<DEL> <DEL<DATA > '4'<ESCAPE> 0034 001A <ELSE> setArray "G"<END>)

如果 array "G"(阵列"G")未被使用,就锁住所说的设备。否则,就不锁住该设备。这种方法可以和资源加锁一道使用。如果所述设备正在使用,则发请求的设备可通过发送一个广播消息向加锁设备请求"打招呼" 以使它放弃被锁住的设备。

c)使用 LOCK 宏 U 2

可用 LOCK 宏 U 2 来锁住描述表和对象。宏消息包括宏 I D,之后是变元表。有两个必需的变元: 1) Locking_Address(加锁地址);以及, 2) persistence(持续时间)。如果未填写变元表,就认为该变元表为 null。在形成加锁之后,更新 Key_Entry(密钥项)和 Unlock_ConditionIV(开锁条件)。

<context_ID><object_ID> " U2 " <data token><size of data><escape token>< Locking_Address > F5<persistence>F5<Key_Entry>

就 C A L 宏消息而言锁住描述表 1 1 的实例是:

"11 011 55U 322 F4 34 F6 0F 32 23 34 F5 01"

变元 0 F f 3 2 2 3 3 4 是加锁地址。持续时间设置为 1 。这时,加锁设备可发送一 Key_Entry(密钥项) IV 数据和 Unlock_Condition(开锁条件) IV 数据。

在未提供 Locking_Address 时,就假定是宏消息的源地址。

"11 01 55 32 F5 F5 01"

加锁释放请求

a) 发往加锁主设备的加锁释放请求

(1) 用 DISINHERIT(解除继承) (选项 1)

如果在对象、描述表或设备层次上锁住了对象,则该对象会请求妨碍加锁的设备放弃加锁。消息是发送给加锁管理器对象的:

<Universal Context><Locking Manager Object> · disinherit

<lock_pointer(6C)>F4[number of bytes]F6 <device_address> <context_id> <object>

解除对<device><context><object>的加锁的继承.

所述 context_id 若不为 A 0 则包括 context_number. 若为 A 0，则可任选地包括描述表号。

（2）用 UNLOCK 宏 U 4 1（选项 2 1）

UNLOCK 宏 U 4 1 被发送给 EVENT Manager Object（事件管理器对象）。该宏与预定的加锁指针相匹配并请求删除该指针。如果存在着匹配，那么，加锁管理器或者对所述请求授权或者拒绝该请求。如果不能找到加锁设备，则被锁住的设备将锁解除。详情节见 2.1.2 节。

监听的实现

监听功能是与前述报告方法相对应的。监听是在对其它设备或网络资源感兴趣的对象中实施的。网络资源的一个实例是时间。

监听功能能使一个设备鉴别包含状态信息的广播消息。监听功能是加锁功能的特殊应用。在监听模式中，所述对象不继承描述表或设备的加锁状态。（这在设备或描述表已锁住时对监听功能正确地实施是必需的）。

在监听过程中，一个设备将自身和另一个设备相结合。这意味着一个设备可以设置它自己的 Locking_AddressIV. Locking_AddressIV 或者是只读的或者是读／写的。写选项实际是松散加锁的一种类型。将持续时间 IV 置为“2”。Key_EntryIV 可用作特殊的使用鉴别器。下表包含了监听 IV 列表。

	对象监听 IV 列表			
IV	R / W	类型	名称	描述表功能
G*(47)	R / W **	D	Lock_Address（加锁地址）	对象监听地址
B*(42)	R	N	Persistence（持续时间）	2 = 监听

当 Lock_Address 是只读变量时，设备就设置它自己的加锁地址。当从外部设置 Locking_Address 时，加锁设备不将 lock_pointer 放进它自己的加锁管理器。

有三种不同的监听方法。第一种方法使用源网络地址以控制使用监听对象。所述对象支持与 Locking_AddressIV 相对应的源设备地址进行鉴别。在这种情况下，将消息写进适当的 Instant Variable（瞬时变量）。

第二种和第三种监听方法使用了 Key_Entry 设备。这种方法需要宏消息。该宏消息 L 1 用于同时传送 Key_Entry 值和数据。所述监听设备验证所说的宏来自适当的来源地址并且所提供的 Key_Entry 值与前述所需值相匹配。

1 多来源监听的实现方法

存在这样的情况即监听对象要在不进行鉴别的情况下监听所有的广播源。当火警传感器广播它正检测火警时就会出现这种情况。该传感器将信息广播到网络上。监听设备要监听所有可能会报告火警的来源。

一般地说，通过允许一个设备接收任何设备（传感器）的状态，可以实现多源监听应用。这一点可通过将 Locking_AddressIV（单元地址 IV）置成一广播地址来表示。在 CEBus 应用中，将监听鉴别器设置成特定房屋代码中的所有 Unit_addressIV 或者设置成网络上的所有地址。警报发出源或区域地址包括在所述消息中；警报发出源或区域的地址包含在该消息中，这可从包含在消息中的 source_address 域或状态矢量中取出。

2 通过 Key_Entry 和 Locking_Address 进行监听

通过这种方式，lock_key 的值会变成一选择器的值，该值允许一个设备将个别描述表与对象区别开。例如，在时间描述表中，可在 Macro Message（宏消息）中提供时间类型标识符。可以检验该值是否为所需的时间类型。如果一个设备具有报告描述表的多个实例，该设备就可在 Key_Entry 域中包括描述表号。接收节点检查到来的描述表号并验证该号是否为所关心的号码。

<context_ID><object_ID>53 31 F4 <size of data><escape token>5< Locking_Address >F5<data token><size of data><escape token><Key_Entry>F5< Argument_listmessage>

在监听过程中，设备将自身与另一个设备相结合。在监听的这种情况下，一个设备可设置它自己的 Locking_AddressIV。Locking_AddressIV 可以是只读的或者是读/写的。

3 通过 Key_Entry 进行监听

还可通过将 Locking_Address 置成“00 00 00 00” 来完成监听。在这种情况下，唯一的鉴别器是 Key_Entry 值。这就允许一个对象根据存取密钥进行鉴别。可按如下方式做到这一点:

<context_ID><object_ID>53 31 F5 00 00 00 00 F5 <data token><size of data><escape token><Key_Entry>F5< Argument_list>

当 Locking_Address 是只读的时，所述设备就以排它的方式设置它自己的加锁地址。当 Locking_Address 是读/写的时，所述加锁是松散型的。加锁设备不将一 locking_pointer 放进它的加锁管理器。

<context_ID><object_ID>L1<data token><size of data><escape token>< Locking_Address >F5 F5 <message>

存在某些实例，在这些实例中，要有唯一的标识符以便在一个单个的设备中鉴别多个描述表或对象。一个例子是包含有多个传感器的温度传感器设备。应将每个传感器与相应的监听对象相结合。这一点是用 key_entry 方法实现的。在所说的方法中，如果 key_entry =空，则不存在鉴别。在将 Key_Entry 设置为一定的值时，监听设备在更新对象之前需要正确的 Key_Entry 值。

实例: 监听变量具有一类似 key_entry 号的鉴别器，它允许用户来设置一鉴别器。在 Macro_ID key_entry F5 Listening DATA 处发送所述数据和鉴别器。如果未提供 Key_entry，则所述鉴别器是网络地址。

尽管参照若干实施例详细地说明了本发明，但是，可以看出，根据对以上内容的阅读及理解，对本技术的专家来说，可以有多种上述实施例的替代形式，后附权利要求的范围内包括了上述替代形式。

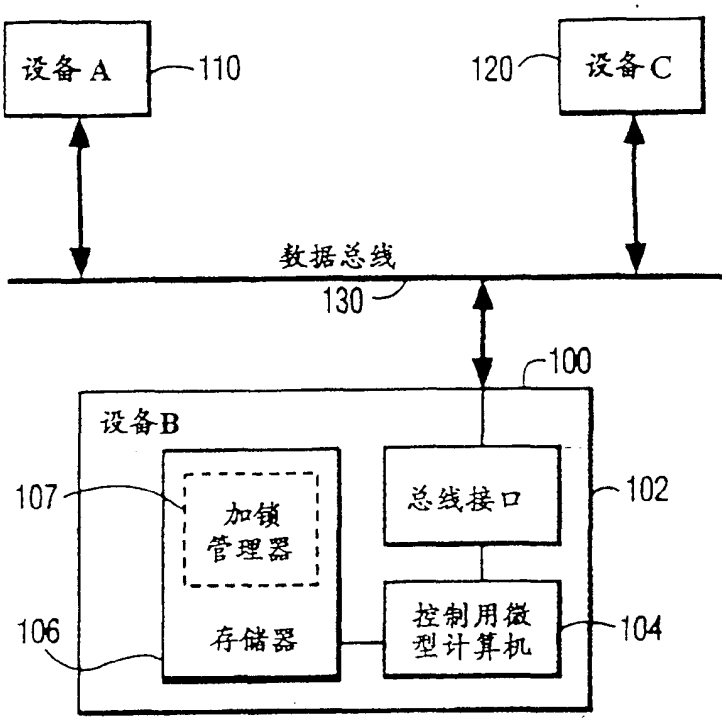

设备 A
110
120
设备C
数据总线
130
100
设备B
107
加锁
管理器
存储器
106
总线接口
102
控制用微
型计算机
104

图 1A

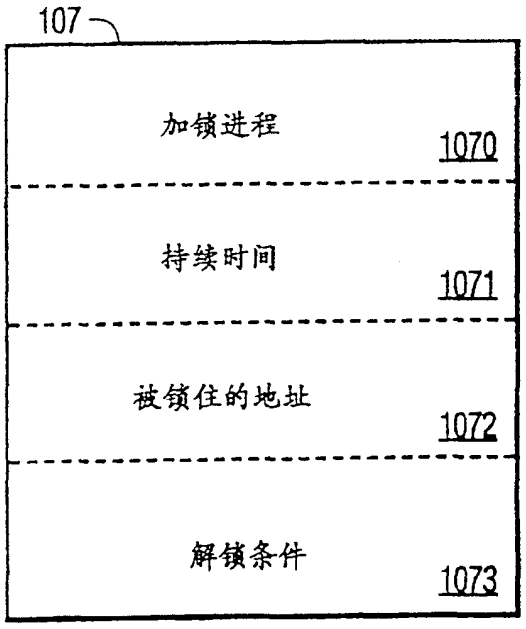

107
加锁进程
1070
持续时间
1071
被锁住的地址
1072
解锁条件
1073

图 1B

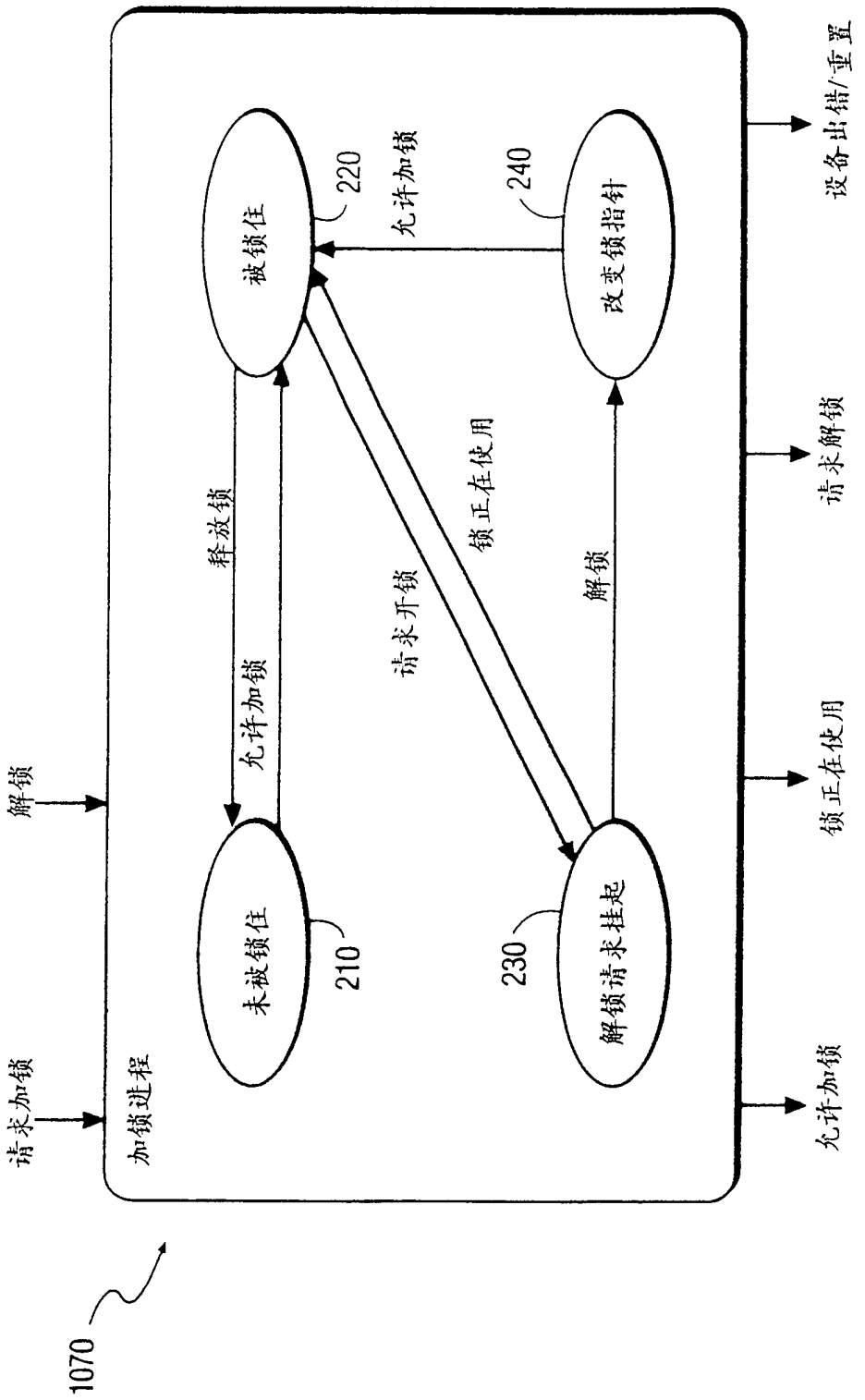

1070
加锁进程
请求加锁
解锁
未被锁住
210
被锁住
220
解锁请求挂起
230
改变锁指针
240
允许加锁
释放锁
允许加锁
请求开锁
锁正在使用
解锁
设备出错/重置
请求解锁
锁正在使用
允许加锁

图 2

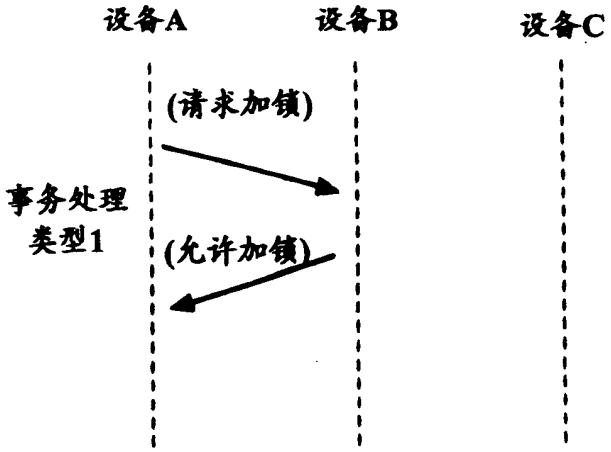

设备A
设备B
设备C
(请求加锁)
事务处理
类型1
(允许加锁)

图 3

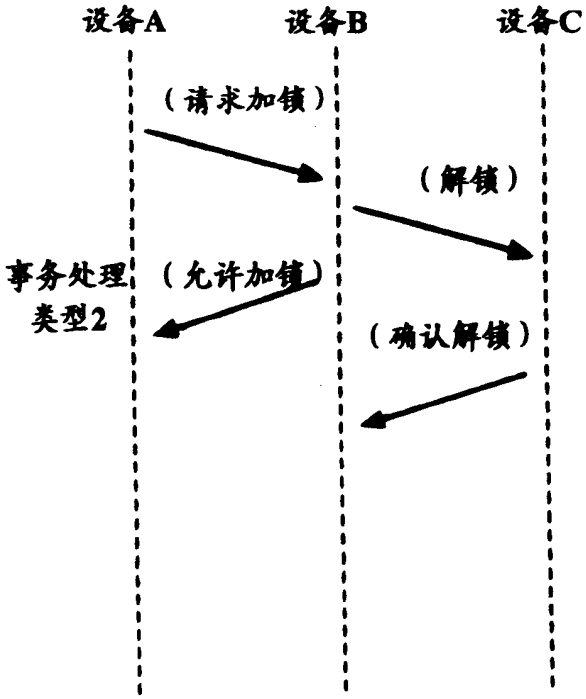

设备A
设备B
设备C
(请求加锁)
(解锁)
事务处理
类型2
(允许加锁)
(确认解锁)

图 4

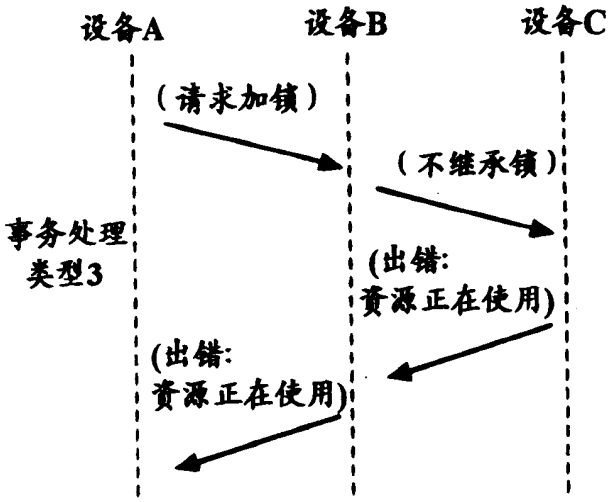

设备A
设备B
设备C
（请求加锁）
（不继承锁）
事务处理
类型3
(出错:
资源正在使用)
(出错:
资源正在使用)

图 5

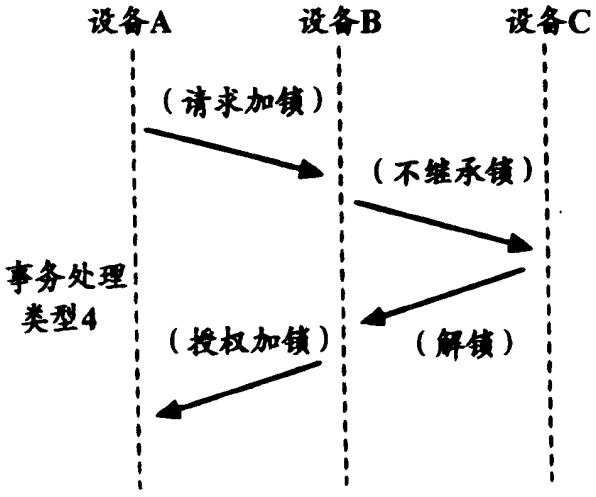

设备A
设备B
设备C
（请求加锁）
（不继承锁）
事务处理
类型4
（授权加锁）
（解锁）

图 6

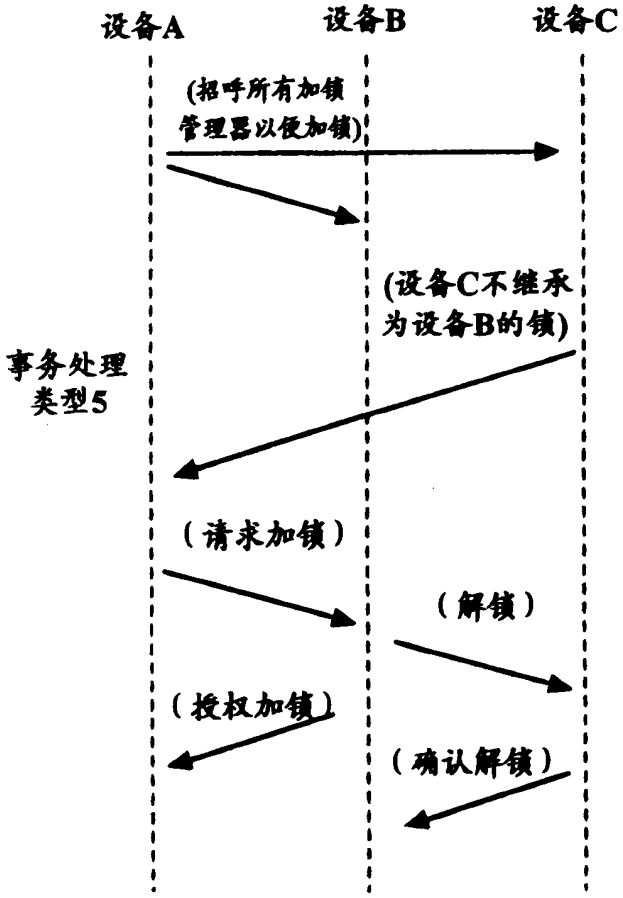

设备A
设备B
设备C
(招呼所有加锁
管理器以便加锁)
(设备C不继承
为设备B的锁)
事务处理
类型5
(请求加锁)
(解锁)
(授权加锁)
(确认解锁)

图 7