[19]中华人民共和国国家知识产权局

[51]Int. Cl[7]
H04N 5/44

[12]发明专利说明书

[21] ZL 专利号 95104704.3

[45]授权公告日 2002 年 12 月 25 日

[11]授权公告号 CN 1097386C

[22]申请日 1995.4.27 [21]申请号 95104704.3

[30]优先权

[32]1994.4.28 [33]US [31]234146

[73]专利权人 开放电视公司

地址 美国加利福尼亚州

[72]发明人 J·R·曼南德 A·德尔普希

审查员 魏 玮

[74]专利代理机构 中国专利代理(香港)有限公司

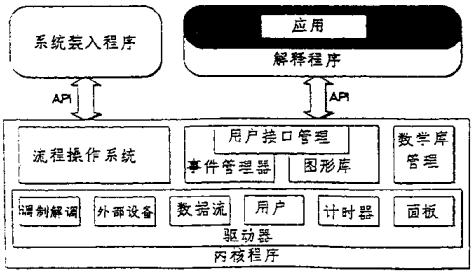
代理人 董 巍 马铁良

权利要求书 5 页 说明书 18 页 附图 4 页

[54]发明名称 控制执行音频视频交互程序的方法

[57]摘要

在接收包括目录和在该目录中具有有关的识别符的 AVI 程序的分组数据流的一种音频视频交互式(AVI)接收机中,一种控制执行 AVI 程序的方法包括下列步骤:首先响应在分组数据流中出现的 AVI 程序将 AVI 程序装入存储器中,然后开始执行已装入的 AVI 程序。然而当在分组数据流中识别不同 AVI 程序的目录被检测到时,对该执行的 AVI 程序进行最小化。



ISSN1008-4274

1. 在接收包括目录和在目录中有相关识别符的音频视频交互式（AVI）程序的分组数据流的一种音频视频交互式的接收机中的一种控制执行AVI程序的方法，其特征是有如下的步骤:

响应在分组数据流中出现的AVI程序，将该AVI程序装入存储器;

开始执行装入的AVI程序；和

响应于检测到第一标识，中止该执行AVI程序，其中所述中止包括将AVI程序的执行状态存储在存储器的方块中。

2. 根据权利要求1所述的方法，其特征是所述中止该执行AVI程序的步骤还包括

最小化该AVI程序;

其中该AVI程序从存储器中清除。

3. 根据权利要求2所述的方法，其特征在于装入AVI程序的步骤中还包括如下步骤:

搜寻包含与那个AVI程序相符的识别符的先前存储的存储块；和

如果是相符的存储块，在该存储块中设定AVI程序的执行状态和环境。

4. 根据权利要求1所述的方法，其特征在于所述第一标识包括分组数据流中的表示不同AVI程序的一个目录的检测。

5. 根据权利要求1的方法，其特征在于分组数据流还包括执行信号，并进一步包括如下步骤:

响应在分组数据流中的暂停执行信号，暂停执行AVI程序；和

响应在分组数据流中的继续执行信号，继续执行先前暂停的AVI程序。

6. 根据权利要求1的方法，其特征在于第一标识包括分组数据流中的具体信号分组的检测。

7. 在接收包括AVI程序和执行信号的分组数据流的音频视频交互式（AVI）接收机中的控制AVI程序执行的方法，其特征在于如下步骤:

响应在分组数据流中出现的AVI程序将AVI程序装入存储器;

开始执行装入的AVI程序；和

响应在分组数据流中的结束执行信号停止执行并从存储器中清除AVI程序。

8. 根据权利要求 7 的方法，其特征在于结束执行信号在分组数据流中是以结束执行信号分组的形式出现的。

9. 根据权利要求 7 的方法，其特征在于如下步骤:

响应在分组数据流中的暂停执行信号暂停执行 AVI 程序；和

响应在分组数据流中的继续执行信号继续执行先前暂停的 AVI 程序。

10. 根据权利要求 9 的方法，其特征在于分组数据流中的暂停信号是由在暂停期间重复的暂停执行信号分组所表现的，而继续执行信号是由不重复的暂停执行信号分组所表现的。

11. 根据权利要求 9 的方法，其特征在于在分组数据流中的暂停执行信号是由暂停执行信号分组所表现的，而由继续执行信号分组表现继续执行信号。

12. 在接收具有 AVI 程序和执行信号的分组数据流以及包括一个用于控制执行 AVI 程序的执行控制器的音频视频交互式（AVI）接收机中，控制执行 AVI 程序的方法，其特征在于如下步骤:

在该执行控制器中:

响应在分组数据流中出现的 AVI 程序，将 AVI 程序装入存储器中；

发出一个启动触发消息给 AVI 程序以响应用户的输入;

响应在分组数据流中的结束执行信号发出一个退出消息给 AVI 程序;

响应在分组数据流中的暂停执行信号发出一个暂停消息给 AVI 程序;

响应在分组数据流中的继续执行信号发出一个继续消息给 AVI 程序；和

在该 AVI 程序中:

执行下列中的一个:

一种非现用（inactive）的状态，在该状态下 AVI 程序不被使用;

一种有效（active）状态，在该状态下正在执行 AVI 程序;

一种暂停状态，在该状态下，暂停执行 AVI 程序;

响应将 AVI 程序装入存储器，而进入非现用状态;

响应退出消息，停止执行 AVI 程序并从存储器中清除 AVI 程序;

响应有效触发消息，当在非现用状态时进入有效状态;

当在有效状态下:

响应有效触发消息进入非现用状态; 和

响应暂停消息进入暂停状态; 和

当在暂停状态:

响应继续消息进入有效状态; 和

响应有效触发消息进入非现用状态。

13. 根据权利要求 12 的方法, 其特征在于:

在分组数据流中的暂停执行信号是由在暂停时间间隔内的重复的暂停执行信号分组所表现的, 而继续执行信号是由不重复的暂停执行信号分组所表现的; 和

在执行控制器中发出暂停消息的步骤包括响应第 1 个暂停执行信号分组发出暂停消息的步骤;

在执行控制器中发出继续消息的步骤包括在没有收到暂停执行信号分组的预定时间期间过后发出一个继续消息的步骤。

14. 根据权利要求 12 的方法, 其特征在于:

在分组数据流中的暂停执行信号是由暂停执行信号分组表现的, 而继续执行信号是由继续执行信号分组表现的; 和

在执行控制器中发出暂停消息的步骤包括响应一个暂停执行信号分组发出暂停消息的步骤; 和

在执行控制器中发出继续消息的步骤包括响应一个继续执行信号分组发出继续消息的步骤。

15. 在一种在用户控制下选择接收多个分组数据流中的一个分组数据流的音频视频交互式接收机中,每个分组数据流包含具有识别符的 AVI 程序以及执行信号, 该接收机包括控制执行 AVI 程序的执行控制器, 在该接收机中控制执行 AVI 程序的方法, 其特征在于如下步骤:

在执行控制器中:

响应在从多个分组数据流中选择的一个中出现的 AVI 程序, 将 AVI 程序装入存储器, 将装入的应用程序的识别符存入存储器的各自存储单元, 搜寻先前存储在存储器中包含最小化 AVI 程序的数据块, 并且如果在该数据块中包含的识别符与装入存储器中的 AVI 程序的识别符相符, 将新的装入的 AVI 程序的执行状态和环境设定为包含在相匹配数据块中的执行状态和环境;

响应用户输入发出一个有效的触发消息给 AVI 程序;

响应从多个分组数据流中选择的一个中的结束执行信号，发一个退出消息给 AVI 程序;

响应从多个分组数据流中选择的一个中的暂停执行信号，发出一个暂停消息给 AVI 程序; 和

响应从多个分组数据中选择的一个中的继续执行信号，发出一个继续消息给 AVI 程序;

响应在具有与存储器中各自存储单元中的识别符不同的识别符的数据流中出现的 AVI 程序; 发出一个最小化消息给 AVI 程序; 和

在 AVI 程序中:

执行下面中的一个:

一种非现用状态，在这种状态中 AVI 程序是非现用的;

一种有效状态，在这种状态中 AVI 程序正在执行;

暂停状态，在这种状态中 AVI 程序暂停; 和

已最小化的状态，有如下步骤:

停止执行 AVI 程序; 然后在存储器的一个存储块中储存 AVI 程序的识别符，以及 AVI 程序的执行状态和环境; 然后

从存储器中清除 AVI 程序;

响应在存储器中装入的 AVI 程序，进入非现用状态，

响应退出消息，停止执行并从存储器中清除 AVI 程序;

当处在非现用状态时，响应有效触发消息进入有效状态;

当处在有效状态时;

响应有效触发消息，进入非现用状态; 和

响应暂停消息，进入暂停的状态; 和

当处在暂停状态时:

响应继续消息，进入有效状态;

响应有效触发消息，进入非现用状态; 和

响应最小化消息，进入最小化状态。

16. 按照权利要求 15 的方法，其特征在于执行应用程序中的最小化状态的存储步骤还包括如下步骤:

在一个存储块中存储表示存储在存储块中的数据要保留在存储器中的时间期间的持续时间，在这个持续时间超过以后，就清除存储在存储

块中的数据。

17. 根据权利要求 15 的方法，其特征在于：

在多个分组数据流所选中的一个中的暂停执行信号由在暂停时间间隙中的暂停执行信号分组表现，而继续执行信号由不重复的暂停执行信号分组表现；和

在执行控制器中的发出暂停消息的步骤包括响应第 1 个暂停执行信号分组发出暂停消息的步骤；和

在执行控制器中的发送继续消息的步骤包括在没有接收到暂停执行信号分组的预定时间期间过后发出一个继续消息的步骤。

18. 根据权利要求 15 的方法，其特征在于：

暂停执行信号由暂停执行信号分组表现，而继续执行信号由继续执行信号分组表示；和

在执行控制器中发送暂停消息的步骤包括响应暂停执行信号分组发出暂停消息的步骤；和

在执行控制器中发送继续消息的步骤包括响应继续执行信号分组发出继续消息的步骤。

控制执行音频视频交互程序的方法

本发明涉及一种控制执行音频视频交互（AVI）可执行的程序（program）成分的方法。

已经提出交互式电视接收系统的方案，其中的电视接收机包括根据广播台并可以响应用户加入的数据而可编程的处理器，产生迭加在广播视频图像上的屏幕上图形显示，产生和广播音频相结合的声音和/或与广播台或其他外部数据处理业务结合的交换数据。在这种系统中，广播地点（location）包括产生交互应用程序信息的计算机系统，该信息包括可执行的代码和数据，将作为附加成分的交互式应用程序信息与相关的视频和音频电视信号分量相结合。在电视接收机中的处理器从广播台接收交互式应用程序信息，执行由那些信息表示的交互式应用程序，产生将与电视视频与音频相结合的图形和声音，并且处理通过遥控单元接收的用户输入。

在建议的 AVI 系统中，来自广播台的复合 AVI 信号是分组数据流形式的广播，它包含多个时分复用的分组业务。每个分组业务载送复合 AVI 信号的不同成分信号。例如，一个业务载送视频成分，另一个载送音频成分，还有一个载送交互式应用程序信息成分。还可以有载送立体声和 SAP 音频频道和/或闭合字幕信息等的其他业务。另外，某些分组数据流可以包含载送多于 1 个 AVI 程序的分量信号的分组业务。每个分组业务具有与它相关的独立的业务成分识别符（SCID），并且在该分组业务中每个分组都有业务识别符。

另外，在提出的 AVI 系统中，每个分组业务载送程序引导并包括预定的业务识别符。由程序引导分组业务载送的数据使载送这些成分的分组业务的业务识别符与 AVI 程序的成分有关联。利用这个数据，载送要求的 AVI 程序成分的分组业务可以从分组的数据流中抽取。

在 AVI 信号分组数据流中的成分信号由一个或多个传输单元载送，每个单元由多个分组组成。在任何传输单元中的第 1 个分组是标题分组，在传输单元中的其余的分组是与数据分组有关的。标题分组包含有

关随后数据的信息，而且相关的数据分组载送构成成分信号部分的数据。不同的传输单元可以包含不同数量的数据分组，将成分信号划分成为传输单元会受到在希望的时间传送不同的成分信号到观看者的地点所必须的时序的影响，或者受到其他的实时因素的影响。

交互式应用程序成分是由一个或多个代码模块（包含可执行的代码）组成，可能是一个或多个数据模块，以及包含描述代码的数据和组成交互式应用程序成分的数据模块的目录模块。在应用程序数据成分流中这些模块连续地重复。这些模块各自被单独地识别，而且由传输单元载送，正如前所述，在传输单元中标题分组还包含模块的识别符和在模块内随后数据分组内的数据所属的位置。该交互式应用程序信息成分还包含用以控制执行 AVI 应用程序的特殊的信号。例如，一个信号可以指示当前执行的 AVI 应用程序暂停执行；另一个信号可以指示当前暂停的 AVI 应用程序重新开始执行；而且还有一个信号可以指示当前执行的应用程序中止执行。这些信号可以结合在 AVI 程序成分分组业务内的信号分组中。

在 AVI 接收机中的处理器在系统装入程序的控制下首先从数据流中提取目录模块，而且利用包含在目录中的信息确定哪个代码模块首先被执行。称之为自动启动模块的代码模块于是从数据流中被取出并装入到存储器中，当自动启动模块被全部装入存储器时，处理器开始执行代码模块。在它的执行过程中，代码模块可以从目录模块中识别的数据模块请求数据。这些数据模块于是被取出来并装入到存储器中。当该模块已经完全装入存储器中时，该请求的代码模块于是被通知，并且执行程序连续地处理这些数据。这还可以使一个代码模块链接到随后的一个。在这种情况下，当前的代码模块发出链接列出在目录模块中新的一个代码模块的请求，而且它的存储空间是空的。被请求的代码模块于是从数据流中被提取出来，并且装入到存储器中。当它完全装入存储器中时，它随后被执行了。其他的功能也是可能的，并且在以下描述。

不象其他的分布式计算机系统，由 AVI 接收机接收的 AVI 程序成分在任何时候可以变化。例如，AVI 程序可以为非商业 AVI 或为商业 AVI 所中断，当然这包括不同的 AVI 节目。或者一个观看者可以从一个 AVI 节目到另一个 AVI 节目改变频道。它必须保持 AVI 可执行的代码和由那个代码产生的声音和图形间的适当的同步，同时接收音频和视

频成分。

根据本发明的原理，在接收包含目录和具有与目录中的识别符有关的 AVI 程序的分组数据流的音频和视频交互式（AVI）接收机中，控制执行 AVI 程序的方法包括下列步骤：第一，响应出现的分组数据流中的 AVI 程序，将 AVI 程序装入存储器。然后开始执行装入的 AVI 程序。当在分组数据流中用于识别不同的 AVI 程序的目录被检测到时，就把执行的 AVI 程序减小到最小。

图 1 是引入的本发明的 AVI 信号解码器的部分的方框图；

图 2 是图 1 所示的处理单元 40 所执行的软件的软件结构图；

图 3 是有益于理解从 AVI 程序中的数据成分中提取模块的流程图和设计图；

图 4 是有益于理解从 AVI 程序中的数据成分抽取模块的部分以方框形式和部分以存储器的设计图形式的示图；

图 5 是表示系统装入程序的初始化功能的流程图；

图 6 是表示系统装入程序的数据流监视功能的状态转换图。

图 1 是结合于本发明中的 AVI 信号解码器的部分的方框图。在图 1 中示出了在每个观看者场所安装的解码器，在该场所它按照要求参与 AVI 程序。在图 1 中，传输机构（未示出）耦合到解码器的输入端 5。输入端 5 耦合到调谐器 10 的输入端。调谐器 10 的输出端接到 AVI 程序成分解码器 30 的数据输入端。程序成分解码器 30 的数据输出端接到处理单元 40 的系统总线 416 上。处理单元 40 包含通过系统总线 416 以公知的方式连接在一起的中央处理单元（CPU）410、读/写存储器（RAM）412 和只读存储器（ROM）414。数据流 I/O 适配器 408 是以双向的方式连接在系统总线 416 和程序成分解码器 30 的控制端之间。

连接到系统总线 416 的音频处理器 418 提供音频信号给 AVI 音频输出端 25，而连接到系统总线 416 的视频处理器 420 提供视频信号给 AVI 视频输出端 15。此外，通过双向端 45 由 I/O 端口 422 将输入和输出设备连接到配置的本地处理器（未示出）上。用户 I/O 适配器 424 用以通过输入端 35 从用户处接收数据；而且通过双向终端 55 将调制解调器 426 连接到外部计算机（未示出），所有这些也是以公知的方式连接到系统总线 416 上。其他的例如数学处理器、其他的 I/O 适配器等等以公知的方式连接到系统总线 416 上。此外，可以还包括总线扩展器用于

连接解码器外部的其他设备。

在操作中，例如可以是直接射频（RF）卫星链路、电缆系统馈送或光纤链路的解码器传输机构载送多个 AVI 信号，任何一个 AVI 信号可以由观看的用户选择观看。在直接卫星链路中，例如多个 AVI 数据流可以通过调制各自的RF载波信号在传输机构上做到频分复用。每个RF载波信号可以由卫星中的相应的转发器再广播给观众的所在地。在处理器 40 的控制下，调谐器 10 用公知的方法选择需要的RF调制信号。例如，在直接卫星系统中，包含载有所要求的 AVI 程序信号的成分的分组业务的RF调制信号由公知的RF调谐器调谐。调谐器 10 的输出是包含这些分组业务的基带数字分组数据流。

通过将希望的业务识别符以及 RAM412 缓冲器存储单元写入适当的业务成分识别符（SCID），CPU410 从程序成分检测器 30 中请求希望的分组业务，而直接存储器存取（DMA）控制器通过数据流 I/O 适配器 408 在程序成分检测器 30 中寄存希望的分组业务。程序成分检测器 30 于是监视分组数据流以得到希望的分组业务。当从任意的希望的分组业务中接收到标题分组时，用公知的 DMA 写技术将它们存入 RAM412 中的预定的标题分组缓冲器，而且产生标题分组中断。当从任意的希望的分组业务中接收到数据分组时，使用公知的 DMA 写技术将其存储在 RAM412 中的先前指定的缓冲器存储单元。当已经接收到了传输单元中的所有数据分组时，产生数据的完全中断。在 CPU410 的控制下，也可启动也可禁止从分组业务中接收的标题和/或数据分组。由 K.E.Bridgewater 等人在 94 年 4 月 22 日申请的题为："分组视频信号反向传送处理器存储器选址电路"的美国专利申请 No.232，787，更详尽地描述了程序成分检测器 30。

例如，当调谐器 10 调谐新的RF调制信号时，包含程序引导的分组业务通过提供固定程序引导业务识别符给程序成分检测器 30 中的业务识别符寄存器而由 CPU410 提出请求。当在程序引导分组中的数据已被接收并储存在存储器中时，该数据允许 CPU 为所希望的 AVI 程序请求分组数据业务。

在由程序成分检测器 30 接收请求的分组业务中的信息分组以后，该分组通过 DMA 写入 RAM412 中的先前指定的缓冲器存储单元，使用公知的 DMA 读技术，在程序成分检测器 30 的控制下，视频处理器

420 和音频处理器 418 从与它们各自的分组业务相关的 RAM412 缓冲器存储单元读出数据。视频处理器 420 和音频处理器 418 于是解码压缩的和编码的数据以便分别在输出端 15 产生 AVI 视频信号和在输出端 25 产生 AVI 音频信号。在解码过程中 CPU40 可能与视频处理器 420 和/或音频处理器 418 合作，用以下插叙的方法，在 CPU410 的控制下处理数据成分分组业务的分组信息。

如上所述，每当程序成分检测器 30 接收到来自被请求的分组业务的标题分组时，它都会被存储在用于分组业务的 RAM412 中的预定存储单元，并且为 CPU410 产生标题分组中断信号。响应标题分组中断信号，中断处理程序执行对标题分组内容的分析，以及要么适当地刷新在程序成分检测器 30 中的 DMA 寄存器中的 RAM412 缓冲器存储单元，并启动 DMA 转换，要么如果传输单元是所不需要的，使 DMA 不转换。一旦 DMA 转换可以执行，数据分组中的数据于是在 DMA 的控制下装入到 RAM412 中。当这个数据分组装入完成时，程序成分检测器 30 产生数据完全中断信号。响应数据完全中断信号，执行中断处理程序以进行“清除”功能并准备下一个标题分组。

图 2 是由图 1 所示处理单元 40 执行的软件 200 的结构图。图 2 示出了组成 AVI 处理多重任务操作系统的主要软件成分。在图 2 中中，除去应用程序外，所有的存储在 ROM414 中的成分用黑色区表示。由 AVI 信号的数据成分载送的应用程序从广播位置接收并存储在 RAM412 中。图 2 中所示的软件成分表示可执行的代码和相关的常量数据。正象代码执行的一样，它可以产生和存取存储在 RAM412 中的可变数据。

在建议的 AVI 广播系统中，不同的解码器可以使用例如来自不同制造商的不同的指令集的 CPU。在这个系统中，应用程序是处理器独立的中间码。在每个解码器中的软件包括解释中间应用码的成分。这就可在包含任何型 CPU410 的解码器中执行广播应用程序。该解释程序将在来自 RAM412 运算存储器的中间码中读取数据成分指令，并借助应用编程接口（API）通过其他软件成分与硬件交互作用。作为基本上是可利用的应用程序的子例程的列表和必要时要访问的信息的 API 被公布，并可被应用程序员利用以对解码器单元进行访问。

数学库执行进行整数和浮点算术的所有的功能。流程操作系统控制监视 AVI 信号的数据成分的所有必要的驱动器，并处理被请求的模块，

以下将详细描述。用户接口管理部分处理所有的用户交互作用，并利用图形库，以及一个事件管理器与用户进行通讯。该图形库执行在接收的AVI视频上迭加的所有图象的产生，并且使用数学库划出复杂的曲线。

通过彼此间非同步地传送消息，解码器软件的不同的软件成分间相互通讯。每个程序成分具有消息队列，并且通过从读消息队列读出下一个消息、处理那个消息、可能的话发送信息给另外的程序成分以及如果不再有未定的消息的话，等待下一个消息来重复地工作。通过适当地路由选择这些消息并保持消息队列，事件管理器管理在其他的软件成分中的这些消息的通讯。

每个硬件适配器还包括相关的软件驱动器。该驱动器通过系统总线416执行CPU410和相关的硬件适配器中的寄存器间实际的交互作用。例如有用于调制解调器426、外部I/O端口422、数据流I/O适配器408和用户 I/O424 的驱动器。另外，分立的驱动器维持软件定时器并操作解码器的前面板。这些驱动器密切依赖于事件管理器。所有以上的成分使用由多重任务核心提供的通用的功能。例如，该核心保持处理优先权、实际任务队列、信号、信号量（semaphores）、优先任务转换时钟信号、中断（硬件和软件）以及处理堆栈存储器。另外，该核心提供硬件预置初始状态和作为系统装入程序的第一系统任务的启动。

在启动时，系统装入程序执行 API 对流程操作系统的请求，它依次通过数据流 I/O 适配器 408 请求数据流驱动器给程序成分检测器 30 发送适当的数据。这些来自系统装入程序的 API 请求以后面将详述的方式启动目录模块的数据成分分组业务的扫描。当目录模块被找到，它被装入 RAM412，并且查看是否所有的执行程序的需要的资源是可用的。如果是可用的，于是系统装入程序启动第一模块 AVI 数据成分的扫描，第一模块又称为自动启动模块，该模块启动 AVI 程序。当自动启动模块被装入，它将从数据成分分组业务中抽取出来并装入RAM412中。这个自动启动模块是中间码的形式，并且由解释程序解释执行。该自动启动模块执行初始状态预置的剩余部分并开始执行 AVI 程序。这个程序可能装入其他的码和数据块，并且链接到另外的代码模块，所有这些都通过API请求。以此方式，该系统装入程序是以与典型的UNIX相同的方式工作的。

另外，系统装入程序连续地扫描将传输的目录模块与在 RAM412

中的当前目录模块相比较的数据成分分组业务。如果传输的目录模块不同于 RAM412 中所存储的，它表示该数据成分分组业务已经改变，例如，观看者改变了频道，或者一个交互式商业节目正在广播。在这种情况下，通过事件管理器使用 API 将一则消息传送给应用程序。响应这一消息，应用程序重新分配它的所有资源，仅仅保持在处理单元 40 中的最小的存在。例如，用于存储所有代码和数据模块的存储器可能是空的，仅有应用的执行状态保持在 RAM412 中。当完成了应用程序最小化时，一个消息传送给系统装入程序。

系统装入程序于是分配执行由新的目录模块表示的 AVI 程序所必需的资源。在 AVI 数据成分分组业务中检测到新的目录模块时，搜索先前的最小化了的应用列表，如果出现了由新的目录表示的应用，那么这个应用通过重新从数据成分流中输入必要的代码和数据模块而重新开始，即从先前已经停止处重新开始。这可能发生在插入的交互式商业节目的末尾。这个过程可以是递归的，在这里第 2 个 AVI 程序本身可以由第 3 个 AVI 程序中断，随后重新启动。

图 3 示出了流程图和存储器布局图，图 4 是有助于理解从 AVI 程序中的数据成分抽取模块的图 1 所示的程序成分检测器 30 的更详细的方框图和存储器布局更详尽的示图。在图 4 中，来自图 1 所示调谐器 10 的基带数字分组流分别地连到数据 DMA 控制器 32 的数据输入端和程序成分检测器 30 内的标题分组 DMA 控制器 34 的数据输入端。数据 DMA 控制器 32 以及标题分组 DMA 控制器 34 的各自的数据输出端连接到处理单元 40 的系统总线 416。数据流 I/O 适配器 408 连接在系统总线 416 与各自的数据 DMA 控制器 32 和标题分组 DMA 控制器 34 的控制输入端之间。在工作中，数据流 I/O 适配器 408 提供控制信息，例如缓冲器存储单元开始和结束的地址、读和写地址以及转换计数，并用公知的方式以图 1 所示的 CPU410 加给数据 DMA 控制器 32 和标题分组 DMA 控制器 34。数据流 I/O 适配器 408 于是能使数据 DMA 控制器 32 和/或标题分组 DMA 控制器 34 分别地以公知的方式在 CPU410 控制下从分组数据流到缓冲器传送数据或标题分组，或者使这种传送不进行。当数据 DMA 控制器 32 完成了数据传输时，它产生一个数据完成中断给 CPU40。当标题分组 DMA 控制器 34 完成输入标题分组时，它产生一个标题分组中断给 CPU410。

仍在图 4 中，RAM412 是由大方框表示的，数据结构是由在大方框内的小方框表示的。图 4 中的方框仅仅是示意，决不意味 RAM412 中数据结构分配的绝对和相对位置以及尺寸大小。在 412 中示出了模块请求队列 322、标题分组缓冲器 324、目录模块缓冲器 326 和模块缓冲器 328 的数据结构。数据结构内的信息字段用水平的条表示，该条包含字段中含有的信息类型的名字。这些在以后将详细讨论。

图 3 示出了随后的从数据成分分组业务中抽取模块并存储在 RAM412 中的缓冲器的步骤。用作其他的模块处理的随后的类似步骤将在以后描述。在图 3 中，应用程序（或系统装入程序）中的动作表示在“标题应用程序（APPLN PROG）”下面的左侧栏中。在方框 302 中，使用 AVI 的应用程序可以请求流程操作系统从 AVI 程序成分分组业务中装入具有识别符 ID 的模块。如上所述，API 请求基本上是对操作系统功能的子例程请求的。该程序执行传送给流程操作系统（FOS）。在 FOS 中的动作表示在（FOS）流程操作系统标题下面的相邻的右栏中。由于该请求包含模块的装入，在方框 312 中的 FOS 请求存储器管理器分配容纳该模块的足够大小的存储空间。例如，如果请求的模块是代码或者数据模块，先前存储的目录模块 326（图 4）包括含有模块 ID 的长度（LENGTH）的字段。在这种情形存储器管理器分配具有开始地址 START 和终止地址 END 的模块存储器缓冲器 328（图 4 中示出）。然后在方框 314 中的描述请求的信息，例如是模块的识别符 ID、请求 REQUEST 的类型（在这种情形中请求抽取和装入模块）以及分配的缓冲器开始地址 START 和终止地址 END 都存储在请求队列（QUEUE）322 的条目处。当他们出现在分组数据流中时，标题分组 DMA 控制器 34 就可以被启动以将标题分组装入 RAM412 中。

如果请求是对于目录模块的，预先不知道其长度。在这种情况下就需要相对大的存储器分配。如果这个分配产生得太小，就要重复请求，直到请求到大的存储器分配，要么目录模块装入，要么确定为没有充分的存储器可以令其装入，在这种情况下运行 AVI 程序的努力就放弃了。

随后 FOS 立即返回到调用应用程序。该应用程序于是可以执行其他的处理，例如向其他的模块发出请求、其他的初始状态预置等等。当请求的模块的存取是所需要的，在方框 304 中的应用程序可以发出 API 请求给核心中的等待功能。这个功能暂停应用程序的执行，直到由应用

程序检测到表示成功装入请求的模块的消息为止。当接收到那个消息时，该应用程序被再次启动以处理那个消息。另外，该应用程序可以保持有效，例如为了响应更快的用户输入，并且周期地询问消息队列表示成功装入请求的模块的消息是否收到，当收到该消息时就处理该消息。

如上所述，标题分组 DMA 控制器在先前由存储器管理器分配的 RAM412 中的标题分组（HDRPKT）缓冲器 324（见图 4）中装入标题分组，并且输出标题分组中断给 CPU410。由位于核心的标题中断处理器执行处理的一部分在图 3 中的标有“HEADERINTR”（标题中断）的栏中示出。在方框 332 中，在传输单元中载送的，作为标题程序分组的程序分组的识别符，从在标题分组缓冲器 324 中的公知存储单元、ID 处搜索到，请求队列 332 被检查以确定是否这里有这个模块的未决的请求。

如果有模块未决的请求，则在方框 336 中，在程序成分检测器 30 中的数据分组 DMA 控制器用下列的值预置初始状态：来自请求队列 322 中的开始地址 START 和终结地址 END 的模块缓冲器 328；作为模块缓冲器 328 开始地址与传输单元数据偏移 OFFSET（即 START + OFFSET）的和的写地址；和作为 START + OFFSET + SIZE 的上一个写地址（换言之，是来自代替上一个写地址的标题分组缓冲器 324 的尺寸大小 SIZE 的装入计数）。该数据分组 DMA 控制器 32 于是可以工作。

在方框 338 中，如果这是完成装入请求以后接收的第 1 个标题分组，存储在请求队列 322 中的指向第 1 写地址的指针 FIRST 被初始化为该第 1 传输单元的写地址（即 FIRST + START + OFFSET）。另外，一个预期的指向下一个写地址的指针也存储在请求队列 332 中，该指针 NEXT 对第 1 个传输单元的写地址（即 NEXT = START + OFFSET）预置初始状态。随后其他的处理在方框 338 中完成，338 将在以后详细描述。例如，指向请求队列 332 中的存储单元的专用指针被当前处理，ＣＵＲＲ ＲＥＱ（即当前请求）存储在 RAM412 中未示出的预定存储单元中。然后，在方框 339 中，中断处理器返回（339）。

数据分组 DMA 控制器 32 对指向先前接收的写地址（START + OFFSET）的写指针进行预置初始状态，并且将来自 AVI 程序成分分组业务中的随后的数据分组装入 RAM412 中模块缓冲器 328 中的顺序

的存储单元。当传输单元中的所有数据已经装入 RAM412，就产生数据完全中断。在核心中的数据完全中断处理器执行的处理的部分示于图 3 中的标有 DATA COMPL INTR 标题的右侧栏中。

在方框 342 中执行有关 DMA 传输的当前状态的清除功能。先前设定在标题分组中的中断处理程序中的当前请求指针（CURRREQ）指向请求队列 332 中的传输单元刚好已经完成了装入的条目。在当前请求中的预期的下一个写地址指针增加了标题分组缓冲器中的 SIZE 值，而且现在为下一个传输单元指向预期的写地址。如果预期的下一个写地址指针的新值等于缓冲器 328 的结束地址 END，它将用返转方式对模块缓冲器 328 的开始地址 START 复位。

在方框 344 中，确定是否所有的请求的模块已经装入存储器了。预期的下一个写地址指针 NEXT 的值与第 1 个装入地址 START 比较。如果相同，即整个模块已经装入了。在方框 346 中，如图 3 中的破折线所示，消息通过事件管理器传送给请求应用程序以指示请求的模块已完全地搜索。另外，从请求队列 322 中除去请求。如果预期的下一个写地址 NEXT 的值和第 1 个装入地址 START 不相同，该数据完全中断处理程序返回（349），包含请求模块的数据的下一个传输单元，如上所述，由标题分组中断处理程序处理。在这两种情形下，当前请求指针（CURRREQ）被清除了。

如果程序成分检测器 30 未能适当地接收传输单元的某些部分，在来自先前的标题分组的数据完全中断信号已由程序成分检测器 30 中的 DMA 电路产生出来前，随后的标题分组将被接收到。这就是在先前数据完全中断信号可以产生之前，依次产生随后的标题分组中断信号。在标题分组中断处理程序和数据完全中断处理程序中的处理可以合作以识别这种状态并提供对差错的处理。

在标题分组中断处理程序中，在数据分组 DMA 控制器可以接收下一个传输单元后，在方框 338（图 3 中示出）进行这种处理。对于每一个接收的标题分组，在由数据完全中断处理程序先前刷新的当前请求队列的条目中预期的下一个写地址 NEXT 被与新接收的标题分组的写地址（START + OFFSET）相比较。如果他们是相同的，先前的传输单元被成功地接收。然而，如果最后的结束地址和新的偏移不相同，这意味着先前的传输单元的 DMA 传输完成得不成功。在这种情况下，

第 1 写地址 FIRST 和预期的下一个写地址 NEXT 二者被刷新成当前的写地址（START + OFFSET）。这就是说先前装入的传输单元基本上被放弃，而且用当前的传输单元重新启动模块的装入。由于先前成功装入的传输单元当重新装入时可能产生差错，从数据遗漏型的差错恢复的方式可能要用更多的时间。然而，使用这种形式的恢复，由标题分组中断处理程序和数据完全中断处理程序完成的任务被最小化，而且只有两个指针需存入存储器中。

作为模块完整的消息处理的一部分，对接收的模块内事件处理程序执行误差检查。例如，循环冗余校验（CRC）码作为模块嵌入的部分被传输。事件处理程序在 RAM412 中的模块缓冲器 328 中的接收的模块内计算 CRC，并将其与嵌入的 CRC 作比较。如果新计算的 CRC 等于嵌入 CRC，则模块被正确地接收，否则则出现差错，如前所述该模块要重新装入。

当请求的模块已经完全装入存储器中时，可以连续进行由应用模块所做的进一步处理，在图 3 中由 API 调用的底部到等待功能 304 的一条虚线表示出。然而，在应用程序中的独立的任务可以被起动以便响应来自应用程序消息队列的消息的接收。

上述的 API 包括借助应用程序、解释程序或系统装入程序而访问数据流的功能。一个应用程序将使用发表了的 API 说明来提出 API 请求以访问所希望的数据流功能。第一组功能涉及到模块目录。第 1 功能 DIR NEW 是对新的目录的请求。如上所述，为了响应这个 API 功能，对存储器做出分配，于是请求被排入队列以便将数据流中的下一个目录模块装入，然后 API 功能返回。当目录已经装入后，发送一个消息给请求程序。另一个功能 DIR_FREE 释放当前目录所占有的存储器空间。功能 DIR_SELECT 表示哪一个目录模块将用于随后的 API 请求。功能 DIR_CURRENT 将处理返回到当前选择的目录。

功能 DIR_SPY 和 DIR_STOP_SPY 类似于 DIR_NEW 功能。响应 DIR_SPYAPI 请求，一个请求被排入目录模块请求队列，而不是当它被装入时进行装入目录模块和发送消息，无论何时在数据流中（目录模块未装入）检测到目录模块，该功能都发送一个消息。另外，该请求一直保留在请求队列中直到完成 DIR_STOP_SPY API 请求。当作出 DIR_STOP_SPY_API 请求时，针对目录窥探请求对请求队列进行搜

寻，而且条目被清除。这些功能用于在数据流中窥探当前目录的任何变化。最后，API 请求抽取关于当前目录的信息：DIR_IDENTIFER、DIR_REQUIREMWNT 和 DIR_NB_MODULES。

由于在模块中嵌入 CRC 码，为了装入模块的任何存储器分配请求必须考虑这种码。提供三个 API 请求用以处理这个（码）。考虑到任何 CRC 或其他的存储器需求，功能 MODULE_ALLOC 将模块识别符作为自变量并请求分配适当数量的存储器以装入模块。功能 MODULE_FREE 释放模块所占有的存储器。MODULE_CHECK执行对装入的模块的 CRC 校验并返回该结果。由于 CRC 是在装入存储器时嵌入在模块中的，所以这就可以在任何时候来执行。

另外一组 API 请求涉及用当前选择的目录去识别模块。API 请求抽取的有关的模块的信息为：MODULE_REQUIREMENT、MODULE_SIZE 和 MODULE_FLAGS。这些能够使系统确定模块是否可能被装入和/或执行。功能 MODULE_RUN 用以装入可执行的模块，如上所述，产生一种新的处理，并且在模块的入口点开始执行该模块。这个功能是由系统装入程序所使用的以便启动 AVI 程序的执行。功能 MODULE_CHAIN 用以装入随后的可执行的模块、结束当前模块的执行以及在它的入口端开始执行新的模块装入。在这种情况下没有产生新的处理。功能 MODULE_LOAD 用以装入模块，但不开始执行。如上所述，当模块装入已经完成时，一个消息就发送给请求程序。功能 MODULE_EXEC 用以产生新的处理并开始执行模块，该模块由 MODULE_LOADAPI 请求在它的入口端先已装入。

功能 MODULE_LINK 使用当前的处理、资源和变量执行新的模块。它允许借用提供的动态链路将模块内的子例程之类的请求接到新的模块。这就允许 AVI 程序划分成为更小的模块，一旦需要，这些更小的模块可以动态地链接。功能 MODULE_LINK 保持再定位和转移表。功能 MODULE_SPY 和 MODULE_STOP_SPY 的工作与 DIRECTORY_SPY 和 DIRECTORY_STOP_SPY 相类似，但与所识别的模块相关。MODULEAPI 请求在包含模块的识别符的请求队列中插入一条目。不论何时在数据流中检测到与标题模块相同的识别符，一个消息就发送给请求程序。这要持续到 ULE_STOP_SPY API 请求完成。响应 MODULE_STOP_SPYAPI 请求，包含对识别的模块作窥探请求

的条目被从请求队列中被除掉。在处理中 MODULE_STOP_LOAD 功能停止任何当前模块装入的请求并从请求队列中除去装入请求的条目。当与数据流相关的具体的信令分组，诸如暂停数据流或结束数据流出现时，FLOW_MESSAGE 和 FLOW_STOP_MESSGE分别产生和除去对消息的请求。当这种事件发生时，一个消息被发送给请求程序。

如上所述，系统装入程序执行系统的初始状态的预置并监视数据流以确保应用程序的执行与接收的音频和视频成分同步。图 5 是表示系统装入程序的预置初始状态功能的流程图。在图 5 中的方框 52 中，解码器(17)的不同的硬件和软件成分被预置初始状态。另外，分配了 RAM412 中的存储单元并对不同的数据结构预置初始状态。这些初始状态的预置功能是公知的，并且依赖于解码器中的其他软件成分。系统程序员将明白，硬件和软件需要预置什么样的初始状态、需要什么样的数据结构以及如何执行预置初始状态。因此，这个方框在下面不准备详述了。

在方框 54 中，如上所述完成 DIR_NEW API 请求。该 API 请求将出现在 AVI 程序成分分组业务中的下个目录模块装进 RAM412 中的分配的缓冲器中。这个 API 请求立即返回到系统装入程序，甚至直到晚些时候目录才装入 RAM412 中。系统装入程序执行其他的功能，而且，然后如果必需，等待 API 请求（未示出）一直到一个消息通过事件管理器被接收到，表示目录模块已经装入。在方框 56 中，在解码器（图 1 中）中可利用的资源在目录模块中与表示需要的资源的数据作比较。如果解码器有充足的资源执行 AVI 程序，则作出一个 MODULE_RUNAPI 请求以便从如前所述地装入在先前装入的目录中识别出的自动启动代码模块。再者，API 请求立即返回，但是直到在稍后一些时间，来自数据流的代码模块才被装入。在自动启动代码模块完全装入后，通过解释程序，使用执行 AVI 程序的多重任务核心程序以公知的方式产生另一个任务。

在方框 58 中，系统装入程序开始监视执行信号的 AVI 程序成分和目录变化，并且如下所述，通过发送消息给 AVI 程序来控制 AVI 程序的执行。图 6 是表示系统装入程序的监视功能以及有利于理解系统装入程序的工作的状态转变图。如果在 AVI 程序成分分组业务中检测到了目录，则观看者已选择了的节目是一个交互式节目。一旦目录以及由 AVI 成分分组业务请求的自动启动代码模块已经装入 RAM412 中，在系统

装入程序的控制下，AVI 程序进入 INACTIVE 状态 61。在该状态 61 中，所有的启动的应用的资源已经分配好，而且该应用可以部分地或全部地装入，但是没有与观看者的交互作用。例如，当自动启动模块被装入时，AVI 程序保持处在 INACTIVE 状态 61。另外，甚至自动启动模块已经装入，观看者仅仅能通过传送 AVI 程序的通道改变频道，并且不打算与 AVI 程序交互作用。或者观看者在作出决定去连接交互作用前恰恰能希望观看 AVI 节目。在这些情况下重要的是，遥控在一般频道改变方式中起作用，在交互式方式中不起作用。这就是 INACTIVE 状态 61 的目的。为了通知观看者正在观看的频道广播的是交互式节目，具体的交互式节目（program）Logo （标志）和 icon（肖像）迭加在 AVI 视频图像上。

为了使观看者实际开始与 AVI 程序（program）的交互作用，在下面的一个称之为 ACTIVATEKEY 的专用键提供用作遥控。当已显示了交互程序(program) Logo 或 icon 时，观看者可以按 ACTIVATE KEY 键，响应按 ACTIVATE KEY 键，系统装入程序发送一个 ACTIVATE 消息给 AVI 程序，该程序然后进入 ACTIVATE 状态 63。在该状态 63 中，解释程序实际开始执行先前在它的入口端装入的 AVI 程序。当 AVI 程序的自动启动模块开始执行时，它在 RAM412 中分配和预置它自己的数据结构的初始状态，装入其他的代码和/或数据模块并控制所有来自遥控器和前控制面板的用户的操作。由于 AVI 程序控制所有的用户的交互作用，它可以防止用户改变频道或执行其他的通常遥控功能。为了恢复通常的遥控功能，观看者必须首先停止当前的 AVI 程序。观看者要再按 ACTIVATEKEY 键，以撤销该程序。响应这个键的按下，系统装入程序发出一个 DEACTIVATE（撤销）消息给执行中的 AVI 程序，该程序于是离开 ACTIVATE 状态 63，并返回到 INACTIVATE 状态 61。再者，具体的交互程序 Logo 或 icon 显示出来，表示 AVI 程序已经装入，但未执行。然后，观看者可以变化频道或操作其他的通常遥控功能，或者可以再按 ACTIVATEKEY 键来重新开启 AVI 程序。ACTIVATEKEY 键当它按下时，起到了 ACTIVE 状态 63 和 INACTIVE 状态 61 间的触发器或开关的作用。ACTIVATE 和 DEACTIVATE 消息也可以被认为开启开关的（ACTIVATE TOGGLE）作用。当 ACTIVATEKEY 键按下，它们的意思是（ACTIVATE 或

DEACTIVATE）依赖于 AVI 程序的状态。

当在 ACTIVE63 中执行 AVI 程序时，当希望暂停它的执行时要重复几次。例如，当非交互式的商业节目被发送，该发送的音频和视频将不再与由解码器 10（见图 1）产生的音频和视频相适应，它希望允许观看者用正常的方式使用遥控器。然而，系统程序员不能提前知道何时需要这种暂停。因此，在这种情况下，独立于 AVI 程序的广播台可以重复地包括在 AVI 程序成分分组业务中的称之为暂停信号分组的专用信号分组（如上面所述）。每个这样的分组包含涉及暂停执行当前执行的 AVI 程序的数据。

系统装入程序通过 FLOW_MESSAGE API 请求接收每当在 AVI 程序成分分组业务中识别出那个分组而发出的消息。例如，当接收到暂停的信号分组时，系统装入程序接收暂停的信号消息，并且响应第一个暂停信号消息而发出一个暂停的消息给 AVI 程序，AVI 程序依次暂停执行而进入 SUSPENDED 暂停状态 65。在该状态 65 中，执行的 AVI 程序是以这样的方式停止的，即它可以从它暂停的点上再次启动。这就是说，所有执行 AVI 程序的必要的所有的资源保持分配的状态，AVI 程序的执行状态存储在 RAM412 中的一个存储单元。另外，表示先前执行的交互式程序暂停，而且准备当允许时再次开始的第 2 个 Logo 或 icon 迭加在当前的视频图象上。

当中断（即非交互式商业节目）过去后，广播台停止包含在 AVI 程序成分分组业务中的信号分组。在没有接收暂停信号消息的时间期间过后，系统装入程序给 AVI 程序发出 CONTINUE（连续）的消息，该 AVI 程序依次从它先前暂停的状态而重新启动从而进入 ACTIVE 的活动状态，这正如前面所述过的一样。

如上所述的 SUSPEND/CONTINUE 信令方案的另一实施例，是当希望暂停执行 AVI 程序时，为广播台在 AVI 程序成分分组业务中包括单一的一个暂停的信号分组。当需要重新启动 AVI 程序时，广播台于是包含在 AVI 程序成分分组业务中的称之为继续信号分组的另一个具体的信号分组。这个分组包含涉及当前暂停的 AVI 程序重新启动的数据。系统装入程序识别了该继续信号分组，并发出 CONTINUE（继续）的信息给 AVI 程序，该程序重新开始并进入 ACTIVE（有效）状态 63，这和以前所述相同。

还可以让观看者停止执行暂停的 AVI 程序。当暂停的程序的 Logo 或 icon 被显示出来，观看者按动 ACTIVATEKEY 键。响应这个按键，系统装入程序发出 DEACTIVATE 消息（停用）给暂停的 AVI 程序，如前面述过的一样，该 AVI 程序依次进入 INACTIVE 状态（静止）。从 INACTIVE 状态 61，当观看者按动 ACTIVE KEY 键时，程序可以仅重新开始执行，该键的按动使得系统装入程序发出一个 ACTIVATE 消息给 AVI 程序，该程序于是进入 ACTIVE 状态 63。如果系统装入程序继续收到暂停信号分组，另一个 SUSPEND 消息立即发给 AVI 程序，该程序再次进入 SUSPENDED 状态（暂停）65。INACTIVE 状态 61、ACTIVE 状态 63 和

SUSPENDED 状态 65 是 AVI 程序中的三种状态，AVI 程序响应发自系统装入程序的消息来转换状态。然而还有两种别的状态，它们在系统装入程序的直接控制下可以进入。

一个 AVI 程序可以到达它的执行的终点。例如，广播台可以包括称之为终点执行信号分组的另一个具体的信号分组于该 AVI 程序成分分组业务中。当通过 FLOW_MESSAGEAPI 请求在 AVI 程序成分分组业务中识别出一个终点执行信号分组时，系统装入程序接到一个终点执行消息。响应该终点执行消息，系统装入程序发出一个 EXIT（退出）的消息给 AVI 程序。不考虑 AVI 程序是处在 INACTIVE61、ACTIVE63 或 SUSPENDED65 的哪种状态，AVI 程序响应 EXIT 消息解除它的所有资源，并且除掉在解码器 10（图 1 中示出）的它的所有记录。这个程序被认为进入 HALTED（停止）状态 69，并且从解码器 10 处消失。在这些情况下，程序本身可以通过用户命令或依靠它自己的执行来识别它的执行是否到达了终点。当 AVI 程序识别出它的执行已结束，它将主动进行例如收到 EXIT 消息后使其进入 HALTED 状态 69 的相同的处理。

当 AVI 程序处在 SUSPENDED 状态时，在 AVI 程序成分数据流中接收不同的交互式 AVI 程序是可能的。例如 AVI 程序可以由于商业节目而暂停，该商业节目本身可以是交互式节目，或者用户可以将频道变到广播不同 AVI 节目的频道上。在这两种情况下，新的 AVI 程序将包括目录模块，该目录模块不同于暂停的 AVI 程序。

每当在 AVI 程序成分分组业务中检测到目录，系统装入程序通过

DIR_SPYAPI 请求，接收一个消息。系统装入程序将当前的有效目录与刚检测的目录相比较。当系统装入程序识别出出现在 AVI 程序成分分组业务中的不同的目录时，它开始装入用这些目录所表示的 AVI 程序。

首先，一则消息发给当前暂停的 AVI 程序以表示该程序成分分组业务不再广播它自己的节目，或者该节目“失去流程”（lost the flow）。这个消息是当前执行的程序最小化它自己的请求，即是 MINMIZE 消息。为了响应该 MINMIZE 消息，当前暂停的 AVI 程序首先将其当前的执行状态和环境存储在 RAM412 的一个小存储块之中；RAM412 包含 AVI 程序的识别以及下面要描述的时间的持续。然后暂停的 AVI 程序开始解除它的资源。最小化 AVI 程序不包括任何代码，因此，不能响应消息改变状态或重新启动自己。

系统装入程序然后装入新检测的目录和自动启动模块，并且将新的 AVI 程序放入前面已述过的显示交互式程序 Logo 或 icon 的 INACTIVE 状态 61。观看者于是可以通过按 ACTIVATEKEY 键来开始和结束与新的 AVI 程序的交互作用。而且该程序本身可能被暂停和继续。

该最小化处理是循环处理。例如，这个新 AVI 程序，如果暂停，如果在 AVI 程序成分分组业务中检测到另外一个 AVI 程序，也可以被最小化。在这种情况下，存储器的另一存储块被分配，于是 AVI 程序的执行状态和环境，根据它的识别符和持续时间存储在这个存储块中。可以同时被最小化的程序的数目仅受到所需存储包括最小化程序的执行状态和环境的所有存储块数量的限制。

如果没有足够的可用的存储空间用以装入新的目录模块，或者执行由先前装入的目录模块所代表的程序，而且如果有表示最小化的程序已分配了的存储器块，系统装入程序可以自动地根据算法重新分配某些或全部的存储器块（例如首先重新分配最早的存储器块，或者首先重新分配标为由最初的应用可扩展的存储器块）以试图派生充分的存储空间。另外，系统装入程序可为观看者表现出最小化应用的列表，以允许观看者选择某些或删除所有的。代表选择的最小化应用的存储块于是被重新分配以导出充分的存储空间。

这时，包含先前最小化 AVI 程序的执行状态和环境的存储块保持分配在存储器中。如前所述，在每个存储块中有一个持续时间。如在任

何块中持续时间超过了，于是先前的最小化 AVI 程序重复进行。在这种情况下，程序被认作已进入停止状态 69，它的包含执行状态和环境的存储块被重新分配，先前的最小化 AVI 程序的所有记录失掉了。

然而，可以让解码器 10 再次接收包含先前的最小化的 AVI 程序的目录、代码和数据模块的 AVI 程序成分，或让 AVI 程序回到该流程（regain the flow）。例如交互式的商业节目可能已经结束，进入 HALTED 停止状态 69，或者观看者可以改变该频道返回到这个频道。系统装入程序开始装入 AVI 程序成分分组业务中的“新”的目录。每当新的目录装入，应用识别符与包含当前存储在 RAM412 中的状态和环境的所有存储块的识别符相比较。如果相符合的块找到了，于是代码和数据模块被装入，AVI 程序处在 INACTIVE 状态 61，但是它的执行状态在它刚被最小化前被刷新。当观看者按动 ACTIVATE KEY 键时，AVI 程序进入 ACTIVE 状态 63，并且在先前停止处开始执行该程序。用这种方式程序可以暂时停止以进行其他的 AVI 程序，然而不需同时将二个程序所需的充分资源保留在存储器中，而重新开始程序。

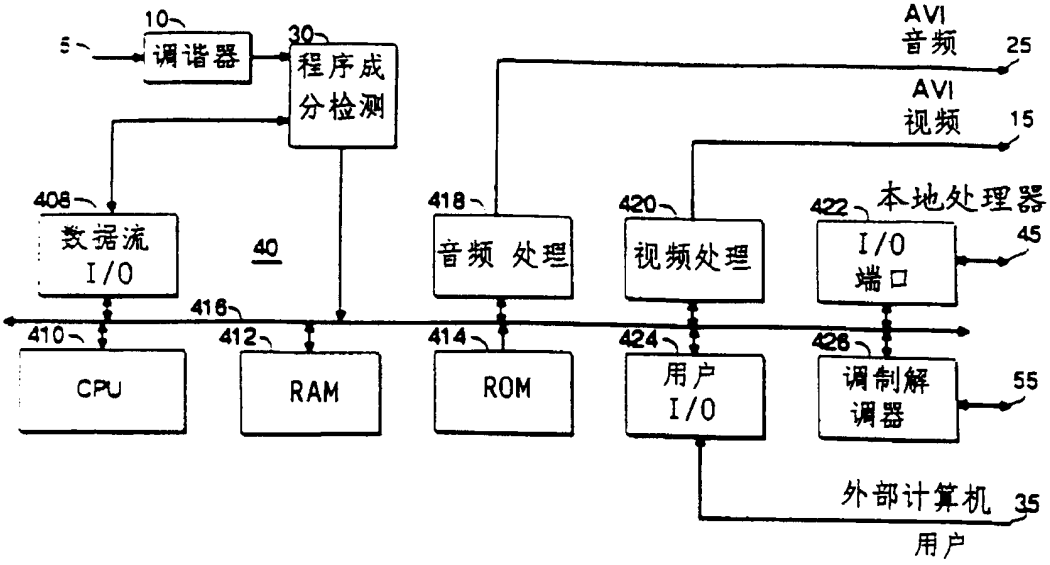

10
5
调谐器
30
程序成分检测
AVI 音频
25
AVI 视频
15
408
数据流 I/O
40
418
音频 处理
420
视频处理
422
本地处理器
I/O 端口
45
416
410
CPU
412
RAM
414
ROM
424
用户 I/O
426
调制解调器
55
外部计算机
35
用户

图 1

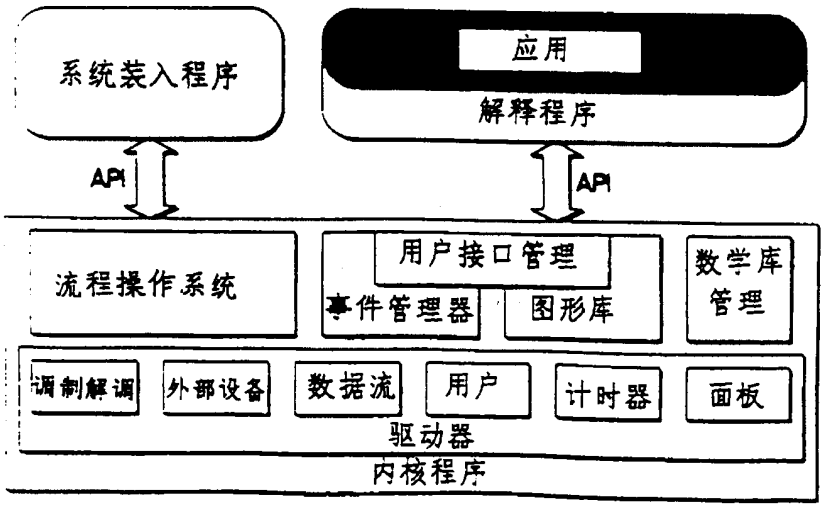

系统装入程序
应用
解释程序
API
API
流程操作系统
用户接口管理
事件管理器
图形库
数学库管理
调制解调
外部设备
数据流
用户
计时器
面板
驱动器
内核程序

图 2

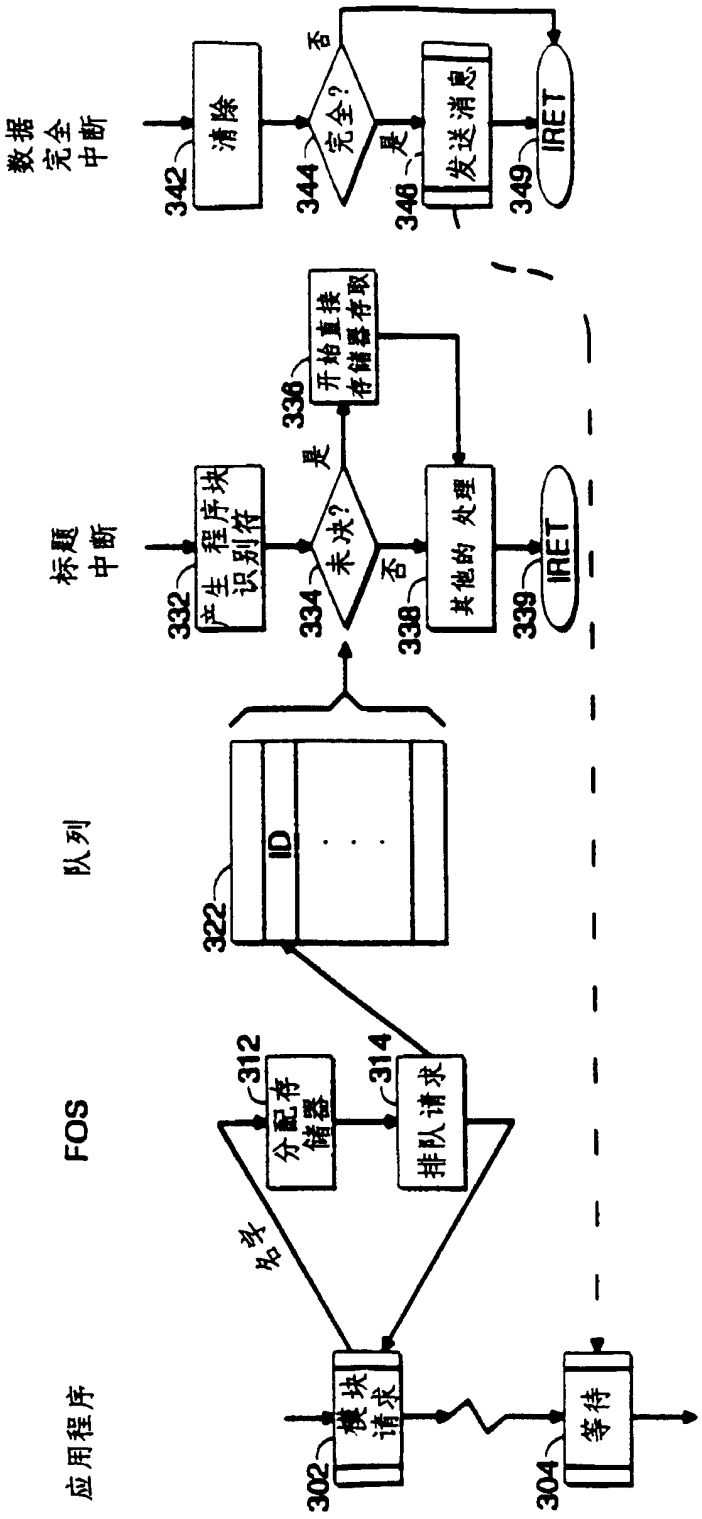

应用程序
FOS
队列
标题中断
数据完全中断
模块请求
302
等待
304
分配存储器
312
排队请求
314
名字
ID
322
产生程序块识别符
332
未决?
334
是
否
开始直接存储器存取
336
其他的 处理
338
IRET
339
清除
342
完全?
344
是
否
发送消息
346
IRET
349

图 3

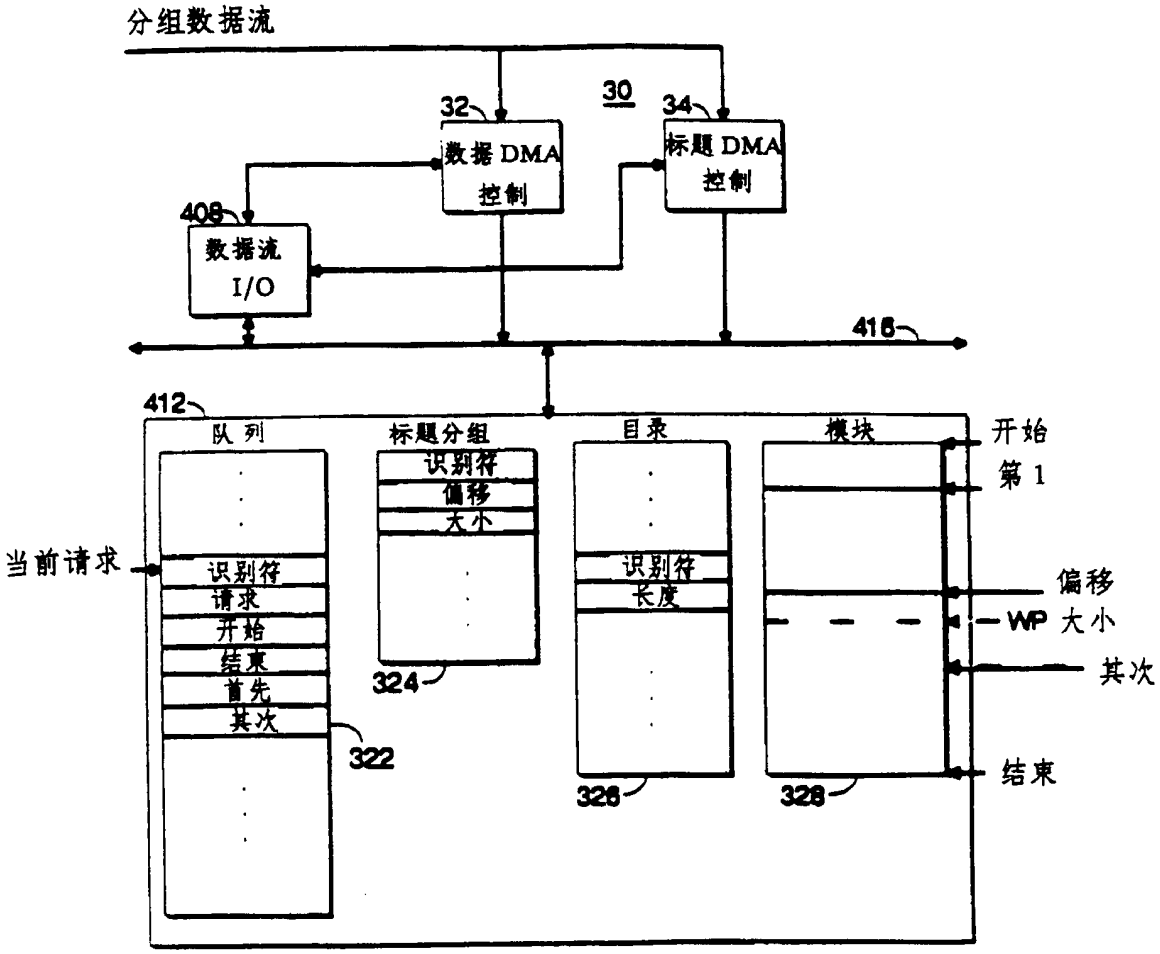

分组数据流
30
32
数据 DMA 控制
34
标题 DMA 控制
408
数据流 I/O
416
412
队列
标题分组
目录
模块
开始
第 1
当前请求
识别符
请求
开始
结束
首先
其次
322
识别符
偏移
大小
324
识别符
长度
326
偏移
WP 大小
其次
结束
328

图 4

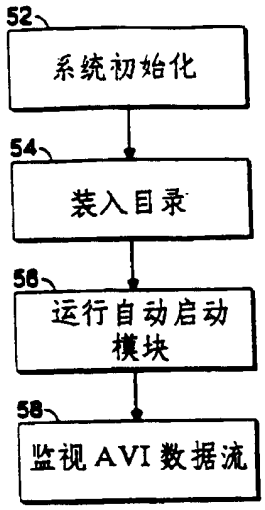

52
系统初始化
54
装入目录
56
运行自动启动模块
58
监视 AVI 数据流

图 5

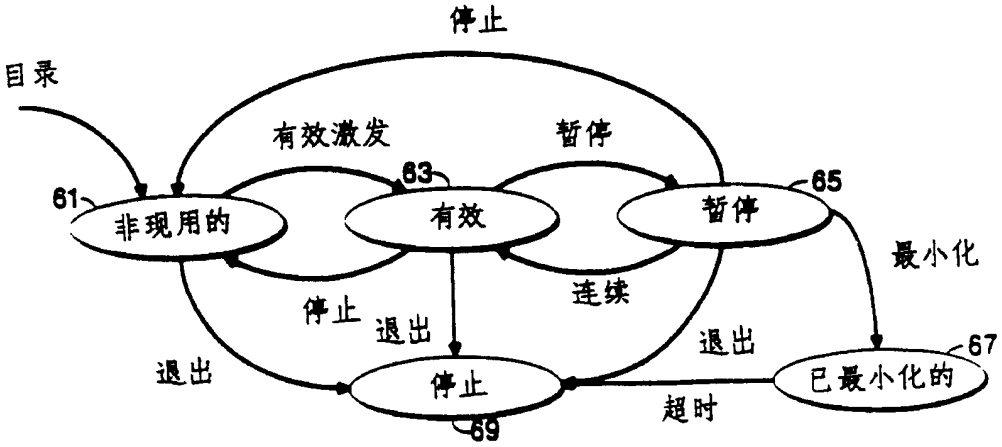

停止
目录
有效激发
暂停
61
非现用的
63
有效
65
暂停
最小化
停止
退出
连续
退出
退出
67
已最小化的
停止
超时
69

图 6