[19]中华人民共和国专利局

[11] 公开号 CN 1125489A

[12] 发明专利申请公开说明书

[21]申请号 94192492.0

[51]Int.Cl[6]

G06F 9/44

[43]公开日 1996 年 6 月 26 日

[22]申请日 94.1.6

[30]优先权

[32]93.9.13 [33]US[31]08/114,640

[86]国际申请 PCT/US94/00143 94.1.6

[87]国际公布 WO95/08150 英 95.3.23

[85]进入国家阶段日期 95.12.18

[71]申请人 塔里根特公司

地址 美国加州

[72]发明人 C·约翰·托别斯

H·罗蒂文·米而恩

[74]专利代理机构 永新专利商标代理有限公司

代理人 程 伟

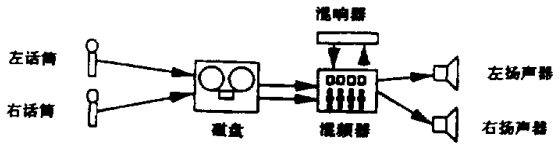
G06F 17/30 G06F 17/21

权利要求书 3 页 说明书 44 页 附图页数 25 页

[54]发明名称 面向目标的乐器数字接口系统

[57]摘要

一种用于把乐器数字接口目标与多个多媒体目标进行连接以便由具有存储器和显示器的计算机对一个多媒体表示进行面向目标的模拟的系统。在显示器上生成的多个多媒体目标包括位于存储器中的至少一个连接目标和至少一个乐器数字接口目标，在显示器上显示的多个多媒体目标包括至少一个乐器数字接口目标。多媒体目标和乐器数字接口目标被连接并且通过在多媒体目标和乐器数字接口目标之间的连接对信息进行路由选择以便生成一个多媒体表示。



权 利 要 求 书

1. 一种用于多媒体表示的系统，其特征在于包括:

   a) 一个处理器;

   b) 一个与所述处理器相连并受其控制的存贮器;

   c) 一个与所述处理器相连并受其控制的显示器;

   d) 存贮在存贮器中并可在所述显示器上显示的多个多媒体目标;

   e) 位于所述存贮器之中并可由所述显示器显示的一个连接目标，该连接目标用于将所述的多个多媒体目标中的第一个多媒体目标与一个乐器数字接口（MIDI）目标相连；以及

   f) 由所述处理器控制通过所述连接目标在所述的第一多媒体目标以及所述的乐器数字接口目标之间确定信息路由的装置。

2. 根据权利要求1所述的系统，还包括至少一个位于多媒体目标中用于接收乐器数字接口信息的端口装置。

3. 根据权利要求2所述的系统，还包括至少一个位于多媒体目标中用于发送乐器数字接口信息的端口装置。

4. 根据权利要求2所述的系统，还包括与一外设接口的装置。

5. 根据权利要求1所述的系统，还包括播放音乐的装置。

6. 根据权利要求1所述的系统，还包括在记录的信息上配音的装置。

7. 根据权利要求1所述的系统，还包括把一个乐器数字接口类型转换成另一种媒介类型的装置。

8. 根据权利要求1所述的系统，还包括把一个乐器数字接口类型转换成多种媒介类型的装置。

9. 根据权利要求1所述的系统，还包括由多媒体目标的每个端口联接一种数据类型并多形地连接多媒体目标的装置。

10. 根据权利要求1所述的系统，还包括联接不同类型的多媒体目标的转换装置。

11. 一种用于在计算机系统上进行多媒体表示的方法，其中的计算机系统包括处理器以及与该处理器相连的存贮器和显示器，该方法的特征在于包括以下步骤:

a) 在所述存贮器中生成多个包括至少一个连接目标和至少一个乐器数字接口目标的多媒体目标;

b) 在所述显示器上显示一个多媒体目标;

c) 在所述显示器上显示一个乐器数字接口目标;

d) 把所述多媒体目标和所述乐器数字接口目标相连；以及

e) 在所述多媒体目标和所述乐器数字接口目标之间确定信息路由以生成一个多媒体表示。

12. 根据权利要求11所述的方法，还包括在一个多媒体目标端口接收乐器数字接口信息的步骤。

13. 根据权利要求11所述的方法，还包括发送乐器数字接口信息到另一多媒体目标的端口的步骤。

14. 根据权利要求11所述的方法，还包括与一外设接口的步骤。

15. 根据权利要求11所述的方法，还包括播放音乐的步骤。

16. 根据权利要求11所述的方法，还包括在记录的信息上配音的步骤。

17. 根据权利要求11所述的方法，还包括把一个乐器数字接口类型转换成另一种媒介类型的步骤。

18. 根据权利要求11所述的方法，还包括把一个乐器数字接口类型转换成多种媒介类型的步骤。

19. 根据权利要求11所述的方法，还包括由多媒体目标的每个端口与一种数据类型联接并多形地连接多媒体目标的步骤。

20. 根据权利要求11所述的方法，还包括联接不同类型的多媒体目标的步骤。

说　明　书

面向目标的乐器数字接口系统

本发明所属技术领域

本发明通常涉及一种改进的计算机系统，特别是涉及一种用于在两个多媒体部件之间对乐器数字接口信息进行路由选择的系统。

本专利申请的部分内容受版权保护。版权所有者允许对出现在专利局的专利文献或记录中或专利公开中的复制再现，但在其它任何情况下保留所有版权。

背景技术

对计算机系统而言，发展最快的应用恐怕就是多媒体了。用户愈来愈多地使用计算机将图形，声音和图象信息表示给终端用户。用户愈来愈需要一种人与处理控制的接口，以用于管理多媒体的表示。过去，使用时间矩阵和编程语言来完成多媒体的表示。然而，模拟一个灵活的混合电路插件板以使得能够把带有信息表示的音乐或声音表示成展开的多媒体表示是不可能的。

不具有本发明能力的当前多媒体系统的例子是在 NEWMEDIA 三月刊(1993)上发表的 “It's Showtime” (第 36-42 页)中所描述的 Apple's Ouicktime 和Microsoft's Video for Windows。在 IEEE Spectrum三月刊(1993)第 22-31 页的 “Interactive Multimedia” 和 IEEE Spectrum (1993)第 32-39 页的 “The Technclogy Framework” 中讨论了解决在现有技术中遇到的路由选择问题的重要性。所述文章指出了用于控制多媒体产生的美学接口的重要性。

本发明的目的

因此，本发明的一个主要目的就是要提供一种系统和一种方法，用于在多媒体表示的过程中由具有存储器和显示器的计算机对乐器数字接口多媒体信息进行连接，路由选择，以及滤波。在该系统中的处理器具有一个与之相连并由其控制的存储器和显示器。在存储器中生成多个多媒体目标并由显示器显示。在存储器中还生成一个连接目标并在显示器上显示，以便把一个多媒体目标连接到一个乐器数字接口目标。最后，通过该连接目标在多媒体目标和乐器数字接口目标之间对信息进行路由选择。

附图的简要说明

图 1 的方框图示出了依据最佳实施例的一个个人计算机系统;

图 2 简单地示出了在现有技术中使用一个磁带机、一个混频器、一个混响单元、一对话筒和一对扬声器的组合家用音响装置;

图 3 示出了根据一个最佳实施例的媒体部件;

图 4 示出了根据一个最佳实施例的音频播放器部件;

图 5 示出了根据一个最佳实施例的扇入和扇出端口;

图 6 示出了根据一个最佳实施例的具有零或多个音频输入端口并具有零或多个音频输出端口的音频部件;

图 7 示出了根据一个最佳实施例通过使用一个音频播放器去记录和播放声音注释所使能的声音注释的应用;

图 8 示出最根据一个最佳实施例的声音通信/耳机应答的应用;

图 9 示出了根据一个最佳实施例的一个被外部同步到主时钟的多媒体播放器;

图 10 示出了根据一个最佳实施例的将三种声音，即音乐、音响效果和话外音混合到一起并通过诸如一个扬声器的输出设备加以输出;

图 11 示出了根据一个最佳实施例的某些音频类型;

图 12 示出了根据一个最佳实施例将一种音频类型转换成另一种音频类型的转换处理;

图 13 示出了根据一个最佳实施例的一个远程过程调用;

图 14 示出了根据一个最佳实施例具有用于从其输入缓冲器中读出数据，对数据进行处理以及将处理结果写入其输出缓冲器的一个相关 Run() 成员功能的音频处理器结构;

图 15 示出了根据一个最佳实施例的音频处理器;

图 16 示出了根据一个最佳实施例在图 15 中所示之网络是如何执行处理的;

图 17 示出了根据一个最佳实施例的一个音频端口;

图 18 示出了根据一个最佳实施例的诸如是一个播放机的音频处理器;

图 19 示出了根据一个最佳实施例与用于激活一个音频处理器相关的递归逻辑;

图 20 示出了根据一个最佳实施例用于表示与一个用于释放所述音频处理器相关的详细逻辑的流程图;

图 21 示出了根据一个最佳实施例表示与运行一个音频处理器相关的逻辑的流程图;

图 22 示出了根据一个最佳实施例将一个视频数字转换器部件连接到一个观查器部件上以便在计算机屏幕上进行显示的例子;

图 23 示出了根据一个最佳实施例在一个效果处理器中对来自两个视频目标的图象进行混合并将其结果显示于计算机屏幕上的一个例子;

图 24 示出了根据一个最佳实施例如何使用图形端口;

图 25 示出了根据一个最佳实施例表示与一个输出端口的 Write 成员功能相关的逻辑的流程图;

图 26 示出了根据一个最佳实施例的一个输入端口的读出处理;

图 27 示出了根据一个最佳实施例的一个输入端口的下一个成员处理;

图 28 示出了根据一个最佳实施例两个部件 MIDI 播放机 2800 和 MIDI 接口 2810 是如何被用于操纵联接到所述计算机上的音乐合成器以便进行演奏的;

图 29 示出了根据一个最佳实施例 MIDI 数据是如何被记录和如何从一个外部音乐合成器被播放的;

图 30 示出最根据一个最佳实施例 MIDI 数据是如何被播放的;

图 31 示出了根据一个最佳实施例的包括有 MIDI 和音频端口的媒体部件;

图 32 示出了根据一个最佳实施例被划分成公用、实时和互斥信息的详细的系统信息;

图 33 示出了根据一个最佳实施例 MIDI 信息的某些格式;

图 34 示出了一种根据最佳实施例的封装 MIDI 信息类型和结构，状态字节和时间标记的 MIDI 包目标;

图 35 示出了根据一个最佳实施例的一个扇入操作的例子;

图 36 示出了根据一个最佳实施例的一个 MIDI 输出端口的 Write() 成员功能;

图 37 示出了根据一个最佳实施例的一个 MIDI 输入端口的 Read() 成员功能;

图 38 示出了根据一个最佳实施例的具有一个单一输入端口和两个输出端口的媒体部件;

图 39 示出了一个根据最佳实施例能够演奏时基媒体顺序的媒体部件;

图 40 示出了根据一个最佳实施例用于使一个音频部件播放和记录音频数据的音频播放机;

图 41 示出了根据一个最佳实施例的一个扬声器部件的例子;

图 42 示出了根据一个最佳实施例的话筒部件的例子;

图 43 示出了根据一个最佳实施例的一个混频器部件的例子;

图 44 示出了根据一个最佳实施例的一个分离设备部件的例子;

图 45 示出了根据一个最佳实施例的一个增益部件的例子;

图 46 示出了根据一个最佳实施例的一个回波部件的例子;

图 47 示出了根据一个最佳实施例的一个抖颤(fuzz)部件的例子;

图 48 示出了根据一个最佳实施例的一个音频类型转换器的例子;

图 49 示出了根据一个最佳实施例的一个音频多路转换器;

图 50 示出了根据一个最佳实施例的一个声音部件;

图 51 示出了根据一个最佳实施例被置入一个声音部件之中的部件;

图 52 示出了根据一个最佳实施例的一个实际的扬声器部件;

图 53 示出了根据一个最佳实施例的一个实际的话筒部件;

图 54 示出了根据一个最佳实施例的一个图形播放机部件;

图 55 示出了根据一个最佳实施例的一个图形观看器部件;

图 56 示出了根据一个最佳实施例的一个视频数字转换器部件;

图 57 示出了根据一个最佳实施例的一个 MIDI 播放机部件;

图 58 示出了根据一个最佳实施例的 MIDI 接口部件;

图 59 示出了根据一个最佳实施例的一个 MIDI 滤波器部件;

图 60 示出了根据一个最佳实施例的一个 MIDI 映象器部件;

图 61 示出了根据一个最佳实施例的一个 MIDI 程序映象器部件;

图 62 示出了根据一个最佳实施例的一个 MIDI 注释映象器部件; 以及

图 63 示出了根据一个最佳实施例的一个 MIDI 频道映象器部件。

本发明的最佳实施例

本发明最好应用于驻留在诸如是 IBM ® PS/2 ® 或 Apple ® Macintosh ® 计算机的个人计算机上的操作系统中。典型的硬件环境示于图 1，该图示出了根据本发明的一个工作站的典型的硬件结构，它包括一个诸如是传统微机的中央处理单元 10 和一些通过系统总线相互连接的其它单元。图 1 所示的工作站包括一个随机存储存储器(RAM)14，一个只读存储器(ROM)16，一个用于将诸如是一个磁盘单元 20 的外围设备连接到

总线的 I/O 适配器 18，一个用于将键盘 24，鼠标 26，扬声器 28，话筒 32 和/或诸如触屏设备的其它用户接口设备(未示出)连接到所述总线上的用户接口适配器 22，一个用于将所述工作站连接到数据处理网络上的通信适配器 34 和一个用于将所述的总线连接到一个显示设备 38 上的显示器适配器 36。所述工作站中具有例如 Apple System/7 ® 操作系统这样的操作系统。

在一个最佳实施例中，本发明是根据使用面向目标编程技术的 C ++ 编程语言加以执行的。如本专业技术领域以内的普通技术人员所理解的那样，面向目标编程(OOP)的目标是多个包含有数据结构和在所述数据上进行操作的软件实体。这些元素一起使得所述目标能够根据由其数据元素所表征的特征和由其数据控制功能所表征的特性去有效地模拟现实存在的任一实体。利用这种方式，目标可以模拟象人和计算机这样的具体事物，并且它们可以模拟象数这样的抽象概念或几何概念。目标技术的优点源于三个基本原理，即：封装、多形性和继承。

目标隐藏或封装它们数据的内部结构和使其功能可以进行工作的算法。目标不展示这些实施细节，目标表示代表没有用多余信息的抽象的接口。多形性是封装的进一步骤，其概念是多种形状，一个接口。软件部件能够产生另一个部件的请求而不必确切了解那个部件是什么。接收所述请求的部件解释它，并根据它的变量和数据确定如何执行所述请求。第三个原则是继承，它允许研制者重复使用予先存在的设计和代码。这种能力使研制者得以避免从无到有的建立软件。进而，通过继承，研制者获得具有继承性的子类别,这样，研制者随后就可以定制以满足他们的特殊需要。

现有技术的方法是将在一个过程环境中的多个目标分出层次并将库进行分类。市场上的很多应用结构都是采取这种设计方法。在这种设计中，在一个单片操作系统之上有一个或多个目标层。虽然这种方法在所述目标层中应用了包括封装，多形性和继承的所有原则并且这种方法相对于过程编程技术而言是一种改进，但这种方法有某些限制。这些困难是由下面这样一个事实所引起的，即虽然研制者很容易重新使用他们自己的目标，但很难使用其它系统的目标，并且所述的研制者仍然需要选用过程操作系统(OS)调用才能进入较低的非目标层。

面向目标编程的另一个概念就是趋向于应用开发的结构处理。对结构最理性的定义之一来自 Illinois 大学的 Ralph E. Johnson 和 Purdue 的 Vincent F.Russ 。在 Illinois 大学教学报告 UIUCDCS91-1696 发表的 1991 论文“重新使用面向目标设计”中，他们作出了如下定义：“一个抽象的类别就是一组共同协作以执行一组职责的执行规定的一组计算职责的一组目标类别”。根据编程的观点，所述的结构是多组用于提供工作应用的预制结构的交互连接目标的类别。例如，一个用户接口结构可以提供绘图窗口，滚动条以及菜单等的支持和“系统预置”特性。由于所述的结构是基于目标技术的，所以这些特性能够被继承和发展，从而允许研制者在一个专门技术的特定领域内拓展所述的结构并建立定制的解决办法。它超过传统编程的主要优点就在于所述编程员未改变原来的代码而代之以扩展所述的软件。另外，由于所述的结构提供了体系引导和模型而同时又允许它们随后自由地将所述特定的作用唯一地提供给出问题的区域，研制者就可以不必盲目地通过代码层进行工作。

从商业的观点来看，可以把结构看作是一种封装或包括一个特定知识领域中专门技术的一种方式。合作发展组织、独立软件供应商以及系统综合者在诸如制造、计算以及流通业务等特殊领域内已经获得了有关的专门知识。这些专门知识被包括在它们的代码之中。通过将其收录在所述组织的编码之内，所述结构允许所述的组织收集和封装那个专门技术的共同特征。首先，这允许研制者建立或扩展利用这种专门技术的应用，其次，所述的问题获得了解决并自始至终贯彻执行和使用所述的商业规划和设计。结构以及结构后面所隐含的专门技术对于那些在垂直市场中已经获得了诸如制造、会计或生物技术的专门知识的组织而言具有至关重要的意义，并且它能够提供一种分布式结构以用于封装、转卖和推广应用他们的专门知识，并促进技术的进步和普及。

从历史上来看，所述结构只是在最近才作为个人计算站的主要概念而出现。这种进展是由于可获得诸如 C ++的面向目标语言而推动的。传统上，C ++大多是从 UNIX 系统和研究者的工作站中发现的，而不是从商业设备的计算机上发现的。正是诸如 C ++和其它诸如 Smalltalk 等面向目标语言的语言，而使一些大学和研究单位研发出导致今天的商用结构和类别库的产品。某些这类产品的例子包括 Stanford 大学的 InterView 和来自 Caunegie-Mellon 大学的 Andrew toolkit，以及来自 Zurich 大学的 ET ++结构。

所述结构种类的范围从用于协助开发用户接口的应用结构直到用于提供诸如通信、打印文件系统支援以及图形等的基本系统软件服务的低级结

构。所述应用结构的商用实例是 MacAPP (Apple), Bedrock (Symantec), OWL (Borland), NeXTStep App kit (NeXT),和 Smalltalk-80MVC (ParcPlace)。

利用所述结构进行编程要求习惯于其它种类型系统的研制人员采取一种新的思维方式。事实上，它根本不象传统意义上的“编程”。在诸如 DOS 或 UNIX 等老式操作系统中，研制者自己的程序提供了所述结构的全部。所述的操作系统通过系统调用来提供服务，即当它需要服务时，研制者的程序将产生一个调用，而在提供了所述的服务时，则控制其返回。所述的程序结构是以控制流程为基础的，所述控制流程包含于所述研制者编写的代码之中。

当使用所述结构时，这一切都是相反的。所述研制者不再决定控制流程。所述研制者必须放弃根据执行流程来理解编程任务的意向。必须根据所述目标的职责来考虑，这就必须依赖结构去确定任务应该在何时被执行。由研制者写入的程序是由研制者未写入甚至是研制者从没有见过的代码激活的。控制流程中的这一转变对于那些仅熟悉过程编程的研制人员来言可能是一个重大的心理障碍。然而，一旦理解了这一点，所述的结构编程所需要进行的工作量要远比其它类型的编程少得多。

利用与一个应用结构向研制者提供予制功能相同的方式，诸如包括于一个最佳实施例中的系统结构通过提供系统级服务使用相同的概念。诸如是系统程编员的研制者使用子类别/置换来建立定制的解决方法。例如，可以认为一个多媒体结构能够提供用于支援诸如音频，视频， MIDI ，动

画片制作等新的各种设备的基础。需要支援一种新设备的研制者应当对一个设备驱动器进行写入。为了利用结构达到这些目的，所述的研制者仅需要将所规定的特征和特性提供给新设备。

在这种情况下，研制者提供一种工具用于建立将要被多媒体网络调用的成员功能。研制者的直接受益是每种设备所需的类属代码已经由多媒体结构所提供。这就意味着研制者需要编写，测试和调试的设备驱动器代码将更少。使用系统结构的另一个例子应当具有一个用于 SCSI 设备、NuBus 插件和图形设备的单独的 I/O 结构。由于具有继承功能，所以每一个结构都为在其设备类别内发现的通用功能提供支援。其它的研制者然后可以根据这些相容的接口以应用于其它种类的设备。

最佳实施例采用了所述结构的概念，并将其应用于整个系统。对于商业或联合研制者、系统综合者或 OEM 而言，这就意味着有关诸如 Mac App 等的结构已经展示了所有优点，即不仅包括诸如文本和用户接口方面的有关应用层级的优点，而且还包括诸如图形，多媒体，文件系统， I/O 以及测试之类的系统级服务的优点。

在一个最佳实施例总体结构中的应用的建立很象写入一个遵守结构协议的特殊区域段。利用这种方式，编程的整个概念都政变了。取代逐行编写调用多 API 层次的代码，通过从在这种环境之中予先存在的结构获取类别，然后附加上新的特性和/或将所述的继承特性置换成所需特性来开发所述软件。

由此，研制者的工作就变成了收集编写的代码并与所有其它的结构应用共享这些代码。由于研制者将能够根据相互间的工作来生成代码，所以这个概念是相当有用的。这也使得研制者可以根据需要的多少来灵活地实现定制操作。某些结构将被用作其本身。在某些情况下，定制量将是最小化，所以研制者所插入的段将很小。在其它的情况下，研制者可以作非常广泛的修改并建立某些全新的设计。

在图 1 所示的最佳实施例中，一个多媒体数据路由选择系统通过计算机系统来管理所述多媒体信息的移动，而驻留在 RAM14 中的多媒体部件在 CPU10 控制下或通过总线 12 或通信适配器 34 与外部相关系以负责显示多媒体信息，不需要一个中心播放机去协同或管理系统的全部操作。这种体系结构提供了灵活性，并为一个新类型媒体的加入提供了增加的可扩展性。所述系统利用各种多媒体目标，其中的某些可用作连接目标。所述的连接目标包括增益，滤波器，放大器，混频器，播放机，和其它在面向目标操作系统中用作目标的多媒体部件。上述所讨论的目标包括一个代码或方法部件和一个数据部件。所述系统包括一个鼠标，以便于拖曳/投放，两次按动(double clicking)，投放启动，光标定位，以及其它典型操作的图符操作。

在视频和音频产生工作室中，诸如声音， MIDI ，和视频的媒体利用实际上的插入线为处于电源，效果处理器，和转接器之间的信号进行路由选择。信号处理算法也经常被表示成所述电阻，转接器和处理器的网络。这两种模型都可以被表示为被连接目标的导向图形。一个最佳实施例允许在个人计算机系统上实现把目标连接到一起模型。图 2 示出一个现有技术

的家用组合音响装置，它包括一个磁带机，一个混频器，一个混响单元，一对话筒机，和一对扬声器。由于所述的话筒被连接到所述磁带机上，所以声音输入的路径选择为从所述的话筒到所述的磁带机，在这里，所述的声音被记录下来。由于所述联接是由磁带机连接到混频器，所以当磁带机进行播放时，其信号将被传送给混频器。同样，混响单元和扬声器被连接到与混频器相连的放大器。

一个最佳实施例使用面向目标技术来表示一个连接模型。多媒体目标可以相互连接，从而生成直接的数据流图形。另外，一组标准的多媒体目标被规定给所述系统。这些目标可以被连接到一起，以便于多媒体数据从一个目标流到另一个目标。通过利用诸如线，线段或其它适当的几何结构的几何图形将多个媒体目标连接起来，以助于所述的连接操作。下面要讨论的图示出了各种多媒体目标的例子，包括连接目标和用于表示内部数据结构和连接多媒体目标的逻辑的几何图。

用于路由选择的类别

基于时间的媒体部件(此后称为媒体部件)的基本类别是用于路由选择的中心抽象。一个媒体部件具有零或多个输入端口和零或多个输出端口。在图 3 中，媒体部件具有一个单一的输入端口 300 和两个输出端口 310 和 320 。端口 300 、 310 和 320 用黑色的三角形来表示。

通过将媒体部件的端口相连接将所述媒体部件的子类别连接到一起。这种处理类似于在一个记录工作室内使用插入码将音频和视频部件相接到

一起。在图 4 中，一个媒体部件的子类别、即一个音频播放机部件目标被连接到另一媒体子类别,即一个扬声器部件目标上。所述的音频播放机具有一个输出端口，且所述扬声器具有一个输入端口。利用成员功能调用对所述的媒体部件进行控制。例如图 4 中的音频播放机就具有与播放音频数据相关的成员功能。当调用所述音频播放机的成员功能 Play( )时，音频数据将从所述的音频播放机传送给扬声器，从而可以从计算机的扬声器中听到所述的音频。由于所述的扬声器部件能够播放传送给它的任何音频数据，所以它不具有 Play( )功能。所述的媒体部件完全可以用软件实现。然而，所述的媒体部件也可表示一个实际的硬件。例如，一个扬声器部件可以用于表示一个计算机的播放硬件。在这种方式之下，诸如是一个视频磁带记录机，混频器和效果处理器的外部媒体设备可以被表示成媒体部件并可以被连接到一起。

连接媒体部件

通过连接媒体部件的端口可以将所述的媒体部件连接到一起。为了避免客户和部件把数据同时写入同一个端口，从而损害数据的完整性，客户机不允许对端口进行直接存取。客户机应当在多线索安全代用目标的基础上执行连接操作。根据上下文的关系，术语“代用” 一词涉及到一种允许多个客户机去安全共享所述类示例的基础类别的特殊表示。在所述媒体部件的情况下，代用端口允许对实际的端口进行有限的间接控制。每一个媒体部件都具有用于建立与其每个输入和输出端口相关的代用的成员功能。这些端口代用是一些极轻的目标并且非常适于通过地址边界，由此有助于连接驻留于不同地址空间的媒体部件。

每个端口和端口代用具有与其相关的数据类型。这些类型的例子是MIDI 数据， 44k 赫兹 16 位的音频数据， 22K 赫兹 8 位的音频数据和用于视频的图形数据。当两个端口被要求连接时，一个类型谈判协议保证所述端口能够支援兼容数据类型。假如所述端口不具有通用类型，就要产生一个异常。

在两个具有互不兼容端口类型的目标之间插入有转换器目标。所述转换器是一种取入一种数据而产生不同类型数据的部件。扇入和扇出端口的例子示于图 5 。可以选择多个特殊的子类别以禁止扇入和/或扇出。例如，音频端口既禁止扇入又禁止扇出，并试图通过将一个端口连接到一个以上的其它端口来产生所述的异常。扇入和扇出性质是利用特定的媒体子类别进行处理的。例如，在 MIDI 中，扇出将相同的数据传送给多个接收者，而扇入则从多个发送器中接收数据。

通过下述步骤将两个媒体部件 A 和 B 连接到一起:

1) 调用 A 的成员功能，以请求一个输出端口代用， Pa;
2) 调用 B 的成员功能以便要求 B 去请求一个输入端口代用， Pb ;
3) 调用 Pa 的成员功能 ConnectTo 并在 Pb 中作为自变量进行传送。

所述端口可以通过调用 Pa 的成员功能 DisconnectFrom 并在 Pb 中作为自变量进行传送而断开连接。当要连接或断开连接任一媒体部件时，无论端口类型是一个音频，图形， MIDI ， 或某些其它多媒体数据类型，都

会激发上述过程。当两个端口都被连接时，没有编译时间检查。例如，没有什么东西可以使软件研制者避免去编写，编译，和链接一个试图将音频端口与所述 MIDI 端口相连的代码。代之，一个异常在运行时间内被产生。这种特性被明显地设计进所述的路由选择系统以便使得可多形性地连接媒体端口。例如，一个插入模块可以在特定的连接时间把一对媒体端口连接到一起，而不需要对不同的端口类型进行特殊处理。利用一个插入器可以用完全相同的方式对音频，视频，和 MIDI 端口进行处理。多个目标间的连接可以用在表示诸如一个显示器上的图符的所述多媒体目标的标记之间划一个线段来可视地加以表示。

音频数据的路由选择

音频数据可在计算机上数字化，存贮，处理,和播放。一个计算机也可以用于合成和播放所述音频。一个模拟/数字转换器(ADC)被用于将一个模拟音频电子信号转换成一系列的称为数字音频取样的数码。所述音频通常以每秒 8000 次取样的取样频率被取样，以用于全程的电话质量音频;每秒钟 44,100 次取样，以用于小型磁盘(CD)质量音频和每秒钟 48,000 次取样，以用于数字音频磁带(DAT)。一旦成为数字形式，所述的音频就可以利用计算机加以存贮和处理。使用数/模转换器(DAC)将所述的数字音频取样转换回模拟音频电子信号。一个最佳实施例定义了用于为所述媒体部件之间的数字音频数据进行路由选择的端口目标（称为音频端口）的子类别。音频输出端口可以被连接到音频输入端口，以为音频数据进行路由选择。若一个媒体部件具有至少一个音频端口，它将被称之为一个音频部件。所有音频部件都是媒体部件基本类别的子类别。如图 6 所示，一个音

频部件具有零或多个音频输入端口或零或多个音频输出端口。通过连接所述音频部件的端口，可以将它们连接到一起。这类似于使用插入码去把利用硬件的立体部件连接到一起。

一个最佳实施例有助于说明所述的音频部件是如何连接以建立各种有趣的应用的。图 7，图 8，图 9 和图 10 示出了能够使用音频部件构成的某些举例应用。图 7 示出了使用一个音频播放机如何写入一个声音注释应用以及如何记录和播放所述的声音注释。一个电话听筒被用于输入和输出。图 8 示出了声音邮件/电话应答应用是如何构成的。一个音频播放机在电话线上播放一个致意输出，而另一个音频播放机记录一个输入信息。图 9 示出了一个音乐应用。作为一个特殊效果的回波被加到一个乐器的声音上并通过一个扬声器被播放出来。图 10 示出了如何将三种声音：一种是音乐，一种是声音效果，一种是画外音，混合到一起并将它们通过诸如扬声器的一个输出设备加以传送。

像所有的媒体部件一样，通过连接所述音频部件的端口便可将它们连接到一起。这个操作是通过选择一个音频部件端口，从一个部件端口向另一个多媒体部件端口延展诸如一条线的几何图形以及建立用于记住在所述显示器上几何表示的所述链接的数据结构来实现的。每一个音频部件都具有用于建立与其输入和输出端口相关的代用的成员功能。客户机通过根据每个部件申请的输入和/或输出端口代用，然后使用由所述代用目标所提供的成员功能来执行连接操作，从而把实际的输入端口连接到实际的输出端口。每个音频端口和端口代用具有与其相关的音频类型。图 11 列表了一些由所述最佳实施例支持的音频类型。

由所述显示器上的链接线段所表示的连接表示一个单一的音频通道。通过使用其中用于左通道而另一个用于右通道的两个连接来处理立体声。当有两个端口被要求连接时，一个谈判协议保证所述端口能够支援可兼容的数据类型。假如所述端口不具有通用类型，那么将产生一个异常。假如发生这种情况，一个音频类型转换器可以被插入到所述两个端口之间以将一种类型转换成另一种类型，如图 12 所示。在一个音频部件的网络中，不允许存在有回路(循环)。任何可能导致回路的连接企图都将导致产生异常。通过在诸如表示多媒体目标的标记的音频输入和输出端口的图形的图符之间延展诸如一个线段的几何图形，便可以在所述的显示器上几何地表示连接音频部件的处理。

实现音频部件

基于客户机--服务器的体系结构被用于实现音频部件。对于每一个音频部件而言，都具有一个驻留于称为音频服务程序的任务之中的音频处理器目标。所述的音频处理器目标负责执行实际的信号处理。所述的音频部件子类别必须写入一个音频部件和一个音频处理器子类别。客户机/服务器方案被用于加强性能。若在单一任务之下执行了所有数据的移动和信号处理，则可以最有效地实现音频的插接线结构。这就避免了不必要的交互处理通信(IPC)和上下文转换。若所述的客户机存在于多项不同的任务之中，就需要一个声音服务器把处理集中在一个单一任务中。

还可以有另外一种实现方法，这种方法根据每个音频部件的不同的任

务对信号进行处理。不需要用于音频处理的服务器。这是一种具有很多优点的替换，但遗憾的是它有一个缺点，即它完成数量级的速度要比本发明最佳实施例所进行的处理慢。在单处理机的情况下，即使是所述的机器具有加强的处理功率，这个比率也基本保持不变。

客户机/服务器模块

音频部件需要方便与它们相对应的音频处理器之间进行通信。一个远程过程调用接口(RPC)被用于完成此任务。通常，一个音频部件子类别的成员功能远程激活在所述音频处理器上的“真实的”成员功能。图 13 是一个方框图，它示出了根据最佳实施例的远程过程调用。

执行连接

与音频部件相反，音频处理器具有音频端口。音频部件具有音频端口代用。只要将在客户地址空间中的两个端口代用连接在一起时，在声音服务器地址空间中的相应音频端口就被连接到一起。所述的子类别不需要起任何使能作用，这些作用都是由所述结构完成的。

处理音频

音频端口之间的连接是利用音频数据缓冲器完成的。每个连接或“插入线”都具有与其相关的缓冲器。音频处理器可以对与该缓冲器的地址和规模相关的音频端口提出要求。图 14 是一个方框图，它示出了一个具有

相关 Run() 成员功能的音频处理器结构。所述的 Run() 成员功能用于从其输入缓冲器中读出数据，对这些数据进行处理以及将处理结果写入其输出缓冲器。所述缓冲器的规模是可变的，但其最佳规模是可容纳 5 毫秒取样。这个规模对于获取从起始到停止时间长达 10 分钟的性能目标是需要的。

基于帧的处理

所述的声音服务器使用称作基于帧的处理的技术来处理声音取样，所述的基本处理如下:

1) 排序音频处理器以便生产者早于消费者。

2) 对于每个音频处理器而言，调用它的 Run() 成员功能。只要当一个 I/O 设备请求数据时，重复这个步骤。

对于处理音频而言，由于一旦指定了所述的运行顺序，就能够通过调用 Run() 很容易地产生所述帧，所以这是一个极为有效的方式。由于它使用了能够在一个单一帧中进行完全处理的固定规模的缓冲器，所以，Run() 的执行是非常有效的。例如，参看图 15，该图示出了一个音频处理器网络。每个音频处理器或结点都被标有一个字母。

为了产生声音，所述的声音服务器执行下列步骤:

1) 排序音频处理器以便生产者早于消费者;

2) 在诸如是所述扬声器的音频 I/O 设备需要更多数据时运行它们。

图 16 示出了对于图 15 所示之网络这种处理是如何执行的。在所述第一个方框中排序所述的音频处理器，然后，以一个规定的顺序运行所述的处理器。同时，每个处理器都获得信息。在这个时间以前，没有处理器被运行。换言之，在所述处理器获得适当信息之前，所述处理器进入等待状态。

排序音频处理器

使用称为模拟数据流的技术来排序音频处理器。由于在大多数场合下，所述的排序都是由一个结构自动确定的，所以大多数子类别成员都不必耽心排序。模拟数据流以拓扑逻辑次序反复通过所述的网络，就像数据从一个处理器通向下一个处理器一样。作这项工作的目的是为了看一看哪一个音频处理器可以运行，而哪一些音频处理器不能够运行。运行的音频处理器被放入运行表中并在所述运行阶段被反复停止运行。

若满足下述条件，则一个音频处理器的结点将能够运行:

1：它的输入端口具有可获得的数据，以及

2：它的输入端口具有可以放置数据的地方。

每个音频处理器都具有一个 CanRun() 成员功能，若所述的的结点可以运行，则它返回到 TRUE 。 CanRun() 缺省执行使用上述法则，所以，若所述的音频处理器需要置换一个法则的话，一个子类别仅需要置换所述的 CanRun()。若一个音频处理器可以运行，那么它将模拟从所有输出端口输出的数据流。 Fire() 实际上恰恰在周转并调用每个输出端口上其它的

Fire()成员功能。输出端口的启动能够模拟所述端口输出的数据流，从而使得模拟数据在另一侧的输入端口处可用。输入端口具有一个IsAvailable()成员功能。当所述的端口具有可获得的模拟数据时，它返回到TRUE。图17示出了根据一个最佳实施例的音频端口。

由于对延迟的输入处于静止状态，所以延迟的优先级必须高于CanRun()。因此，所述的延迟必须仍然能够产生输出，直到所述的延迟用尽为止。所述的输入端口不再具有可获得的数据，但是所述的音频处理器必须仍然在运行。当没有输入可以适用并且在所述延迟期间内仍然有必须输出的数据时，必须修改CanRun()以使其返回到TRUE。即使是在数据流准则被满足的情况下，由于当所述播放机处于停止状态时，所述播放机不能运行，所以播放机也必须置换CanRun()。播放机必须修改所述法则，从而使得在所述播放机处于停止状态时，CanRun()总是能够返回到FALSE，而与所述的数据流准则是否得到了满足无关。音频处理器能够询问它的音频输出端口以确定从数据被写入所述端口的缓冲器开始到所述音频实际上被用户听到为止的延迟。这就允许音频处理器被同步到任一时基。图18，图19和图20示出了与音频处理器相关的逻辑流程。一个诸如是播放机的音频处理器确定哪一种处理是需要的并激励示于图18中的RequestOrder。

图18是一个流程图，它示出了根据一个最佳实施的用于调用一个顺序请求的详细逻辑。处理是在端1800处开始并直接通向功能块1810以使得运行表失效。通过丢弃启动一个多媒体目标，在一个多媒体目标上连扣两下鼠标或使用任意一个表示操作启动的其它图符操作可以使所述的运行

表失效。而后在判定块 1820 处执行一个测试以确定所述的音频处理器是否能够运行。假如是，那么在功能块 1830 处释放音频处理器，控制转向端 1850 。但是，若所述的音频处理器不能运行，那么在功能块 1840 中激活所述音频处理器，控制转向端 1850 。

图 19 示出了根据一个最佳实施例与激活一个音频处理器相关的递归逻辑的流程图。处理是在端 1900 处开始的，这里的控制来自功能块 1840 或由终端 1990 处产生的递归调用。在二者之一的情况下，控制立即去往功能块 1910 ，在这里，变址 i 为零，计数 N 被设置得等于音频处理器输出端口的数。然后在判断块 1920 执行一个测试，以确定是否计数 i=N 。如果是，控制返回到端 1930 。如果不是，所述输出端口在功能块 1940 处被启动。所述输出端口在功能块 1950 处被标记为已经启动。在功能块 1960 处把所述输出端口连接到音频处理器，在功能块 1970 增加计数 i ，并在功能块 1980 执行一个测试，以确定所述的音频处理器是否能够运行，通过端 1900 为图 19 所示之逻辑产生一个递归调用。如果不能运行，控制转向判断块 1920 以继续进行处理。

图 20 的流程图详细地示出了与释放所述音频处理器相关的逻辑。处理是在端 2000 处开始的，在这里，控制来自功能块 1830 或从端 2018 或 2080 产生的一个递归调用。在这两种情况下，控制都直接导向功能块 2004 。在功能块 2004 ，变址 i 等于 0 而计数 N 被设置等于音频处理器输出端口的数。然后，在判断块 1920 执行一个测试，以确定计数 i 是否等于 N 。如果是，控制转向功能块 2020 ，以重置变址和计数的初值。然后在判断块 2024 执行一个测试以确定是否 i=N ，如果不是，那么在功能块

2040 中，输入端口 i 被标记为没有被启动，在功能块 2050 获得连接到所述输入端口的音频处理器。在功能块 2060，计数增加，并且在功能块 2070 处执行一个测试，以确定所述的音频处理器是否能够运行。假如是，控制在端 2040 处返回。如果所述的音频处理器不能够运行，通过端 2080 为图 20 所示之逻辑提供一个递归调用。若所述的音频处理器能够运行，控制转向判断块 2024 以对变址进行再次测试。

若在判断块 2006 处变址 i 不等于 N，那么在功能块 2008 内所述输出端口被标记为 NOT 启动。在功能块 2010 处获得连接到所述输出端口的音频处理器，在功能块 2012 处计数增加，并在判断块 2014 处执行一个测试以确定所述的音频处理器是否能够运行。如果所述的音频处理器不可以运行，通过端 2018 为图 20 所示之逻辑提供一个递归调用。若不是这样，则控制转向判断块 2006 以再次测试所述变址并继续进行处理。

图 21 是一个流程图，它表示了根据一个最佳实施例与运行一个音频处理器相关的逻辑。处理开始于端 2100 并直接通向判断块 2104，以确定所述的运行表是否失效。若运行表没有失效，那么在功能块 2106 执行一个拓扑逻辑排序，从而对所述音频处理器根据其启动顺次进行排序。然后在功能块 2110，所述的排序表被标为有效并且控制转向功能块 2120。假如在判断块 2104 处所述运行表是无效的，那么控制转向功能块 2120，以复位所述变址 i 和重新对计数 N 置数。接着，在判断块 2122 中执行一个测试，以判断所述计数 i 是否达到了计数值 N。如果是，处理完成，控制转向端 2130。如果不是，音频处理器 i 如同在功能块 2124 中描述的那样运行，且变址 i 如在功能块 2126 中描述的那样增加。

视频和图形数据的路由选择

视频信息可以在一个计算机上被数字化，存贮，处理和播放。一个视频数字转换器被用于把一个模拟视频电子信号转换成一系列被称作一个帧的数字图象。每一秒钟被进行数字转换的帧的数量被称之为帧速率。每秒 15 帧或 13 帧是典型的帧速率。一旦处于数字形式，就可以利用计算机对所述的视频进行存贮和处理。通过在计算机屏幕上以原来的帧速率按顺序显示所述的数字图象，所述的视频就可以被播放出来。

一个图形目标是用于表示能够在计算机屏幕上进行显示的任一目标的基本类别。所述图形目标的子类别包括(但不受其限制)有多边形、曲线和数字图象。数字化视频的每一帧是一个数字图象，因此可以利用一个图形目标来表示。

图形输入和输出端口被用于对从一个媒体部件到另一个媒体部件的图形目标进行路由选择。当每一个视频帧都是一个图形目标时，数字视频可以这种方式进行路由选择。由于图形端口可以对任何图形目标进行路由选择，所以动画数据也能够使用图形端口进行路由选择。使用包含有图形端口的媒体部件，可以建立其间具有视频流的视频目标网络。利用这种方式，可以构成各种有趣的应用。图 22 示出了根据本发明一个实施例把一个视频数字转换器部件插入到观看者部件之中以在计算机的屏幕上进行显示的例子。

图 23 是一个较复杂的例子，它示出了根据一个最佳实施例如何将来自两个视频目标的图象在一个效果处理器中进行混合并在计算机屏幕上进行显示。借助于所述的端口代用将图形端口连接到一起。每一个视频部件都具有用于建立与其输入和输出端口相关的代理的成员功能。客户机通过申请来自每个部件的输入和/或输出端口代用执行所述的连接操作。然后，利用由所述代用目标提供的成员功能将实际输入端口连接到实际输出端口上。每一个图形端口和端口代用具有与其相关的图形类型。当两个端口被要求连接时，一个类型谈判协议保证所述的端口能够支援兼容的数据类型。若所述端口不具有通用类型，将产生一个异常。

所述连接视频部件的处理可以通过在表示视频输入和输出端口的标记之间延展诸如一个线段的几何图符在一个显示器上几何地加以表示。

数据的路由选择

通过指向图形目标的写指针和读指针可以使图形端口控制图形数据。用于写入的端口执行是完全同步的：所述图形输出端口的 Write() 成员功能被阻塞，直到所述图形目标被传送给接收机为止。由于性能方面的原因，所述的图形端口不使用拷贝语义。由于在这种操作之中包含有一定数量的处理器和存贮器，所以，应避免 RGB 位映象的拷贝，代之以将指向所述图形目标的指针从所述的图形输出端口传送至所述图形输入端口。如果一个共享存贮器被用于存贮正在被写入和读出的所述图形目标，或者如果在一个任务的整个地址空间中拷贝所述的图形目标，所述同步接口就在所述整个地址空间中起作用。在所述连接被破坏所引起的事件中，被封锁的

图形端口将不再被封锁。

图 24 示出了根据一个最佳实施例是如何使用所述图形端口的。 1) 在源媒体部件中的任务将一个图形目标写入到所述媒体部件的输出端口中; 2) 指向所述图形目标的指针被传送给所述目地媒体部件的一个已连接的输入端口; 3) 在所述目地媒体部件中的任务调用它的输入端口的 Read() 成员功能，并且被封锁、解封锁和阅读所述的图形目标指针; 4) 当这个任务完成了对所述图形目标的处理时，它调用所述目地媒体部件输入端口的 Next() 成员功能; 5) 源媒体部件的任务被释放，并从写调用返回。由于所述的目地已伴随它而结束，所以，可以安全地处置所述的图形目标。图 25 示出了一个表示与一个输出端口 Write 成员功能相关的逻辑流程图。指向图形目标的指针被归入这个成员功能中。处理是在端 2500 开始的，并直接通向判断块 2510 ，以确定所述的端口是否已被连接。如果所述的端口未被连接，那么，如在端 2520 所示，就要产生一个异常。如果所述端口已被连接，那么在判断块 2350 处执行一个检测，以确定所连接的端口不处于同一地址空间。若所述的端口不处于同一地址空间，那么在功能块 2540 处，整个的图形目标被拷贝到共享存贮器中。若所述地址处于同一地址空间，则指向所述图形目标的指针被拷贝到存贮器中。在这两种情况的任一种情况下，下一步都是要将一个标志信息如在功能块 2560 处所描述的那样传送给所述的输入端口，并封锁所述的任务直到如在功能块 2570 处所描述的那样的所述输入端口通知所述的端口处理已经完成为止，同时在端 2580 处结束。

图 26 示出了根据一个最佳实施例的输入端口的读处理。所述处理开

始于端 2600,并直接通往判断块 2610 以确定所述的端口是否已被连接。如果所述的端口未被连接，则所述结构在端 2620 处产生一个异常。如果所述端口已被连接，将在判断块 2630 处执行另处一个测试，以确定所述图形是否已以准备好。如果不是，执行一个块任务，直到如在功能块 2640 处所描述的那样准备好所述的目标，且控制转到功能块 2650 。如果所述的图形已经准备好，指针返回到功能块 2650 中的所述图形目标，控制通过端 2660 返回。

图 27 示出了根据一个最佳实施例的一个输入端口的下一个成员处理。所述处理开始于端 2700，并直接通往判断块 2710 以确定所述的端口是否已被连接。如果所述的端口未被连接，那么如在端 2720 处所示产生一个异常。如果所述的端口已被连接，那么如在功能块 2730 内所示要发送一个适当的标志信息，并且在判断块 2740 处要执行另一个测试，以确定已经被连接的端口是否处于同一地址空间。若所述的端口处于同一地址空间，那么如在功能块 2750 中所示的删除所述图形目标的拷贝，且所述处理在端 2760 处结束。

MIDI 数据的路由选择

如同在 MIDI Manufacturers Association Publication (1990) 的题目为“MIDI 1.0 Detailed Specification”中所讨论的那样，乐器数字接口(MIDI) 规定一个接口，用于在电子乐器、计算机、序列发生器、亮度控制器、混频器和磁带记录机之间交换信息。 MIDI 广泛地应用于记录工作室和日常演出之中，并在工作室记录、自动控制、音频视频生成以

及合成领域中具有极大影响。利用其本身和其它媒体相结合， MIDI 对于计算机在多媒体应用方面扮演着极其重要的角色。与数字式音频相比较，MIDI 文件占据更少的空间，且所述信息是便于控制和观察的符号。例如，典型的 3 分钟 MIDI 文件可以在盘上要求 30 到 60K 字节。另一方面，一个 CD 优质立体声文件可能每秒要求大约 200K 字节，或 3 分钟 36 兆字节。 MIDI 数据可以以音乐注释，图形钢琴滚动(graphic piano-roll)或适于对不同乐器进行编辑和再赋值的信息表的形式出现。一般的 MIDI 具有标准化的乐器分配，从而极大地鼓励多媒体产品的生产者。

MIDI 输入和输出端口被用于为时间特征 MIDI 信息包从一个媒体部件到另一个媒体部件进行路由选择。 MIDI 端口作为一个信箱区，用于跨越地址空间的 MIDI 信息包的通信。通过连接包含有 MIDI 端口的媒体部件，可以构成很多有趣的 MIDI 的应用。图 28 示出了两个部件，即一个 MIDI 播放机 2800 和一个 MIDI 接口 2810 是如何被用来演奏一个被连接到所述计算机上的音乐合成器的。所述的 MIDI 接口被用于连接诸如是一个音乐合成器的外部设备。所述的 MIDI 信息包被从 MIDI 播放机送到所述的 MIDI 接口。所述 MIDI 接口 2810 将所述的 MIDI 信息包转换成将要传送给所述音乐合成器以用于播放的 MIDI 数据。

图 29 示出了所述的 MIDI 数据是如何被记录和如何从一个外部音乐合成器中播放出来的。所述的 MIDI 端口 2910 具有一个用于根据从所述的音乐合成器接收的数据产生所述 MIDI 信息包的 MIDI 输出端口。所述的 MIDI 播放机具有一个 MIDI 输入端口，用于读取这些信息包并把它们存贮到计算机上。图 30 示出了 MIDI 数据是如何被播放 3000，滤波 3010 和传

送给一个外部音乐合成器 3020 的。滤波器 3010 能够在其输入端 3000 执行一个操作，并将操作结果传送给它的输出端 3010 。多个特定的滤波器可以被写入，以例如增加额外的注释，从而建立 MIDI 回波、延迟，或者截去音调带以减少带宽负载。

图 31 示出了包含有 MIDI 和音频端口的媒体部件。以软件为基础的音乐合成器从它的输入端口读取 MIDI 信息包并输出表示在其输入端上读入的音符的数字音频。借助于所述的端口代理将 MIDI 端口连接到一起。每个 MIDI 部件具有多种成员功能，以用于建立和其输入输出端口相关的代理。通过向每一个部件申请输入和/或输出端口代理并利用代理目标提供的成员功能，客户机执行连接操作，把实际的输入端口连接到实际的输出端口上。每一个 MIDI 端口和端口代理都具有与其相关的 MIDI 类型。当要求连接两个端口时，一个类型谈判协议保证所述的端口能够支援兼容的数据类型。若所述端口不具有通用类型，将产生一个异常。

通过在代表 MIDI 输入和输出端口的标记之间延展诸如是一个线段的几何图形，可以几何地表示连接 MIDI 部件的处理。

MIDI 信息包

用于通过传送和接收信息支援 MIDI 彼此相互通信的部件。 MIDI 标准定义了两种信息：通道信息和系统信息。所述的通道信息被进一步分成声音信息和模式信息。系统信息还进一步被分成通用，实时和互斥信息，如图 32 所示。通道声音信息包括一个 MIDI 设备能够听到的通道号(0-15)。

通道模式信息在基本通道上进行传送，以确定一个乐器对于所述通道声音信息的响应。系统通用信息送往所有的接收机，系统实时信息将同步信息载送给时基乐器。系统互斥信息允许制造者在标准规定之外提供 MIDI 支援。除了可以有选择地归入状态字节的同类连续信息(运行状态)以外，所有的信息都可以用一个状态字节开始。除了所述系统互斥以外的所有信息都具有 0 、 1 或 2 个数据字节。所述的系统互斥信息由任意数量的数据字节组成，并可由 EOX 字节结束。图 33 示出了根据最佳实施例的 MIDI 信息的格式。

MIDI 信息包封装所述的标准

MIDI 包目标封装所有的 MIDI 信息类型及结构。另外，如图 34 所示，所有的 MIDI 包目标都具有一个状态字节和一个时间标记。通过定义具有方便结构和存取成员功能的信息类型， MIDI 信息包的子类别反映了所述的 MIDI 协议。

MIDI 端口

MIDI 端口用于在媒体部件之间交换 MIDI 信息包。一个 MIDI 输出端口能够写一个 MIDI 信息包，而一个 MIDI 输入端口能够读一个 MIDI 信息包。一个输出端口可以被连接到具有一个代用端口的输入端口上。所述的代用端口不能读或写 MIDI 信息包，它们是一些能够通往其它地址空间以支援连接的被动目标。提供成员功能以便在某一时刻将一个或多个信息写给一个输出端口。类似地，当从一个输入端口读出时，所有被缓存信息的

下一个信息或下一个计数都可以被请求。在所述缓存器未被腾空之前，禁止读出，并且利用另一个任务来取消被禁止的 Read 调用。被写入一个输出端口的所述信息包的拷贝从一个输入端口被读出。

信息包按出现的次序被读出并且不依时间标记而排序。假如需要两个数据流的时序归并程序，那么将需要一个排序归并目标。为了理解为什么一个输入端口不被排序，就要首先回忆一下其时间既可以朝前以可以朝后流动的信息包。考虑包括有以 6， 8 和 10 秒这个顺序所产生事件的特殊重放的情况。假如时间从 5 开始直到 11 周转并返到到 5，那么事件的次序将是 6， 8， 10， 10， 8， 6 而不是 6， 6， 8， 8， 10， 10。由于时间是在两个方向上行进，因而使得所述的排序不足以用于缓存所述的信息包，因而它是不精确的。

所述连接的扇入和扇出被支援，扇入和扇出提供了一种能力，即它可以使一个 MIDI 输出端口被连接到一个以上的 MIDI 输入端口，并且可以使得一个输入端口具有连接到一个以上的 MIDI 输出端口。图 35 示出了根据一个最佳实施例的一个扇形展开操作。

实施 MIDI 端口

一个最佳实施例使用了一个被连接端口的表，以保持追踪 MIDI 端口的连接，并使用了一个共享缓冲器，以用于传送和接收信息包。每个端口都保持有一个与其相连接的所有其它端口的表。这个表利用 Connect 和 Disconnect 调用进行修改，并且当所述端口被破坏时，利用这个表通知

所有与其相连的端口该端口已不复存在。当事实上这个输入端口被破坏时，这种处理避免了一个输出端口在其连接表内含有一个输入端口并试图向该输入端口写入。由此，一个输出端口保持有一个用于执行 Write 调用的输入端口表，并且它向其表中的每一个端口进行写入。而一个输入端口保持有与其相连的输出端口的表，并且当其脱离时，这些端口必须被告知。一个共享缓冲器支援生产者和消费者之间的协议。当所述的缓冲器是空的时，一个任务可以被中断，直到它不空为止。随后被置入其缓冲器的另一项任务可以通知被中断的任务它已被写入。替换之，其它的任务也可以取消所述的中断，以产生另一种类型的通知。

图 36 示出了一个 MIDI 输出端口的 Write ()成员功能。所述处理开始于端 3600，并直接通往功能框 3610 以对计数器置初值并限制用于一个回路的值。然后，在判断块 3620 处执行一个测试以确定所述的计数是否已经达到了所述的限定值。如果所述的计数已经达到了限定值，那么在端 3670 处理完成。若所述的计数值未达到所述的限定值，那么在功能块 3630 处，所述信息包的拷贝被插入一个端口的缓冲器，并且在判断块 3640 处执行一个测试，以确定所述的缓冲器是否具有一个消息。如果所述缓冲器不是空的，那么将如功能块 3650 所示发送一个适当的消息。若不是这种情况，就不必传送消息。在这两种情况的任一种情况下，都要在功能块 3660 处增加所述的计数，并且控制通过端 3670 返回。

图 37 示出了一个 MIDI 输出端口的 Read ()成员功能。处理开始于判断块 3700，以确定所述缓冲器是否是空的。若是，所述任务被中断，直到所述缓冲器包含有信息，或者是在功能块 3710 处产生一个取消。在判

断块 3730 处执行一个测试确定所述的等待是否已被取消。若是，则在端 3750 处产生一个异常。若不是或者是在判断块 3700 处所述缓冲器不是空的，则在功能块 3720 处将信息包从所述缓冲器拷贝到调用程序中，并通过在端 3740 处的返回完成了所述处理。

抽象多媒体部件

媒体部件

被称之为媒体部件基本类别的基于时间的媒体部件是用于路由选择的主要抽象。一个媒体部件具有零或多个输入端口和零或多个输出端口。例如在图 38 中，所述的媒体部件具有一个单一的输入端口和两个输出端口。

媒体顺序

媒体顺序是表示媒体内容（包括音频顺序）的抽象基本类别。媒体顺序的子类别用于表示音频、视频和 MIDI 的削减。媒体顺序的特征在于宽度和类型表。由浮点值所表示的宽度指出所述的数据有多长。所述数据还被类型化以指出例如视频或音频等的哪一种类型的声音由所述数字所表示。可以用一个子类别去支援多种类型。例如一个音频子类别可以利用线性形式和压缩形式等两种形式提供数据。由于这种可能，所以媒体顺序具有一个类型表。

播放机

图 39 示出的一个基于时间的媒体播放机(player)基本类别的例子是一个能够演奏时基媒体顺序(称之为媒体顺序)的媒体部件。一个媒体顺序是一个能够被用于表示音频，视频，和动画制作或 MIDI 数据的削减的抽象基本类别。所述时基媒体顺序的子类别被用于执行音频，视频，动画制作，以及 MIDI 数据。一个播放机类似于一个磁带记录机，而一个媒体顺序类似于一个卡型盒式磁带。一个播放机具有用于演奏所述媒体顺序的 play()成员功能，具有用于将其记录到所述顺序中的 Record()成员功能以及用于停止播放或记录的 Stop()成员功能。它还具有一个用于在一个顺序中寻找一个位置的 Seek()成员功能以及一个允许所述播放机被同步到其它播放机或软件时钟上的成员功能。所述的播放机与数据分离后，该播放机可以被重复使用。在演奏了一个盒带以后，一个播放机可以被构成以演奏或记录另外一个盒带。

标准的音频部件

音频播放机

图 40 示出了一个用于使一个音频部件播放和记录音频数据的音频播放机。该音频播放机具有一个相关的基本类别。所述的音频数据被存贮于是媒体顺序的子类别并称为音频顺序的目标中。音频播放机类似于一个磁带记录机，所述的音频顺序类似于一个卡型盒式磁带。音频播放机是播放机的子类别。同所有播放声音，视频，或 MIDI 的播放机一样，一个音频

播放机具有 play() 成员功能，所以声音可以被听到。它还具有一个 Record() 成员功能，所以所述的顺序可以被记录(如果允许写的话)，同时还具有一个 Stop() 成员功能，用以停止记录和演奏。它还具有 Seek() 成员功能，用以对声音中的一个点进行随机存取。另外，它还具有允许所述的播放机被同步到其它媒体播放机或一个软件时钟上的成员功能。

扬声器

图 41 示出了一个扬声器部件的例子。扬声器部件是一个与音频输出设备相关的抽象基本类别。一个系统可以具有一系列与其相连的声音输出设备。一个扬声器部件可以被子分类，以表示所述输出设备的各种特性。例如，存在一个扬声器部件的子类别用于允许使用电话送受话器或电话线进行播放。一个扬声器部件是一个逻辑输出设备。可以有很多扬声器部件的示例存在，所有这些部件都可被混合到一起并在计算机的实际扬声器上放出。一个立体声输出设备由所述扬声器部件的两个子类别表示，其一用于左声道，而另一个用于右声道。

话筒

图 42 示出了根据一个最佳实施例的话筒部件。一个话筒部件是一个表示诸如话筒或线路输入的声音输入源的抽象基本类别。一个系统可以具有与其相连的一系列声音输入设备。话筒部件可以被子分类以表示这些设备。例如，所述话筒部件的子类别被用于从所述电话送受话器或电话线上进行记录。同所述扬声器部件一样，所述的话筒部件是一个逻辑设备。可

以有很多话筒部件的示例存在，所有的这些示例都产生来自与相同实际话筒相同的声音。一个话筒部件具有一个输出。由于话筒部件是一个数据源，所以，必须通过调用 Start() 来明白无误地启动它。它可以通过调用 Stop() 加以停止。利用两个话筒部件的子类别来表示立体声输入设备，其一是左声道，其二是右声道。

混频器

图 43 示出了一个混频器部件的例子。混频器部件是一种把两个或多个音频输入总和成一个音频输出的抽象基本类别。输入的数量是由对所述构成设备的输入所确定的。

分离器

图 44 示出了一个分离器部件的例子。所述的分离器部件是一个抽象基本类别，它仅有一个输入，并将该输入分离成两个或多个输出。所述输出的数量取决于什么传送给了所述的构成设备。

增益

图 45 示出了一个增益部件的例子。增益部件是一个抽象基本类别，它能够增加或衰减一个信号的幅值。它被用于音量控制并具有 SetGain() 和 GetGain() 功能。一个增益部件将每一个输入声音取样数学地乘以增益值，并将其结果传送给它的输出端。增益值为 1.0 基本不能改变所述信号，

而增益值为 2.0 则使其音量增大两倍,增益值为 0.5 则使其信号的音量变成原来的一半。太高的信号电平则受到削减。

回波

图 46 示出了根据一个最佳实施例的一个回波部件。一个回波部件是用于将一个回波加到其输入端并在其输出端上产生结果的抽象基本类别。一个回波部件具有三个可以选择的参数:延迟长度,反馈,和混合速率。较长的延迟长度使得所输入的声音很象它处于一个大峡谷之中,而短的延迟能够产生类似于喷气飞机猛冲的卷边(flanging)效果;反馈确定所听到的回波的数量;而混合速率确定有多少回波的信号被混合回来。

抖颤(Fuzz)

图 47 示出了一个根据最佳实施例的抖颤部件。抖颤部件是一个把抖颤加入到一个声音内的抽象基本类别。这个效果在吉它或乐器声音上最有用了。

音频类型转换器

图 48 示出了一个音频转换器的例子。音频类型转换器是一个把一种音频类型转换成另一种音频类型的抽象基本类别。

音频多路转换器

图 49 示出了根据一个最佳实施例的一个音频多路转换器的例子。一个音频多路转换器部件把多种音频类型根据一个特定选择转换成其它的多种音类型。

声音

图 50 示出了根据一个最佳实施例的一个声音部件。声音部件是用于方便地记录和播放声音的目标部件。图 51 示出了嵌入所述声音部件中的多个部件。声音部件 5100 包括有一个话筒部件 5110，一个(用于控制输入电平的)增益部件 5120、一个音频播放机部件 5130、(用于控制输出电平的)另一增益部件 5140 和扬声器部件 5150。给出一个声音文件名，声音部件将自动地建立正确的音频顺序，且所需的音频部件就能演奏和记录所述文件。

实际的扬声器

图 52 示出了根据一个最佳实施例的一个实际的扬声器部件。一个实际的扬声器被用于所述计算机的输出放大器和扬声器的代用。它被用于控制整个计算机的音量。如果所述计算机还有另外的声音输出设备，那么将存在有实际扬声器的相应子类别。实际扬声器部件具有 SetVolume() 和 GetVolume() 成员功能，用于建立和获得整个音量级。

实际话筒

图 53 示出了根据一个最佳实施例的一个实际话筒的例子。一个实际话筒部件表示与话筒或线路输入相关的实际硬件。可以设置进入计算机的整个输入电平。

标准的视频部件

图形播放机

图 54 示出了根据最佳实施例的一个图形播放机部件。一个图形播放机部件是一个抽象基本类别，该类别涉及到能够演奏和记录随时间而变化的图形目标的时间图形目标的部件。图形顺序是时间图形的子类别。一个图形顺序是图形目标的顺序序列。其中，每一个图形目标都具有一个时间宽度。一个图形播放机类似于一个视频磁带记录机，而一个图形顺序类似于一个视频卡式盒带。图形播放机是播放机的子类别。像所有的用于播放声音，视频或 MIDI 的播放机一样，一个图形播放机具有 Play() 成员功能，所以，能够看到所述的图形目标；图形播放机还具有 Record() 成员功能，所以能够记录所述的顺序(若果允许写)；它还具有 Stop() 成员功能，所以能够停止记录或播放；它还具有 Seek() 成员功能，可以对所述顺序中的一个点进行随机存取；同时，它还具有允许所述播放机被同步到其它媒体播放机或一个软件时钟的成员功能。

图形观察器

图 55 示出了根据一个最佳实施例的一个图形观察器部件的例子。图形观察器部件被用于在计算机显示器上观察图形目标。一个图形播放机必须被连接到一个图形观察器上，以便观看播放的图形目标。由于使用了对消费者电子视频的模拟，所以，一个图形播放机就像一个视频磁带记录机，而图形观察器就像是一个视频显示监示器。

视频数字转换器

图 56 示出了根据一个最佳实施例的一个视频数字转换器的例子。视频数字转换器通过一个连接到所述计算机上的硬件视频数字转换器把模拟视频转换成图形目标。视频数字转换器可与图形播放机连接以记录输入到所述计算机的模拟视频。由于模拟了消费者的电子视频，所以，所述的视频数字转换器类似于一个视频摄影机，而图形播放机则类似于一个视频磁带记录机。

标准的 MIDI 部件

MIDI 播放机

图 57 示出了根据一个最佳实施例的 MIDI 播放机部件。 MIDI 播放机部件是一种部件的抽象基本类别，所述部件能够演奏和记录 MIDI 顺序。MIDI 顺序是 MIDITrack 的集合， 一个 MIDITrack 是一个顺序序列的 MIDI

信息包。 MIDI 播放机是播放机的子类别。像所有能够播放声音、视频或 MIDI 的播放机一样，一个 MIDI 播放机具有 Play ()成员功能。所以，MIDI 信息包可以被传送给一个外部音乐合成器；它具有 Record ()成员功能，所以能够记录来自外部键盘的 MIDI 信息(假设写是被允许的)；它还具有 Stop ()成员功能，用于停止记录或播放；它还具有 Seek ()成员功能，用于对所述顺序中的一个点进行随机存取。同时，它还具有允许所述播放机被同步到其它媒体播放机或一个软件时钟上的成员功能。

MIDI 接口

图 58 示出了根据一个最佳实施例的一个 MIDI 接口部件的例子。MIDI 接口部件既能将 MIDI 信息包传送给一个外部音乐合成器又能接收它们。一个 MIDI 播放机必须被连接到一个 MIDI 接口上，以便演奏或记录 MIDI 数据。

MIDI 滤波器

图 59 示出了根据一个最佳实施例的一个 MIDI 滤波器部件的例子。MIDI 滤波器部件是一个与具有一个 MIDI 输入端口和一个 MIDI 输出端口的目标的抽象基本类别。子类别提供了一种将输入数据转换成输出数据的变换算法。

MIDI 映象器

图 60 示出了根据一个最佳实施例的一个 MIDI 映象器部件的例子。所述 MIDI 映象器部件是使用一个字典将输入 MIDI 信息包映象给输出 MIDI 信息包的 MIDI 滤波器的一个子类别。所述字典内的每个入口都是一对 MIDI 信息包，即一个输入信息包(称为键入)和一个输出信息包(称为值)。进入所述 MIDI 映象器的输入 MIDI 信息包被用作查寻所述字典的键入，查寻的结果就是要从输出端口输出的输出 MIDI 信息包。

MIDI 程序映象器

图 61 示出了根据一个最佳实施例的一个 MIDI 程序映象。MIDI 程序映象器部件是 MIDI 映象器的一个子类别。它把 MIDI 程序变化信息转换成另外的 MIDI 程序变化信息。它能够用于把任一乐器映象到任一其它乐器上。

MIDI 注释映象器

图 62 示出了根据一个最佳实施例的一个 MIDI 注释映象器部件。MIDI 注释映象部件是 MIDI 映象器的子类别，它转换 MIDI 注释通和 MIDI 注释断，从而使所述注释可以被调换。

MIDI 通道映象器

图 63 示出了根据一个最佳实施例的一个 MIDI 通道映象器部件。MIDI 通道映象器部件是 MIDI 的映象器的子类别。它转换 MIDI 通道声音和模式信息，并能用于通用目的的通道变化。

虽然本发明是根据最佳实施例而予叙述的，但本发明领域内的一般技术人员可根据本发明的原理进行不同的变型及改变。因此，本发明的范围受权利要求的保护。

说 明 书 附 图

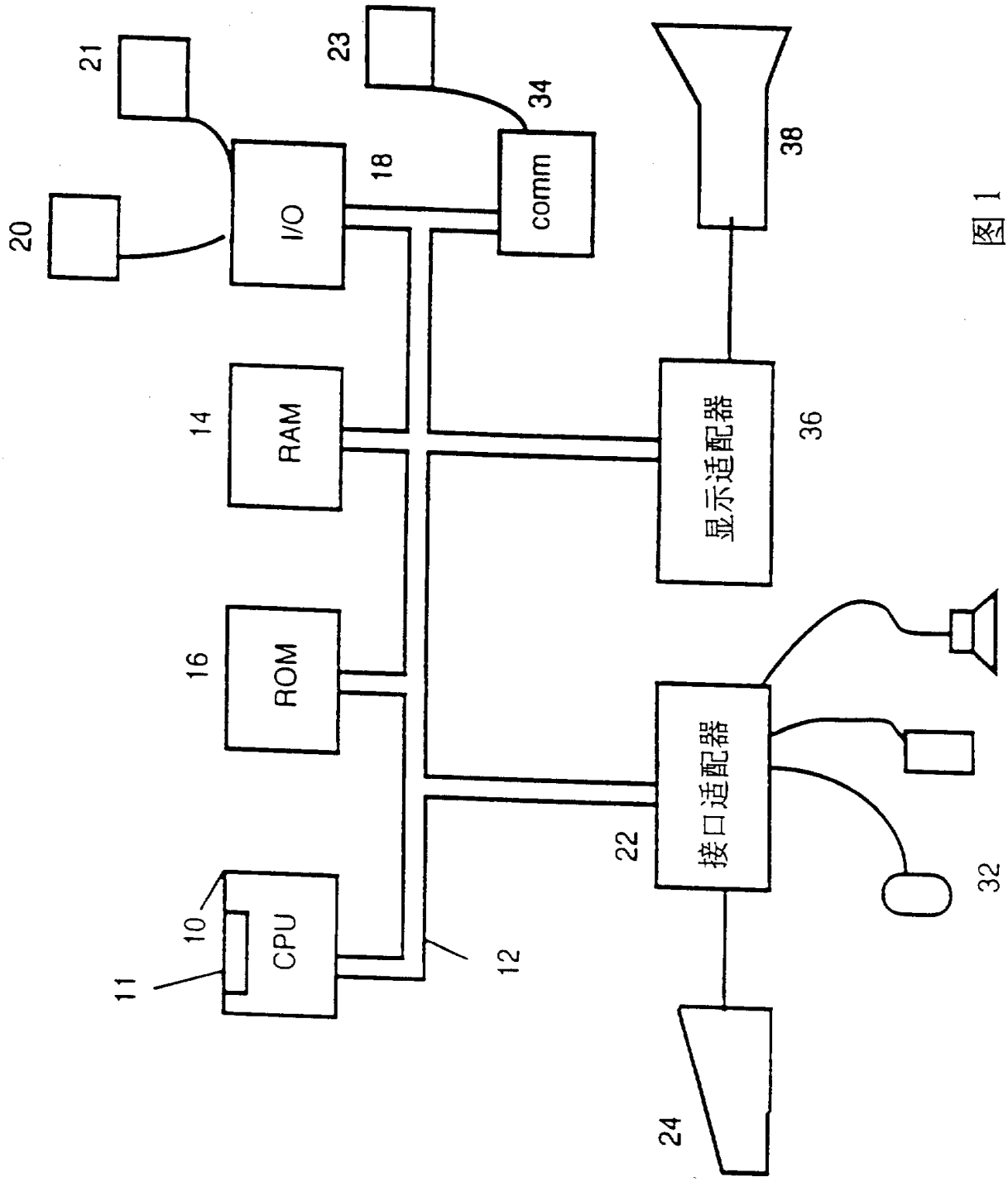

CPU
ROM
RAM
I/O
comm
显示适配器
接口适配器
10
11
12
14
16
18
20
21
22
23
24
26
28
32
34
36
38

图 1

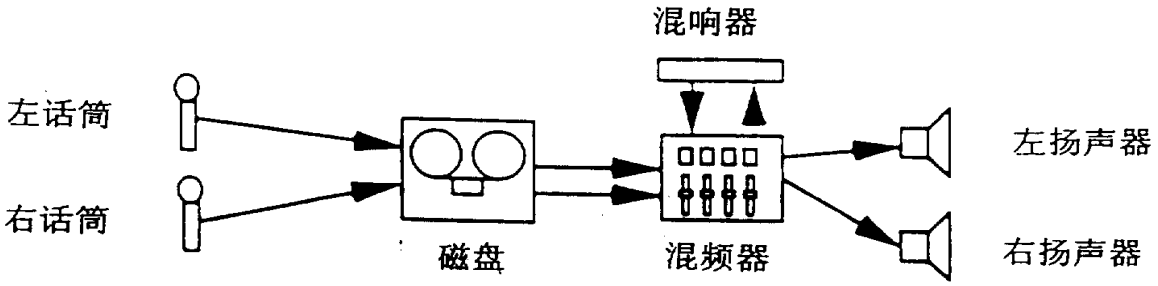


图 2

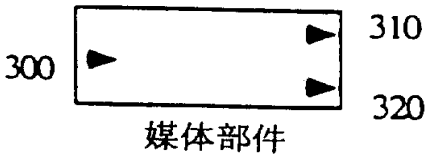


图 3

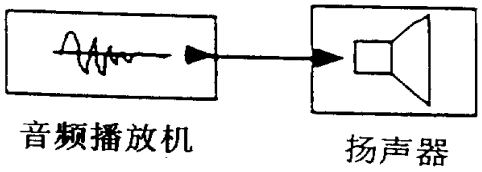


图 4

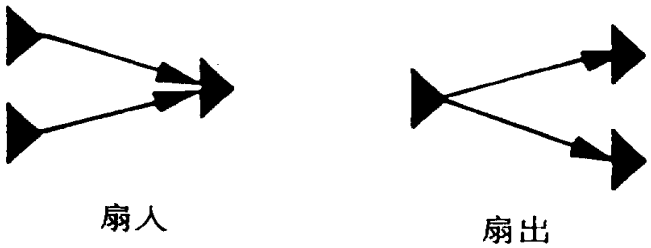


图 5

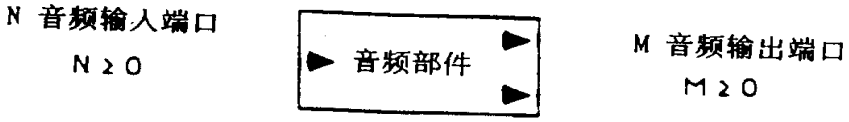

N 音频输入端口
N ≥ 0
音频部件
M 音频输出端口
M ≥ 0

图 6

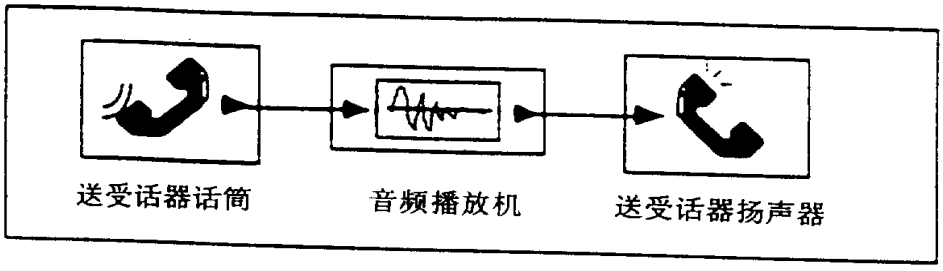

送受话器话筒
音频播放机
送受话器扬声器

图 7

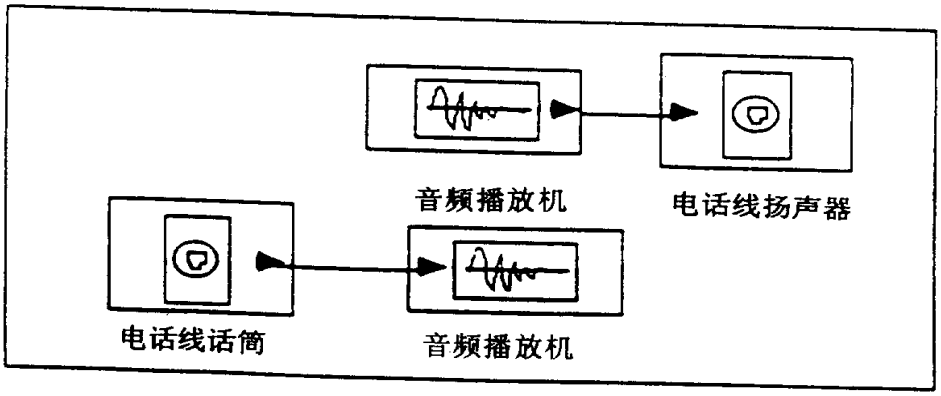

音频播放机
电话线扬声器
电话线话筒
音频播放机

图 8

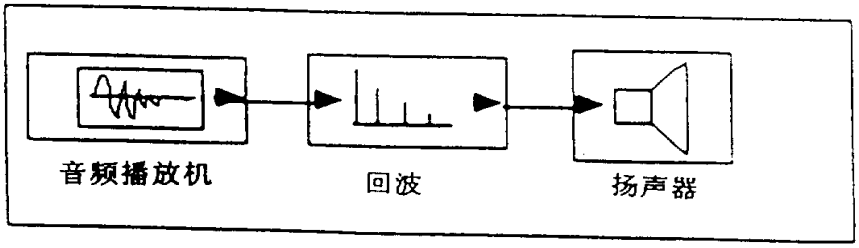


图 9

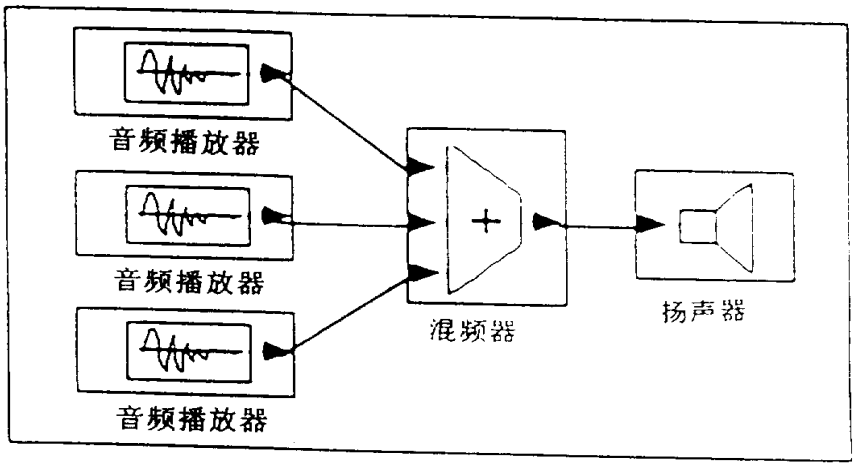


图 10

音频类型	取样速率（取样/秒）	取样宽度（字节）	说明
K44KHz16BitLinear	44100	16	小型磁盘质量 两个部件整体取样
K22KHz8bitOffsetBinary	22050	8	媒体质量，编制二进制（128是静）取样
K8KHzMuLaw	8000	8	北美电话标准 编码取样
K8KHzALaw	8000	8	欧洲电话标准 编码取样

图 11


K8KHzMuLaw
K8KHzMuLaw
K22KHz8bitLinear
K22KHz8bitLinear
音频播放机
音频类型转换器
扬声器

图 12

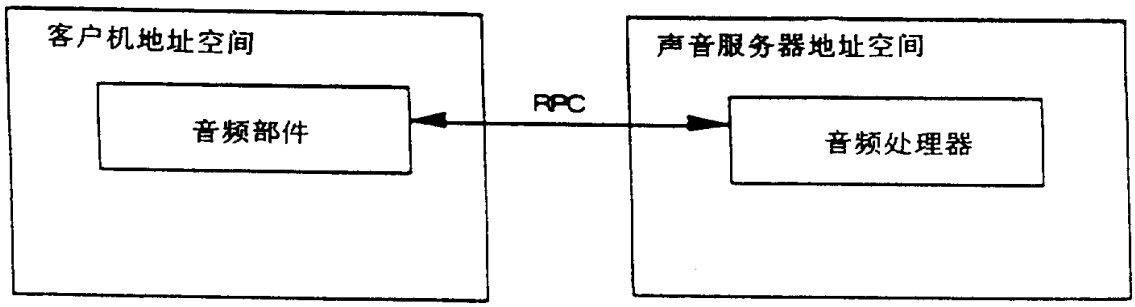

客户机地址空间
音频部件
RPC
声音服务器地址空间
音频处理器

图 13

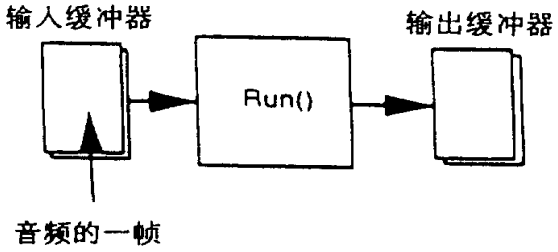

输入缓冲器
Run()
输出缓冲器
音频的一帧

图 14

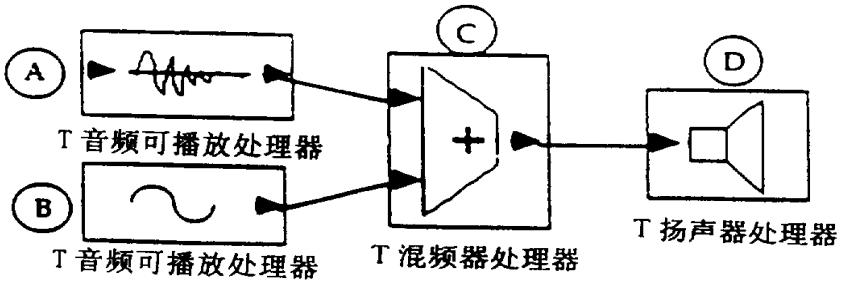

A
B
C
D
T 音频可播放处理器
T 音频可播放处理器
T 混频器处理器
T 扬声器处理器

图 15

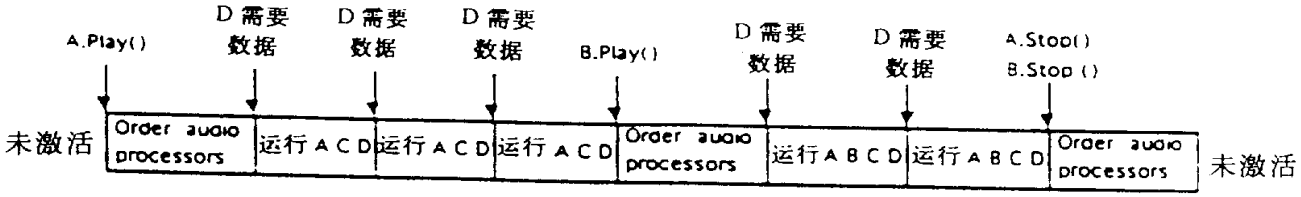

A.Play()
D 需要数据
D 需要数据
D 需要数据
B.Play()
D 需要数据
D 需要数据
A.Stop()
B.Stop ()
未激活
Order audio processors
运行 A C D
运行 A C D
运行 A C D
Order audio processors
运行 A B C D
运行 A B C D
Order audio processors
未激活

图 16

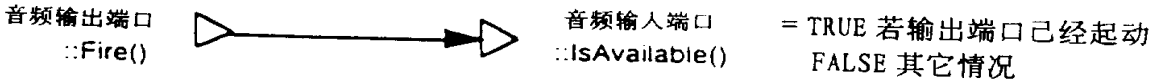

音频输出端口
::Fire()
音频输入端口
::IsAvailable()
= TRUE 若输出端口已经起动
FALSE 其它情况

图 17

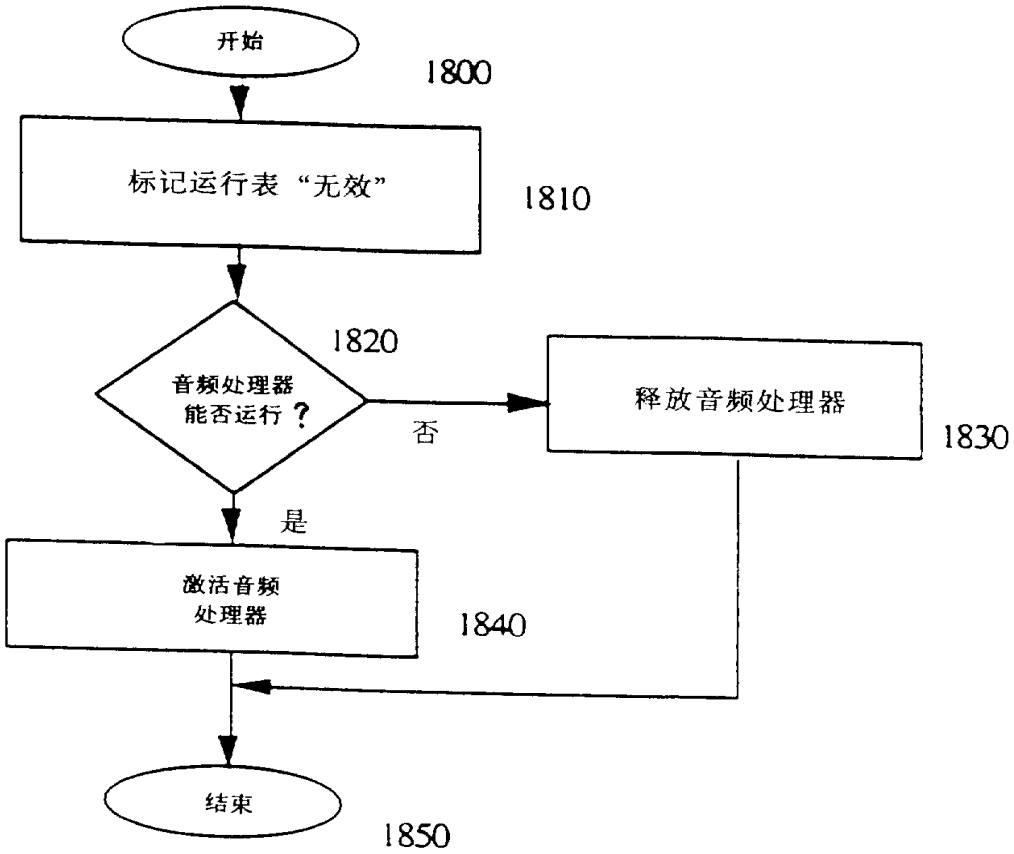

开始
1800
标记运行表“无效”
1810
1820
音频处理器
能否运行？
否
释放音频处理器
1830
是
激活音频
处理器
1840
结束
1850

图 18

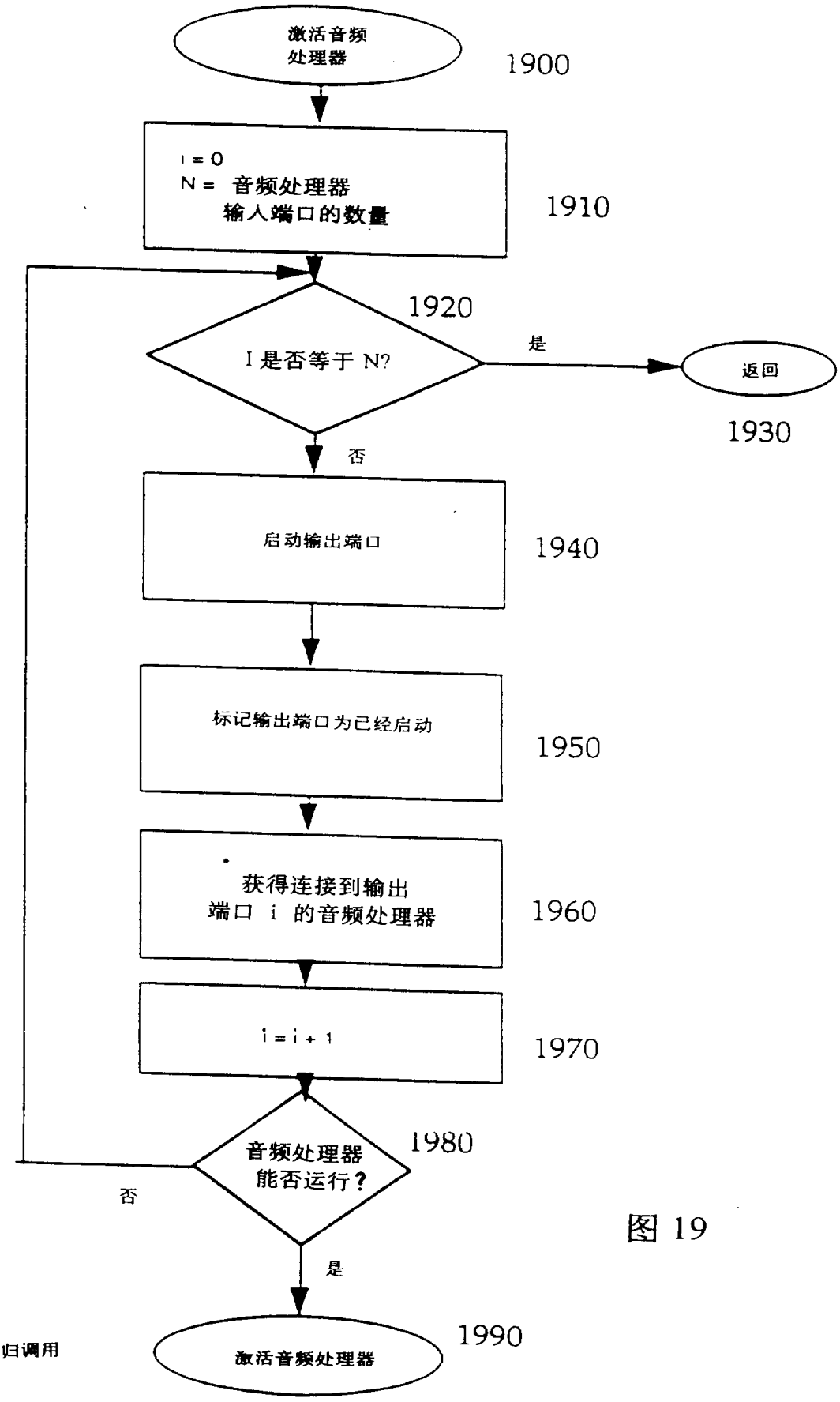

激活音频
处理器
1900
i = 0
N = 音频处理器
输入端口的数量
1910
1920
I 是否等于 N?
是
返回
1930
否
启动输出端口
1940
标记输出端口为已经启动
1950
获得连接到输出
端口 i 的音频处理器
1960
i = i + 1
1970
音频处理器
能否运行?
1980
否
是
递归调用
激活音频处理器
1990

图 19

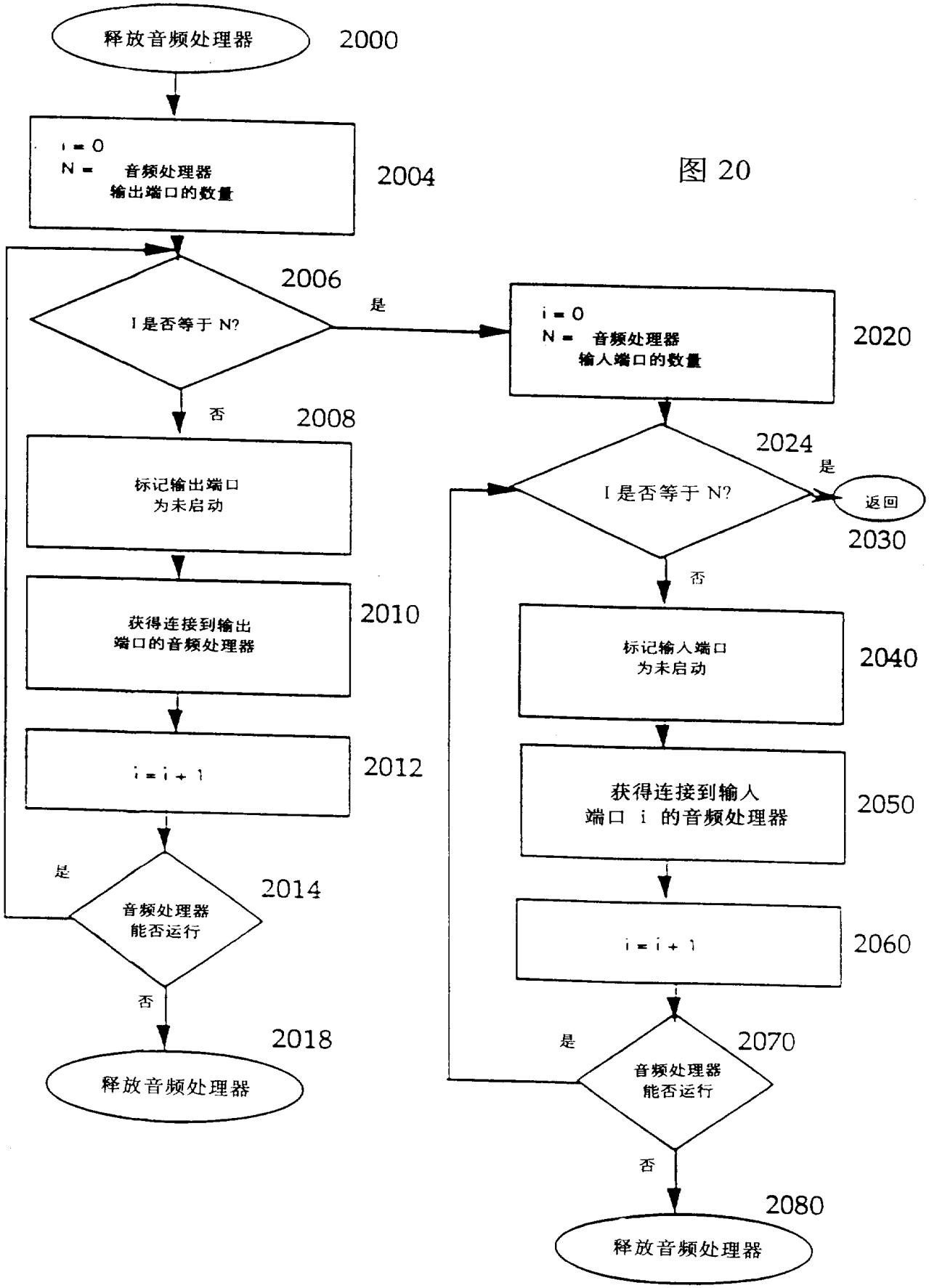

释放音频处理器
2000
i = 0
N = 音频处理器
输出端口的数量
2004
2006
I 是否等于 N?
是
否
2008
标记输出端口
为未启动
获得连接到输出
端口的音频处理器
2010
i = i + 1
2012
是
音频处理器
能否运行
2014
否
2018
释放音频处理器
i = 0
N = 音频处理器
输入端口的数量
2020
2024
I 是否等于 N?
是
返回
2030
否
标记输入端口
为未启动
2040
获得连接到输入
端口 i 的音频处理器
2050
i = i + 1
2060
是
音频处理器
能否运行
2070
否
2080
释放音频处理器

图 20

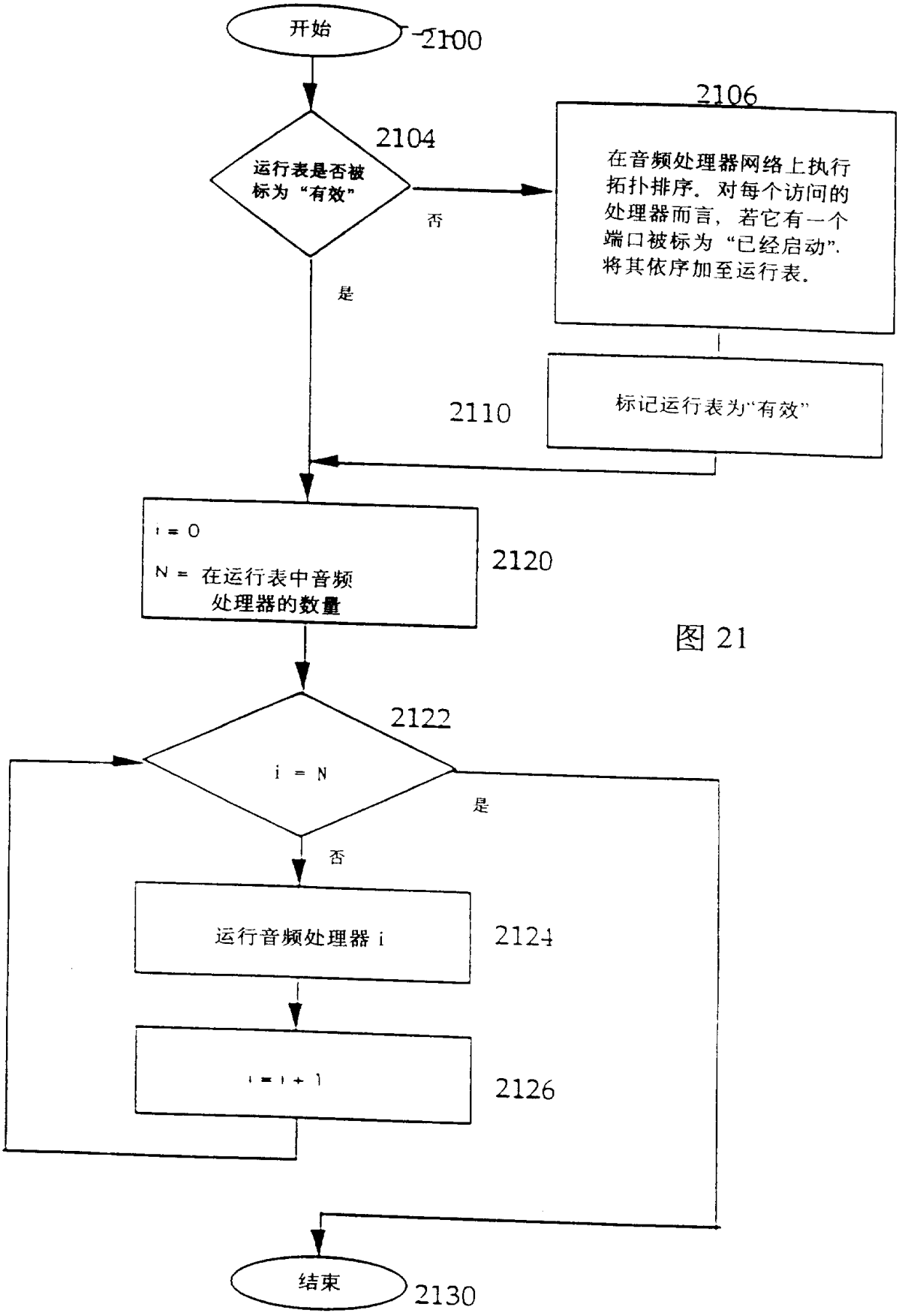

开始
2100
2104
运行表是否被标为“有效”
否
是
2106
在音频处理器网络上执行拓扑排序。对每个访问的处理器而言，若它有一个端口被标为“已经启动”，将其依序加至运行表。
2110
标记运行表为“有效”
i = 0
N = 在运行表中音频处理器的数量
2120
2122
i = N
是
否
运行音频处理器 i
2124
i = i + 1
2126
结束
2130

图 21

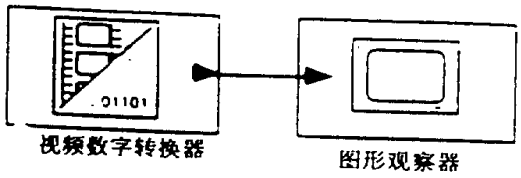

视频数字转换器
图形观察器

图 22

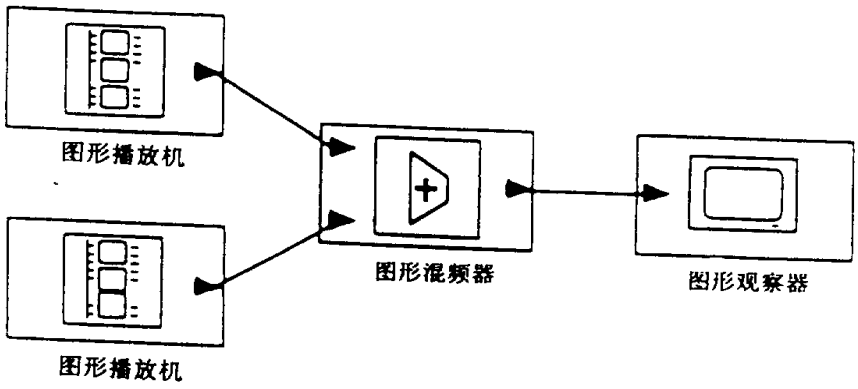

图形播放机
图形播放机
图形混频器
图形观察器

图 23

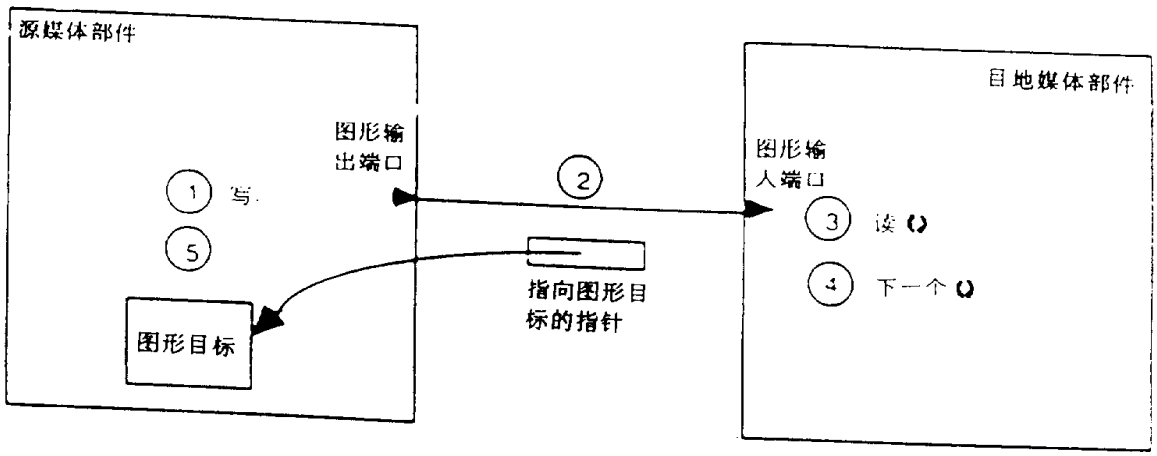

源媒体部件
图形输出端口
1 写
5
图形目标
2
指向图形目标的指针
目地媒体部件
图形输入端口
3 读()
4 下一个()

图 24

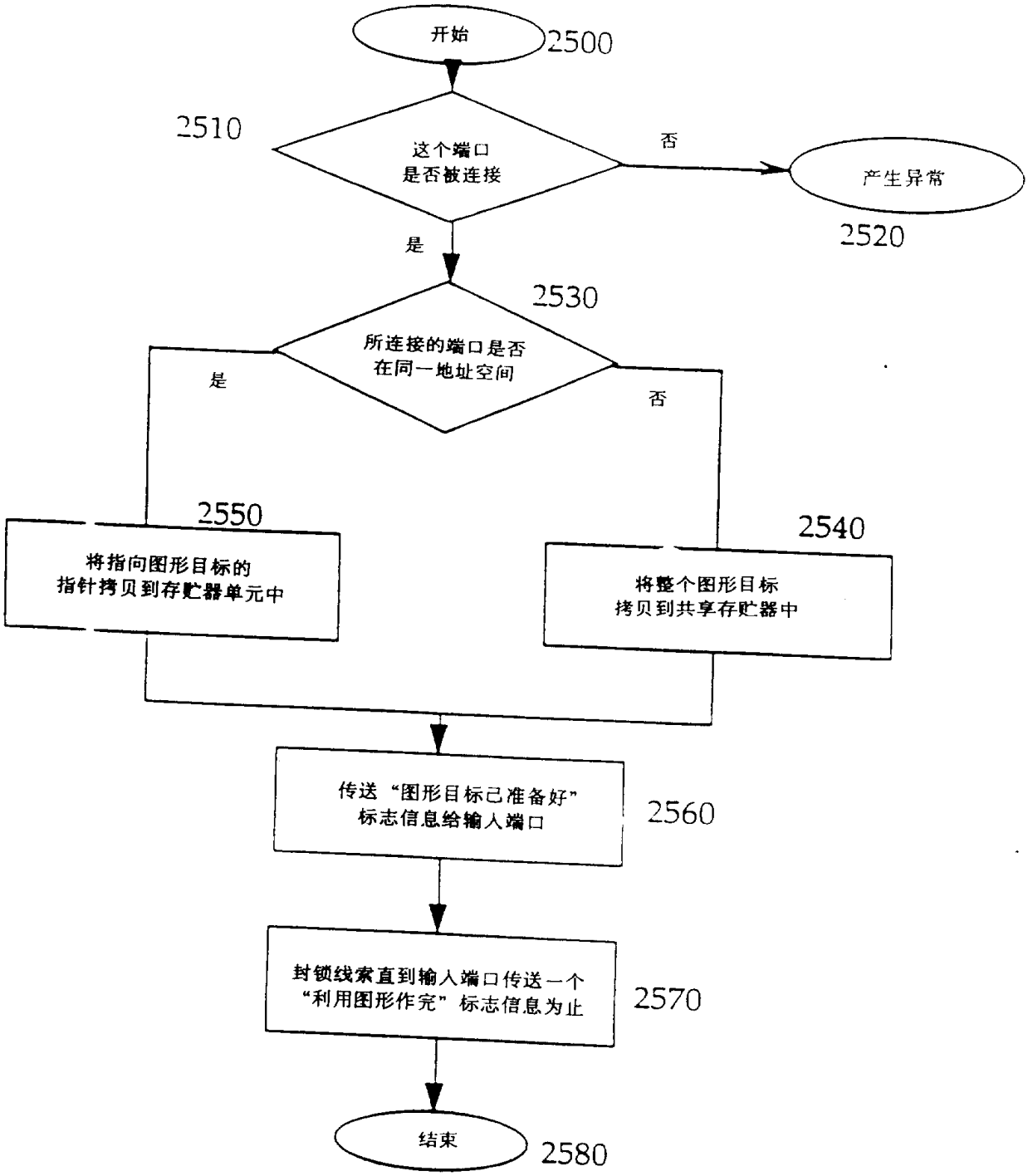

开始
2500
2510
这个端口
是否被连接
否
产生异常
2520
是
2530
所连接的端口是否
在同一地址空间
是
否
2550
将指向图形目标的
指针拷贝到存贮器单元中
2540
将整个图形目标
拷贝到共享存贮器中
传送“图形目标已准备好”
标志信息给输入端口
2560
封锁线索直到输入端口传送一个
“利用图形作完”标志信息为止
2570
结束
2580

图 25

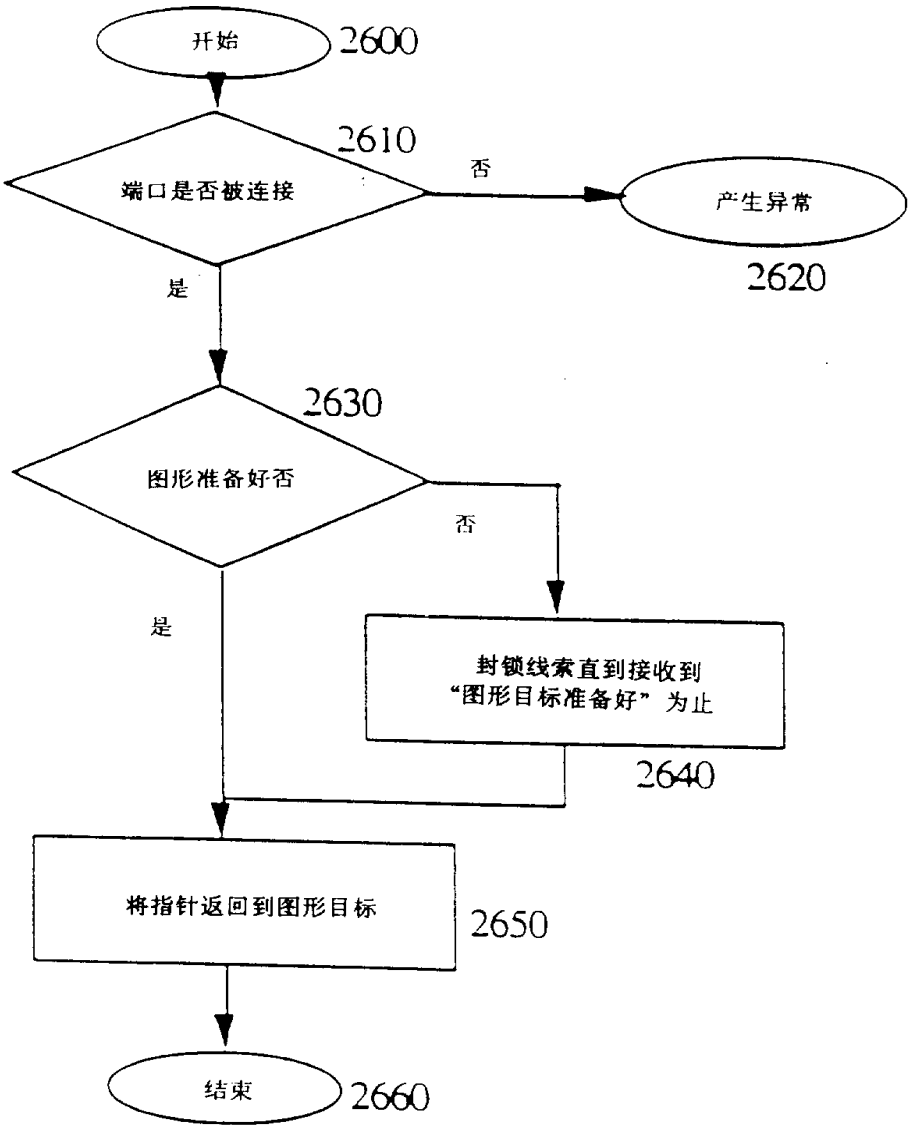

开始
2600
端口是否被连接
2610
否
产生异常
2620
是
图形准备好否
2630
否
是
封锁线索直到接收到
“图形目标准备好”为止
2640
将指针返回到图形目标
2650
结束
2660

图 26

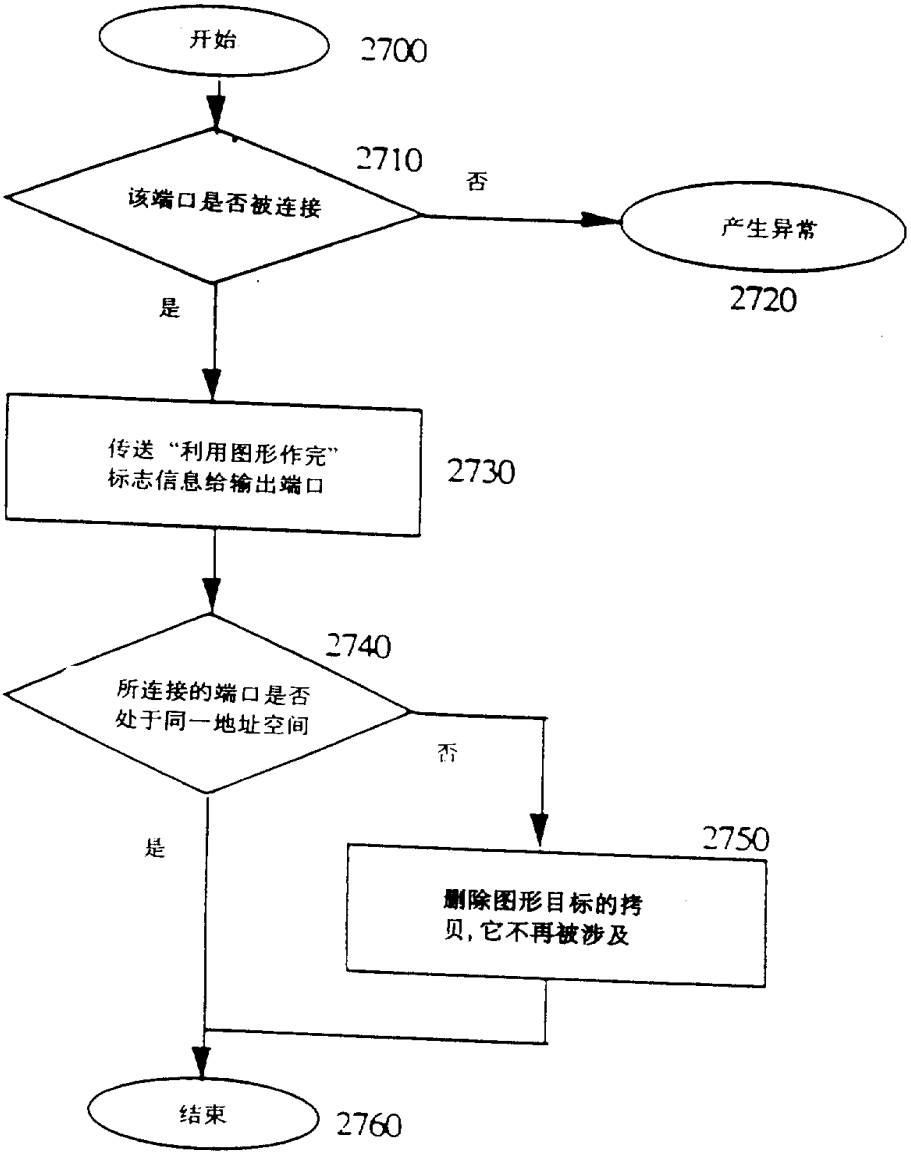

开始
2700
该端口是否被连接
2710
否
产生异常
2720
是
传送“利用图形作完”
标志信息给输出端口
2730
所连接的端口是否
处于同一地址空间
2740
否
是
2750
删除图形目标的拷
贝，它不再被涉及
结束
2760

图 27

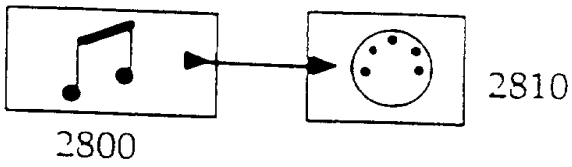


图 28

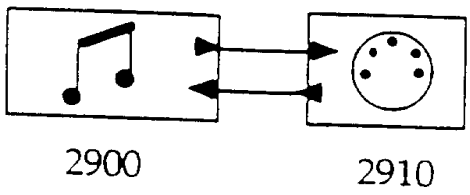


图 29

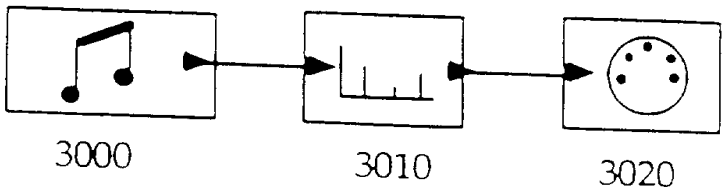


图 30

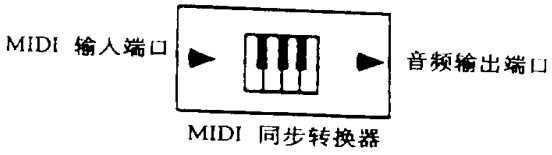


图 31

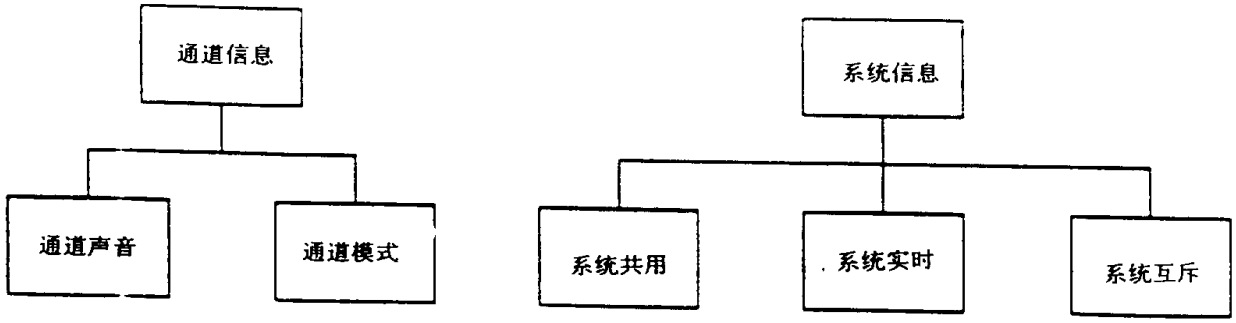


图 32

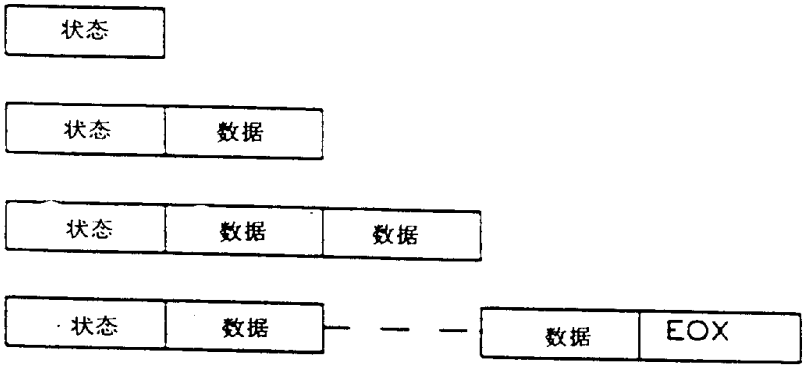


图 33

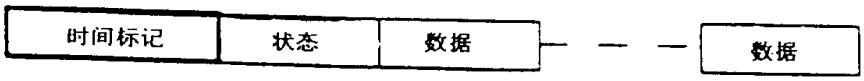

时间标记
状态
数据
数据

图 34

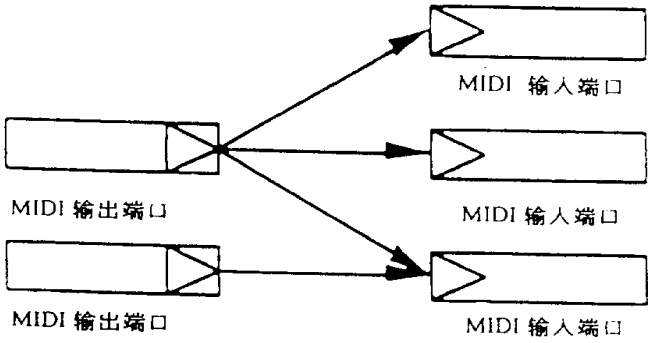

MIDI 输入端口
MIDI 输出端口
MIDI 输入端口
MIDI 输出端口
MIDI 输入端口

图 35

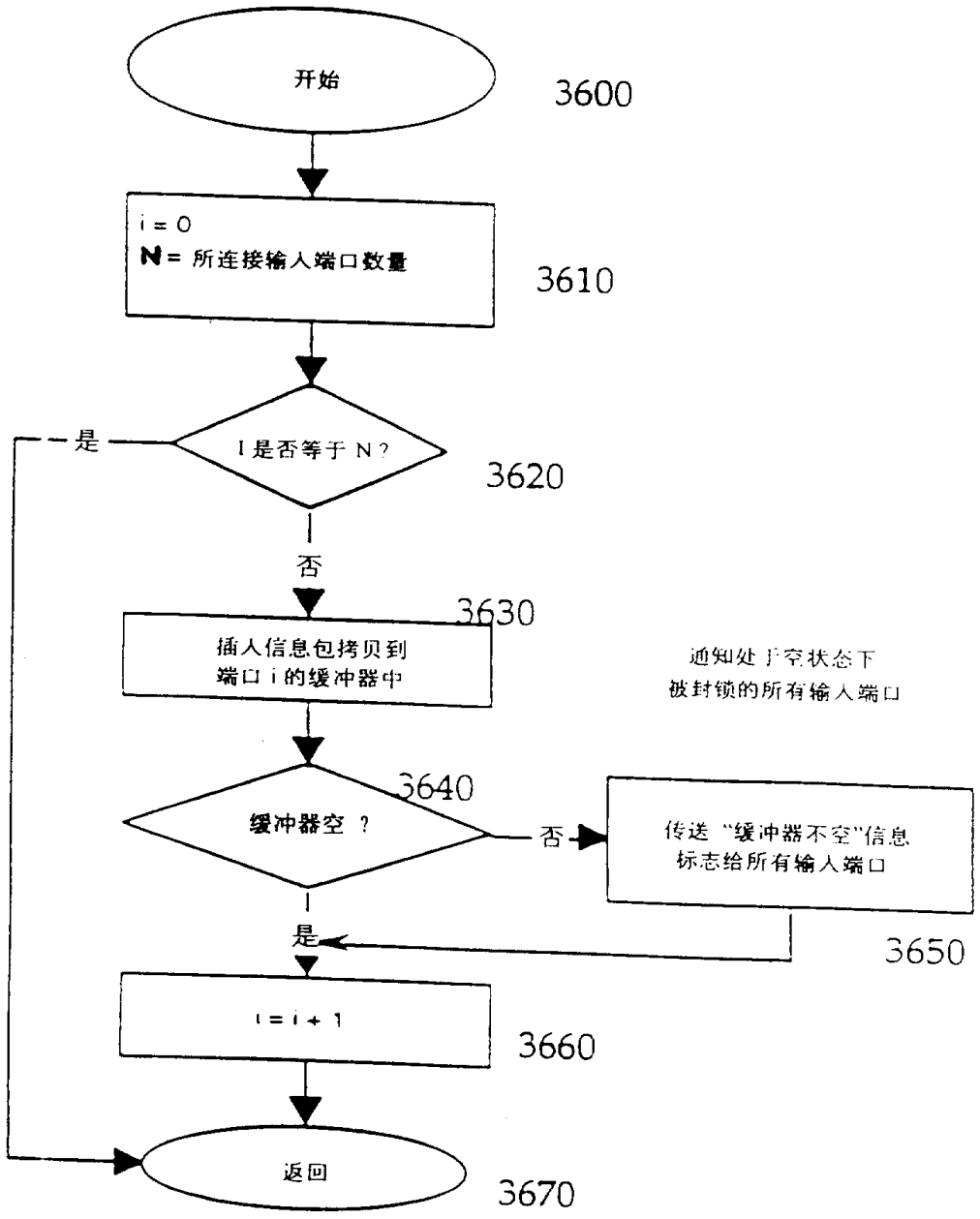

开始
3600
i = 0
N = 所连接输入端口数量
3610
I 是否等于 N ?
是
3620
否
3630
插入信息包拷贝到
端口 i 的缓冲器中
通知处于空状态下
被封锁的所有输入端口
3640
缓冲器空 ?
否
传送 "缓冲器不空"信息
标志给所有输入端口
是
3650
i = i + 1
3660
返回
3670

图 36

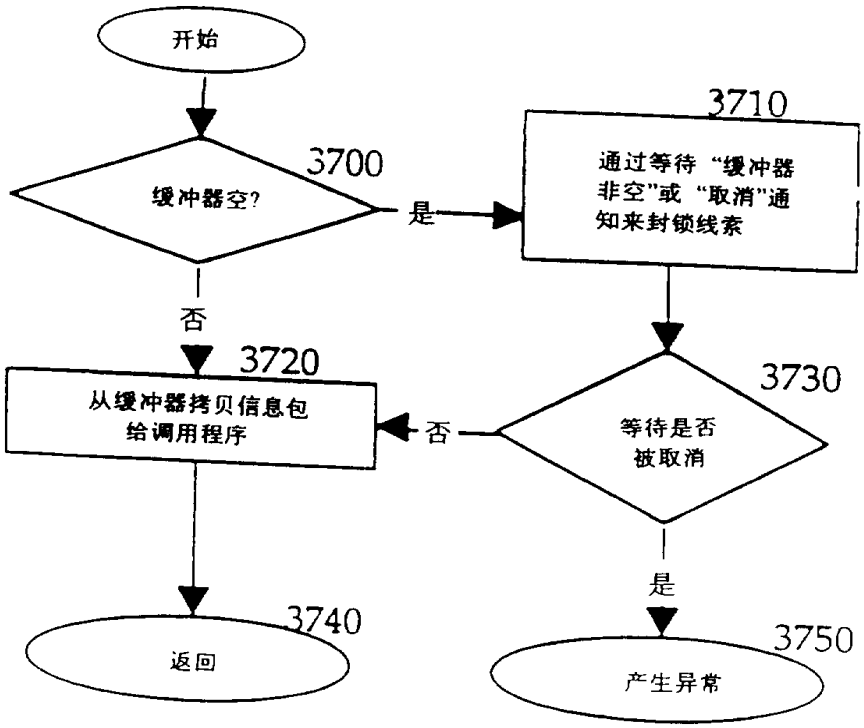

开始
3700
缓冲器空?
是
3710
通过等待“缓冲器非空”或“取消”通知来封锁线索
否
3720
从缓冲器拷贝信息包给调用程序
否
3730
等待是否被取消
是
3740
返回
3750
产生异常

图 37

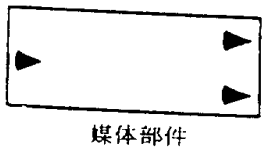

媒体部件

图 38

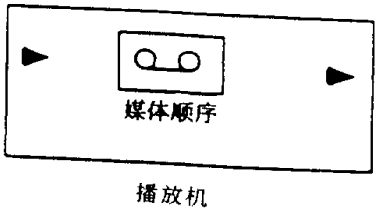

媒体顺序
播放机

图 39



图 40

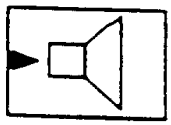


图 41

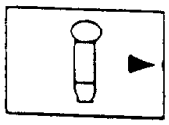


图 42

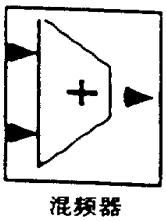
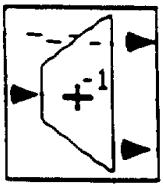
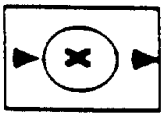


图 43



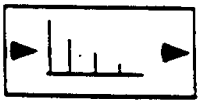
分离器

图 44



增益

图 45



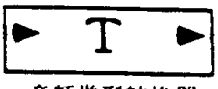
回波

图 46



抖颤

图 47



音频类型转换器

图 48



图 49

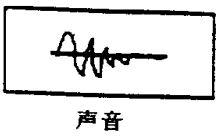


图 50

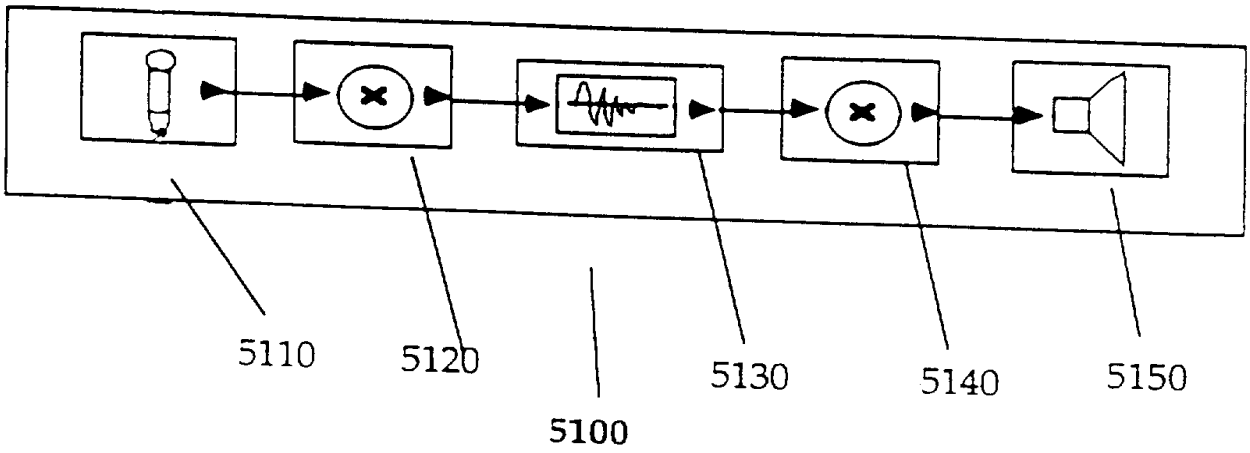


图 51

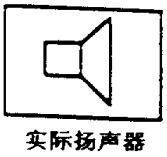
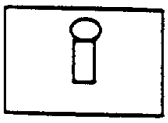
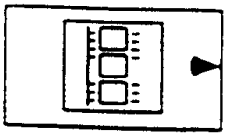


图 52



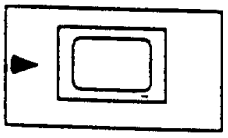
实际话筒

图 53



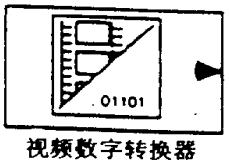
图形播放机

图 54



图形观察器

图 55



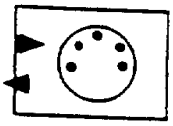
视频数字转换器

图 56



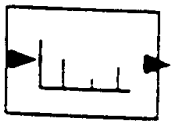
MIDI 播放机

图 57



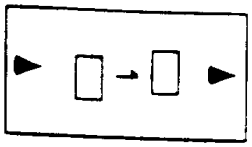
MIDI 接口

图 58



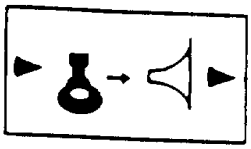
MIDI 滤波器

图 59



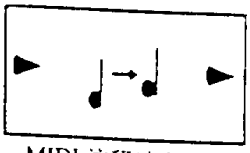
MIDI 映象器

图 60



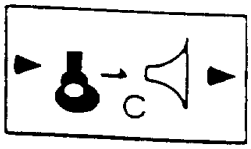
MIDI 程序映象器

图 61



MIDI 注释映象器

图 62



MIDI 通道映象器

图 63