[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H04N 5/92 (2006.01)

[12] 发明专利说明书

专利号 ZL 200480018976.6

[45] 授权公告日 2010 年 1 月 6 日

[11] 授权公告号 CN 100579194C

[22] 申请日 2004.7.5

[21] 申请号 200480018976.6

[30] 优先权

[32] 2003.7.3 [33] US [31] 60/485,207

[86] 国际申请 PCT/JP2004/009873 2004.7.5

[87] 国际公布 WO2005/004478 英 2005.1.13

[85] 进入国家阶段日期 2006.1.4

[73] 专利权人 松下电器产业株式会社

地址 日本大阪府

[72] 发明人 约瑟夫·麦克罗森 冈田智之 持永和宽

[56] 参考文献

EP0924934A1 1999.6.23

US2003/0117529A1 2003.6.26

US6469718B1 2002.10.22

EP1035735A2 2000.9.13

The MPEG-4 systems and description languages: A wayahead in audio visual information representation. AVARO O ET AL. SIGNAL PROCESSING. IMAGE COMMUNICATION, ELSEVIER SCIENCE PUBLISHERS, AMSTERDAM, NL, Vol.9. 1997

审查员 陈柳叶

[74] 专利代理机构 永新专利商标代理有限公司

代理人 王 英

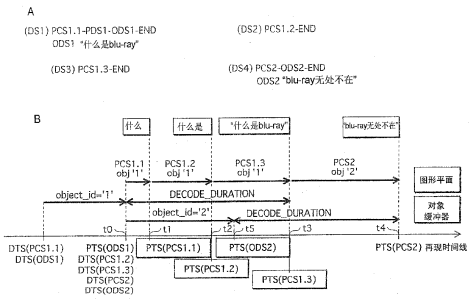
权利要求书 2 页 说明书 73 页 附图 80 页

[54] 发明名称

记录装置、再现装置、记录方法和再现方法

[57] 摘要

通过多路复用图形流和视频流获得记录在 BD-ROM 中的 AV 剪辑。该图形流是 PES 包序列，包括 1) 存储图形数据(ODS)的 PES 包，和 2) 存储控制信息(PCS)的 PES 包。每个 ODS 中，DTS 和 PTS 的值分别指示相应图形数据的解码开始的定时和相应图形数据的解码结束的定时。每个 PCS 中，PTS 的值指示与视频流组合的相应解码图形数据的显示定时。



1、一种再现装置，包括

获得单元，用于获得视频流和图形流，所述图形流包括数据包和控制包；

视频解码器，用于通过解码所述视频流获得运动图片数据；以及

处理器，用于在由所述数据包中的解码时间标记指示的时间开始解码所述数据包中包含的图形数据的过程，并且在由所述数据包中包含的表示时间标记指示的时间之前结束所述图形数据的解码；

控制器，用于在由所述控制包中的表示时间标记指示的时间之前将所述解码后的图形数据写到图形平面中，所述解码后的图形数据与所述运动图片数据组合，并且当所述控制包包含指示时元开始的类型信息时，在由所述控制包中的解码时间标记指示的时间清除所述图形平面。

2、一种记录方法，包括：

生成视频流，所述视频流是通过编码运动图片数据获得的；

生成图形流，所述图形流包含数据包和控制包，在所述数据包中包含第一解码时间标记和第一表示时间标记，所述第一解码时间标记指示开始解码所述数据包中包含的图形数据的过程的时间，并且所述第一表示时间标记指示结束所述图形数据的解码的最后时间，以及，在所述控制包中包含第二解码时间标记和第二表示时间标记，所述第二表示时间标记指示将所述解码后的图形数据写到图形平面中的最后时间，所述解码后的图形数据与所述运动图片数据组合，所述第二解码时间标记指示当所述控制包包含指示时元开始的类型信息时、清除所述图形平面的时间；

生成应用数据，所述应用数据包括所述图形流和所述视频流；

将所述生成的应用数据记录到记录介质中。

3、一种再现方法，包括：

获得视频流和图形流，所述图形流包括数据包和控制包；

解码所述视频流以获得运动图片数据；以及

在由所述数据包中的解码时间标记指示的时间开始解码所述数据包中包含的图形数据的过程，

在由所述数据包中包含的表示时间标记指示的时间之前结束所述图形数据的解码；

在由所述控制包中的表示时间标记指示的时间之前将所述解码后的图形数据写到图形平面中，所述解码后的图形数据与所述运动图片数据组合，以及

当所述控制包包含指示时元开始的类型信息时，在由所述控制包中的解码时间标记指示的时间清除所述图形平面。

4、一种记录装置，包括

生成单元，用于：

生成视频流，所述视频流是通过编码运动图片数据获得的；

生成图形流，所述图形流包含数据包和控制包，在所述数据包中包含第一解码时间标记和第一表示时间标记，所述第一解码时间标记指示开始解码所述数据包中包含的图形数据的过程的时间，并且所述第一表示时间标记指示结束所述图形数据的解码的最后时间，在所述控制包中包含第二解码时间标记和第二表示时间标记，所述第二表示时间标记指示将所述解码后的图形数据写到图形平面中的最后时间，所述解码后的图形数据与所述运动图片数据组合，所述第二解码时间标记指示当所述控制包包含指示时元开始的类型信息时、清除所述图形平面的时间；以及

生成应用数据，所述应用数据包括所述图形流和所述视频流；以及

记录单元，用于将所述生成的应用数据记录到记录介质中。

记录装置、再现装置、记录方法和再现方法

技术领域

本发明涉及一种诸如 BD-ROM 这样的记录介质和一种再现装置。本发明尤其涉及利用图形实现字幕显示和交互显示的技术。

技术背景

利用图形的字幕显示的一个重要任务是将一个作品中通过字符的讲话内容传送给世界各处的人们。用于实现字幕显示的一种传统技术是 ETSI EN 300 743 标准的字幕应用(ETSI：欧洲电信标准协会)。该字幕应用是一种与利用图形的字幕显示一起被再现的视频流。这里，与字幕对应的图形作为 MPEG2 标准的数据流被显示。该数据流是一序列 PES 包，其中每个 PES 包具有一个 PTS(表示时间标记)。ETSI EN 300 743 标准定义字幕应用中字幕显示的时序。该标准在运动图片和图形之间建立同步，其中当显示视频流中对应图像时显示图形。

当为 BD-ROM 提供字幕应用时，需要进一步提高图形的分辨率级别。更具体地，需要将分辨率级别提高到 1920×1080 的级别。然而，实现这么高的分辨率导致再现时的大量解码工作量。

ETSI EN 300 743 标准定义了再现控制，用于在 PTS 指定的时间执行解码，并立刻进行显示。当执行这种操作时，在显示之前再现装置中将集中大量解码工作量。这种工作量的集中迫使再现装置的硬件/软件能力必须很高，以实现图形显示。如果这种条件成为再现装置的关键，则再现装置的生产成本将显著提高，这将会阻止这种再现装置的普遍使用。

发明内容

本发明的目的是提供一种记录介质，其既实现高分辨率级别的图形显示，又避免了生产成本的升高。

为了实现上述目的，本发明提供了一种记录介质，其中存储了通过多路复用图形流和视频流获得的数字流，该图形流是包括存储图形数据的数据包和存储控制信息的控制包的包序列，其中

该数据包具有一个时间标记，其值指示图形数据的解码时间，以及

该控制包具有一个时间标记，其值指示图形数据在解码后与视频流相组合地被显示的时间。

由存储图形的包的时间标记指示图形被解码的周期，且由分配给对应控制信息的时间标记的值定义图形的显示。因此本发明中，再现时间线上定义了“已经解码但还没有显示的状态”即，解压缩图形被缓冲的状态。

通过定义这种缓冲周期，可以避免在一点上集中大量解码工作量。另外，如果用于解码的硬件资源的使用同时与其它处理冲突，则可以提供缓冲周期来重新分配图形解码周期，从而避免这种冲突。

这里，如果引入这种缓冲概念来实现上述目的，则研发这种再现装置的技术人员的困惑将是为保证正常操作所要安装的存储器的扩展。同时，生产字幕应用的技术人员也将感到焦虑的是，它们自己的字幕应用是否可以确保被该再现装置再现。所有这些是因为在该再现时间线上再现的过程中，该缓冲的存储器占用将随时间改变。如果存储器占用的这种随时间改变保持未知，则这些技术人员的焦虑不会消除。

为了解决这个问题，期望具有一种结构，其中控制信息包括指示存储器管理开始的类型信息，控制包的时间标记是表示时间标记，以及控制包还包括解码时间标记，其值指示该数字流的再现时间线上对应于存储器管理开始的点，和控制信息被读到存储器的时间。

根据该结构，由存储控制信息的包的解码时间标记指示存储器管理的开始。从而，通过参考解码时间标记，可以知道应该在再现时间线的哪一点清除(flash)解码器模型的每个缓冲器。如果认为该清除点

是存储器管理的开始点，则容易掌握存储控制信息的缓冲器、存储解码前图形的缓冲器和存储解码后图形的缓冲器的随时间的占用随时间的变化。通过改变该解码时间标记的值，可以调节缓冲器状态的随时间变化。根据这种调节，可以避免再现装置中的缓冲器溢处。因此，在再现装置的开发阶段实现硬件/软件变得容易。

另外，因为掌握和调整随时间的变化变得容易，所以验证由创作获得的图形流是否满足 BD-ROM 标准假定的解码器模型的限制相应变得容易。从而，负责创作的人可以在确保他创作的图形被正常操作的条件下，继续进行创作工作。

假定 BD-ROM 的解码器模型时，为了实施本发明，另一个构成要素必不可少。在 BD-ROM 的解码器模型中，图形的解码器主体(即，处理器)与用于更新图形的控制器主体(即，控制器)相互独立。与更新控制器主体独立地提供解码器主体的原因是为了执行诸如逐渐地显示和删除图形这样的改进更新，这在例如图形是字幕的情况下是有用的。当更新控制器主体与解码器主体相互独立时，处理器和控制器的连接需要更接近。这是因为，当处理器完成图形数据的解码后，控制器必须没有延迟地执行更新。

将处理器的解码完成通知给控制器的方式取决于在装置中实现处理器和控制器的方式。如果处理器和控制器作为程序实现，则将通过进程内部(intra-process)通信执行通知。如果处理器和控制器作为相互独立的硬件单元实现，那么由中断信号执行通知。这种通知的延时量也取决于该装置中的实现方式。如果该实现的通知需要很大延时，则将产生图形更新不能与运动图片的显示速率同步的情况。

为了防止这种情况发生，期望具有一种构造，其中通过使解码时间标记的值加一个预定值得到表示时间标记的值，其中该预定值基于：清除屏幕所需要的周期、解码图形数据所需要的周期和向该屏幕写该图形数据所需要的周期中较长的一个周期。

存储图形的包的表示时间标记指示解码结束时间，存储控制信息的包的表示时间标记指示通过使该解码结束时间加一个预定周期获得的时间。从而只有通过参考该表示时间标记，控制器才可以在恰当

的定时处执行更新，而不从处理器接收任何图形数据的解码完成通知。如果执行这种更新，则无论再现装置中实现的方式如何，可以确保与运动图片的显示速率同步地更新。

由于不管再现装置中处理器和控制器实现的方式来实现较近的处理器和控制器连接，所以变得可以保持装置设计中一定程度的灵活性，并有助于以低成本制造装置。

附图说明

图 1 示出了根据本发明的存储介质的使用的例子；

图 2 示出了 BD-ROM 的结构；

图 3 示意性示出了 AV 剪辑的结构；

图 4A 示出了显示图像流的结构；

图 4B 示出了功能段被转换后获得的 PES 包；

图 5 示出了由各种类型的功能段组成的逻辑结构；

图 6 示出了字幕显示位置与信号出现时间之间的关系；

图 7A 示出了在对象定义段(ODS)中定义图形对象的语法；

图 7B 示出了调色板定义段(PDS)的语法；

图 8A 示出了窗口定义段(WDS)的语法；

图 8B 示出了表示组合段(PCS)的语法；

图 9 示出了字幕的显示集合的描述的例子；

图 10 示出了在 DS1 中 WDS 和 PCS 的描述的例子；

图 11 示出了在 DS2 中 PCS 的描述的例子；

图 12 示出了在 DS3 中 PCS 的描述的例子；

图 13 沿时间线示出了当执行 Cut-in/out 时显示集合的描述的例子；

图 14 沿时间线示出了当执行 Fade-in/out 时显示集合的描述的例子；

图 15 沿时间线示出了当执行 scrolling 时显示集合的描述的例子；

图 16 沿时间线示出了当执行 wipe-in/out 时显示集合的描述的例子；

图 17 比较了两种情况：一个窗口具有四个图形对象，和一个窗口具有两个图形对象；

图 18 示出了用于计算 decode_duration 的算法的例子；

图 19 是图 18 的算法的流程图；

图 20A 和 B 是图 18 的算法的流程图；

图 21A 示出了其中每个窗口具有一个对象定义段的情况；

图 21B 和 C 是时序图，示出了图 18 中指出的标号的次序；

图 22A 示出了其中每个窗口具有两个对象定义段的情况；

图 22B 和 C 是时序图，示出了图 18 中指出的标号的次序；

图23A描述了其中两个窗口中的每个窗口包括一个ODS的情况；

图 23B 示出了其中解码周期(2)比清除周期(1)和写周期(31)的总和长的情况；

图 23C 示出了其中清除周期(1)和写周期(31)的总和比解码周期(2)长的情况；

图 24 示出了本说明书中一个例子中描述的更新随时间的变化；

图 25A 示出了被描述成执行上述更新的四个显示集合；

图 25B 是表示包含在四个显示集合中的功能端的 DTS 和 PTS 的设置的时序图；

图 26 示出了根据本发明的再现装置的内部结构；

图 27 示出了写速率 Rx、Rc 和 Rd、图形平面 8、编码数据缓冲器 13、对象缓冲器 15 和合成缓冲器 16 的大小；

图 28 示出了再现装置的流水线处理的时序图；

图 29 示出了在图形平面的清除完成以前 ODS 的解码结束的情况下流水线处理的时序图；

图 30 是表示图形平面 8 上积累的量按时间顺序转变的时序图；

图 31 是表示功能段的装载操作的处理流程图；

图 32 表示多路复用的一个例子；

图 33 示出了 DS10 被装载再现装置的编码数据缓冲器 13 的方式；

图 34 示出了执行正常再现的情况；

图 35 示出了在图 34 中执行的正常再现中 DS1、DS10、和 DS20

的装载；

图 36 是表示由图形控制器 17 执行的处理的流程图；

图 37 是表示由图形控制器 17 执行的处理的流程图；

图 38 是表示由图形控制器 17 执行的处理的流程图；

图 39 示出了再现装置基于 PDS 的 PTS 的流水线处理；

图 40 是描述再现装置的流水线处理中 END 的意义的示意图；

图 41 示意性示出了根据第二实施例的 AV 剪辑的结构；

图 42A 和图 42B 是关于根据第二实施例的交互屏幕的示意图；

图 43 示出了交互合成段的数据结构；

图 44 表示包含在 DSn 中的 ODS 与 ICS 之间的关系；

图 45 表示任意图片数据“pt1”的显示定时处的屏幕合成；

图 46 表示 ICS 中按钮信息集合的例子；

图 47 示出了按钮 A-D 的状态转变；

图 48 示出了 ODS11、21、31 和 41 的图像，作为一个例子；

图 49 示出了用于按钮 A 的 ODS11-19 的图像，作为一个例子；

图 50 示出了显示集合中的按钮状态组和 ODS 的次序；

图 51 示出了布置了图 50 的按钮状态组的交互屏幕的状态转变；

图 52 示出了显示集合中 ODS 的次序；

图 53 表示 default_selected_button_number=0 的情况与 default_selected_button_number=B 的情况之间，S-ODSs 中 ODS 排列的不同；

图 54A 和 54B 表示当 N-ODSs 包含构成按钮 A-D 的多个 ODS 和 S-ODSs 包含构成按钮 A-D 的多个 ODS 的情况下，ΣSIZE (DSn [ICS. BUTTON [i]])的值；

图 55 表示依靠 ICS 的同步显示的时序；

图 56 表示由多个 ODS 构成交互屏幕的初始显示和默认选中按钮为有效的情况下如何设置 DTS 和 PTS；

图 57 表示由多个 ODS 构成交互屏幕的初始显示和默认选中按钮为无效的情况下如何设置 DTS 和 PTS；

图 58 示出了对象缓冲器 15 的内容与图形平面 8 的比较；

图 59 示出了在初始显示时间由图形控制器 17 执行的操作；

图 60 示出了当根据第一个用户动作(右移)执行交互屏幕更新时，由图形控制器 17 执行的操作；

图 61 示出了当根据第一个用户动作(下移)执行交互屏幕更新时，由图形控制器 17 执行的操作；

图 62 示出了当根据第一个用户动作(激活)执行交互屏幕更新时，由图形控制器 17 执行的操作；

图 63 是表示由再现装置执行的流水线处理的时序图；

图 64 是表示动态地改变默认选中按钮的情况下由再现装置执行的流水线处理的时序图；

图 65 是表示图形平面 8、对象缓冲器 15、编码数据缓冲器 13 和合成缓冲器 16 的占用按时间顺序转变的时序图；

图 66 是表示段的装载操作的处理流程图；

图 67 表示多路复用的一个例子；

图 68 示出了 DS10 被装载再现装置的编码数据缓冲器 13 的一种方式；

图 69 示出了执行正常再现的情况；

图 70 示出了在如图 69 中执行的正常再现中，DS1、DS10 和 DS20 的装载；

图 71 是表示图形控制器 17 执行的处理的主程序的流程图；

图 72 是实现使用时间标记的同步控制的处理流程图；

图 73 是表示向图形平面 8 写操作的处理流程图；

图 74 是表示默认选中按钮的自动激活处理的流程图；

图 75 是表示动画显示处理的流程图；

图 76 是表示当前按钮改变操作的处理流程图；

图 78 是表示数值输入操作的处理流程图；

图 79 示出了制造用于记录第一实施例中说明的 PCS 的 BD-ROM 的方法；

图 80 示出了制造用于记录第二实施例中说明的 PCS 的 BD-ROM 的方法。

本发明的优选实施方式

(第一实施例)

以下说明根据本发明的记录介质的第一实施例。

图 1 示出了使用该记录介质的一个例子。图中，BD-ROM 100 是根据本发明的记录介质。BD-ROM 100 用于向由再现装置 200、电视 300 和遥控器 400 构成的家庭影院系统提供电影作品数据。

通过改进 BD-ROM 的应用层制造了根据本发明的记录介质。图 2 示出了 BD-ROM 的结构。

该图中，图底部示出了 BD-ROM，其上示出了该 BD-ROM 上的轨道。轨道在盘上实际是螺旋形状，但在图中表示为直线。轨道包括导入区、容量区和导出区。该图中的容量区具有物理层、文件系统层和应用层。在该图上部利用目录结构示出了 BD-ROM 的应用格式。如图所示，BD-ROM 在根目录下具有目录 BDMV，且 BDMV 目录包含以扩展名 M2TS 存储 AV 剪辑的文件(XXX. M2TS)、以扩展名 CLPI 存储管理信息的文件(XXX. CLPI)、和以扩展名 MPLS 为 AV 剪辑定义逻辑播放列表(PL)的文件(YYY. MPLS)。通过形成上述应用格式，可以制造根据本发明的记录介质。每种类型具有一个以上文件的情况下，优选地在 BDMV 下提供名为 STREAM、CLIPINF 和 PLAYLIST 的三个目录，将具有相同扩展名的文件存储在一个目录中。具体地，期望将扩展名为 M2TS 的文件存储在 STREAM 中，将扩展名为 CLPI 的文件存储在 CLIPINF 中，将扩展名为 MPLS 的文件存储在 PLAYLIST 中。

以下说明上述应用格式中的 AV 剪辑 (XXX. M2TS)。

AV 剪辑 (XXX. M2TS)是通过多路复用视频流、至少一个音频流和表示图形流得到的 MPEG-TS(TS 是传输流)格式的数据流。视频流代表电影中的图片，音频流代表电影中的声音，表示图形流代表电影中的字幕。图 3 示意性示出了 AV 剪辑的结构。

AV 剪辑 (XXX. M2TS)按照以下方式构成。由多个视频帧(图片 pj1、 pj2 和 pj3)组成的视频流和由多个音频帧(图中顶行)组成的音频

流被分别变换成一行 PES 包(图中第二行)，然后被变换成一行 TS 包(图中第三行)。表示图形流(图中底行)被变换成一行 PES 包(图中下数第二行)，然后被变换成一行 TS 包(图中下数第三行)。多路复用这三行 TS 包，构成 AV 剪辑(XXX. M2TS)。

该图中，仅多路复用了一个表示图形流。然而，BD-ROM 兼容多种语言的情况下，每种语言的表示图形流都被多路复用以构成 AV 剪辑。以上述方式构成的 AV 剪辑如同普通计算机文件被分成多于一个区(extent)，并被存储在 BD-ROM 的区域内。

接下来说明表示图形流。图 4A 示出了表示图形流的结构。顶行表示要被多路复用到 AV 剪辑 中的一行 TS 包。第二行表示构成图形流的一行 PES 包。通过从具有预定 PID 的 TS 包中取出净荷并连接所取出的净荷来构成该行 PES 包。

第三行表示该图形流的结构。该图形流由称为合成表示合成段(PCS)、窗口定义段(WDS)、调色板定义段 (PDS)、对象定义段(ODS)和显示集合段的结束(END)的这些功能段组成。以上这些功能段中，PCS 被称为屏幕合成段，WDS、PDS、ODS 和 END 被称为定义段。PES 包和每个功能段一对一的对应或一对多的对应。即，或者一个功能段在变换成一个 PES 包后被记录到 BD-ROM，或者一个功能段在分成片段并变换成多于一个 PES 包后被记录到 BD-ROM。

图 4B 示出了通过变换功能段得到的 PES 包。如图所示，PES 包由包头和净荷组成，且净荷是功能段的实质主体。包头包括与功能段对应的 DTS 和 PTS。包含在包头中的 DTS 和 PTS 在下文中被称为功能段的 DTS 和 PTS。

上述各种类型的功能段构成了如图 5 所示的逻辑结构。图 5 示出了由各种类型功能段组成的逻辑结构。该图中，顶行示出了时元时元(Epoch)，中间行示出了显示集合(DS)，底行示出了功能段。

中间行示出的每个 DS 是一组功能段，在构成图形流的所有功能段中，这一组功能段组成一个屏幕的图形。图中虚线表示底行中的功能段所属于的 DS，并表示一系列功能段 PCS、WDS、PDS、ODS 和 END 构成一个 DS。再现装置通过读取构成该 DS 的这些功能段能够

为一个屏幕产生图形。

顶行中的时元指示时间周期，且存储器管理是沿一个时元中 AV 剪辑再现时间线的连续时间方式。一个时元还代表分配到同一时间周期的一组数据。这里所指的存储器是为一个屏幕存储图形的图形平面(Graphics Plane)和存储解压缩图形数据的对象缓冲器。存储器管理的连续性表明在时元中不会发生图形平面或对象缓冲器的闪除(flash)，且仅在图形平面上预定矩形区域内执行图形的擦除和呈现(这里，闪除表示清除平面或缓冲器内存储数据的所有内容)。一个时元期间举行区域的大小和位置是固定的。只要仅在图形平面的预定矩形区域内执行图形的擦除和呈现，就能保证所述图片和所述图形之间的同步再现。即，时元是再现时间线中的一个单元，在该单元中确保同步再现所述图片和所述图形。当将其中图形被擦除和呈现的区域移动到不同位置时，必须在时间线上定义一点来移动该区域，且该点之后的周期成为新的时元。在两个时元之间的边界处不保证同步再现。

观看真实电影时，一个时元是在屏幕上同一矩形区域内显示字幕的时间周期。图 6 示出了字幕位置与时元之间的关系。在该图示出的例子中，五个字幕“实际上...”、“我隐藏了”、“我的感情”、“我一直”和“爱你。”被示出的位置随着电影中的图片移动。具体地，字幕“实际上...”、“我隐藏了”、“我的感情”出现在屏幕底部，而字幕“我一直”和“爱你。”显示在屏幕顶部。考虑到电影的可视性，移动矩形区域的位置是为了当观看屏幕是字幕不妨碍图片。字幕出现在底部的时间周期是时元 1，字幕出现在顶部的下一时间周期是时元 2。时元 1 and 2 具有不同的呈现字幕的区域。时元 1 中的区域是位于屏幕底部的窗口 1，时元 2 中的区域是位于屏幕顶部的窗口 2。在时元 1 and 2 的每一个中存储器管理是连续的，从而窗口 1 和 2 中字幕的呈现与图片同步。

接下来描述显示集合(DS)的细节。

图 5 中虚线 hkll 和 hkl2 表示中间行的哪个功能段属于哪个时元。一系列 DS “时元开始”、 “获取点”和“正常情况”组成了顶行的时元。“时元开始”、“获取点”、“正常情况”和“时元继续”是 DS

的类型，“获取点”与“正常情况”之间的次序不重要，它们中任一个都可以首先出现。

“时元开始”是具有“新显示”显示效果的一种 DS，指是一个新时元的开始。因此，“时元开始”包含显示新的屏幕合成所需要的所有功能段。“时元开始”被提供在 AV 剪辑的跳转操作目标的位置上，例如电影的一章。

“获取点”是具有“显示刷新”显示效果的一种 DS，其用于呈现图形的内容与作为先前 DS 的“时元开始”相同。“获取点”不提供在时元的开始点处，但是它包含显示新的屏幕合成所需要的所有功能段。所以，当执行跳转到获取点的操作时，可以确保显示图形。从而，利用获取点，可以在时元的中间合成屏幕。

获取点被提供在可以是跳转操作目标的位置。这种位置的一个例子是当执行时间搜索时可以被指定的位置。时间搜索是对应于用户输入一个时间以从与该用户指定的时间对应的再现点开始再现的操作。粗略地指定该时间，例如 10 分钟或 10 秒钟，相应地在例如 10 分钟间隔或 10 秒钟间隔处提供再现开始的点。通过在再现可以开始的点提供获取点，可以在时间搜索之后平稳地执行再现。

“正常情况”是具有“显示更新”显示效果的一种 DS，且只包含与先前屏幕合成不同的元素。具体地，当 DSv 中的字幕与 DSu 中的字幕相同，但是屏幕在 DSv 和 DSu 中被不同地显示时，提供仅包含 PCS 的 DSv，并使 DSv 为“正常情况”。这样，就不再需要提供与先前 DS 中 ODS 的内容相同的 ODS，并可以减小 BD-ROM 中的数据量。另一方面，因为“正常情况”的 DS 仅包含不同内容，所以不可能单独使用“正常情况”来合成屏幕。

“时元继续”表示时元跨过 AV 剪辑的边界继续。如果一个 DSn 的合成状态被设置为时元继续，则如果该 DSn 所在的 AV 剪辑与该 DSn 之前的 DSn-1 所在的 AV 剪辑不同，DSn 和 DSn-1 将属于同一个时元。从而，即使在这两个 DS 之间出现 AV 剪辑转移，也不会有图形平面/对象缓冲器的闪除。

以下详细说明定义段(ODS、WDS 和 PDS)的细节。对象定义段

(ODS)是定义图形对象的功能段。首先说明图形对象。记录在BD-ROM中的AV剪辑的卖点是它的分辨率与高清电视一样高，从而图形对象的分辨率设置为1920×1080像素。因为该1920×1080像素的高分辨率，所以可以在屏幕上清楚地显示字幕的特定字体。关于字幕的颜色，每个像素(红色差Cr、蓝色差Cb、亮度Y和透明度T)的给定值的比特长度是8比特，从而可以为字幕从全部颜色(16,777,216色)中选择任何256种颜色。通过将文本定位在透明背景上来呈现由图形对象实现的字幕。

图7A示出了用于定义图形对象的ODS语法。ODS由以下组成：指示该段为ODS的segment_type、指示ODS数据长度的segment_length、唯一识别与时元中该ODS对应的图形对象的object_id、指示时元中ODS版本object_version_number、last_in_sequence_flag、和与一部分或全部图形对象对应的字节的连续序列object_data_fragment。

object_id用于唯一地识别与时元中该ODS对应的图形对象。该图形流的时元包含一个以上具有相同ID(标识符)的OSD。具有相同ID的ODS还具有相同的宽和高，并在对象缓冲器中被分配了公共区域。具有相同ID的ODS中的一个在该公共区域中被读取后，下一个具有相同ID的ODS盖写该读取的ODS。随着视频流再现的进行，通过由该下一个具有相同ID的ODS盖写该读取到对象缓冲器的ODS，相应地更新了ODS的图形。具有相同ID的图形对象的宽和高应该相同，这个尺寸限制仅应用于一个时元期间，且不同时元中的图形对象可以具有不同大小。

接下来说明last_in_sequence_flag和object_data_fragment。一些情况下，由于PES包的净荷限制，不可能在一个ODS中存储构成字幕的解压缩图形。这种情况下，将该图形分成一系列连续的片段，将一个片段设置为object_data_fragment。当以多于一个片段来存储一个图形对象时，除最后一个片段之外的每个片段具有相同大小。该最后一个片段小于或等于先前片段的大小。携带这些片段的ODS以相同序列次序出现在DS中，由具有last_in_sequence_flag的ODS指示该

序列的结束。虽然上述 ODS 的语法基于的前提是从先前 PES 堆叠这些片段，但是这些片段可以被堆叠成使得每个 PES 包含空白部分。

接下来说明调色板定义段(PDS)。使用 PDS 定义用于颜色转换的调色板。图 7B 示出了 PDS 的语法。PDS 由以下组成：指示该段为 PDS 的 segment_type、指示 PDS 数据长度的 segment_length、唯一识别该 PDS 中包含的调色板的 palette_id、指示时元中 PDS 版本的 palette_version_number、和指定调色板项目编号的 palette_entry_id 组成。palette_entry_id 指示红色差(Cr_value)、蓝色差(Cb_value)、亮度(Y_value)和透明度(T_value)。

接下来说明窗口定义段(WDS)。

使用 WDS 在图形平面上定义矩形区域。如上所述，只有当在图形平面的特定区域内执行擦除和呈现时存储器管理才是顺序的。图形平面上的该区域由 WDS 定义，称为“窗口”。图 8A 示出了 WDS 的语法。如图所示，WDS 由以下组成：指示该段是 WDS 的 segment_type、指示 WDS 数据长度的 segment_length、在图形平面上唯一识别该窗口的 window_id、指定在图形平面上窗口左上角像素的水平地址的 window_horizontal_position、指定在图形平面上窗口左上角像素的垂直地址的 window_vertical _position、指定在图形平面上窗口的宽度的 window_width 和指定在图形平面上窗口的高度的 window_height。

以下说明 window_horizontal_position、window_vertical _position、window_width 和 window_height 可以取值的范围。这些值的坐标系在图形平面上一个区域内，且其大小由关于高度的 window_height 和关于宽度的 window_width 两维地表示。

window_horizontal_position 指定图形平面上窗口左上角像素的水平地址，且在 0 至(window_width)-1 范围内。而且，window_vertical _position 指定图形平面上窗口左上角像素的垂直地址，在 0 至(window_height)-1 范围内。

window_width 指定图形平面上窗口的宽度。该指定宽度在 1 至(video_width)-(window_horizontal_position)范围内。另外，window_height 指定图形平面上窗口的高度，且该指定高度在 1 至(video

_height)-(window_horizontal_position)范围内。

对于每个时元，图形平面上窗口的位置和大小由 window_horizontal_position、window_vertical _position、window_width 和 window_height 定义。从而，在创作时可以调节窗口的位置和大小，使得在一个时元中窗口出现在当观看电影时不妨碍图片的位置。这样，字幕的可视性更好。因为对于每个时元都定义 WDS，所以即使图片随时间改变，可以根据图片调节窗口的位置。结果，可以保持电影质量与将字幕合并在电影主体中的情况下一样高。

接下来说明显示集合段的结束(END)。END 指示 DS 的传输已经完成。在一个 DS 中，END 在最后一个 ODS 之后插入流中。END 由指示该段是 END 的 segment_type 和指示 END 数据长度的 segment_length 组成。END 不包括需要进一步说明的其它元素。

接下来说明表示合成段(PCS)。

PCS 是一个用于合成交互显示的功能段。图 8B 示出了 PCS 的语法。如图所示，PCS 由 segment_type、segment_length、composition_number、composition_state、palette_update_flag、palette_id 和 composition_object 1-m 组成。

composition_number 利用 0 至 15 范围的值识别 DS 的图形更新。如果在时元的头和 PCS 之间存在图形更新，则每次图形更新发生都会是 composition_number 增加。

composition_state 指示包含 PCS 的 DS 的类型：正常情况、获取点或时元开始。

palette_update_flag 指示 PCS 描述“仅显示更新的调色板”。该“仅显示更新的调色板”指示只有该调色板从前一个调色板更新。如果执行“仅显示更新的调色板”，则 palette_update_flag 字段被设置为 1。

palette_id 用于识别“仅显示更新的调色板”中使用的调色板。

composition_object 1-m 指示如何控制该 PCS 所属于的 DS 中的每个窗口。图 8B 中虚线 wdl 详细说明了 composition_object i 的内部语法。composition_object i 由 object_id、window_id、object _cropped_flag、object_horizontal_position、object_vertical_position 和 cropping_

rectangle_information 1-n 组成。

object_id 识别与 composition_object i 对应的窗口中的 ODS。

window_id 识别 PCS 中为图形对象分配的窗口。一个窗口中可以分配总共两个图形对象。

object_cropped_flag 用于在对象缓冲器中显示和不显示剪切图形对象之间进行切换。当 object_cropped_flag 设置为“1”时，在对象缓冲器中显示剪切图形对象，如果设置为“0”，则不显示该图形对象。

object_horizontal_position 指定图形平面中该图形对象左上角像素的水平地址。

object_vertical_position 指定图形平面中该图形对象左上角像素的垂直地址。

cropping_ rectangle_information 1-n 是当 object_cropped_flag 被设置为“1”时使用的元素。虚线 wd2 详细说明了 cropping_ rectangle_information i 的内部语法。如虚线 wd2 所示，cropping_ rectangle_information i 由四个字段组成，分别是 object_cropping_ horizontal_position、object_ cropping_vertical_position、object_cropping _width 和 object_cropping_height。

object_cropping_horizontal_position 指定在图形平面中呈现图形对象期间使用的剪切矩形左上角的水平地址。该剪切矩形是用于指定和剪切一部分图形对象的剪切框，与 ETSI EN 300 743 标准中的区域(Region)对应。

object_cropping_vertical_position 指定在图形平面中呈现图形对象期间使用的剪切矩形左上角的垂直地址。

object_cropping _width 指定剪切矩形的宽度。

object_cropping_height 指定剪切矩形的高度。

以下详细说明 PCS 的一个具体例子。该例中，通过随着图片的进行向图形平面中写入 3 次，逐个出现图 6 中所示的字幕"实际上..."、"我隐藏了"和"我的感情。"。图 9 是描述实现这种字幕显示的一个例子。该图中一个时元包括 DS1(时元开始)、DS2(正常情况)和 DS3(正常情况)。DS1 包含一个 WDS、一个 ODS 和第一 PCS，其中 WDS

用于指定其中显示字幕的窗口，ODS 用于指定"Actually... I was hiding my feelings. "行。DS2 包含第二 PCS，DS3 包含第三 PCS。

图 10-12 示出了 DS 中包含的 WDS 和 PCS 的例子。图 10 示出了 DS1 中 PCS 的例子。

图 10 中，WDS 的 window_horizontal_position 和 window_vertical _position 由图形平面上窗口左上角像素的位置 LP1 表示。window_idth 和 window_eight 分别表示窗口的宽度和高度。

图 10 中，object_cropping_horizontal_position 和 object_cropping_ vertical_position 指示坐标系中剪切矩形的参考点 ST1，该坐标系中原点是图形对象的左上角像素。该剪切矩形是这样一个区域，该区域具有从 ST 至 object_cropping_width 的宽度和从 ST 至 object_cropping_height 的高度(粗线框所示的矩形)。该剪切图形对象位于虚线框 cpl 所示的矩形内，其参考点在图形平面中以 window_horizontal_position 和 window_vertical _position(图形对象的左上角像素)为原点的坐标系中。这样，字幕"实际上..."被写到图形平面上的窗口中，然后与电影图片组合并显示在屏幕上。

图 11 示出了 DS2 中 PCS 的一个例子。因为 DS2 中的 WDS 与 DS1 中的 WDS 相同，所以不对 DS2 中的 WDS 进行说明。DS2 中剪切信息的描述与图 10 中所示的剪切信息的描述不同。

图 11 中，剪切信息中的 object_cropping_horizontal_position 和 object_cropping_vertical_position 指示对象缓冲器中“实际上...我隐藏了我的感情。”中的字幕“我隐藏了”的右上角像素。object_cropping_width 和 object_cropping_height 指示包含字幕“我隐藏了”的矩形的宽度和高度。这样，字幕“我隐藏了”被写入图形平面上的窗口中，然后与电影图片组合并显示在屏幕上。

图 12 示出了 DS3 中 PCS 的一个例子。因为 DS3 中的 WDS 与 DS1 中的 WDS 相同，所以不对 DS3 中的 WDS 进行说明。DS3 中剪切信息的描述与图 10 中所示的剪切信息的描述不同。

图 12 中，剪切信息中的 object_cropping_horizontal_position 和 object_cropping_vertical_position 指示对象缓冲器中“实际上...我隐藏

了我的感情。”中的字幕“我的感情”的右上角像素。object_cropping_width 和 object_cropping_height 表示包含字幕“我的感情”的矩形的宽度和高度。这样，字幕“我的感情”被写入图形平面上的窗口中，然后与电影图片组合并显示在屏幕上。

通过描述以上 DS1、DS2 和 DS3，可以实现在屏幕上显示字幕的效果。还可以实现其它效果，以下说明用于实现其它效果的描述协议。

首先说明切入/切出(Cut-In/Out)效果的描述协议。图 13 沿时间线示出了当执行切入/切出时 DS 的描述的例子。

该图中，窗口(x, y, u, v)中的 x 和 y 分别表示 window_vertical_position 和 window_horizontal _position 的值，u 和 v 分别表示 window_width 和 window_height 的值。该图中，剪切矩形(a, b, c, d)中的 a 和 b 分别表示 object_cropping_ vertical_position 和 object_cropping_horizontal_position 的值，c 和 d 分别表示 object_cropping_width 和 object_cropping_height 的值。显示集合 DS11、DS12 和 DS13 位于该图中再现时间线上点 t11、t12 和 t13 处。

点 t11 处的 DS11 包括：PCS#0，其中 composition_state 是“时元开始”且 object_cropped_flag 是“0”(no_cropping_rectangle _visible)；WDS#0，具有在图形平面中(100,100)处宽度 700×高度 500 的窗口的声明；PDS#0；ODS#0，表示字幕“Credits:”；和 END(结束)。

点 t12 处的 DS12 包括 PCS#1，其中 composition_state 是“正常情况”，并指示在对象缓冲器中对于从(0,0)点处 600×400 大小的图形对象的剪切操作(cropping_rectangle#0 (0,0,600,400))，且将该剪切图形对象定位在图形平面中坐标(0,0)处(on Window#0(0, 0))。

点 t13 处的 DS13 包括 PCS#2，其中 composition_state 是“正常情况”，且其中 object_cropped_flag 被设置为“0”，从而擦除该剪切图形对象(no_cropping_rectangle_visible)。

通过上述显示集合，字幕“Credits:”在 t11 没有显示，在 t12 出现，然后在 t13 再次没有显示，从而实现切入/切出效果。

其次，说明用于淡入/淡出(Fade-In/Out)效果的描述协议。图 14 沿时间线示出了当执行淡入/淡出时 DS 的描述的例子。该图中，显示

集合 DS21、DS22、DS23 和 DS24 位于再现时间线上点 t21、t22、t23 和 t24 处。

点 t21 处的 DS21 包括：PCS#0，其中 composition_state 是“时元开始”，并指示对象缓冲器中对于(0,0)点处 600×400 大小的图形对象的剪切操作(cropping_rectangle#0 (0,0,600,400))，且将该剪切图形对象定位在图形平面中坐标(0,0)处(on Window#0(0, 0))；WDS#0，具有在图形平面中(100,100)处宽度 700×高度 500 的窗口的声明；PDS#0；ODS#0，指示字幕“Fin”；和 END(结束)。

点 t22 处的 DS22 包括：PCS#1，其中 composition_state 是“正常情况”；和 PDS#1，指示与 PDS#0 相同级别的 Cr 和 Cb，但是由 PDS#1 指示的亮度高于 PDS#0 中的亮度。

点 t23 处的 DS23 包括 PCS#2、PDS#2 和 END。PCS#2 的 composition_state 是“正常情况”；PDS#2 指示与 PDS#1 相同级别的 Cr 和 Cb，但是由 PDS#2 指示的亮度低于 PDS#1 中的亮度。

点 t24 处的 DS24 包括一个 PCS，其 composition_state 是“正常情况”，且 object_cropped_flag 是“0”(no_cropping_rectangle_visible)，该 DS24 还包括 END。

每个 DS 指定与先前 DS 不同的 PDS，从而在一个时元中以多于一个 PCS 呈现的图形对象的亮度逐渐变高或变低。这样，可以实现淡入/淡出效果。

接下来说明用于滚动(Scrolling)的描述协议。图 15 沿时间线示出了当执行滚动时 DS 的描述的一个例子。该图中，显示集合 DS31、DS32、DS33 和 DS34 位于再现时间线上点 t31、t32、t33 和 t34 处。

点 t31 处的 DS31 包括：PCS#0，其 composition_state 被设置为“时元开始”，且 object_cropped_flag 为“0”(no_cropping_rectangle_visible)；WDS#0，具有在图形平面中(100,100)处宽度 700×高度 500 的窗口的声明；PDS#0；ODS#0，指示字幕“Credits:Company”；和 END(结束)。

点 t32 处的 DS32 包括 PCS#1，其 composition_state 是“正常情况”，并指示在对象缓冲器中对于(0,0)点处 600×400 大小的图形对象的剪切操作(cropping_rectangle#0 (0,0,600,400))，且将该剪切图形对象

定位在图形平面中坐标(0,0)处(on Window#0(0, 0))。在对象缓冲器中点(0,0)处 600×400 大小的区域包含两行显示的字幕“Credits: Company”中的“Credits:”部分，从而在图形平面上出现“Credits:”部分。

点 t33 处的 DS33 包括 PCS#2，其 composition_state 是“正常情况”，并指示在对象缓冲器中对于(0,100)点处 600×400 大小的图形对象的剪切操作(cropping_rectangle#0 (0,100,600,400))，且将该剪切图形对象定位在图形平面中坐标(0,0)处(on Window#0(0, 0))。在对象缓冲器中(0,100)点处 600×400 大小的区域包含两行显示的字幕“Credits: Company”中的“Credits:”部分，从而在图形平面上以两行出现“Credits:”部分和“Company”部分。

点 t34 处的 DS34 包括 PCS#3，其 composition_state 是“正常情况”，并指示在对象缓冲器中对于 (0,200)点处 600×400 大小的图形对象的剪切操作(cropping_rectangle#0 (0,200,600,400))，且将该剪切图形对象定位在图形平面中坐标(0,0)处(on Window#0(0, 0))。在对象缓冲器中(0,200)点处 600×400 大小的区域包含两行显示的字幕“Credits: Company”中的“Company”部分，从而在图形平面上显示出“Company”部分。通过以上 PCS 描述，可以以两行滚动显示字幕。

最后，说明用于划入/划出(Wipe-In/Out)效果的描述协议。图 16 沿时间线示出了当执行划入/划出时 DS 的描述的例子。该图中，显示集合 DS21、DS22、DS23 和 DS24 位于再现时间线上点 t51、t52、t53 和 t54 处。

点 t51 处的 DS51 包括：PCS#0，其 composition_state 被设置为“时元开始”，且 object_cropped_flag 为“0”(no_cropping_rectangle_visible)；WDS#0，具有在图形平面中(100,100)处宽度 700×高度 500 的窗口的声明；PDS#0；ODS#0，指示字幕“Fin”；和 END(结束)。

点 t52 处的 DS52 包括 PCS#1，其 composition_state 是“正常情况”，并指示在对象缓冲器中对于(0,0)点处 600×400 大小的图形对象的剪切操作(cropping_rectangle#0 (0,0,600,400))，且将该剪切图形对象定位在图形平面中坐标(0,0)处(on Window#0(0, 0))。在对象缓冲器中

(0,0)点处 600×400 大小的区域包含字幕“Fin”，从而在图形平面上出现字幕“Fin”。

点 t53 处的 DS53 包括 PCS#2，其 composition_state 是“正常情况”，并表示在对象缓冲器中对于(200,0)点处 400×400 大小的图形对象的剪切操作(cropping_rectangle#0 (200,0,400,400))，且将该剪切图形对象定位在图形平面中坐标(200,0)处(on Window#0(200, 0))。这样，窗口中由坐标(200,0)和(400,400)指示的区域成为显示区域，且由坐标(0,0)和(199,400)指示的区域成为非显示区域。

点 t54 处的 DS54 包括 PCS#3，其 composition_state 是“正常情况”，并指示在对象缓冲器中对于(400,0)点处 200×400 大小的图形对象的剪切操作(cropping_rectangle#0(400,0,200,400))，且将该剪切图形对象定位在图形平面中坐标(400,0)处(on Window#0(400, 0))。这样，由坐标(0,0)和(399,400)指示的区域成为非显示区域。

这样，随着非显示区域变大，显示区域变小，实现划入/划出效果。

如上所述，利用相应的脚本可以实现诸如切入/切出、淡入/淡出、划入/划出和滚动等各种效果，从而可以以各种方式呈现字幕。

对于实现上述效果具有以下限制。为了实现滚动效果，必须进行清除和重画窗口的操作。对于图 15 的例子，在 t32 和 t33 间隔期间，必须执行“窗口清除”，以从图形平面中擦除 t32 处的图形对象“Credits:”，然后执行“窗口重画”，以将“Credits:”的下半部分和“Company”的上半部分写在图形平面上。假定该间隔与视频帧的间隔相同，则用于滚动效果的对象缓冲器与图形表面之间的传输速率成为重要因素。

这里，考察关于窗口大小的限制。Rc 是对象缓冲器与图形平面之间的传输速率。这里最坏情况是以传输速率 Rc 同时执行窗口清除和窗口重画。这种情况下，需要以 Rc 的一半速率(Rc/2)来执行窗口清除和窗口重画中的每一个。

为了使窗口清除和窗口重画与视频帧同步，需要满足下面的方程式：

窗口大小×帧速率=Rc/2

如果帧速率为 29.97，则 Rc 由下式表示：

Rc =窗口大小×2×29. 97

呈现字幕时，窗口大小至少占图形平面的 25%至 33%。图形平面中的像素总数为 1920×1080。由于每个像素的给定比特长度为 8 比特，所以图形平面的总容量是 2M 字节(=1920×l080×8 比特)。

使窗口大小为图形平面总容量的 1/4，则窗口大小为 500k 字节(=2 M 字节/4)。将该值代入上面方程式，计算出 Rc 为 256Mbps (=500K 字节×2×29.97)。如果窗口清除和窗口重画的速率可以是帧速率的一半或四分之一，则即使 Rc 相同也可以使窗口大小放大两倍或四倍。

通过将窗口大小保持在图形平面的 25%至 33%并以 256 Mbps 传输速率显示字幕，则不管要实现什么类型的显示效果，都可以保持图形和电影图片之间的同步显示。

接下来说明窗口的位置、大小和区域。如上所述，一个时元中窗口的位置和区域不改变。窗口的位置和大小被设置成在一个时元期间相同，因为如果改变位置和大小就必须改变图形平面的目标写地址，且改变该地址导致系统开销，进而降低了从对象缓冲器到图形平面的传输速率。

每个窗口的图形对象数目具有限制。提供该数目限制是为了降低传输解码后图形对象时的系统开销。当设置图形对象的边缘地址时产生该系统开销，且边缘数目越多，产生的系统开销越大。

图 17 示出了相对比的两个例子，一个例子中一个窗口具有四个图形对象，另一个例子中一个窗口具有两个图形对象。具有四个图形对象的例子的边缘数目是具有两个图形对象的例子的边缘数目的两倍。

如果没有图形对象数目的限制，就变得不知道传输图形时会产生多少系统开销，从而传输负荷会剧烈的增加或减少。另一方面，当窗口中图形对象的最大数目为 2 时，则可以考虑总共 4 个系统开销来设置传输速率。从而，较容易设置最小传输速率。

接下来说明具有 PCS 和 ODS 的 DS 怎样被分配到 AV 剪辑的时

间线上。时元是再现时间线上存储器管理连续的一个时间周期。因为时元由多于一个 DS 组成，所以如何将 DS 分配到 AV 剪辑的再现时间线上是很重要的。AV 剪辑的再现时间线是这样一条时间线，用于指定对构成多路复用到AV剪辑的视频流的每一块图片数据进行解码和再现的定时。以 90KHz 精度表示再现时间线上的解码和再现定时。DS 中 PCS 和 ODS 上附加的 DTS 和 PTS 指示用于再现时间线上同步控制的定时。显示集合在再现时间线上的分配指利用 PCS 和 ODS 上附加的 DTS 和 PTS 执行同步控制。

首先，以下说明如何使用 ODS 上附加的 DTS 和 PTS 执行同步控制。

DTS 以 90 KHz 精度指示开始解码 ODS 的时间，PTS 指示解码结束的时间。

ODS 解码不会立刻完成，其具有一定时间长度。由于要求清楚地指示解码过程的开始点和结束点，ODS 的 DTS 和 PTS 分别指示解码开始和结束的时间。

PTS 值指示最后时间，从而需要在 PTS 指示的时间之前必须完成 ODS 的解码，且将解压缩图形对象写入再现装置的对象缓冲器。

由 DTS (DSn [ODS]) 以 90 KHz 精度指示 DSn 中任何 ODSj 的解码开始时间。DTS (DSn [ODS])加上最长解码持续时间就是 ODSj 解码结束的时间。

当 ODSj 的大小是“SIZE (DSn[ODSj])”且该 ODS 的解码速率是“Rd”时，以秒表示的解码所需最大时间表示为“SIZE(DSn[ODSj])//Rd”。该符号“//”表示在小数点后上舍入的除法操作符。

通过将最大时间周期变换为以 90 KHz 精度表示的数字并加上 ODSj 的 DTS，计算出由 PTS 指示的解码结束(90 KHz)的时间。

DSn 中 ODSj 的 PTS 由下面方程式表示：

$$PTS(DSn[ODSj])=DTS(DSn[ODSj])+90{,}000\times(SIZE(DSn[ODSj])//Rd)$$

另外，两个连续 ODS：ODSj 和 ODSj+1 之间的关系必须满足下面方程式：

PTS (DSn[ODSj]) ≤DTS(DSn[ODSj+l])

接下来说明 PCS 的 DTS 和 PTS 的设置。

在 DSn 中第一个 ODS(ODS1)的解码开始时间(DTS (DSn[ODSl]))之前，且在 DSn 中第一个 PDS(PDS1)变有效的时间(PTS (DSn[PDSl]))之前，PCS 必须被加载到再现装置的对象缓冲器中。从而，DTS 必须被设置成满足下面方程式：

DTS(DSn[PCS])≤DTS(DSn[ODSl])

DTS(DSn[PCS])≤PTS(DSn[PDSl])

另外，DSn 中的 PCS 的 PTS 由以下方程式表示：

PTS (DSn [PCS])≥DTS (DSn [PCS]) +decode_duration(DSn)

“decode_duration(DSn)”表示对用于更新 PCS 的所有图形对象进行解码的持续时间。该解码持续时间不是固定值，但不随着再现装置和设备或安装在再现装置上的软件的状态改变。当用于组成 DSn. PCSn 的屏幕的对象是 DSn. PCSn. OBJ [j]时，decode_duration(DSn)受以下因素影响：(i) 用于清除窗口需要的时间；(ii) 用于解码 DSn. PCSn.OBJ 的解码持续时间；(iii)写 DSn.PCSn.OBJ 需要的时间。当设置了 Rd 和 Rc 时，decode_duration(DSn)总相同。从而，创作时通过计算这些持续时间长度，计算出 PTS。

基于图 18 所示的程序来执行 decode_duration 的计算。图 19、20A 和 20B 是示意性示出该程序算法的流程图。下面参考这些图说明 decode_duration 的计算。图 19 所示的流程图中，首先调用 PLANEINITIALZE 函数(图 19 中步骤 S1)。使用 PLANEINITIALZE 函数调用一个函数，用于计算对呈现 DS 的图形平面进行初始化所必需的时间周期。图 19 的步骤 S1 中，以参数 DSn、DSn.PCS.OBJ[0]和 decode_duration 调用该函数。

以下参考图 20A 说明 PLANEINITIALZE 函数。该图中，initialize_duration 是表示 PLANEINITIALZE 函数返回值的变量。

图 20A 中的步骤 S2 是一个 IF 语句(判断语句)，用于根据 DSn 中 PCS 的 page_state 是否指示“时元开始”来切换操作。如果 page_state 指示“时元开始”(图 18 中 DSn.PCS.page_state==epoch_start, 步骤

S2=Yes)，则清除图形平面必需的时间周期被设置为 initialize_duration (步骤 S3)。

如上所述当对象缓冲器与图形平面之间传输速率为 256,000,000，且图形平面的总大小被设置为 video_width×video_height 时，清除所需的时间周期为“video_width×video_height//256,000,000”。当乘以 90,000Hz 从而以 PTS 的时间精度表达时，清除图形平面所需的时间周期为“90,000×video_width×video_height //256,000,000”。该时间周期与 initialize_duration 相加。

如果 page_state 不指示“时元开始”(步骤 S2=否)，则将所有窗口的用于清除由 WDS 定义的 Window[i]所需的时间周期加到 initialize_duration 上 (步骤 S4)。当如上所述对象缓冲器与图形平面之间的传输速率 Rc 为 256,000,000 且属于 WDS 的 Winodow [i]的总大小为 ΣSIZE (WDS.WIN [i]) 时，清除所需的时间周期为“ΣSIZE(WDS.WIN [i])//256,000,000”。当乘以 90,000Hz 从而以 PTS 的时间精度表示时，清除属于该 WDS 的这些窗口所需的时间周期为“90,000×ΣSIZE(WDS.WIN[i])//256,000,000”。该时间周期被加到 initialize_duration 上，并返回作为结果的 initialize_duration。以上是 PLANEINITIALZE 函数。

图 19 中的步骤 S5 用于根据 DSn 中图形对象数目为 2 或 1 来切换操作(图 18 中，if(DSn.PCS.num_of_object==2)，if(DSn.PCS.num_of_object==1))，如果该数目为 1(步骤 S5)，用于解码该图形对象的等待时间被加到 decode_duration 上(步骤 S6)。通过调用 WAIT 函数(图 18 中，decode_duration+=WAIT(DSn,DS.PCS.OBJ[0], decode_duration))执行该等待时间的计算。使用参数 DSn、DSn.PCS.OBJ[0]、decode_duration 调用该函数，返回值为 wait_duration。

图 20B 是表示该 WAIT 函数操作的流程图。

该流程图中，调用者的 decode_duration 被设置为 current_duration。Object_definition_ready_time 是设置到 DS 中图形对象的 PTS 的变量。

current_time 是设置为 current_duration 和 DSn 中 PCS 的 DTS 的总和的变量。当 Object_definition_ready_time 大于 current_time 时(如果(current_time < Object_definition_ready_time)，步骤 S7 中为是)，作为返回值的 wait_duration 被设置为 Object_definition_ready_time 与 current_time 之间的差(步骤 S8, wait_duration += object_definition_ready_time-current_time)。decode_duration 被设置为 WAIT 函数的返回值加上重画窗口所需时间周期(90,000×(SIZE(DSn.WDS.WIN[0]))//256,000,000)得到的时间周期。

上述说明是图形对象数目为 1 的情况。在图 19 的步骤 S5 中，判断图形对象数目是否为 2。如果 DSn 中图形对象数目等于 2 (图 18 中，if (DSn.PCS.num_of_object==2))，使用 PCS 中的 OBJ[0]作为参数调用 WAIT 函数，并将返回值加到 decode_duration 上(步骤 S10)

在接下来的步骤 S11 中，判断 DSn 的 OBJ[0]所属于的窗口是否与图形对象[1]所属于的窗口相同(if(DSn.OBJ[0].window_id==DSn.PCS.OBJ[l].window_id)。如果窗口相同，则使用 OBJ[1] 作为参数调用 WAIT 函数，并将返回值 wait_duration 加到 decode_duration 上(步骤 S12)，并将重画 OBJ[0]所属于的窗口需要的时间(90,000×(SIZE(DSn.WDS.OBJ[0].window_id))//256,000,000)加到 decode_duration 上(步骤 S13)。

如果判定窗口不同(步骤 Sll，“不同”)，则重画 OBJ[0]所属于的窗口需要的时间(90,000×(SIZE(DSn.WDS.OBJ[0].window_id))//256,000,000)被加到 decode_duration 上(步骤 S15)，使用 OBJ[1]作为参数调用 WAIT 函数，并将返回值 wait_duration 加到 decode_duration 上(步骤 S16)，且将重画 OBJ[1]所属于的窗口需要的时间(90,000×(SIZE(DSn.WDS.OBJ[0].window_id))//256,000,000) 加到 decode_duration 上 (步骤 S17)。

由上述算法计算出 decode_duration。以下说明设置 PCS 的 PTS 的具体方式。

图 21A 示出了一个窗口中包含一个 ODS 的情况。图 21B 和 21C 是以时间顺序表示图 18 中涉及的值的时序图。每个图中末行“ODS

解码”和中间行“图形平面访问”表示再现时同时执行的两个操作。上述算法的描述假定这两个操作并行执行。

“图像平面访问”包括清除周期(1)和写周期(3)。清除周期(1)或指示清除整个图形平面所需的时间周期(90,000×(图形平面大小//256,000,000))，或指示清除图形平面上所有窗口所需的时间周期(Σ(90,000× (窗口[i] 的大小//256,000,000))。

写周期(3)指示呈现整个窗口所需的时间周期(90,000×(窗口[i]的大小 //256,000,000))。

另外，解码周期(2)指示该 ODS 的 DTS 与 PTS 之间的时间周期。

根据要清除的范围、要解码的 ODS 的大小和要写到图形平面上的图形对象的大小，清除周期(1)、解码周期(2)和写周期(3)可以改变。为了方便，该图中解码周期(2)的开始点与清除周期(1)的开始点相同。

图 21B 示出了解码周期(2)较长，且 decode_duration 等于解码周期(2)和写周期(3)的总和的情况。

图 21C 示出了清除周期(1)较长，且 decode_duration 等于清除周期(1)和写周期(3)的总和的情况。

图 22A 至 22C 示出了一个窗口中包含两个 ODS 的情况。图 22B 和 22C 中的解码周期(2)指示解码两个图形所需的总时间周期。同样，写周期(3)指示向图形平面写两个图形所需的总时间周期。

即使 ODS 数目为 2，也可以以与图 21 的情况中相同方式计算 decode_duration。当解码两个 ODS 的解码周期(2)较长时，如图 22B 所示 decode_duration 等于解码周期(2)和写周期(3)的总和。

当清除周期(1)较长时，decode_duration 等于清除周期(1)和写周期(3)的总和。

图 23A 描述了两个窗口中的每一个窗口包含一个 ODS 的情况。如同前面的情况，当清除周期(1)比用于解码两个 ODS 的解码周期(2)长时，decode_duration 等于清除周期(1)和写周期(3)的总和。然而，当清除周期(1)比解码周期(2)短时，可以在解码周期(2)结束前向第一个窗口写。从而，decode_duration 既不等于清除周期(1)和写周期(3)的总和，也不等于解码周期(2)和写周期(3)的总和。

当解码第一个ODS所需的时间周期是写周期(31)且解码第二个ODS所需的时间周期是写周期(32)时，图23B示出了解码周期(2)比清除周期(1)和写周期(31)的总和长的情况。这种情况下，decode_duration等于解码周期(2)和写周期(32)的总和。

图23C示出了清除周期(1)和写周期(31)的总和比解码周期(2)长的情况。这种情况下，decode_duration等于清除周期(1)、写周期(31)和写周期(32)的总和。

从再现装置的模型可以预先知道图形平面的大小。而且，在创作时也可以知道窗口大小和ODS的大小和数目。从而，可以确定decode_duration等于哪种时间周期的组合：清除周期(1)和写周期(3)，解码周期(2)和写周期(3)，解码周期(2)和写周期(32)，或清除周期(1)、写周期(31)和写周期(32)。

通过基于上述decode_duration的计算设置ODS的PTS，可以以高精度同步显示图形和图片数据。通过定义窗口和限制重画窗口的区域有可能以高精度实现这种同步显示。从而，将窗口概念引入创作环境中具有很大意义。

以下说明DSn中WDS的DTS和PTS的设置。WDS的DTS可以被设置成满足以下方程式：

DTS (DSn [WDS]) ≥ DTS (DSn [PCS])

另一方面，DSn中WDS的PTS指示向图形平面开始写的最后时间。因为向图形平面上的窗口写的时间充分，所以通过从由PCS的PTS指示的时间减去写WDS所需的时间周期来确定向图形平面开始写的时间。当WDS的总大小为ΣSIZE(WDS.WIN[i])时，清除和重画所需的时间为“ΣSIZE(WDS.WIN[i])//256,000,000”。当以90.000 KHz时间精度表示时，该时间为“90,000×ΣSIZE(WDS.WIN[i])//256,000,000”。

从而，可以由以下方程式计算出WDS的PTS：

PTS (DSn [WDS]) =

PTS (DSn[PCS])-90000×ΣSIZE (WDS.WIN [i])//256,000,000

WDS中指示的PTS是最后时间，在PTS之前可以向图形平面

开始写。即，如图 23 所示，一旦对这些窗口之一中要呈现的 ODS 进行解码，可以在该点开始写由解码获得的图形对象。

如上所述，使用 WDS 上附加的 DTS 和 PTS，可以将窗口分配到 AV 剪辑的再现时间线上任何时间点处。

参考图 24-25 中所示的具体例子，以下说明在显示集合中 DTS 和 PTS 的设置。该例关于以下情况，其中通过向图形平面写四次来显示字幕，并且为两个字幕“什么是 blu-ray”和 blu-ray 无处不在”中每一个字幕的显示执行更新。图 24 示出了该例中按时间顺序的更新转变。到 t1 点显示“什么”，t1 之后直到 t2 显示“什么是”，然后在 t3 显示“什么是 blu-ray.”。第一个字幕的整个句子出现之后，在 t4 显示第二个字幕“blu-ray 无处不在。”。

图 25A 示出了被描述用于执行上述更新的四个显示集合。DS1 包含用于在 t1 处控制更新的 PCS1.1、用于着色的 PDS1、与字幕“什么是 blu-ray”对应的 ODS1，和作为 DS1 结束码的 END。

DS2 包含用于在 t2 处控制更新的 PCS1.2 和 END。DS3 包含用于在 t3 处控制更新的 PCS1.3 和 END。DS4 包含用于在 t4 处控制更新的 PCS2、用于色彩变换的 PDS2、与字幕“blu-ray 无处不在。”对应的 ODS2、和 END。

参考图 25B 的时序图，说明四个显示集合中每个功能段的 DTS 和 PTS 的设置。

该时序图中的再现时间线与图 24 中的时间线相同。在图 25A 的时序图中，分别在显示“什么”的显示点 t1、显示“什么是”的显示点 t2、显示“什么是 blu-ray.”的显示点 t3 和显示“blue-ray 无处不在。”的显示点 t4 设置 PTS(PCS1.1)、PTS (PCS1.2)、PTS (PCS1.3) 和 PTS (PCS2)。如上设置每个 PTS，因为必须在每个字幕的显示点执行诸如每个 PCS 中描述的剪切这样的控制。

PTS(ODS1)和 PTS(ODS2)被设置成通过分别从由 PTS(PCS1.3) 和 PTS(PCS2)指示的点减去 decode_duration 计算出的点，因为需要将 PTS(PCS)设置成满足以下公式：

PTS(DSn[PCS]) >DTS(DSn [PCS]) +decodeduration(DSn)

图 25B 中，PTS(ODS2)被设置成指示点 t4 之前的点 t5，PTS (ODS1)被设置成指示点 t1 之前的点 t0。

DTS(ODS1)和 DTS(ODS2)被设置成指示通过分别从由 PTS (ODS1)和 PTS(ODS2)指示的点减去 decode_duration 所计算出的点，因为需要将 DTS(ODS)设置成满足以下方程式：

PTS(DS[ODSj])=DTS(DSn[ODSj])+90,000×(SIZE(DSn[ODSj])//Rd)

在图 25B 中，PTS(ODS2)被设置成指示点 t4 之前的点 t5，PTS (ODS1) 被设置成指示点 t0 之前的一个点。这里满足由 DTS(ODS2) =PTS(ODS1)指示的关系。

通过在较早显示的前一个 ODS 的 PTS 之后立刻设置一个 ODS 的 PTS，再现装置将该后一个 ODS 读出至存储器以覆盖该前一个 ODS，从而可以通过最小的存储器大小来执行再现处理。通过实现这样一个再现过程，对于再现装置的存储器大小的选择变宽。

PCS1.1 的 DTS 被设置成 DTS (PCS1.1) =DTS(ODS1)，因为 PCS1.1 的 DTS 值可以是由 DTS(ODS1)指示的点之前的任意点。

ODS1 的 PTS，ODS2 的 DTS，和 PCS1.2、PCS1. 3、PCS2 的 PTS 被设置在点 t0，以满足以下方程式表示的关系：

PTS(ODS1)=DTS(ODS2)=DTS(PCS1. 2) =DTS(PCS1. 3) =DTS (PCS2)

这是因为 PCS1.2 和 PCS1.3 的 DTS 值可以是由 PTS(PCS1.3)指示的点之前的任意点，PCS2 的 DTS 可以是由 DTS(PCS2)指示的点之前的任意点。

如上所述，通过在同一时间读出多于一个 PCS，可以在完成前一个 PCS 更新后尽可能快地执行后一个 PCS 的更新。

PCS 的 DTS 和 PTS 以及 ODS 的 DTS 和 PTS 满足由上面公式表示的关系是充分的。从而，可以将这些值设置成 DTS(ODS2) =PTS(ODS1)或 PTS(ODS1)=DTS(ODS2)= PTS(PCS1.2) =PTS(PCS1.3) =DTS (PCS2)。通过这样设置的时间标记，可以调节解码负担增加或需要更多缓冲器的周期的时间长度。这样的调节扩大了再现期间控制的可能性，并更有利于执行创作或制造再现装置的人们。

上述显示集合(PCS, WDS, PDS, ODS)的数据结构是编程语言中

描述的类结构的一个例子。执行创作的制造者通过根据蓝光光盘预录格式(Blu-ray Disc Prerecording Format)中提供的语法描述类结构，可以在 BD-ROM 上获得该数据结构。

接下来，以下说明根据本发明的再现装置的实际例子。图 26 示出了根据本发明的再现装置的内部结构。根据本发明的再现装置是基于该图中所示结构进行工业生产的。根据本发明的再现装置主要由三部分构成：系统 LSI、驱动设备和微计算机系统，通过将这三部分安装在该装置的壳体和基底上可以对该再现设备进行工业生产。该系统 LSI 是一个集成电路，其中集成了执行该再现装置功能的各种处理单元。以上述方式生产的该再现装置包括 BD 驱动器 1、读缓冲器 2、PID 过滤器 3、传输缓冲器 4a-4c、外围电路 4d、视频解码器 5、视频平面 6、音频解码器 7、图形平面 8、CLUT 单元 9、加法器 10、图形解码器 12、编码数据缓冲器 13、外围电路 13a、流图形处理器 14、对象缓冲器 15、合成缓冲器 16 和图形控制器 17。

BD 驱动器 1 执行 BD-ROM 的装入/读取/弹出，以及对 BD-ROM 的访问。

读缓冲器 2 是 FIFO 存储器，用于以先进先出顺序存储从 BD-ROM 读出的 TS 包。

PID 过滤器 3 对从读缓冲器 2 输出的多于一个 TS 包进行过滤。PID 过滤器 3 的该过滤仅将具有期望 PID 的 TS 包写入传输缓冲器 4a-4c。PID 过滤器 3 的过滤不必须进行缓冲，从而输入 PID 过滤器 3 中的 TS 包被没有延迟地写入传输缓冲器 4a-4c。

传输缓冲器 4a-4c 以先入先出顺序存储从 PID 过滤器 3 输出的 TS 包。从传输缓冲器 4a-4c 输出 TS 包的速率是速率 Rx。

外围电路 4d 是布线逻辑，用于将从传输缓冲器 4a-4c 读出的 TS 包变换成功能段。由该变换获得的功能段被存储在编码数据缓冲器 13 中。

视频解码器 5 将从 PID 过滤器 3 输出的多于一个 TS 包解码成解压缩图片并写入视频平面 6。

视频平面 6 是用于运动图片的平面存储器。

视频解码器 7 对从 PID 过滤器 3 输出的 TS 包进行解码并输出解压缩音频数据。

图形平面 8 是具有用于一个屏幕的区域的平面存储器，能够存储用于一个屏幕的解压缩图形。

CLUT 单元 9 基于由 PDS 指示的 Y、Cr 和 Cb 值来变换图形平面 8 中存储的解压缩图形的索引颜色。

加法器 10 用由 PDS 指示的 T 值(透明度)乘以已经由 CLUT 单元 9 执行颜色变换的解压缩图形，并对于每个像素加上存储在视频平面中的该解压缩图片数据，然后获得并输出合成图像。

图形解码器 12 解码图形流以获得分解图形，并将该分解图形作为图形对象写入图形平面 8。通过解码图形流，字幕和菜单出现在屏幕上。该图形解码器 12 包括编码数据缓冲器 13、外围电路 13a、流图形处理器 14、对象缓冲器 15、合成缓冲器 16 和图形控制器 17。

编码数据缓冲器 13 是将功能段连同 DTS 和 PTS 一起存储的缓冲器。通过从存储在传输缓冲器 4a-4c 中的传输流的每个 TS 包去除 TS 包头和 PES 包头并顺序排列净荷得到该功能段。来自被去除的 TS 包头和 PES 包头的 PTS 和 DTS 在 PES 包之间进行对应之后被存储。

外围电路 13a 是布线逻辑，其实现编码数据缓冲器 13 与流图形处理器 14 之间的传输和编码数据缓冲器 13 与合成缓冲器 16 之间的传输。在传输操作中，如果当前时间是由 ODS 的 DTS 指示的时间，则 ODS 被从编码数据缓冲器 13 传输到流图形处理器 14。如果当前时间是由 PCS 和 PDS 的 DTS 指示的时间，则 PCS 和 PDS 被传输到合成缓冲器 16。

流图形处理器 14 解码 ODS，并将由解码获得的索引颜色的解压缩图形作为图形对象写入对象缓冲器 15。流图形处理器 14 执行的解码在与 ODS 对应的 DTS 时间开始，并在与 ODS 对应的 PTS 指示的解码结束时间之前结束。图形对象的解码速率 Rd 是流图形处理器 14 的输出速率。

对象缓冲器 15 是与 ETSI EN 300 743 标准中的像素缓冲器对应的缓冲器，对由流图形处理器 14 执行解码获得的图形对象进行布置。

对象缓冲器 15 需要被设置成图形平面 8 的两倍或 4 倍大，因为在执行滚动效果的情况下，对象缓冲器 5 需要存储的图形对象是该图形平面的两倍或四倍大。

合成缓冲器 16 是其中布置 PCS 和 PDS 的缓冲器。

图形控制器 17 对布置在合成缓冲器 16 中的 PCS 进行解码，并基于该 PCS 执行控制。执行该控制的定时基于该 PCS 上附加的 PTS。

接下来说明对于构造 PID 过滤器 3、传输缓冲器 4a-4c、图形平面 8、CLUT 单元 9、编码数据缓冲器 13 和图形控制器 17 所需的传输速率和缓冲器大小的建议值。图 27 示出了写速率 Rx、Rc 和 Rd、图形平面 8、编码数据缓冲器 13、对象缓冲器 15 和合成缓冲器 16 的大小。

对象缓冲器 15 与图形平面 8 之间的传输速率 Rc 是本实施例的再现装置中的最高传输速率，由窗口大小和帧速率计算为 256Mbps(=500 K 字节× 29.97 ×2)。

与 Rc 不同，流图形处理器 14 与对象缓冲器 15 之间的传输速率 Rd(像素解码速率)不需要每个视频帧周期都更新，Rc 的 1/2 或 1/4 对于 Rd 已经足够。从而，Rd 为 128 Mbps 或 64 Mbps。

传输缓冲器 4a-4c 与编码数据缓冲器 13 之间的传输缓冲器漏率 Rx 是 ODS 在压缩状态下的传输速率。从而，用压缩律乘以传输速率 Rd 对于传输缓冲器漏率 Rx 来说是足够的。假定 ODS 的压缩率是 25%，则 16 Mbps (=64 Mbps×25%)已经足够。

该图中所示的传输速率和缓冲器的大小是最小标准，也可以将其设置为较高速率和较大大小。

上述结构的再现装置中，每个部件在流水线结构中执行解码操作。

图 28 是示出了该再现装置进行流水线处理的时序图。图中的第 5 行是 BD-ROM 中的显示集合，第 4 行表示从 PCS、WDS、PDS 和 ODS 到编码数据缓冲器 13 的读周期。第 3 行表示流图形处理器 14 执行的每个 ODS 的解码周期。第 1 行表示图形控制器 17 执行的操作。

ODS1 和 ODS2 上附加的 DTS(解码开始时间)分别指示图中的 t31

和 t32。因为解码开始时间由 DTS 设置，所以需要将每个 ODS 读出到编码数据缓冲器 13。从而，在用于解码 ODS1 的解码周期 dp1 之前完成读取 ODS1 至编码数据缓冲器 13。并且，在用于解码 ODS2 的解码周期 dp2 之前完成读取 ODS2 至编码数据缓冲器 13。

另一方面，ODS1 和 ODS2 上附加的 PTS(解码结束时间)分别指示图中的 t32 和 t33。流图形处理器 14 对 ODS1 的解码在 t32 前完成，对 ODS2 的解码在 t33 指示的时间前完成。如上所述，流图形处理器 14 在 ODS 的 DTS 指示的时间之前将 ODS 读到编码数据缓冲器 13 中，并在 ODS 的 PTS 指示的时间之前对读到编码数据缓冲器 13 中的 ODS 进行解码，并将解码后的 ODS 写入对象缓冲器 15。

该图中第 1 行的周期 cd1 表示图形控制器 17 清除图形平面所需的周期。而且，周期 td1 表示将对象缓冲器中获得的图形对象写入图形平面 8 的周期。WDS 的 PTS 指示开始写的最后时间，PCS 的 PTS 指示写结束和显示的定时。在 PCS 的 PTS 指示的时间处，在图形平面 8 上获得组成一个交互屏幕的解压缩图形。

CLUT 单元 9 执行解压缩图形的颜色变换以及加法器 10 执行该分解图形和存储在视频平面 6 中的分解图片的合成之后，获得合成图像。

在图形解码器 12 中，在图形控制器 17 执行图形平面 8 的清除的同时，流图形处理器 14 连续地执行解码。通过上述流水线处理，可以执行图形的即时显示。

图 28 中，说明了在完成 ODS 解码之前结束图形平面清除的情况。图 29 示出了在完成图形平面清除之前结束 ODS 解码的情况下流水线处理的时序图。这种情况下，不可能在完成 ODS 解码时向图形平面写。当图形平面的清除完成时，就可以向图形平面写由解码获得的图形。

接下来说明缓冲器占用随时间的变化。图 30 是时序图，示出了图 26 中所示的以下部件随时间的变化：合成缓冲器 16、对象缓冲器 15、编码数据缓冲器 13 和图形平面 8。第一至第四行分别示出了图形平面 8、对象缓冲器 15、编码数据缓冲器 13 和合成缓冲器 16 的占

用随时间的变化。这里，使用线图描述随时间的变化，其中横轴表示时间线，纵轴表示占用。

图30的第四行表示合成缓冲器16的占用随时间的变化。如第四行所示，合成缓冲器16的随时间变化包含“vf0”部分，其表示由于存储从编码数据缓冲器13输出的PCS引起的单调上升。

第三行表示编码数据缓冲器13的占用随时间的变化。如第三行所示，编码数据缓冲器13的随时间变化包含两个单调上升部分vfl和vf2，以及两个单调下降部分vgl和vg2。单调上升部分vfl和vf2的梯度取决于传输缓冲器4a、b、c至编码数据缓冲器13的输出速率Rx，单调下降部分vgl和vg2的梯度代表由流图形处理器14立刻执行的解码。即，立即执行对ODS的解码，且流图形处理器14保留通过解码得到的该解压缩图形。从流图形处理器14至对象缓冲器15的传输路径的写速率是128 Mbps。从而对象缓冲器15的占用随该写速率升高。

第二行表示对象缓冲器15的占用随时间的变化。如第二行所示，对象缓冲器15的随时间变化包含由于存储从流图形处理器14输出的ODS引起的单调上升部分vhl和vh2。单调上升部分vhl和vh2的梯度取决于从流图形处理器14至对象缓冲器15的传输速率Rc。第三行的单调下降部分和第二行的单调上升部分出现的周期对应于“解码周期。该解码周期的开始由ODS的DTS指示，该解码周期的结束由ODS的PTS指示。如果解压缩图形存储在对象缓冲器15中直到由ODS的DTS表示的时间，这意味着对ODS的解码完成。只要解压缩图形存储在对象缓冲器15中直到由ODS的PTS表示的时间，则该解码周期中的单调上升部分和单调下降部分可以是任何形式。

第一行代表图形平面8的占用随时间的变化。如第一行所示，图形平面8的随时间变化包含由于存储从对象缓冲器15输出的已解码ODS引起的单调上升部分vf3。单调上升部分vf3的梯度取决于从对象缓冲器15至图形平面8的传输速率Rd。该单调上升部分的结束由ODS的PTS表示。

使用分配到ODS的DTS和PTS、分配到ICS的DTS和PTS、

图 27 中所示的每个缓冲器的大小和传输速率，来说明诸如图 27 中的图。另外，通过创建诸如该图中的示意图，用于在创作阶段就能够知道每个缓冲器的状态如何改变。

由于通过更新 DTS 和 PTS 可以调节每个缓冲器的状态变化，所以就有可能避免向再现装置施加超过解码器规格的解码负荷，并可以避免再现过程中易于发生的缓冲器溢出。由此，在再现装置的开发阶段，硬件/软件实现变得容易。

接下来，以下说明如何实现控制单元 20 和图形解码器 12。通过写用于执行图 30 所示操作的程序并由通用 CPU 来执行该程序，实现控制单元 20。参考图 30 说明由控制单元 20 执行的操作。

图 31 是表示功能段的装载操作处理的流程图。该流程图中，段 K 是表示再现 AV 剪辑时读出的每个段(PCS, WDS, PDS 和 ODS)的变量。忽略标识是用于确定段 K 被忽略或装载的标识。该流程图具有循环结构，其中，首先将忽略标识初始化为 0，然后对于每个段 K 重复步骤 S21-S24 和 S27-S31(步骤 S25 和步骤 S26)。

步骤 S21 判断段 K 是否为 PCS，如果段 K 是 PCS，则执行步骤 S27 和步骤 S28 的判断。

步骤 S22 判断忽略标识是否为 0。如果忽略标识为 0，则操作进行到步骤 S23，如果忽略标识是 1，则操作进行到步骤 S24。如果忽略表示为 0(步骤 S22：是)，则在步骤 S23 将段 K 载入编码数据缓冲器 13。

如果忽略表示为 1(步骤 S22：否)，则在步骤 S24 忽略段 K。这样，因为步骤 S22 为否，所以属于该 DS 的所有功能段的剩余功能段被忽略(步骤 S24)。

如上所述，由忽略标识决定段 K 被忽略还是被装载。步骤 S27-S31、S34 和 S35 是用于设置忽略标识的步骤。

步骤 S27 中，判断段 K 的 segmet_type 是否是“获取点”。如果段 K 是获取点，则操作进行到步骤 S28，如果段 K 是“时元开始”或“正常情况”，则操作进行到步骤 S31。

步骤 28 中，判断是否有先前的 DS 存在于图形解码器 12 的任何

缓冲器中(编码数据缓冲器 13、流图形处理器 14、对象缓冲器 15 和合成缓冲器 16)。当步骤 S27 中的判断为是时，进行步骤 S28 中的判断。先前 DS 不存在于图形解码器 12 中的情况表示执行跳转操作的情况。这种情况下，显示从作为“获取点”的该 DS 开始，从而操作进行到步骤 S30(步骤 S28：否)。步骤 S30 中，将忽略标识设置为 0，操作进行到步骤 S22。

先前 DS 存在于图形解码器 12 中的情况表示执行正常再现的情况。这种情况下，操作进行到步骤 S29(步骤 S28：是)。步骤 S29 中，将忽略标识设置为 1，操作进行到步骤 S22。

步骤 S31 中，判断 PCS 的 composition_state 是否为“正常情况”。如果 PCS 是“正常情况”，则操作进行到步骤 S34，如果 PCS 是“时元开始”，则在步骤 S30 中将忽略标识设置为 0。

步骤 S34 中，如同步骤 S28 中，判断先前 DS 是否存在于图形解码器 12 的任何缓冲器中。如果存在先前 DS，则将忽略标识设置为 0(步骤 S30)。如果不存在先前 DS ，则不可能获得足够的功能段来组成交互屏幕，且忽略标识被设置为 1(步骤 S35)。

通过以上述方式设置忽略标识，当图形解码器 12 中不存在先前 DS 时，构成“正常情况”的功能段被忽略。

参考如图 31 所示对 DS 进行多路复用情况下的一个例子，说明对 DS 进行读取的方式。在图 31 的例子中，三个 DS 与运动图片多路复用。DS1 的 composition_state 是“时元开始”，DS10 的 composition_state 是“获取点”，且 DS20 的 composition_state 是“正常情况”。

假定，在多路复用了三个 DS 和运动图片的 AV 剪辑中，执行向由箭头 am1 表示的图片数据 pt10 的跳转操作，DS10 与跳转目标最接近，从而 DS10 是图 31 的流程图中表示的 DS。虽然在步骤 S27 中 composition_state 被判断为获取点，但是由于编码数据缓冲器 13 中不存在先前的 DS 所以忽略标识被设置为 0，且如图 32 中箭头 mdl 所示 DS10 被载入再现装置的编码数据缓冲器 13。另一方面，在跳转目标在 DS10 之后的情况中(图 32 中的箭头 am2)，DS20 被忽略，因为 DS20

是“正常情况”显示集合，且在编码数据缓冲器 13 中不存在先前 DS(图 33 中的箭头 md2)。

图 34 示出了在正常再现中 DS1、DS10 和 DS20 的装载。DS1 的 PCS 的 composition_state 为“时元开始”，DS1 被装载编码数据缓冲器 13 (步骤 S23)。然而，DS10 的 PCS 的 composition_state 为“获取点”，因为 DS10 的忽略标识被设置为 1(步骤 S29)，所以构成 DS10 的功能段被忽略，并不载入编码数据缓冲器 13(图 35 中的箭头 rd2 和步骤 S24)。另外，DS20 被载入编码数据缓冲器 13，因为 DS20 的 PCS 的 composition_ state 是“正常情况”(图 35 中 rd3)。

接下来说明图形控制器 17 的操作。图 36-38 示出了由图形控制器 17 执行的操作的流程图。

步骤 S41-S44 是该流程图的主程序的步骤，等待步骤 S41-S44 中指定的任何事件的发生。

步骤 S41 判断当前再现时间是否为 PCS 的 DTS 指示的时间，如果判断为是，则执行步骤 S45-S53 中的操作。

步骤 S45 判断 PCS 的 composition_state 是否为“时元开始”，如果判断为“时元开始”，则在步骤 S46 清除所有图形平面 8。如果判断为“时元开始”之外的状态，则清除由 WDS 的 window_horizontal_position 、window_vertical_position、 window_width 和 window_height 指示的窗口。

在步骤 S46 和步骤 S47 中执行清除后执行步骤 S48，然后判断由任意 ODSx 的 PTS 指示的时间是否已经经过。因为整个图形平面 8 的清除占用时间，所以在清除结束的时间之前应该已经完成任何 ODSx 的解码。

从而，在步骤 S48 中，判断在清除结束时是否已经完成任何 ODSx 的解码。如果判断为否，则操作返回主程序。如果由任意 ODSx 的 PTS 指示的时间已经经过，则执行步骤 S49-S51 的操作。在步骤 S49，判断 object_crop_flag 是否为 0，如果该标识为 0，则图形对象被设置为“不显示”(步骤 S50)。

如果步骤 S49 中该标识不为 0，则基于 object_cropping_horizontal_

position 、 object_cropping_vertical_position 、 cropping_width 和 cropping_height 剪切的对象被写入图形平面 8 的窗口中由 object_cropping_horizontal_position 和 object_cropping_vertical_position 指示的位置(步骤 S51)。通过以上操作，一个或多个图形对象呈现在窗口中。

步骤 52 中，判断与另一个 ODSy 的 PTS 对应的时间是否已经经过。当向图形平面 8 写 ODSx 时，如果已经完成 ODSy 的解码，则 ODSy 变成 ODSx (步骤 S53)，操作进行到步骤 S49。这样，对另一个 ODS 执行步骤 S49-S51 的操作。

接下来参考图 37，以下说明步骤 S42 和步骤 S54-S59。

步骤 S42 中，判断当前再现点是否在 WDS 的 PTS 处。如果判断为当前再现点在 WDS 的 PTS 处，则在步骤 S54 判断窗口数目是否为一个。如果判断为两个，则操作返回到主程序。如果判断为一个，则执行步骤 S55-S59 的循环处理。在该循环处理中，对于窗口中显示的两个图形对象中的每一个图形对象执行步骤 S55-S59 的操作。在步骤 S57 中，判断 object_crop_flag 是否指示 0。如果指示 0，则不显示图形(步骤 S58)。

如果不指示 0，则基于 object_cropping_horizontal_position、object_cropping_vertical_position、cropping_width 和 cropping_height 的剪切对象被写入图形平面 8 的窗口中由 object_cropping_horizontal_position 和 object_cropping_vertical_ position 指示的位置(步骤 S59)。通过重复以上操作，一个以上图形对象呈现在窗口中。

在步骤 S44 中，判断当前再现点是否在 PDS 的 PTS 处。如果判断为当前再现点在 PDS 的 PTS 处，则在步骤 S60 判断 pallet_update_flag 是否为 1。如果判断为 1，则在 CLUT 单元中设置由 pallet_id 指示的 PDS(步骤 S61)。如果判断为 0，则跳过步骤 S61。

之后，CLUT 单元对要与运动图片组合的图形平面 8 上的图形对象执行颜色变换(步骤 S62)。

接下来，参考图 38，以下说明步骤 S43 和步骤 S64-S66。

在步骤 43 中，判断当前再现点是否在 ODS 的 PTS 处。如果判断

为当前再现点在 ODS 的 PTS 处，则在步骤 S63 判断窗口数目是否为 2。如果判断为 1，则操作返回主程序。如果判断为 2，则执行步骤 S64-S66 的操作。在步骤 S64 中，判断 object_crop_flag 是否指示 0。如果指示 0，则不显示图形(步骤 S65)。

如果不指示 0，则基于 object_cropping_horizontal_position、object_cropping_vertical_ position、cropping_width 和 cropping_height 的剪切对象被写入图形平面 8 的窗口中由 object_cropping_horizontal_position 和 object_cropping_vertical_position 指示的位置(步骤 S66)。通过重复以上操作，图形对象呈现在每个窗口中。

上述说明关于属于 DSn 的 PSC 的 DTS 和 PTS 以及 ODS 的 DTS 和 PTS。没有说明 PDS 的 DTS 和 PTS 以及 END 的 DTS 和 PTS。首先，说明属于 DSn 的 PDS 的 DTS 和 PTS。

对于属于 DSn 的 PDS，在第一个 ODS 的解码开始点(DTS (DSn[ODSl]))之后，在 PCS 被载入合成缓冲器 16 (DTS(DSn[PCS])) 时在 CLUT 单元 9 中可以使用 PDS 就足够了。从而，DSn 中每个 PDS(PDSl-PDSlast)的 PTS 值需要被设置成满足下面关系：

DTS (DSn [PCS]) ≤ PTS (DSn[PDSl])

PTS (DSn[PDSj])≤ PTS (DSn[PDSj+l]) ≤PTS (DSn [PDSlast])

PTS (DSn [PDSlast]) ≤ DTS (DSn[ODSl])

注意，再现期间不参考 PDS 的 DTS，ODS 的 DTS 被设置成与 PDS 的 PTS 具有相同值，以满足 MPEG2 标准。

以下说明当 DTS 和 PDS 被设置成满足上面关系时，DTS 和 PTS 在再现装置的流水线处理中的作用。图 39 示出了基于 PDS 的 PTS 的再现装置的流水线。图 39 基于图 26。图 39 中第一行表示在 CLUT 单元 9 中设置 ODS。第一行以下与图 26 的第一至第五行相同。在 CLUT 单元 9 中对 PDSl-last 的设置(箭头 up2、up3)与 ODS1 解码的开始同时执行。(在传输 PCS 和 WDS 后且在 ODS1 解码前执行对 CLUT 单元 9 的 PDSl-PDSlast 设置，从而在由如箭头 up2 和 up3 所示的 ODS1 的 DTS 指示的点之前执行对 CLUT 单元 9 的 PDSl-PDSlast 设置)。

如上所述，在 ODS 解码之前执行 PDS 的设置。

接下来说明 DSn 中显示集合段的 END 的 PTS 的设置。属于 DSn 的 END 指示 DSn 的结束，从而需要 END 的 PTS 指示 ODS2 的解码结束时间。该解码结束时间由 ODS2(ODSlast)的 PTS(PTS(DSn [ODSlast]))指示，从而需要 END 的 PTS 被设置成满足以下方程式的值：

PTS (DSn [END]) =PTS (DSn [ODSlast])

按照 DSn 与属于 DSn+1 的 PCS 之间的关系，DSn 的 PCS 在第一个 ODS(ODS1)的装载时间之前被载入合成缓冲器 16，从而 END 的 PTS 应该在 DSn 的 PCS 的装载时间之后且在属于 DSn+1 的 PCS 的装载时间之前。从而，END 的 PTS 需要满足下面关系：

DTS (DSn [PCS])≤ PTS (DSn [END]) ≤ DTS (DSn+l [PCS])

另一方面，第一个 ODS(ODS1)的装载时间在最后一个 PDS (PDSlast)的装载时间之前，从而 END 的 PTS (PTS (DSn [END]))应该在属于 DSn 的 PDS 的装载时间 (PTS (DSn [PDSlast]))之后。从而，END 的 PTS 需要满足下面关系：

PTS (DSn [PDSlast]) < PTS (DSn [END])

下面说明该再现装置的流水线处理中 END 的 PTS 的重要性。图 40 是描述 END 在再现装置的流水线处理中的重要性的图。图 40 基于图 28，除了图 39 的第一行表示合成缓冲器 16 的内容之外，图 40 的每一行都基本上与图 28 相同。另外，图 40 中示出了两个显示集合 DSn 和 DSn+1。DSn 中的 ODSlast 是 A-ODSs 中的最后一个 ODS，从而由 END 的 PTS 指示的点在 DSn+1 中 PCS 的 DTS 之前。

由 END 的 PTS，可以确定再现期间 DSn 中 ODS 的装载在什么时间完成。

注意，虽然再现期间没有参考 END 的 DTS，但是 END 的 DTS 被设置成与 END 的 PTS 具有相同值，从而满足 MPEG2 标准。

如上所述，根据本实施例，图形平面的一部分被指定为显示图形的窗口，从而再现装置不必须为整个平面呈现图形。再现装置可以仅为预定大小的窗口呈现图形，例如图形平面的 25%至 33%的窗口。

因为不需要呈现窗口中图形之外的图形，所以再现装置中软件的工作量降低。

即使在最坏情况下，其中在例如图形平面的 1/4 上执行图形的更新，通过再现装置以诸如 256Mbps 的预定传输速率向图形平面写，并通过设置窗口大小以确保与图片的同步显示，也可以与图片同步地显示图形。

从而，因为容易保证同步显示，所以可以为各种再现装置实现高分辨率的字幕显示。

(第二实施例)

以上说明的第一实施例是关于专用于字幕显示的图形。相反，第二实施例针对用于交互显示的图形。

在根据本发明的记录介质的实施例中，以下说明使用记录介质的一个例子。如同第一实施例中，也可以通过改进 BD-ROM 的应用层来制造第二实施例的记录介质。图 41 是示意性示出第二实施例的 AV 剪辑的结构的示意图。

按照下面方式构成 AV 剪辑(图中间行所示)。由多个视频帧(图片 pj1, pj2 和 pj3)组成的视频流和由多个音频帧(图中顶行)组成的音频流，分别被变换成一行 PES 包(图中第二行)，然后被变换成一行 TS 包(图中第三行)。交互图形流(图中下数第一行)被变换成一行 PES 包(图中下数第二行)，然后被变换成一行 TS 包(图中下数第三行)。将这三行 TS 包多路复用，从而构成 AV 剪辑。

接下来说明交互图形流。交互图形流具有交互合成段(ICS)代替 PCS，且没有 WDS。交互图形流与表示图形流类似，其也具有称为调色板定义段(PDS)、对象定义段(ODS)和显示集合结束段(END)的功能段。

屏幕上布置的 GUI 部分生成了由这些功能段定义的交互屏幕。图 42A 示出了由交互图形流实现的这种交互屏幕。该交互屏幕包括四个 GUI 部分，称为按钮 A-按钮 D。依靠交互图形流的交互表示根据用户操作改变这些 GUI 部分(即按钮)的状态。如图 42A 所示，这

些 GUI 部分(按钮)的状态包括“正常状态 bt1”、“选中状态 bt2”和“激活状态 bt3”。正常状态是仅提供显示的状态。与之相反，选中状态是按照用户操作做出选择但还没收到确认的状态。激活状态是接收到确认的状态。通过按下第一实施例中所示的遥控器 400 的键可以改变按钮状态。

图 42B 示出了遥控器 400 的键，通过这些键可以接收到用户对交互屏幕的操作。如该图所示，该遥控器 400 提供了上移键、下移键、右移键和左移键。

当交互屏幕中一个按钮处于选中状态时，上移键用于将该选中按钮上边的按钮设置为选中状态。下移键用于将该选中按钮下边的按钮设置为选中状态。右移键用于将该选中按钮右边的按钮设置为选中状态。左移键用于将该选中按钮左边的按钮设置为选中状态。

激活键用于将选中按钮设置为活动状态(即，激活)。数字键“0”-“9”用于将被分配了对应数字的按钮设置为选中状态。“+10”键用于接收将已输入的数值加 10 的操作。这里应该注意，“0”键和“+10”键都用于接收不小于 10 的数字的输入。从而它们中任何一个对于遥控器 400 都是足够的。

每种状态(即，正常状态、选中状态和激活状态)由多个解压缩状态下的图形组成。代表每个按钮状态的每个解压缩图形被称为“图形对象”。由多个解压缩图形代表一个按钮状态的原因是考虑到执行每个按钮的每个状态的动画显示。

接下来说明本实施例中对于定义段(ODS、PDS)的改进。ODS 和 PDS 具有与第一实施例中相同的数据结构。唯一不同是关于 ODS 的“object_ID”。第二实施例中的 ODS 利用由多个 ODS 定义的多个图形对象构造动画。在构造动画过程中，在一序列 ODS 上添加作为序号的 object_ID。

接下来说明 ICS。交互合成段是构成交互屏幕的功能段。交互合成段具有图 43 中所示的数据结构。如该图所示，由 segment_type、segment_length、composition_number、 composition_state、command_update_flag 、 composition_time_out_pts 、 selection_time_out_pts 、

UO_mask_table、animation_frame_rate_code、default_selected_button_number、default_activated_button_number 和按钮信息集合(button_info (1) (2) (3)...)组成 ICS。

“composition_number”代表 0 至 15 的数值，其指示执行更新。

“composition_state”代表与当前 ICS 一同开始的 DS 为正常状态、获取点或时元开始。

“command_update_flag”表示当前 ICS 中的按钮命令与先前 ICS 相比是否已经改变。例如，如果某一 ICS 所属于的 DS 是“获取点”，则原理上该 ICS 与它之前的 ICS 具有相同内容。然而，如果“command_update_flag”被设置为开，则与该先前 ICS 不同的按钮命令可以被设置为 ICS。当应用图形对象的同时期望改变命令时，可以将该命令更新标识设置为有效。

“composition_time_out_pts”描述交互屏幕的结束时间。在该结束时间，交互屏幕显示不再有效，所以不再执行显示。期望以运动图片数据的再现时间线的时间精度描述 composition_time_out_pts。

“selection_time_out_pts”描述有效按钮选择周期的结束时间。在 selection_time_out_pts 期间，由 default_activated_button_number 指定的按钮被激活。该 selection_time_out_pts 的周期等于或短于 composition_time_out_pts 的周期。以视频帧的时间精度描述 selection_time_out_pts。

“UO_ mask_table”表示对于与 ICS 对应的显示集合的用户操作的允许/禁止。如果该屏蔽字段被设置为禁止，则对于再现装置的使用操作将无效。

“animation_frame_rate_code”描述要应用到动画类型按钮上的帧速率。通过由视频帧速率除以该字段的值获得动画帧速率。如果该字段的值为“00”，则仅显示由 start_object_id_xxx 指定的 ODS，而不显示动画，该 ODS 在定义用于按钮的这些图形对象的 ODS 中。

“default_selected_button_number”表示当交互屏幕显示已经开始时，应该被设置为默认选中状态的按钮编号。如果该字段为“0”，具有存储在再现装置的寄存器中的按钮编号的按钮将被自动设置为激

活状态。如果该字段不是“0”，说明该字段指示按钮的有效值。

“default_activated_button_number”代表在由 selection_time_out_pts 定义的时间之前用户没有将任何按钮设置为激活状态时，被自动设置为激活状态的按钮。如果 default_activated_button_number 是“FF”，则在由 selection_time_out_pts 定义的时间将自动选择当前处于选中状态的按钮。如果该 default_activated_button_number 是“00”，则不执行自动选择。如果不是“00”和“FF”，则该字段被解释为表示有效按钮编号。

按钮信息(button_info)定义交互屏幕上组成的每个按钮。该图中的导出线描述按钮信息 i 的内部结构，按钮信息 i 是 ICS 控制的第 i 个按钮。下面说明构成按钮信息 i 的信息元素。

“button_number”是 ICS 中唯一识别按钮 i 的值。

“numerically_selectable_flag”指示是否允许对于按钮 i 的数值选择。

“auto_action_flag”指示是否自动设置按钮 i。如果 auto_action_flag 被设置为开(即，比特值为 1)，则按钮 i 被设置为激活状态，而非选中状态。相反，如果 auto_action_flag 被设置为关(即，比特值为 0)，则按钮 i 被设置为只选中状态，即使该按钮已经被选择。

“object_horizontal_position”和“object_vertical_position”分别表示交互屏幕中按钮 i 的左上角像素的水平位置和垂直位置。

“upper_button_number”指示当按钮 i 处于选中状态时按下上移键后，代替按钮 i 被设置为选中状态的按钮的编号。如果对应于按钮 i 的编号已经被设置在该字段中，则对该上移键的按下操作将被忽略。

“lower_button_number”、“left_button_number”和“right_button_number”指示当按钮 i 处于选中状态期间分别按下下移键、左移键和右移键时，代替按钮 i 处于选中状态的按钮的编号。如果对应于按钮 i 的编号已经被设置在该字段中，则对这些键的按下操作将被忽略。

“start_object_id_normal”是这样一个字段，当由动画呈现正常状态的按钮 i 时，在该“start_object_id_normal”字段中描述分配给构成动画的多个 ODS 的一系列编号中的第一个编号。

“end_object_id_normal”是这样一个字段，当由动画呈现正常状态的按钮 i 时，在该 end_object_id_normal 字段中描述分配给构成动画的多个 ODS 的一系列编号(即 object_id)中的最后一个编号。如果 end_object_id_normal 中指示的 ID 与 start_object_id_normal 中指示的 ID 相同，则图形对象中与该 ID 对应的静止图像就是按钮 i 的图像。

“repeated_normal_flag”指示是否重复继续正常状态下的按钮 i 的动画显示。

“star_object_id_selected” 是这样一个字段，当由动画呈现选中状态的按钮 i 时，在该“star_object_id_selected”字段中描述分配给构成动画的多个 ODS 的一系列编号中的第一个编号。

“end_object_id_selected”是这样一个字段，当由动画呈现选中状态的按钮 i 时，在该 end_object_id_selected 字段中描述分配给构成动画的多个 ODS 的一系列编号中的最后一个编号。如果 end_object_id_selected 中指示的 ID 与 start_object_id_selected 中指示的 ID 相同，则图形对象中与该 ID 对应的静止图像就是按钮 i 的图像。

“repeated_ selected _flag”指示是否重复继续选中状态下的按钮 i 的动画显示。如果 start_object_id_selected 与 end_object_id_selected 具有相同值，则该字段被设置为 00。

“star_object_id_activated”是这样一个字段，当由动画呈现激活状态的按钮 i 时，在该 star_object_id_activated 字段中描述分配给构成动画的多个 ODS 的一系列编号中的第一个编号。

“end_object_id_ activated”是这样一个字段，当由动画呈现激活状态的按钮 i 时，在该 end_object_id_activated 字段中描述分配给构成动画的多个 ODS 的一系列编号(即 object_id)中的最后一个编号。

接下来说明按钮命令。

一个按钮命令(button_command)指当按钮 i 被设置为激活状态时执行的命令。

以下说明通过 ICS 进行交互控制的例子。该例子假定 ODS 和 ICS 如图 44 所示。图 44 表示包含在 DSn 中的 ODS 与 ICS 之间的关系。假定该 DSn 包含 ODS11-19、21-29、31-39 和 41-49。这些 ODS 中，

分别由 ODS11-19 表示按钮 A 的状态，ODS21-29 表示按钮 B 的状态，ODS31-39 表示按钮 C 的状态，ODS41-49 表示按钮 D 的状态。(参考大括号“}”)。然后假定分别在 button_infos (1), (2), (3), (4)中描述这些按钮 A-D 的状态控制。

通过 ICS 的控制的执行时序与图 45 的运动图片中任意图片数据 pt1 的显示时序一致，这说明通过将由按钮 A-D 组成的交互屏幕 tm1 合成(gsl)到图像数据 pt1 中(gs2)，来显示该交互屏幕 tm1。

由于由多个按钮组成的交互屏幕与运动图片的内容一致，所以利用按钮和通过 ICS 可以呈现十分真实的图像。

图 46 示出了按钮 A-D 的状态如图 47 所示改变的情况下 ICS 的描述例子。图 47 中的箭头 hh1 和 hh2 用代表由 button_info (l)的 neighbor_info ()产生的状态改变。该 button_info (l)的 neighbor_info () 的 lower_button_number 被设置为按钮 C。从而，如果按钮 A 处于选中状态时产生下移键被按下的 UO(用户操作)(图 47,upl)，则按钮 C 将处于选中状态(图 47,sjl)。由于 button_info(l)的 neighbor_info ()中的 right_button_number 被设置为按钮 B，所以如果按钮 A 处于选中状态时产生右移键被按下的 UO(图 47,up2)，则按钮 B 将处于选中状态(图 47,sj2)。

图 47 中的箭头 hh3 表示对由于 neighbor_info ()引起的 button_info (3)的状态改变的控制。由于 button_info(3)的 neighbor_info()中的 upper_button_number 被设置为按钮 A，所以如果按钮 C 处于选中状态时产生上移键被按下的 UO，则按钮 A 将返回选中状态。

接下来说明按钮 A-D 的图像。这里假定 ODS11、21、31 和 41 具有如图 49 所示的图像等。由于 ICS 中，button_info(l)的 normal_state _info()具有指示 ODS11-13 的 start_object_id_normal、end_object_id_ normal，所以按钮 A 的正常状态由 ODS11-13 的动画表示。另外，由于 button_info (l)的 selected_state_info()具有指示 ODS14-16 的 start_ovject_id_selected 和 end_object_id_selected，所以按钮 A 的选中状态由 ODS14-16 表示。按钮 A 被用户设置为选中状态的结果是，作为按钮 A 图像的图将从 ODS11-13 的图改变为 ODS14-16 的图。这里，

如果分别在 normal_state_info() 和 selected_state_info() 中将 repeat_normal_flag 和 repeat_select_flag 设置为 1，则 ODS11-13 的动画和 ODS14-16 的动画将被重复连续地显示，如图中“→(A)”、“(A)→”、“→(B)”和“(B)→”所示。

如果能够呈现动画的多个 ODS 被分配到按钮 A-D，且在 ICS 中描述对应的控制，则可以更精细和快速地实现按钮状态控制(例如，随着用户操作的改变来改变图像的特征表情)。

接下来说明显示集合中 ODS 的顺序。如上所述，由 ICS 指示属于显示集合的 ODS，以代表按钮的一种状态。根据应该代表按钮的哪种状态的指示，来确定显示集合中 ODS 的次序。

更具体地，显示集合中的 ODS 根据它们代表的状态进行分组：(1)代表正常状态，(2)代表选定状态，(3)代表激活状态，等等。这些组中的每一个组代表一种按钮状态，被称为“按钮状态组”。然后以诸如“正常状态-选中状态-激活状态”这样的次序排列这些按钮状态组。定义 ODS 的次序将相应地定义显示集合中 ODS 的次序。

图 50 示出了属于显示集合的 ODS 的顺序。在该图的第二行中，示出了显示集合的三个按钮状态组。该图示出了三组 ODS 设置：呈现正常状态的 ODS 组(用于“正常状态”的 ODS)；呈现选中状态的 ODS 组(用于“选中状态”的 ODS)；和呈现激活状态的 ODS 组(用于“激活状态”的 ODS)。这些按钮状态组以“正常状态→选中状态→激活状态”这样的次序排列。该次序的目的是，在读取构成更新的屏幕显示的其它组成部分之前，读取构成交互屏幕的初始显示的组成部分。

图 50 的第一行表示图形对象“An, Bn, Cn, Dn, As, Bs, Cs, Ds, Aa, Ba, Ca, Da”。An, Bn, Cn, Dn 的下表“n”代表对应按钮的正常状态。同样，As, Bs, Cs, Ds 的下表“s”代表对应按钮的选中状态，以及下表“a”代表对应按钮的激活状态。图 50 的第二行表示第一行的图形对象所属于的按钮状态组。应该注意，该图中，每一组 ODS1-0SDn 被分配相同编号，如 1 和 n。然而，这些组彼此不同，分别属于 N-ODSs、S-ODSs 和 A-ODSs。这也应用于以下每个相似的图中。

图 51 示出了布置了图 50 的按钮状态组的交互屏幕的状态转移。

该图的交互屏幕具有多个状态，它们是“初始显示”、“根据第一个用户动作的更新显示”和“根据第二个用户动作的更新显示”。该图中的箭头代表触发相应状态改变的用户动作。根据该图，四个按钮 A、B、C 和 D 分别具有“正常状态”、“选中状态”、“激活状态”。可以理解，为了执行初始显示，需要呈现三个正常状态的图形对象和呈现一个选中状态的图形对象。

即使在没有定义默认选中按钮且哪个按钮要被设置为选中状态将动态改变的情况下，一旦对于每个按钮代表正常状态和选中状态的图形对象的解码完成，就将实现初始显示。考虑到上述情况，本实施例将按钮状态组布置成每个按钮状态组对应于一个不同状态，这些状态以“正常状态-选中状态-激活状态”次序排列，如图 50 的第二行所示。即使当没完成读取和解码构成激活状态的 ODS 时，这种布置也可以实现初始显示，并有助于缩短从开始读显示集合到完成初始显示为止的周期。

接下来说明图 48 和 49 中所示的 ODS 以何种次序布置。图 52 示出了显示集合中 ODS 的顺序。该图中，用于正常状态的 ODS 由 ODS11-13、ODS21-23、ODS31-33 和 ODS41-43 构成。用于选中状态的 ODS 由 ODS14-16、ODS24-26、ODS34-36 和 ODS44-46 构成。用于激活状态的 ODS 由 ODS17-19、ODS27-29、ODS37-39 和 ODS47-49 构成。ODS11-13 用于呈现图 49 中所示的特征表情的改变。ODS21-23、ODS31-33 和 ODS41-43 的作用相同。从而通过将这些 ODS 布置在上面按钮状态组中，甚至在读取显示集合的过程中就可以准备初始显示。由此，动画形式的交互屏幕可以被没有延迟地执行。

接下来说明由多个按钮多重参考的 ODS 的次序。这里，多重参考指由 normal_state_info、selected_state_info 和 activated_state_info 中的两个或更多个指示一个 ODS 的 object_id。通过采用这种多重参考方法，利用用于呈现一个按钮正常状态的图形对象可以呈现另一个按钮的选中状态。这允许共享图形对象的图像。这种共享有助于减少 ODS 的数目。这种情况下，关于多重参考中使用的 ODS 的一个问题

是，该 ODS 属于哪个按钮状态组。

更具体地，当由一个 ODS 呈现一个按钮的正常状态和另一个按钮的选中状态时，要考虑的事情是该 ODS 属于与正常状态对应的按钮状态组还是属于与选中状态对应的按钮状态组。

这种情况下，仅在与最早出现的状态对应的按钮状态组中布置一次该 ODS。

如果在正常状态和选中状态的多重参考中使用某一 ODS，则该 ODS 将被布置在与正常状态(N-ODSs)对应的按钮状态组中，而不被布置在与选中状态(S-ODSs)对应的按钮状态组中。另外，如果在选中状态和激活状态的多重参考中使用另一 ODS，则该 ODS 将被布置在与选中状态(S-ODSs)对应的按钮状态组中，而不被布置在与激活状态(A-ODSs)对应的按钮状态组中。总之，这种多重参考方法中的 ODS 将仅在与最早出现的状态对应的按钮状态组中被设置一次。

接下来说明 S-ODSs 中 ODS 的次序。S-ODSs 中，哪个 ODS 最早出现取决于默认选中按钮是被静态地确定还是被动态地确定。被静态确定的该默认选中按钮具有设置在 ICS 的 default_selected_button_number 中的有效值(00 除外)，且该值指定该按钮。当 default_selected_button_number 指示有效值且 N-ODSs 中没有代表默认选中按钮的 ODS 时，将首先布置代表默认选中按钮的 ODS。

如果 default_selected_button_ number 指示值 00，则被设置为默认选中状态的按钮将根据再现装置侧的状态动态地改变。

default_selected_button_ number应该被设置为指示值0的情况为，例如，将显示集合多路复用得到的 AV 剪辑作为多个再现路径的结合点。例如，如果先前多个播放路径分别是第一、第二和第三章，且作为结合点的显示集合用于显示与该第一、第二和第三章对应的按钮，在default_selected_button_ number中决定将处于默认选中状态的按钮是不充分的。

这种情况下，理想的是根据到达该显示集合为止先前多个再现路径中的哪个路径是刚刚经历的路径，来改变将处于选中状态的按钮，(例如，当从第一章到达时选中第二章的按钮，当从第二章到达时选

中第三章的按钮，当从第三章到达时选中第四章的按钮)。要改变将处于选中状态的按钮的情况下，default_selected_button_ number 将被设计成无效，即将其设为值 0。因为要改变将处于选中状态的按钮，所以不需要在按钮状态组的开始处布置某一 ODS。

图 53 表示 default_selected_button_ number 为“=0”的情况与“=按钮 B”的情况之间 S-ODSs 中 ODS 排列的不同。该图中，虚线 ssl 表示 default_selected_button_number 指示按钮 B 的情况下 S-ODSs 中 ODS 的布置；虚线 ss2 表示 default_selected_button_number 指示值 0 的情况下 S-ODSs 中 ODS 的布置。如图中注释所示，当 default_selected_button_ number 指示按钮 B 时，指示按钮 B 选中状态的 ODSB 被布置在 S-ODSs 的第一位，其它按钮的 ODS 布置在其后。相反，当 default_selected_button_number 指示值 0 时，指示按钮 A 选中状态的 ODSA 被布置在第一位。这样，default_selected_button_number 是否有效导致 S-ODSs 的次序显著变化。

接下来说明在 AV 剪辑的再现时间线上如何布置具有这些 ICS 和 ODS 的显示集合。可以基于第一实施例中所示的表达式设置 ODS 的 DTS 和 PTS。相反，ICS 的 DTS 和 PTS 将与第一实施例中所示的不同。以下说明 ICS 的 DTS 和 PTS 值。

“时元开始”之后，ICS 中的 PTS 被设置为等于或大于对以下(1)、(2)、(3)求和得到的值：(1)在构成 DSn 的初始显示的 ODS 中解码时间最晚的 ODS 的 PTS 值，(2)清除图形平面需要的时间，和(3)向图形平面写由 ODS 解码获得的图形对象的写时间。另一方面，当处于“获取点”时，该 ICS 中的 PTS 将被设置为等于或大于通过将(3)平面写时间周期和(1)ODS 的 PTS 值相加得到的值。

当 ICS 中指示 default_selected_button_ number 时，只要 i) ODS 解码用于呈现所有按钮的正常状态和 ii) ODS 解码用于呈现默认按钮的选中状态，就可以执行初始显示。在初始显示时用于呈现多个按钮的选中状态的 ODS 被称为 S-ODSs，且这些 ODS 中解码时间出现最早的 ODS(这种情况下，该 ODS 用于呈现默认按钮)被称为 S-ODSsfirst。该 S-ODSsfirst 的 PTS 值被设置为其解码时间出现最晚

的 ODS 的 PTS 值，且被用作 ICS 中 PTS 的参考值。

当 ICS 中没指示 default_selected_button_number 时，任意按钮都可以处于选中状态。所以直到准备好呈现所有按钮的正常状态和选中状态后，才能完成初始显示的准备。在初始显示时用于呈现多个按钮的选中状态的 S-ODSs 中，其解码时间出现最晚的一个被称为 S-ODSslast。该 S-ODSslast 的 PTS 值被设置为其解码时间出现最晚的 ODS 的 PTS 值，且被用作 ICS 中 PTS 的参考值。

如果用于解码 S-ODSsfirst 的结束时间被假定为 PTS(DSn[S-ODSsfirst])，则 PTS(DSn[ICS])是通过将以下(1)、(2)和(3)相加得到的值：(1)PTS(DSn[S-ODSsfirst])，(2)清除图形平面需要的时间和(3)向图形平面写由 ODS 解码获得的图形对象的写时间。

这里假定在图形平面中，用于呈现图片的矩形区域的宽度和高度分别被定义为“video_width”和“video_height”，且向图形平面写的速率为 128Mbps。则清除图形平面需要的时间表示为“8×video_width×video_height//128,000,000”。当用 90KHz 时间精度表示时，图形平面的清除时间(2)将为 90,000×(8×video_width×video_height//128,000,000)。

另外，这里假定由包含在 ICS 中的所有按钮信息指定的图形对象的总大小为 ΣSIZE(DSn[ICS.BUTTON [i]])，且向图形平面写的速率为 128Mbps，则向图形平面写所需要的时间表示为 ΣSIZE(DSn[ICS.BUTTON[i]])//128,000,000。如果用 90KHz 时间精度表示时，该图形平面的清除时间(2)为 90,000×(SIZE(DSn[ICS.BUTTON[i]]))//128,000,000)。

这里，ΣSIZE(DSn[ICS.BUTTON[i]])为代表所有按钮的图形对象中第一个显示的图形对象的总大小。默认选中按钮已经确定的情况下，该 ΣSIZE(DSn[ICS.BUTTON[i]]将产生与默认选中按钮动态改变情况下不同的值。当默认选中按钮已经被静态确定时，ΣSIZE(DSn[ICS.BUTTON[i]])将是以下(1)和(2)的总和：(1)用于代表默认选中按钮的选中状态的多个 ODS 中第一显示的 ODS 和(2)用于代表除默认选中按钮之外按钮的正常状态的多个 ODS 中第一显示的 ODS。

相反，当默认选中按钮动态改变时，因为很难知道哪个按钮将是默认选中按钮，所以应该假定写时间最长的情况。这种情况下，认为第一个显示的图形对象是在以下(1)和(2)中具有最大大小(Max(ODSnl, ODSsl))的图形对象：(1)用于代表任意按钮 x 的正常状态下的第一页的图形对象(ODSnl)和(2)用于代表任意按钮 x 的选中状态下的第一页的图形对象(ODSsl)。

将每个按钮的该 Max(ODSn1,ODSsl)相加的结果将为 ΣSIZE(DSn[ICS.BUTTON[i]])。

图 54A、54B 示出了在 N-ODSs 包含构成按钮 A-D 的多个 ODS 且 S-ODSs 包含构成按钮 A-D 的多个 ODS 的情况下，ΣSIZE(DSn[ICS.BUTTON[i]])采用哪些值。这里，当 default_selected_button_number 指示有效值时，ΣSIZE(DSn[ICS.BUTTON[i]])为如粗线框所示的四个 ODS 的总大小。“Asl”是代表按钮 A 选中状态的多个 ODS 中第一个显示的 ODS。“Bnl”、“Cnl”和“Dnl”为代表按钮 B-D 正常状态的多个 ODS 中第一个显示的相应 ODS。当这些大小由 size()表示时，ΣSIZE(DSn[ICS.BUTTON[i]])将是：

size(Asl) +size(Bnl) +size(Cnl) +size(Dnl)。

另一方面，当 default_selected_button_number 为“=0”时，SIZE (DSn[ICS.BUTTON[i]])将是：

Anl, Asl 中较大的 ODS + Bnl, Bsl 中较大的 ODS + Cnl, Csl 中较大的 ODS + Dnl, Dsl 中较大的 ODS。

从而 SIZE(DSn[ICS.BUTTON[i]])表示为以下方程式：

SIZE(DSn[ICS.BUTTON[i]])=

max(size(Cnl),size(Csl))+max(size(Dnl),size(Dsl))

使用上式，“时元开始”刚刚开始后的 PTS(DSn[ICS])表示为以下方程式：

PTS(DSn[ICS]) ≥ PTS(DSn[S-ODSsfirst])

+90,000×(8×video_width×video_heght//128,000,000)

+90,000×(ΣSIZE(DSn[ICS.BUTTON[i]])//128,000,000)

图 55 示出了通过如上设置 PTS 和 DTS 实现同步显示的例子。该

图假定在运动图片中任意图片数据 pyl 的显示定时处显示按钮的情况。这种情况下，ICS 中的 PTS 值应该被设置为与相应图片数据的显示时间点一致。

另外，ODS 中的 PTS 值应该被设置在该图的时间点(1)处，因为，在通过从 ICS 中的 PTS 减去屏幕清除周期“cd1”和图形对象传输周期“td1”获得的时间之前，构成 DSn 初始显示的 ODS 中解码时间最晚的 ODS 的解码应该完成。另外，因为 ODS 的解码需要 dd1 周期，所以 ODS 的 DTS 值应该被设置为在该 PTS 之前 dd1 周期的时间。

图 55 只有一个与运动图片组合的 ODS，这是一个简化的例子。为了在多个 ODS 当中与运动图片组合地实现交互屏幕的初始显示，应该如图 56 所示设置 ICS 中的 PTS 和 DTS，ODS 中的 DTS。

图 56 示出了交互屏幕的初始显示由多个 ODS 构成且静态确定默认选定按钮的情况下如何设置 DTS 和 PTS。如果用于实现初始显示的 ODS 中最后执行解码的 S-ODSsfirst 的解码将在该图的周期 dd1 期间结束，则该 S-ODSsfirst 的 PTS(DSn[S-ODSsfirst])应该被设置为指示周期 dd1 的时间。

另外，初始显示之前，应该执行已解码图形对象的屏幕清除和传输。从而 ICS 的 PTS(DSn[ICS])应该被设置为在将该 PTS(DSn[S-ODSsfirst]) 的值、屏幕清除需要的周期(90,000×(8×video_width×video_height//128,000,000)) 和解码图形对象的传输周期(90,000×(ΣSIZE(DSn[ICS.BUTTON [i]])//128,000,000))相加后得到的时间之后。

图 57 表示交互屏幕的初始显示由多个 ODS 构成且不确定默认选中按钮的情况下如何设置 DTS 和 PTS。如果用于实现初始显示的 ODS 中最后执行解码的 S-ODSslast 的解码将在该图的周期 dd2 期间结束，则该 S-ODSslast 的 PTS(DSn[S-ODSslast])应该被设置为指示周期 dd2 的时间。

另外，初始显示之前，应该执行已解码图形对象的屏幕清除和传输。从而 ICS 的 PTS(DSn[ICS])应该被设置为在将该 PTS(DSn[S-ODSslast])的值、屏幕清除需要的周期(90,000×(8×video_width×

video_height//128,000,000)) 和解码图形对象的传输周期(90,000×(ΣSIZE(DSn[ICS.BUTTON[i]])//128,000,000))相加后得到的时间之后。

这里应该注意，上述依靠 ICS 中的 PTS 的同步控制不仅包括控制在再现时间线上某一定时处显示按钮，还包括控制在再现时间线上某一周期期间显示弹出菜单。该弹出菜单是通过按下遥控器 400 上提供的菜单键弹出显示的菜单。依靠 ICS 中 PTS 的同步控制还包括在 AV 剪辑中特定图片数据的显示定时处允许弹出显示。如同构成按钮的 ODS，构成该弹出菜单的 ODS 首先被解码然后被写入图形平面。除非已经完成向图形平面的写操作，否则不可能应答用户的菜单调用。鉴于此，在弹出菜单的同步显示时，弹出显示变得可以实现的时间被写入 ICS 中的 PTS。

在以上说明本发明的记录介质之后，下面说明根据本发明的再现装置。除了对象缓冲器 15 和图形控制器 17 的一些改进之外，根据第二实施例的再现装置的内部结构基本上与第一实施例相同。从而以下详细说明对象缓冲器 15 和图形控制器 17 的改进。

通过流图形处理器 14 执行解码获得的用于构成交互屏幕的图形对象被布置在根据第二实施例的对象缓冲器 15 中。图 58 示出了对象缓冲器 15 的内容与图形平面 8 的比较。对象缓冲器 15 的内容假定图 48 和 49 所示的 ODS 被写入对象缓冲器 15 的情况。图 48 和 49 的例子由 36 个 ODS(ODS11-ODS49)实现四个按钮动画，其中代表该动画所有帧的 ODS 被存储在对象缓冲器 15 中，且存储在对象缓冲器 15 中的这些 ODS 中的每一个的显示位置在图形平面 8 中被定义。该显示位置由相应按钮信息的 Button_horizontal_position 和 Button_vertical_position 定义。通过每次传输一帧将存储在对象缓冲器 15 中的多个 ODS 写入图形平面 8 的相应显示位置实现动画。

第二实施例的图形控制器 17 解释合成缓冲器 16 的 ICS 布置单元，并基于该 ICS 执行控制。该控制的执行定时基于分配给 ICS 的 PTS 值。该图形控制器 17 的重要任务是在交互屏幕的初始显示时间和更新时间的写操作。以下参考图 59 描述在交互屏幕的初始显示时

间和更新时间的写操作。图 59 示出了图形控制器在初始显示时间执行的操作。如该图所示，图形控制器 17 执行控制，使得属于按钮 A 中 S-ODSs 的 ODS 被写入由按钮 A 的按钮信息中 Button_horizontal_position 和 Button_vertical_position 定义的显示位置；同样，属于按钮 B、C、D 中 N-ODSs 的 ODS 分别被写入由按钮 B、C、D 的按钮信息中相应 Button_horizontal_position 和 Button_vertical_position 定义的各个显示位置。注意到箭头 wl, w2, w3 和 w4 指示上述写操作。通过执行写操作，图 51 中所示的初始显示被执行。这里应该注意的是，不是所有 ODS 都需要用于实现交互屏幕的初始显示，只要对象缓冲器 15 包含属于默认选择按钮中 S-ODSs 的 ODS 和属于其它按钮中 N-ODSs 的 ODS，就足够完成交互屏幕的初始显示。从而当属于默认选中按钮中 S-ODSs 的 ODS 和属于其它按钮中 N-ODSs 的 ODS 已经被解码时，可以说已经为图形控制器 17 执行用于交互屏幕初始显示的写操作做好准备。

图 60 示出了当依照第一个用户动作(右移)执行交互屏幕更新时，由图形控制器 17 执行的操作。如该图所示，图形控制器 17 执行控制，使得属于按钮 B 中 S-ODSs 的 ODS 被写入由按钮 B 的按钮信息中 Button_horizontal_position 和 Button_vertical_position 定义的显示位置；同样，属于按钮 A 中 N-ODSs 的 ODS 被写入由按钮 A 的按钮信息中 Button_horizontal_position 和 Button_vertical_position 定义的显示位置。这里注意到箭头 w5, w6, w7 和 w8 指示上述写操作。通过执行写操作，图 51 中所示状态改变将实现。如同初始显示时，按钮 C 和 D 处于正常状态，但是图形平面 8 的写操作被连续地执行以继续该动画。

以上面的相同方式，图 61 和 62 示出了当第一个用户动作为“下移”和“激活”情况下的交互屏幕更新中，由图形控制器 17 执行的操作。在交互屏幕更新时间，需要除了默认选中按钮之外按钮的“S-ODSs”和“A-ODSs”，所以期望所有 ODS 已经存储在对象缓冲器 15 中了。

按照以上构造的再现装置中，如同第一实施例，每个部件以流水

线处理方法执行解码操作。

图 63 是表示由再现装置执行的流水线处理的时序图。第四行表示 BD-ROM 的显示集合，第三行表示 ICS、PDS、ODS 被读到编码数据缓冲器 13 的读周期。第二行表示 ODS 的解码周期，其中由流图形处理器 14 执行解码。第一行表示图形控制器 17 的操作周期。ODS 的解码开始时间分别由 DTS11、DTS12 和 DTS13 表示。属于 N-ODSs 的那些 ODS 中第一个 ODS(N-ODSs[ODS1])在 DTS11 之前被全部存储到编码数据缓冲器 13 中。属于 N-ODSs 的那些 ODS 中最后一个 ODS(N-ODSs[ODSn])在 DTS12 之前被全部存储到编码数据缓冲器 13 中。这样，每个 ODS 将在它自己的 DTS 所示时间之前已经被读入编码数据缓冲器 13。

另一方面，每个 ODS 的解码结束时间由该图中的 PTS11、PTS12 和 PTS13 表示。由流图形处理器 14 执行的 N-ODSs(ODSl)的解码在 PTS11 之前完成；N-ODSs(ODSn)的解码在 PTS12 之前完成。这样，在每个 ODS 的 DTS 所示时间处，该 ODS 已经被读入编码数据缓冲器 13，被读入编码数据缓冲器 13 的每个 ODS 将在相应 PTS 所示时间之前被解码并写入对象缓冲器 15。流图形处理器 14 以流水线处理方法执行这些序列操作。

当默认选中按钮被静态确定时，当对于 1)与正常状态对应的按钮状态组和 2)与选中状态对应的按钮状态组的第一个 ODS 的编码完成时，交互屏幕初始显示所需要的所有图形对象在对象缓冲器 15 中已经就绪。该图中，在 PTS13 所示时间，交互屏幕初始显示所需要的所有图形对象就绪。

该图中，第一行的周期 cd1 是清除图形平面 8 所需要的周期。另外，周期 td1 是向图形平面 8 写入构成交互屏幕第一页的图形对象所需要的周期，这些图形对象在对象缓冲器 15 中获得的图形对象之中。这些图形对象在图形平面 8 中的确切存储位置是由 button_horizontal_position 和 button_vertical_position 表示的位置。即，ODS 的 PTS13 加上 cd1(屏幕清除周期)、td1(已解码图形对象的写周期)，在获得的周期内，将在图形平面 8 中获得构成交互屏幕的解压缩图形。然后，

通过(1)使 CLUT 单元 9 执行解压缩图形的颜色变换和 2)使加法单元 10 将存储在视频平面 6 中的解压缩图片进行组合，将获得合成图像。

与在对包含在显示集合中的所有 ODS 进行解码之后执行初始显示的情况相反，上述情况中，可以不管与选中状态对应的按钮状态组的解码是否已经完成，或与激活状态对应的按钮状态组的解码是否已经完成，来执行初始显示。从而这种情况下，将在到达该图中的周期 hy1 时较早地执行初始显示。

应该注意，该图中每组 ODSl-ODSn 被分配给相同编号，如 1 和 n。然而，这些组彼此不同，分别属于 N-ODSs、S-ODSs 和 A-ODSs。下面每幅相似图中也是这种情况。

图形解码器 12 中，甚至当图形控制器 17 连续执行图形平面 8 的清除或向图形平面 8 写操作的同时，流图形处理器 14 连续执行解码(第二行中，ODSn 的解码周期、ODS1 的解码周期和 ODSn 的解码周期 n)。从而，变得可以比常规情况下更早地完成除了由图形控制器 17 处理的 ODS 之外其它 ODS 的解码，因为该其它 ODS 的解码将与由图形控制器 17 处理的 ODS 解码同时进行。因为通过完成该其它 ODS 的解码可以较早地准备好更新交互屏幕，所以要使用该其它 ODS 的交互屏幕更新将相应地将比常规情况更早完成。上述流水线处理使得交互屏幕的初始显示及其更新能够被没有延迟地执行。

图 63 假定默认选中按钮已经被静态确定的情况。相反，图 61 是表示默认选中按钮动态改变情况下再现装置的流水线处理的时序图。当默认选中按钮动态改变时，当属于按钮状态组的所有 ODS 已经被解码且在图形平面中获得图形对象时，初始显示所需要的图形对象就绪。与在对包含在与激活状态对应的按钮状态组中的所有 ODS 进行解码之后执行初始显示的情况相反，上述情况中，可以不管与激活状态对应的按钮状态组的解码是否已经完成，来执行初始显示。从而，这种情况下初始显示可以在到达该图中的周期 hy2 时被较早执行。

图 65 是表示图形平面 8、对象缓冲器 15、编码数据缓冲器 13 和合成缓冲器 16 的占用随时间变化的时序图。该图中使用的占用的注释与图 30 中使用的注释一致。由于在第二实施例中要解码构成

N-ODSs,S-ODSs, A-ODSs 的 ODS，所以单调上升部分和单调下降部分的数目比图 30 中多。除了这个不同之外，图 65 与图 30 相同。如同第一实施例，利用分配给 ODS 的 DTS 和 PTS、分配给 ICS 的 DTS 和 PTS、图 27 中所示的每个缓冲器的大小和传输速率，来说明图 65 中例子的示意图。另外，通过创建这些图，用户在创作阶段就可以知道每个缓冲器如何改变状态。因为通过更新 DTS 和 PTS 可以调节每个缓冲器状态的转变，所以在该实施例中也可以避免产生超过再现装置侧解码器技术规范的解码负担，并避免再现过程中易于发生的缓冲器溢出。由此，在再现装置的开发阶段硬件/软件实现将变得容易。

接下来说明实现第二实施例的再现装置所需要的软件改进。

图 66 是表示功能段装载操作的处理流程图。该图基于图 31 的流程图。不同之处在于，图 66 中在步骤 S29 之后添加了步骤 S36、S37。

步骤 S36 用于判断 command_update_flag 是否为 1。如果为 1(步骤 S36：是)，则只有按钮信息中的按钮命令被载入编码数据缓冲器 13，其它命令被忽略(步骤 S37)。如果为 0，则控制进行到步骤 S22，从而忽略代表“获取点”的 ICS(步骤 S24)。

接下来，假定如图 67 中所示执行多路复用的情况，以下说明怎么读出 DS。图 67 的例子将三个 DS 与运动图片多路复用。在这三个 DS 中，第一个 DS1 的 Composition_state 为“时元开始”，包括称为 LinkPL(PL#5)的按钮命令，且 Command_update_flag 被设置为 0。

DS10 是 DS1 的“复制”，其 Composition_state 为“获取点”，包含称为 LinkPL(PL#5)的按钮命令，且 Command_update_flag 被设置为 0。

DS20 是 DS1 的“继承”，其 Composition_state 为“获取点”。与 DS1 不同之处在于按钮命令 LinkPL(PL#10)，从而为了代表该命令，其 Command_update_flag 被设置为 1。

这里假定这三个 DS 与运动图片被多路复用在一个 AV 剪辑中，并执行 ms1 的向图片数据 pt10 的跳转操作。这种情况下，与跳转目标最接近的 DS10 成为图 66 的目标。在步骤 S27，Composition_state 被判断为“获取点”，但图形解码器 12 中不存在先前 DS。从而忽略

标志被设置为 0，并将该 DS10 载入再现装置的编码数据缓冲器 13(图 68 的 hsl)。另一方面，当跳转操作目标在显示集合存在的位置之后时(ms2)，显示集合 20(图 68 的 hs2)将被读入编码数据缓冲器 13。

图 70 示出了如图 69 中执行的正常再现中 DS1、DS10 和 DS20 的装载。在这三个 DS 中，其 ICS 的 Composition_state 为“时元开始”的 DS1 被载入编码数据缓冲器 13(步骤 S23)。然而，其 ICS 的 Composition_state 为“获取点”的 DS10 的忽略标识为 1(步骤 S29)。从而构成 DS10 的功能段将不被载入编码数据缓冲器 13，而是被忽略(步骤 S24)。另外，对于 DS20，ICS 的 Composition_state 为“获取点”，但是其 Command_update_flag 被设置为 1。从而步骤 S36 得到“是”，所以只有按钮命令被装载，且编码数据缓冲器 13 中只有 DS 的 ICS 的按钮命令被 DS20 的按钮命令代替(步骤 S37)。然而，忽略标志仍代表 1，所以与该按钮命令不同的其他内容将不被装载，而是被忽略。

当到达 DS20 时，现实内容保持相同，但是按钮命令已经从 DS 的 LinkPL(PL#5)变成 LinkPL(PL#19)。这种按钮命令的代替允许改变按钮命令内容的控制。接下来说明图形控制器执行的处理。图 71 是表示图形控制器 17 执行的处理主程序的流程图。该流程图中，重复执行以下三个操作：时间标记同步操作(步骤 S35)，动画显示操作(步骤 S36)，和 UO 操作(步骤 S37)。

这里，说明图形控制器 17 执行的处理。图 36-38 中所示的与图 71-78 中所示的图形控制器 17 执行的处理发生很大改变。图 71 是表示图形控制器 17 执行的处理主程序的流程图。图 72 是表示使用时间标记实现同步控制的处理流程图。该流程图中，执行步骤 S41、43-47 中的任意条件的判断。如果这些条件中的任意条件成立，则执行相应操作，然后返回到主程序。该处理是子程序。

步骤 S41 用于判断当前再现点是否为由 S-ODSsfirst 的 PTS 代表的时间和由 S-ODSslast 的 PTS 代表的时间之一。如果该当前再现点被判断为上述时间之一，则计算其周期 α。通过将(2)清除图形平面所需要的周期和(1)向图形平面写由 ODS 解码获得的图形对象所需要的周期相加得到周期 α。

在步骤 S42，图形控制器 17 参考 ICS 中的 Composition_state，以及(a)如果 Composition_state 为“时元开始”，则将 α 设置为“平面清除周期(2)+平面写周期(3)”；(b)如果 Composition_state 为“获取点”，则将 α 设置为平面写周期(3)。平面写周期(3)的计算如下执行：如果 default_selected_button_number 为有效值，则使用图 54A 的计算方法；如果 default_selected_button_number 为 0，则使用图 54B 的计算方法。当计算出 α 后，控制返回循环处理。

步骤 S43 用于判断当前再现点是否是 ICS 中 PTS-α 代表的时间。如果判断结果是肯定的，则执行向图形平面 8 的写操作(步骤 S51)，且控制返回主程序。

步骤 S45 用于判断当前再现点是否是 ICS 中的 PTS。如果判断结果是肯定的，则指示输出图形平面 8 的内容。这些内容的目的地是 CLUT 单元 9。CLUT 单元 9 对这些内容执行颜色变换。然后交互屏幕将与视频平面 9 的内容组合。结果是，执行初始显示(步骤 S52)。然后，变量“animation(p) (p=1,2,3... n)”被设置为 0(步骤 S53)，且控制返回主程序。这里，变量 animation (p)是用于执行按钮(p)的动画显示的全局变量，其指示帧序列中当前显示帧的编号(全局变量是在多个流程图中都有效的变量)。从而在步骤 S53，所有按钮的按钮(p)被设置为 0。

步骤 S46 和 S47 用于判断当前再现点是否到达 ICS 中描述的时间信息。

步骤 S46 用于判断当前再现点是否是由 selection_Time_Out_PTS 代表的时间，如果判断结果是肯定的，则激活由 default_activated_button_number 代表的按钮的操作被执行，且控制返回主程序(步骤 S54)。

步骤 S47 用于判断当前再现点是否是 Composition_Time_Out_PTS，如果判断结果是肯定的，则清除屏幕，然后控制返回主程序(步骤 S55)。在上述同步操作中，步骤 S51 和 S54 的每一个操作被作为子程序执行。然后参考图 73 说明步骤 S51 的子程序。

图 73 是表示向图形平面 8 写菜单初始显示的操作流程图。步骤

S64 用于判断 ICS 中的 Composition_state 是否为“时元开始”，如果判断结果为肯定的，则在步骤 S65 清除图形平面，并执行步骤 S66-S73 的操作。清除图形平面 8 所需要的周期是图 56 和图 57 中的周期 cd1。如果步骤 S64 的判断结果为否定的，则跳过步骤 S65，并执行步骤 S66-S73 的操作。

步骤 S66-S73 形成循环处理，对于 ICS 的每条按钮信息重复该循环处理(步骤 S66、S67)。应该经历该循环处理的按钮信息被称为 button-info(p)。

步骤 S67 用于判断 default_selected_button_number 的指示是否有效。步骤 S68 用于判断 button-info(p)是否是与 default_selected_button_number 指示的默认选中按钮对应的按钮信息。

如果步骤 S68 的判断是否定的，则从对象缓冲器 15 中找到由 button-info(p)的 normal_state_info 指示的 start_object_id_normal 的图形对象，并将该对象识别为图形对象(p)(步骤 S69)。

如果步骤 S68 的判断是肯定的，则从对象缓冲器 15 中找到由 button-info(p)的 selected_state_info 指示的 start_object_id_selected 的图形对象，并将该对象识别为图形对象(p)(步骤 S70)，然后按钮(p)被设置为当前按钮(步骤 S71)。当前按钮是在当前显示的交互屏幕中被设置为选中状态的按钮。再现装置将该当前按钮的标识符存储为 PSR(10)。

一旦作为步骤 S69 和步骤 S70 的结果识别出图形对象(p)，就将该图形对象(p)写到图形平面 8 上由 button_info(p)的 button_horizontal_position 和 button_vertical_position 指示的位置上(步骤 S72)。通过对于每条按钮信息重复上述操作，多个图形对象中的第一个图形对象被写入图形平面 8，其中该多个图形对象中的每一个表示相应按钮的状态。对于对象缓冲器 15 中至少用于初始显示所需的图形对象执行该操作所需要的周期由图 56 和 57 中的周期 td1 表示。

当 default_selected_button_number 为“=0”且默认选中按钮动态改变时，步骤 S67 将为否，并判断 button_info(p)是否与当前按钮对应。如果步骤 S67 的判断结果为肯定的，则控制进行到步骤 S70；如果判

断结果为否定的，则控制进行的步骤 S69。

接下来参考图 74 说明步骤 S54 的子程序处理。

图 74 是表示默认选中按钮的自动激活处理的流程图。首先，判断 default_activated_button_number 为 0 还是 FF(步骤 S75)。如果步骤 S75 的判断结果为“00”，则不执行处理且控制返回主程序；如果步骤 S75 的判断结果为“FF”，则当前按钮 i 被改变成激活状态(步骤 S77)，变量 animation(i)被设置为 0，且控制返回到主程序(步骤 S78)。

如果步骤 S75 的判断结果不是“00”或“FF”，则由 default_activated_button_number 指定的按钮被设置为当前按钮(步骤 S76)，当前按钮 i 被改变成激活状态(步骤 S77)，与当前按钮 i 对应的变量 animation(i)被设置为 0，且控制返回到主程序(步骤 S78)。

上述处理使得选中状态下的按钮在预定时间后能够被改变成激活状态。

接下来说明依靠菜单的动画(步骤 S36)。图 75 是说明动画显示处理的流程图。

这里，通过向图形平面 8 写图形对象实现初始显示，该图形对象由每个 button_info 的(1) normal_state_info 的 start_object_id_normal 和(2) selected_state_info 的 start_object_id_selected 指定。这里，“动画”是一种处理，在每次完成步骤 S35-S37 的循环处理的一个循环后，利用每个按钮的任意帧(即，第 q 帧图形对象)更新图形平面。通过将 button_info 的 normal_state_info 和 selected_state_info 指示的图形对象一个接一个地写入图形平面 8，然会返回主程序来执行该更新。这里，对于每条按钮信息，在识别由 button_info 的 normal_state_info 和 selected_state_info 指示的每个图形对象时使用变量 q。

参考图 75 详细说明用于实现该动画显示的处理。该流程图假定 ICS 的 repeat_normal_flag 和 repeat_selected_flag 被设置成指示“必须重复”，以简化说明。

步骤 S80 用于判断是否已经完成初始显示。如果步骤 S80 的判断结果是否定的，则控制返回，不执行任何处理，如果步骤 S80 的判断结果是肯定的，则执行步骤 S81-S93。步骤 S81-S93 构成循环处理(步

骤 S81、S82)，对于 ICS 中的每个 button_info 重复步骤 S83-S93 的操作。

步骤 S83 将与 button_info(p)对应的变量 animation(p)设置到变量 q。通过执行该步骤，变量 q 将指示与 button_info(p)对应的当前编号的帧。

步骤 S84 用于判断 button_info(p)是否与当前处于选中状态的按钮(以下称为“当前按钮”)对应。

如果 button_info(p)被判断为不同于当前按钮，则执行步骤 S86 的判断。

步骤 86 用于判断当前按钮是否处于激活状态，如果判断为肯定的，则通过将变量 q 与 button_info(p).actioned_state_info 中的 start_object_id_actioned 相加得到的标识符被设置为 ID(q)。然后，执行包含在 button_info(p)中的那些按钮命令中的一个(步骤 S88)。

如果当前按钮被判断为不处于激活状态，则通过将变量 q 与 button_info(p). selected_state_info 中的 start_object_id_ selected 相加得到的标识符被设置为 ID(q) (步骤 S89)。

一旦 ID(q)作为上述操作结果被确定，则具有 ID(q)并存在于对象缓冲器 15 中的图形对象(p)被写到图形平面 8 中由 button_info(p)的 button_horizontal_position 和 button_vertical_position 指示的位置上(步骤 S90)。

通过上述循环处理，在构成当前按钮选中状态(或激活状态)和其它按钮正常状态的多个图形对象中，与第 q 页对应的图形对象被写入图形平面 8。

步骤 S91 用于判断 start_object_id_normal+q 是否已经达到了 end_object_id_normal。如果步骤 S91 的判断结果是否定的，则将变量 q 加 1 获得的值设置为变量“animation(p)” (步骤 S92)。如果步骤 S91 的判断结果是肯定的，则变量“animation(p)”被初始化为值 0(步骤 S93)。对于 ICS 中的所有 button_info 重复上述操作(步骤 S81、S82)。当所有 button_info 都经历了上述操作时，控制返回主程序。

以上说明的步骤 S80-S93 期间，每次主程序(步骤 S35-S37)被执

行一次，交互屏幕的每个按钮的图像将被更新成一个新的图形对象。这说明，当上述主程序(步骤 S35-S37)被执行几次后，就实现了所谓的动画。在该动画中，图形控制器 17 调节时间，使得一帧图形对象的显示间隔为 animation_frame_rate_code 指示的值。

这里应该注意，在步骤 S88，包含在 button_info(p)中的按钮命令被逐条地执行。然而，在与激活状态对应的图形对象序列已经被显示后，也可以共同执行这些按钮命令。接下来参考图 76 说明在主程序的步骤 S37 中执行的 UO 操作的处理。

图 76 是说明 UO 操作的处理流程图。该流程图中，判断步骤 S100-S103 中的任意条件是否成立。如果这些条件中的任一个成立，则执行相应处理，然后返回到主程序。步骤 S100 用于判断 UomaskTable 是否被设置为“1”，如果该判断为肯定的，则控制返回到主程序，不执行任何操作。

步骤 S101 用于判断上移健/下移健/左移健/右移健是否被按下。如果判断为肯定的，则当前按钮被改变(步骤 S104)，然后判断当前按钮的 auto_action_flag 是否为 1(步骤 S108)。如果步骤 S108 的判断为否定的，控制返回到主程序。如果步骤 S108 的判断为肯定的，控制进行到步骤 S105。

步骤 S102 用于判断激活键是否被按下。如果判断为肯定的，当前按钮 i 被改变为激活状态(步骤 S105)。然后，变量“animation (i)”被设置为 0(步骤 S106)。

步骤 S103 用于判断是否是输入数值的情况。如果判断为肯定的，则执行对应的数值输入操作(步骤 S107)，且控制返回到主程序。图 76 的处理中，步骤 S104 和步骤 S107 是子程序。图 77 和 78 中表示这些子程序的处理。以下说明这些流程图。

图 77 是表示当前按钮改变操作的处理流程图。首先，在属于当前按钮的 neighbor_info 的 upper_button_number、lower_button_number、Left_button_number 和 right_button_number 中，识别与被按下的键对应的一个(步骤 S110)。

然后当前按钮被设置为“按钮 i”，新的当前按钮被设置为“按钮

j”(步骤 S111)。步骤 S112 用于判断步骤 S111 中设置的按钮 j 是否与按钮 i 对应。如果它们彼此对应，则控制返回主程序，不执行任何处理。如果它们不互相对应，则按钮 j 被设置为当前按钮(步骤 S113)，变量“animation(i)”和变量“animation (j)”被设置为 0，控制返回主程序(步骤 S114)。

图 78 是表示数值输入操作的处理流程图。判断是否具有其按钮编号与输入数值匹配的 button info. j(步骤 S121)。然后判断按钮 info.j 的 numerically_selectable_flag 是否为 1(步骤 S122)。如果步骤 S121 和步骤 S122 为“是”，则当前按钮被改变为正常状态，且按钮 j 被设置为当前按钮(步骤 S123)，变量“animation(i)”和变量“animation (j)”被设置为 0(步骤 S124)。这些操作之后，判断按钮 info.j 的 auto_action_flag 是否为 1(步骤 S125)。如果判断为否定的，则控制返回到主程序。

如果判断为肯定的，则在步骤 S126 中当前按钮被改变为激活状态，然后控制返回到主程序。

如果步骤 S121-S122 中的任一个为否，则控制返回主程序。

图形控制器 17 执行上述操作，从而进行同步显示。这里，注意如果利用弹出显示等执行由用户操作触发的交互屏幕显示，则流图形处理器 14 和图形控制器 17 执行以下操作，这些操作与执行同步显示的操作一样。通过执行以下操作，在图形平面 8 中获得图形对象。在如上所述获得图形对象之后，等待直到当前再现点经过分配给 ICS 的 PTS 指示的时间为止。然后，上述时间之后，如果 UO 控制器 18 接收到指示菜单调用的 UO，则将其输出到 CLUT 单元 9，并指示 CLUT 单元 9 操作存储在图形平面 8 中的图形对象。如果这种输出与 UO 同步执行，则将实现与按下该菜单调用一致的弹出显示。

上述列举了 ICS 中 PTS 的设置和 ODS 中 DTS 和 PTS 的设置，这些都属于 DSn。然而，没有提及 ICS 中的 DTS、PDS 中的 DTS 和 PTS、END 中的 DTS 和 PTS。鉴于此，以下说明与这些相关的时间标记。因为第二实施例中不存在 WDS，所以在 1)DSn 中第一个 PDS(PDS1)的解码开始时间(即 DTS(DSn[ODS1]))和 2)DSn 中第一个

PDS(PDS1)变得可用的时间(即，PTS(DSn[PDSl]))之前，ICS 应该被载入合成缓冲器 16 中。即，应该设置满足以下方程式的值：

DTS (DSn [ICS]) ≤DTS (DSn [ODSl])

DTS (DSn [ICS])≤PTS (DSn[PDSl])

接下来说明对于属于 DSn 的每个 PDS，DTS 和 PTS 的设置。

属于 DSn 的每个 PDS 在 CLUT 单元 9 中变得有效的时间为从(1)ICS 被载入合成缓冲器 16 的时间到(2)第一个 ODS 的解码开始时间(DTS(DSn[ODSl])。鉴于此，属于 DSn 的每个 PDS(即，PDS1-PDSlast)的 PTS 值应该被设置为满足以下关系的值：

DTS (DSn [ICS]) ≤ PTS (DSn[PDS1])

PTS (DSn [PDSj]) ≤PTS (DSn[PSj+l]) ≤ PTS (DSn [PDSlast])

PTS (DSn [PDSlast]) ≤ DTS (DSn[ODSl])

接下来说明属于 DSn 的“显示集合段的 END”的 PTS。属于 DSn 的该 END 表示 DSn 的结束。从而它应该是 DSn 的最后一个 ODS (ODSlast)的解码结束时间。该解码结束时间由 PTS (PTS(DSn [ODSlast]))指示，所以 END 的 PTS 应该被设置为由下式指示的值：

PTS (DSn [END]) =PTS (DSn [ODSlast])

考虑到属于 DSn 和 DSn+1 的 ICS 之间的关系，DSn 中的 ICS 在第一个 ODS (即 ODS1)的装载时间之前被载入合成缓冲器 16。从而 END 中的 PTS 应该在 1)属于 DSn 的 ICS 的装载时间(即 DTS(DSn [ICS]))之后，并在 2)属于 DSn+1 的 ICS 的装载时间(即 DTS(DSn+1 [ICS]))之前。从而，END 中的 PTS 应该满足以下关系：

DTS (DSn [ICS]) ≤ PTS (DSn [END]) ≤ DTS (DSn+l [ICS])

另一方面，第一个 ODS(即 ODS1)的装载时间、END 的 PTS(即 PTS(DSn[END]))应该在属于 DSn 的 PDS 的装载时间之后。从而，END 中的 PTS 应该满足以下关系：

PTS (Dsn [PDSlast]) ≤ PTS (DSn [END])

由于被设置了 DTS 和 PTS 的 ICS、PDS、ODS 被预先合并在 AV 剪辑中，所以很方便描述使再现装置在某一帧运动图片出现在屏幕中时执行某一操作的交互控制。即，上述布置方便描述与运动图片内容

精确同步的交互操作。另外，ICS、PDS 和 ODS 本身被多路复用到 AV 剪辑中。从而，在要被用户执行再现控制的段很多如几百段的情况下，不需要将与所有段对应的所有 IDS、PDS 和 ODS 都存储在存储器中。因为要从 BD-ROM 中读出这些 ICS、PDS 和 ODS，所以以下布置足够了。即，与此刻要播放的运动图片部分对应的 ICS、PDS 和 ODS 驻留在存储器中。该运动图片部分的播放结束后，从存储器删除相应的 ICS、PDS 和 ODS，而将与之后的运动图片部分对应的 ICS、PDS 和 ODS 存储在存储器中。因为 ICS、PDS 和 ODS 被多路复用到 AV 剪辑上，所以即使 ICS、PDS 和 ODS 的数目变成几百，也可以将存储器的占用限制在最小需要级别上。

如上所述，本实施例具有 360 页 ODS 用来实现动画。从而当以三种状态来分组按钮材料时，ODS 会以 120 页分组(即分为三个按钮状态组)。按钮状态组被布置成，与对应于较晚出现状态的组相比，在开始处更多地安排对应于较早出现状态的组。因此，再现时，与对应于较晚出现状态的按钮状态组相比，与较早出现状态对应的按钮状态组被相应较早的载入再现设备。因此，即使所有 360 页 ODS 的解码没有完成，只要完成全部 ODS 的约 1/3-2/3，则至少已经准备好执行初始显示。因为完成全部 ODS 的约 1/3-2/3 就可以开始初始显示操作，所以即使要读取和解码很多 ODS，初始显示也不会被延迟。从而，即使交互屏幕包含用来取悦用户的动画，也会快速地执行交互屏幕。

(第三实施例)

本实施例涉及 BD-ROM 的制造方法。图 79 示出了制造第一实施例中说明的 PCS 的方法。

该 BD-ROM 的制造方法包括：材料生产步骤 S201，例如拍摄图像图片和记录相应音频；创作步骤 S202，用于产生应用格式；和印制步骤 S203，通过执行印制/层压来完成 BD-ROM。

这些步骤中，对 BD-ROM 的创作步骤包括以下步骤 S204-S210。

在步骤 S204，描述控制信息、窗口定义信息、调色板定义信息

和图形。在步骤 S205，将控制信息、窗口定义信息、调色板定义信息和图形分别转换成功能段。在步骤 S206，根据要被同步显示的图片出现的时间设置 PCS 的 PTS。在步骤 S207，根据 PTS[PCS]值设置 DTS[ODS]和 PTS[ODS]。在步骤 S208，根据 DTS[ODS]值设置 DTS[PCS]、PTS[PDS]、DTS[WDS]和 PTS[WDS]，且在步骤 S209，将播放器模型的每个缓冲器中占用随时间的变化表示为一个图形。在步骤 S210，判断图中表示的随时间变化是否满足对该播放器模型的限制。如果步骤 S210 的判断结果为肯定的，则在步骤 S212 产生图形流，并通过将图形流与视频流和音频流多路复用获得一个 AV 剪辑，视频流和音频流与图形流分开产生。然后使 AV 剪辑与 BD-ROM 的格式一致，从而完成一种应用格式。

以上说明了根据第一实施例的记录介质的制造方法。图 80 表示根据第二实施例的记录介质的制造方法。图 80 中，步骤 S304-S308 代替图 79 中的步骤 S204-S208。

以下说明步骤 S304-S308。在步骤 S304，描述控制信息、调色板定义信息和图形。在步骤 S305，将控制信息、调色板定义信息和图形分别转换成功能段。在步骤 S306，根据要被同步显示的图片出现的时间设置 ICS 中的 PTS。然后在步骤 S207，根据 PTS[ICS]值设置 DTS[ODS]和 PTS[ODS]。在步骤 S308，根据 DTS[ODS]值设置 DTS[ICS]和 PTS[PDS]。

(注意)

不用说，以上说明没有示出本发明的所有实施例和使用形式。通过添加了任意以下修改(A)、(B)、(C)、(D)、...等的实施例也可以实现本发明。请注意，本发明权利要求中的发明是上述实施例或基于以下修改的修改实施例的扩展和一般化的描述。这种扩展和一般化的程度反映了申请时本领域的状态。

(A)在所有实施例中，根据本发明的记录介质被假定为 BD-ROM。然而，本发明的记录介质的特征在于存储在该记录介质中的图形流，且该特征不依赖于 BD-ROM 的物理特性。即，能够记录图形流的任

何记录介质都可以用来实现本发明。这些例子包括：光盘，如DVD-ROM、DVD-RAM、DVD-RW、DVD-R、DVD+RW、DVD+R、CD-R、CD-RW；光磁盘，如 PD 和 MD。这些例子还包括半导体存储卡，如压缩闪存卡(compact flash card)、智能媒体卡(smart media)、记忆棒、多媒体卡和 PCM-CIA 卡。另外，这些例子包括：磁记录盘，如软盘、SuperDisk、Clik；可拆卸硬盘驱动器，如 ORB、Jaz、SparQ、SyJet、EZFley 和微硬盘(micro drive)。这些例子还包括集成在装置中的硬盘。

(B)在所有实施例中，再现装置解码存储在 BD-ROM 中的 AV 剪辑，然后将其输出到电视机。然而，还可能是一种结构，其中再现装置仅是 BD-ROM 驱动器，而其它部件包含在电视机中。这种情况下，再现装置和电视机可以经由 IEEE 1394 相互连接来构成本地网络。这些实施例的再现装置与相连接的电视机一起使用。然而，再现装置可以与显示器整合在一起。另外，在每个实施例的再现装置中，只有系统 LSI(集成电路)可以被认为是本发明，它是处理的本质。该再现装置和集成电路都是本说明书描述的发明，所以，基于第一实施例的再现装置的内部结构制造具有上述形式和方式的再现装置的行为也构成本发明的一个实施方式。不管是否有偿地转让（如果有偿，即为销售，如果无偿，则为礼物）、出租和进口第一个实施例所涉及的再现装置也是实施本发明的行为。此外，同样，通过橱窗展示、商品目录拉客或分发宣传册来许诺转让和出租的行为也是实施本再现装置的行为。

(C)使用硬件资源具体实现每个流程图中所示的信息处理。从而，执行这些流程图中所示处理的任何程序也分别构成独立的发明。与这些程序相关的所有实施例中，程序的形式是结合在相应再现装置中。然而，第一实施例中所示的这些程序本身可以是独立于相应再现装置的实施例。程序本身的实施例包括：(1)制造该程序的行为；(2)不管是否有偿地转让该程序的行为；(3)出租的动作；(4)出口行为；(5)通过交互式电子通信电路提供给公众的行为；(6) 通过橱窗展示、商品目录拉客或分发宣传册来转让给一般用户的行为。

(D)如果在每个流程图中按时间顺序执行的每一步骤中存在的时间概念被认为是指定本发明的必不可少的因素，那么该流程图中的每个处理被解释为揭示再现方法的使用模式。如果通过按时间顺序执行上述流程图中的每一步骤来执行该流程图的处理，从而有效地并仪器地达到本发明的目的，则这将与本发明的记录方法的实施例对应。

(E)当 TS 包被记录到 BD-ROM 时，期望构成 AV 剪辑的每个 TS 包被分配一个额外的头。该额外的头被称为“TP_extra_header”，包括“Arrival_Time_Stamp”和“copy_permission_indicator”，并具有 4 字节长度。分配了 TP_extra_header 的 TS 包(分配了 EX 的 TS 包)被分成每组包含 32 个 TS 包的组，并被写入三个扇区。每组包括 32 个分配了 EX 的 TS 包的这些组的总大小为 6144 字节 (=32×192)，等于三个扇区的总大小(6144 字节(=2048×3)。存储在一个扇区中的 32 个分配了 EX 的 TS 包被称为“对准单元(aligned unit)”。

当在通过 IEEE 1394 连接的本地网络中使用时，该再现装置通过以下传输处理执行对准单元的传输。即，发送装置从对准单元中 32 个分配了 EX 的 TS 包中的每一个包去除 TP_extra_header，对 TS 包的主体编码，然后输出它们。输出 TS 包时，在 TS 包之间许多位置插入同步包。准确的插入位置基于由 TP_extra_header 的 Arrival_Time_Stamp 表示的时间。响应于 TS 包的输出，再现装置输出 DTCP_Descriptor(数据传输内容保护描述符)。DTCP_Descriptor 表示 TP_extra_header 的复制允许/禁止设置。从而，如果 DTCP_Descriptor 被描述成表示“禁止复制”，则当在通过 IEEE 1394 连接的本地网络中使用这些装置时，TS 包将不能被记录在另一装置上。

(F)每个实施例中的数字流是 BD-ROM 标准中的 AV 剪辑。然而，该数字流也可以是 DVD-Video 标准或 DVD-Video 记录标准中的 VOB(视频对象)。VOB 是符合 ISO_IEC13818-1 标准的程序流，并通过多路复用视频流和音频流得到。或者，AV 剪辑中的视频流可以是 MPEG4 方法或 WMV 方法。另外，音频流或者可以是杜比=AC3 方法、MP3 方法、MPEG-AAC 方法或 dts 方法。

(G)这些实施例中的电影作品可以通过编码模拟图像信号得到，或对通过数字广播传播的传输流组成的流数据进行编码得到。另外，内容可以通过对记录在录像带上的模拟/数字电影信号编码得到，或可以是从发布服务器发布的数字作品。

(H)第一和第二实施例中所示的图形对象是已经被用游程编码方法编码的栅格数据。使用游程编码方法作为图形对象的压缩/编码方法的原因是，游程编码方法最适合于压缩/解压缩字幕。字幕的特征在于水平方向上相同像素值的连续长度比较长。从而，如果使用游程编码压缩，可以获得高压缩率。另外，压缩的负荷不重，所以适合产生用于解码处理的软件。本发明中，用于字幕的压缩/解压缩方法也用于图形对象，从而用于执行解码的一个装置结构被字幕和图形对象共享。然而，对图形对象采用游程编码方法不是本发明的必不可少的特征，或者图形对象可以是 PNG 数据。另外，栅格数据可以是矢量数据，或透明图像。

(I)对于根据装置侧语言设置选择的字幕的图形，可以产生 PCS 的显示效果。这样，过去常通过由当前 DVD 中的运动图片主体代表的字符实现的显示效果，能够使用根据装置侧语言设置显示的字幕图形来实现。

(J) 对于根据显示设置由装置侧选择的字幕的图形，可以产生 PCS 的显示效果。具体地，用于诸如宽显示、全景扫描(pan scan)和信箱模式(letter box)这样的多种显示模式的图形被记录在 BD-ROM 中，且根据相连接的电视机的设置，该装置从这些显示模式中选择一种。这种情况下，以上述方式显示的字幕图形将会产生一种显示效果。从而字幕图形看起来更好。这样，过去常通过由当前 DVD 中的运动图片主体代表的字符实现的显示效果，能够使用根据装置侧显示设置的字幕图形显示来实现。这实际上很有价值。

(K)在第一实施例中，向图形平面写的写速率 Rc 被定义为使得窗口大小为整个大小的 25%，从而在一个视频帧内有可能实现图形平面清除和再呈现。然而，或者如果假定垂直回扫时间为 1/29.93 的 25%，那么 Rc 将为 1Gbps。通过这样设置 Rc，将易于图形显示。这实际上

很有价值。

除了在垂直回扫时间进行写操作以外，可以同步地执行与写扫描同步的写操作。这样，即使写速率为 Rc=256Mbps，也易于显示。

(L)在每个实施例中，再现装置配备了图形平面。然而，可以在再现装置上安装行缓冲器代替该图形平面，该行缓冲器可以存储一行解压缩像素。由于可以为每一水平行(即，行)执行图像信号的变换，所以如果提供这种行缓冲器，则再现装置能够执行图像信号的变换。

(M)以上说明了图形形式的字幕，该字幕为代表电影作品中讲话的字符序列。然而，这些字幕可以包含图画、字符和颜色的组合，如同构成商标。另外，这些字幕可以包含所有种类的国家标记、由国家采用的用于监督和授权的官方标记、国际组织的标记、代表具体物品起源地的标记、等等。

(N)第一实施例假定字幕被显示在屏幕的上部/下部，所以窗口相应地被定义在图形平面的上部/下部。然而，也可以将窗口定义在图形平面的左部/右部。这在纵向显示日文字幕时有用。

(O)每个实施例中的 AV 剪辑构成电影作品。然而，AV 剪辑可以用于实现“karaoke”(预录磁带的附属物)。这种情况下，在一首歌的过程中，PCS 可以实现例如改变字幕颜色的显示效果。

(P)在多个再现路径彼此连接，且默认选中按钮根据要采用哪条再现路径而改变的情况下，以下安排是优选的。即，动态方案中的再现控制被描述成，使得在经过每个再现路径时，将该再现路径的特征值设置在再现装置的寄存器中，且再现处理被描述成，使得根据寄存器中设置的值，将按钮设置为选中状态。通过这种安排，可以根据要经过哪条再现路径来改变将处于选中状态的按钮。

工业应用性

本发明的记录介质和再现装置实现了具有显示效果的字幕显示和包含动画的交互显示，从而有助于为市场提供具有高附加值的电影产品，有助于激励电影市场和生活消费品市场。从而，本发明的记录

介质和再现装置在电影工业和生活消费品工业中十分有用。

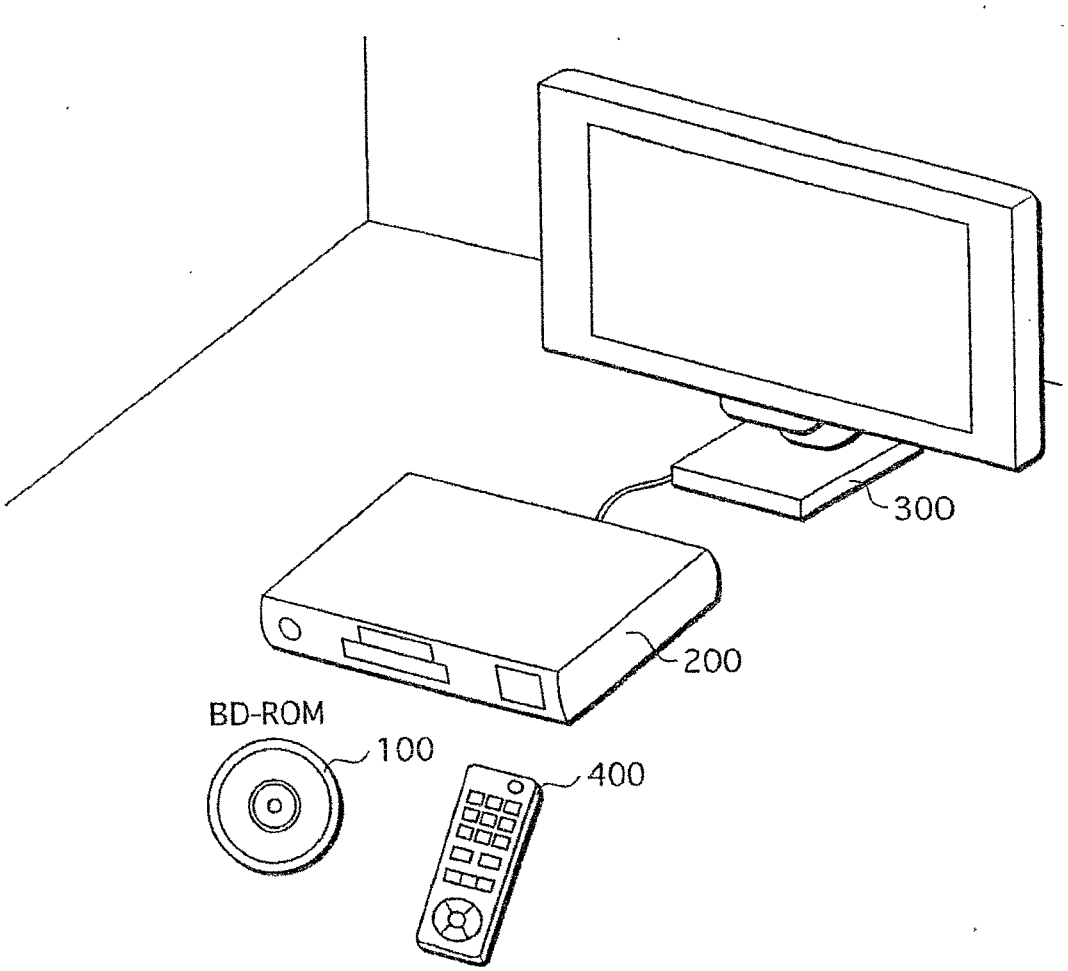


图1

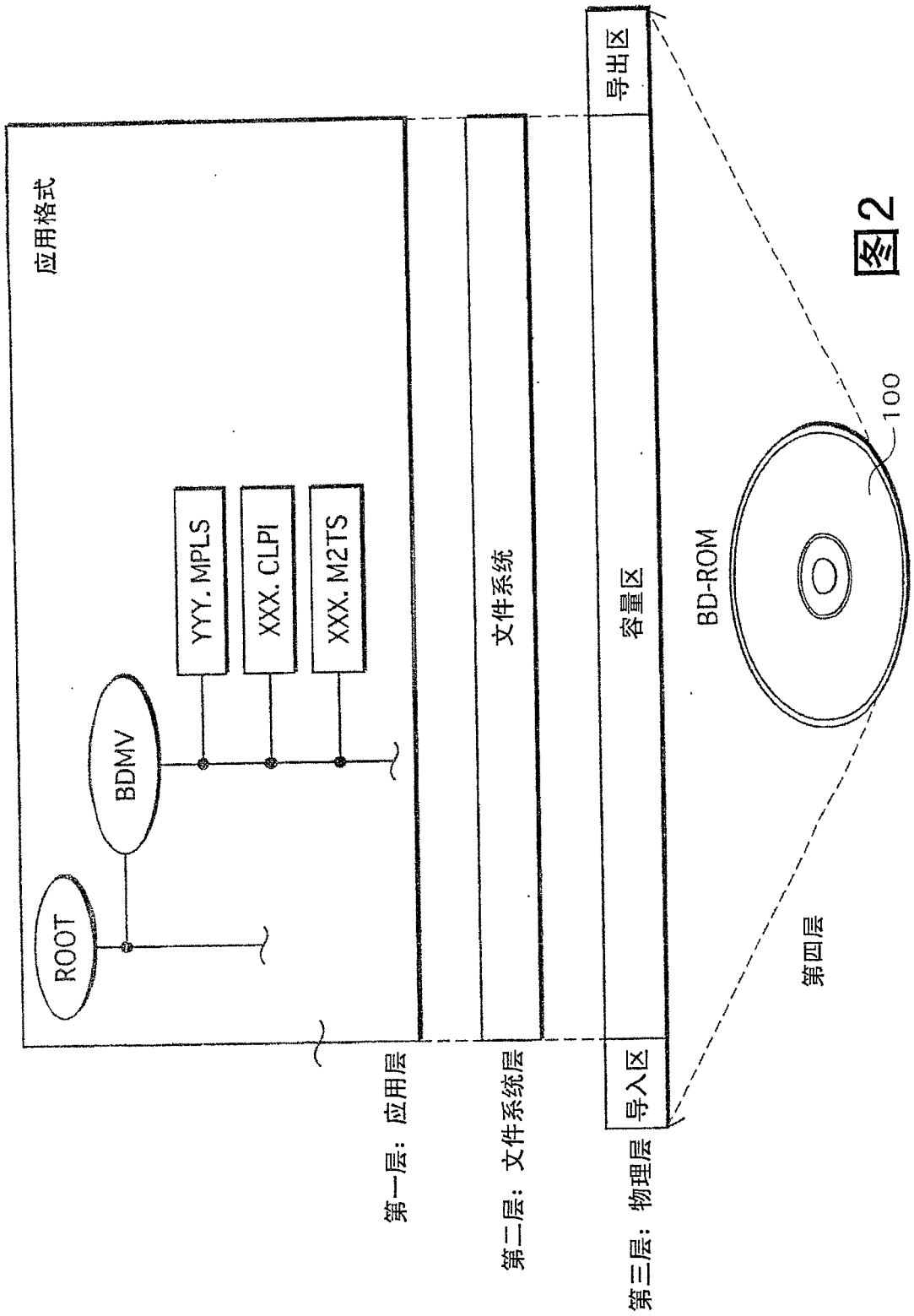

应用格式
ROOT
BDMV
YYY. MPLS
XXX. CLPI
XXX. M2TS
第一层：应用层
文件系统
第二层：文件系统层
导出区
容量区
导入区
第三层：物理层
BD-ROM
100
第四层

图2

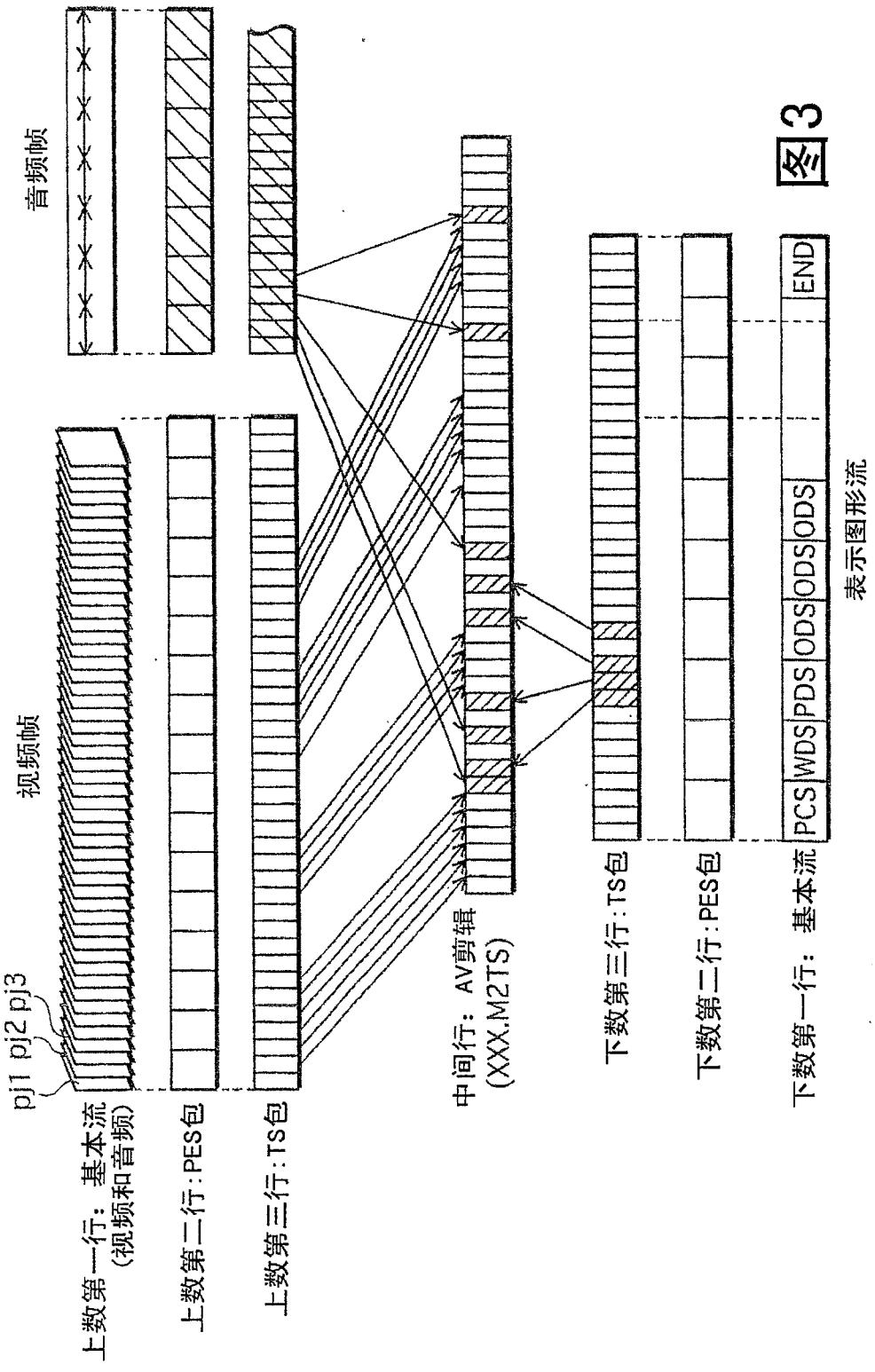

音频帧
视频帧
pj1 pj2 pj3
上数第一行：基本流
(视频和音频)
上数第二行:PES包
上数第三行:TS包
中间行：AV剪辑
(XXX.M2TS)
下数第三行:TS包
下数第二行:PES包
下数第一行：基本流
PCS
WDS
PDS
ODS
ODS
ODS
END
表示图形流

图3

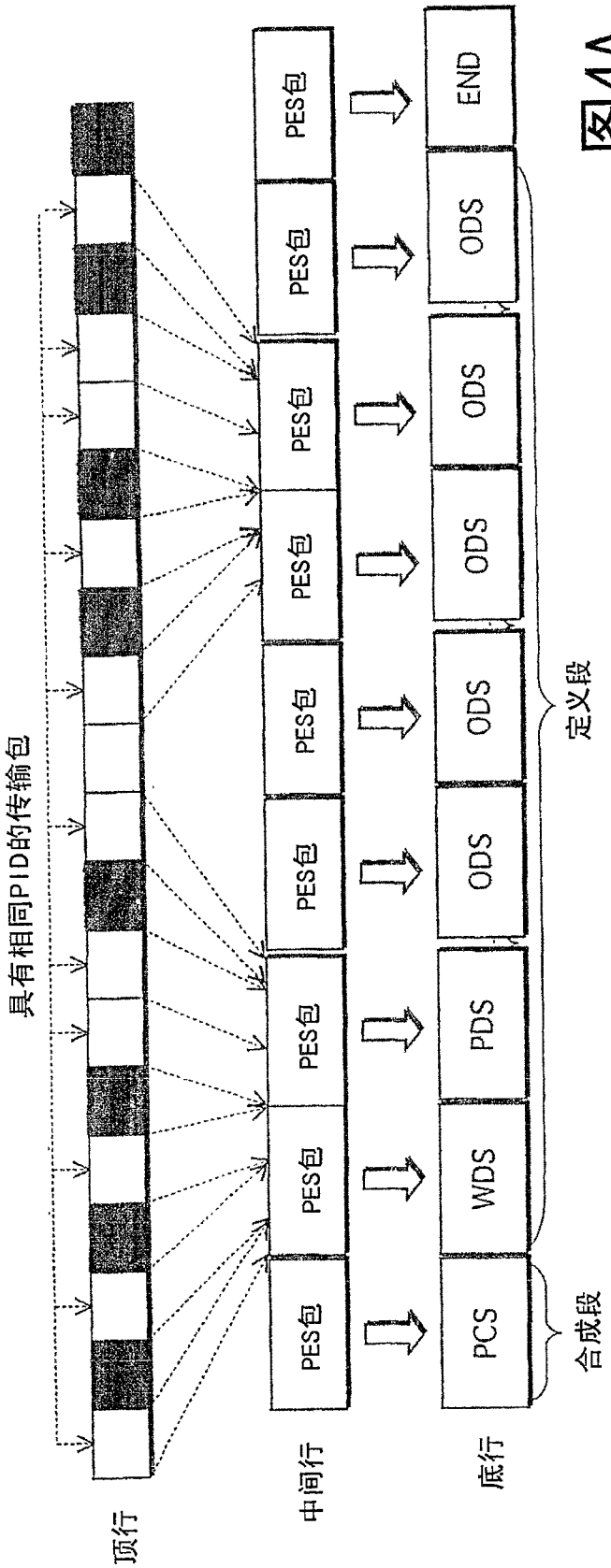

具有相同PID的传输包
顶行
中间行
底行
PES包
PCS
WDS
PDS
ODS
END
定义段
合成段

图4A

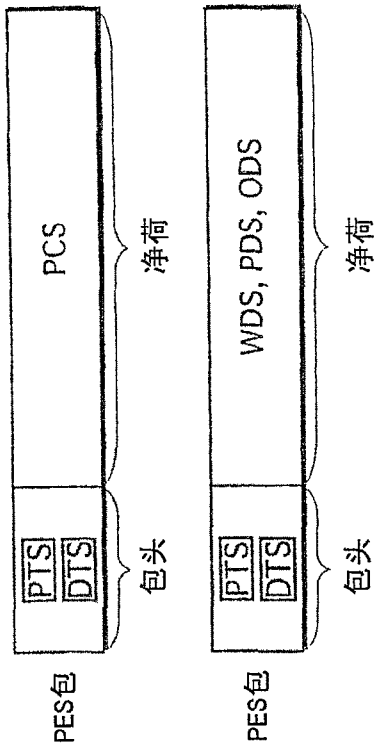

PES包
PTS
DTS
PCS
WDS, PDS, ODS
包头
净荷

图4B

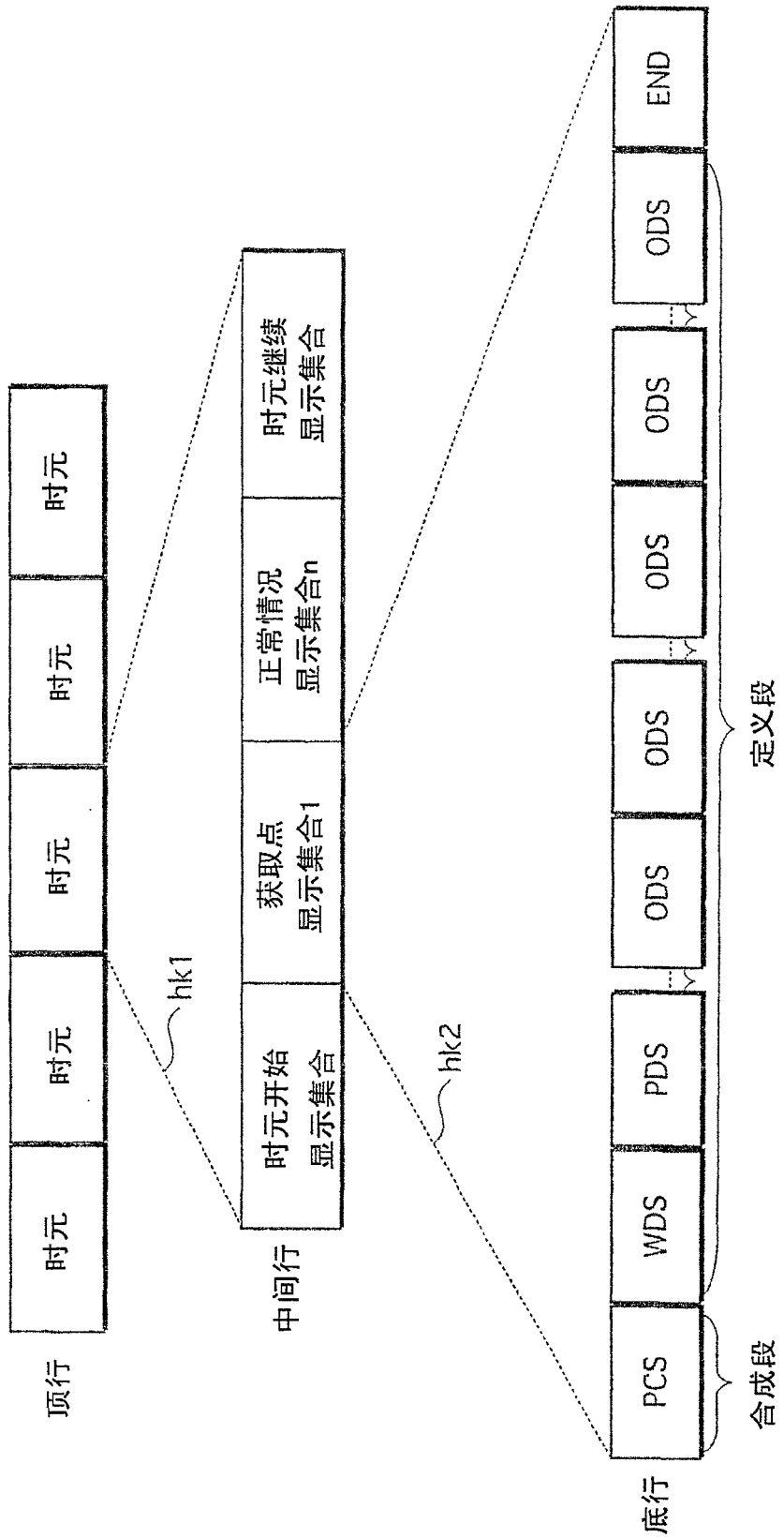

顶行
时元
时元
时元
时元
时元
hk1
中间行
时元开始
显示集合
获取点
显示集合1
正常情况
显示集合n
时元继续
显示集合
hk2
底行
PCS
WDS
PDS
ODS
ODS
ODS
ODS
ODS
END
合成段
定义段

图5

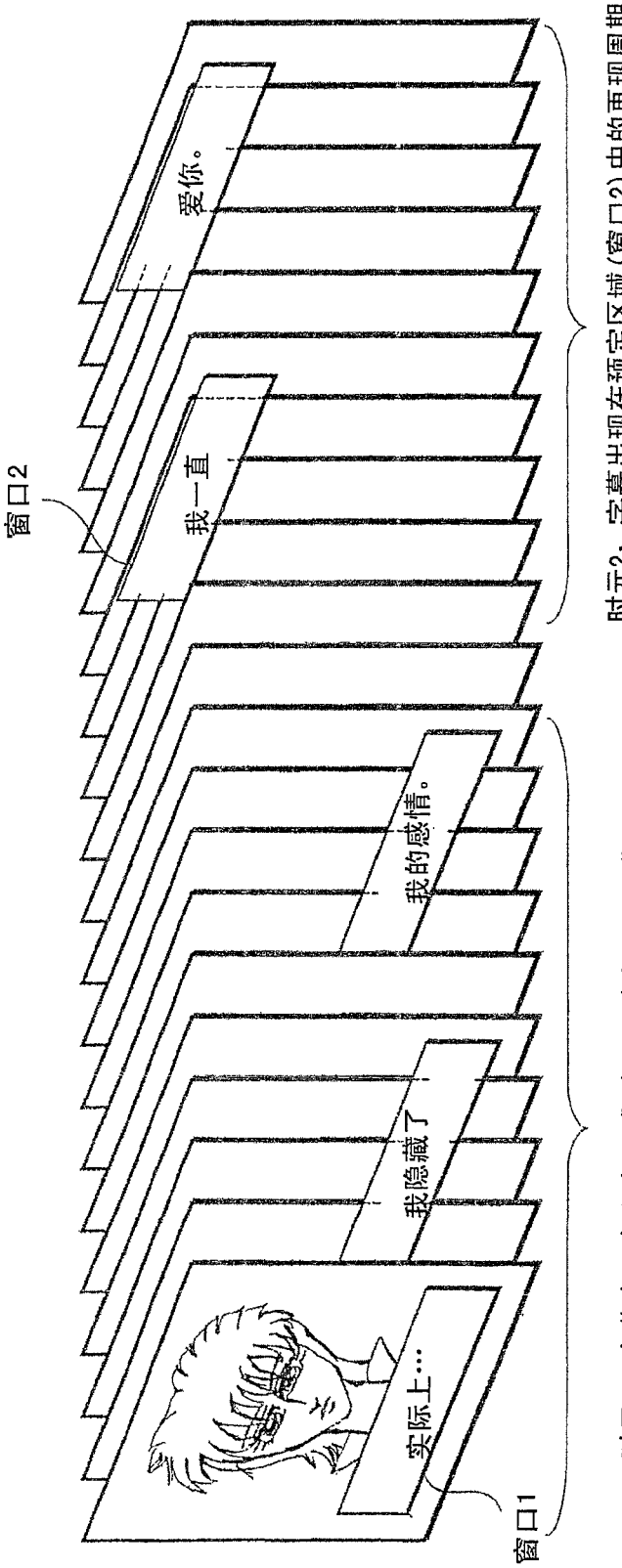

窗口2
爱你。
我一直
我的感情。
我隐藏了
实际上…
窗口1
时元2：字幕出现在预定区域(窗口2)中的再现周期
时元1：字幕出现在预定区域(窗口1)中的再现周期

图6

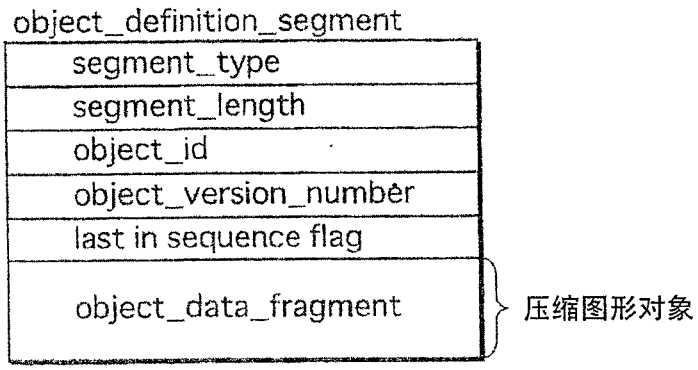

object_definition_segment
segment_type
segment_length
object_id
object_version_number
last in sequence flag
object_data_fragment
压缩图形对象

图7A

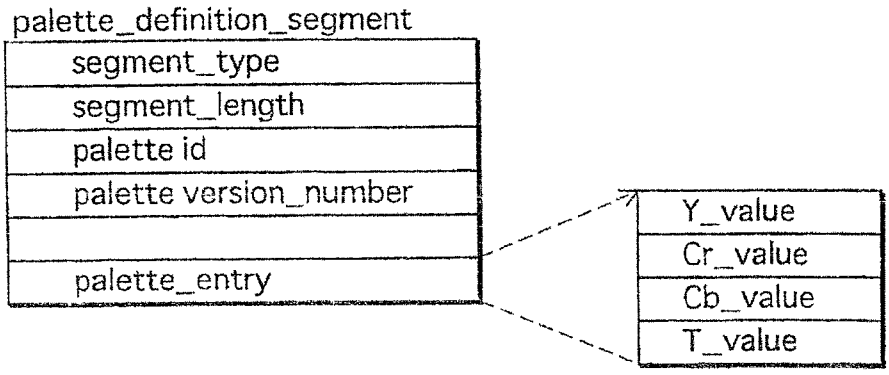

palette_definition_segment
segment_type
segment_length
palette id
palette version_number
palette_entry
Y_value
Cr_value
Cb_value
T_value

图7B

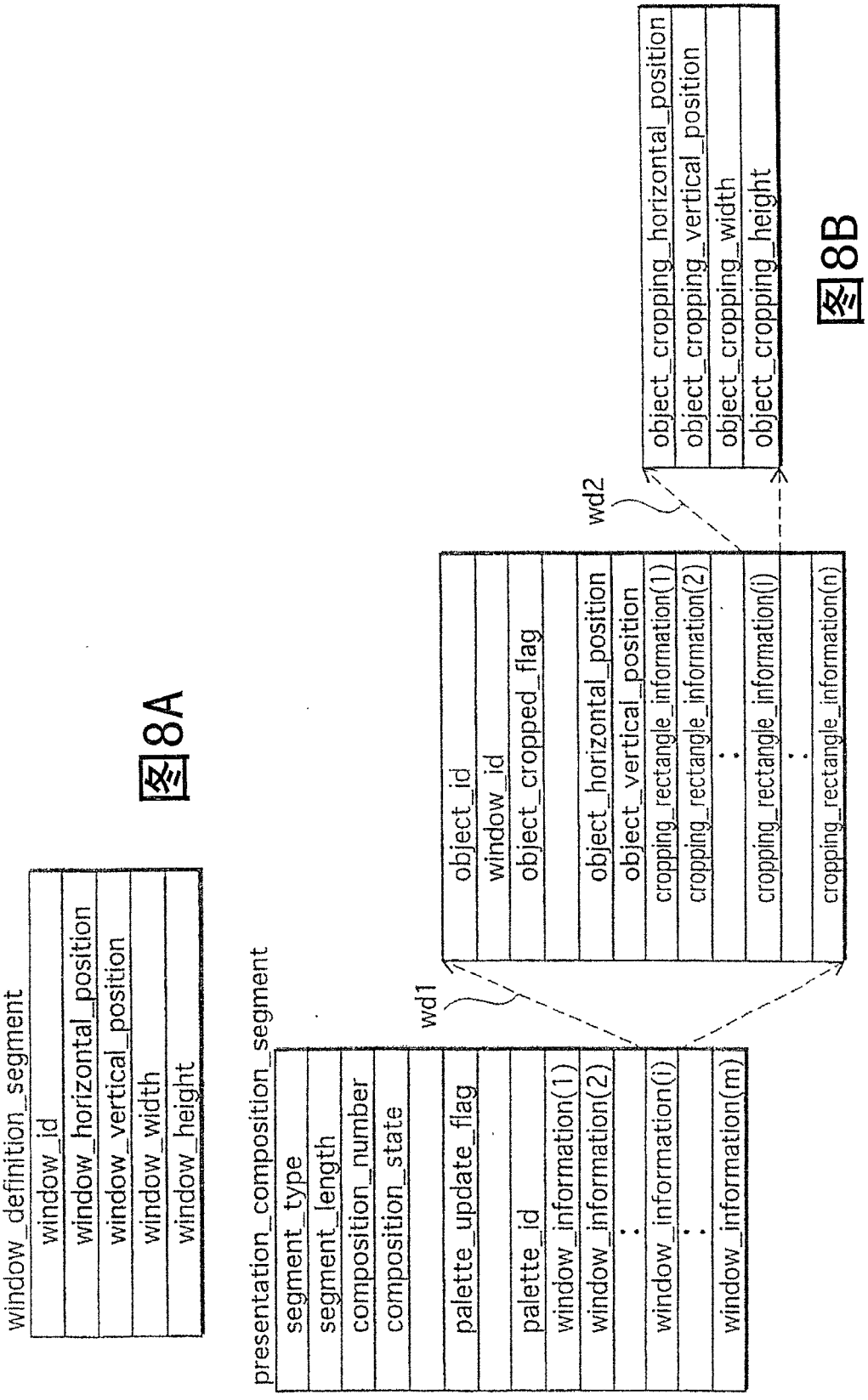

window_definition_segment
window_id
window_horizontal_position
window_vertical_position
window_width
window_height
presentation_composition_segment
segment_type
segment_length
composition_number
composition_state
palette_update_flag
palette_id
window_information(1)
window_information(2)
window_information(i)
window_information(m)
wd1
object_id
window_id
object_cropped_flag
object_horizontal_position
object_vertical_position
cropping_rectangle_information(1)
cropping_rectangle_information(2)
cropping_rectangle_information(i)
cropping_rectangle_information(n)
wd2
object_cropping_horizontal_position
object_cropping_vertical_position
object_cropping_width
object_cropping_height

图8A

图8B

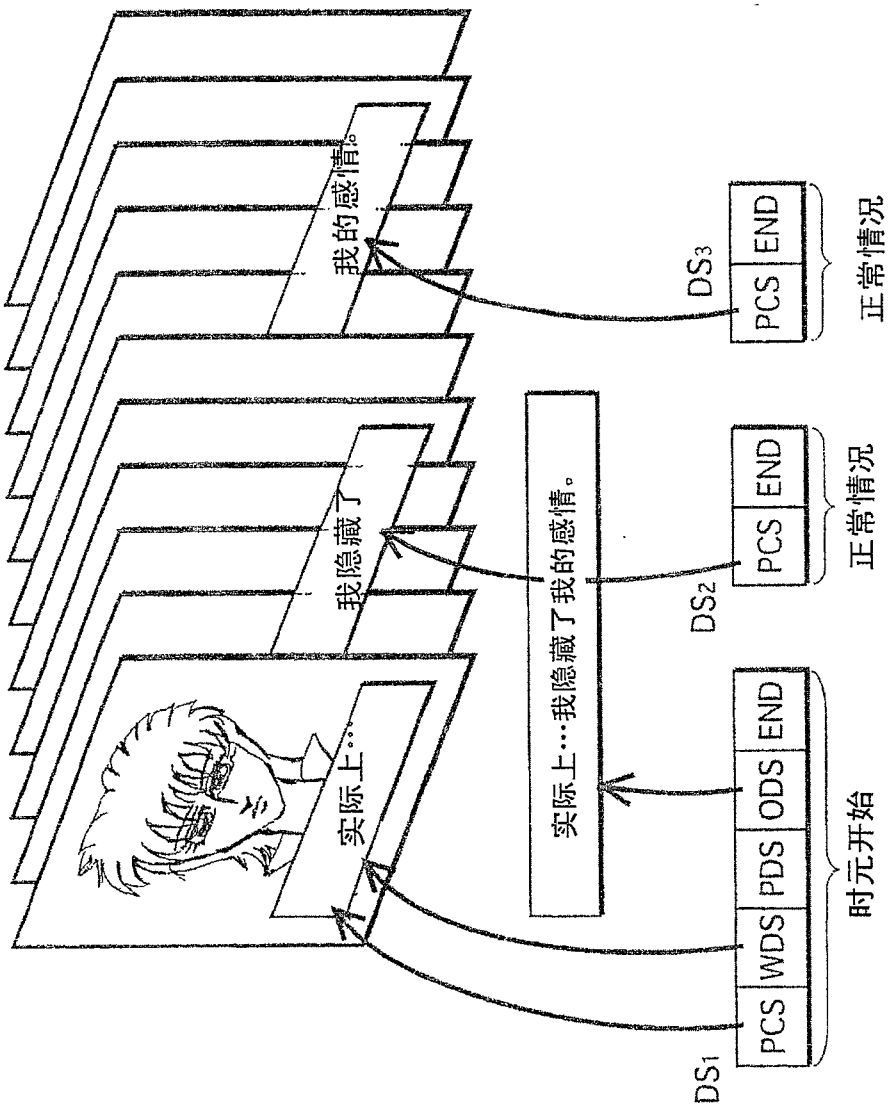

我的感情。
我隐藏了
实际上…
实际上…我隐藏了我的感情。
DS1
PCS
WDS
PDS
ODS
END
时元开始
DS2
PCS
END
正常情况
DS3
PCS
END
正常情况

图9

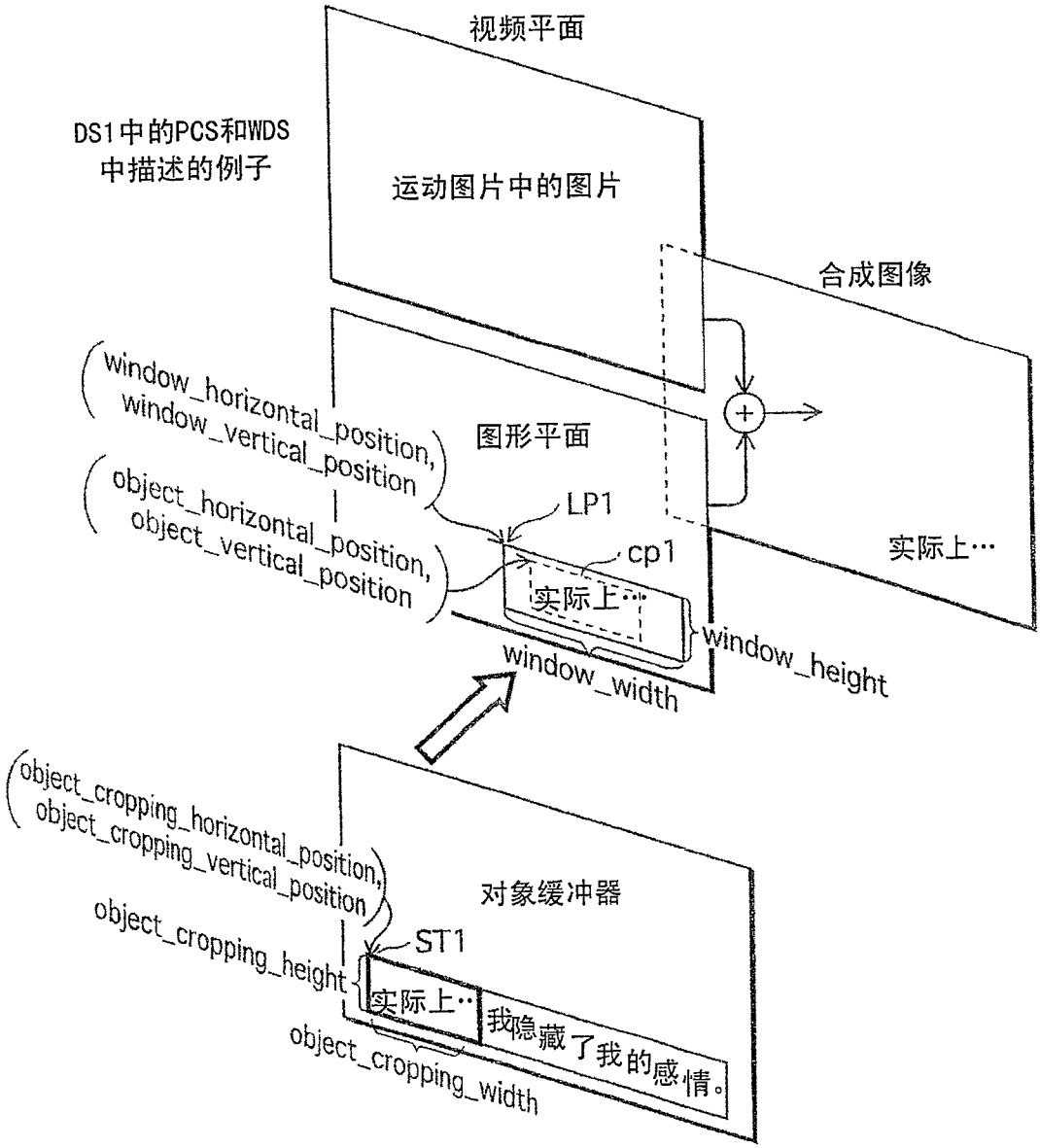

视频平面
DS1中的PCS和WDS
中描述的例子
运动图片中的图片
合成图像
window_horizontal_position,
window_vertical_position
图形平面
object_horizontal_position,
object_vertical_position
LP1
cp1
实际上…
实际上…
window_height
window_width
object_cropping_horizontal_position,
object_cropping_vertical_position
对象缓冲器
ST1
object_cropping_height
实际上…
我隐藏了我的感情。
object_cropping_width

图10

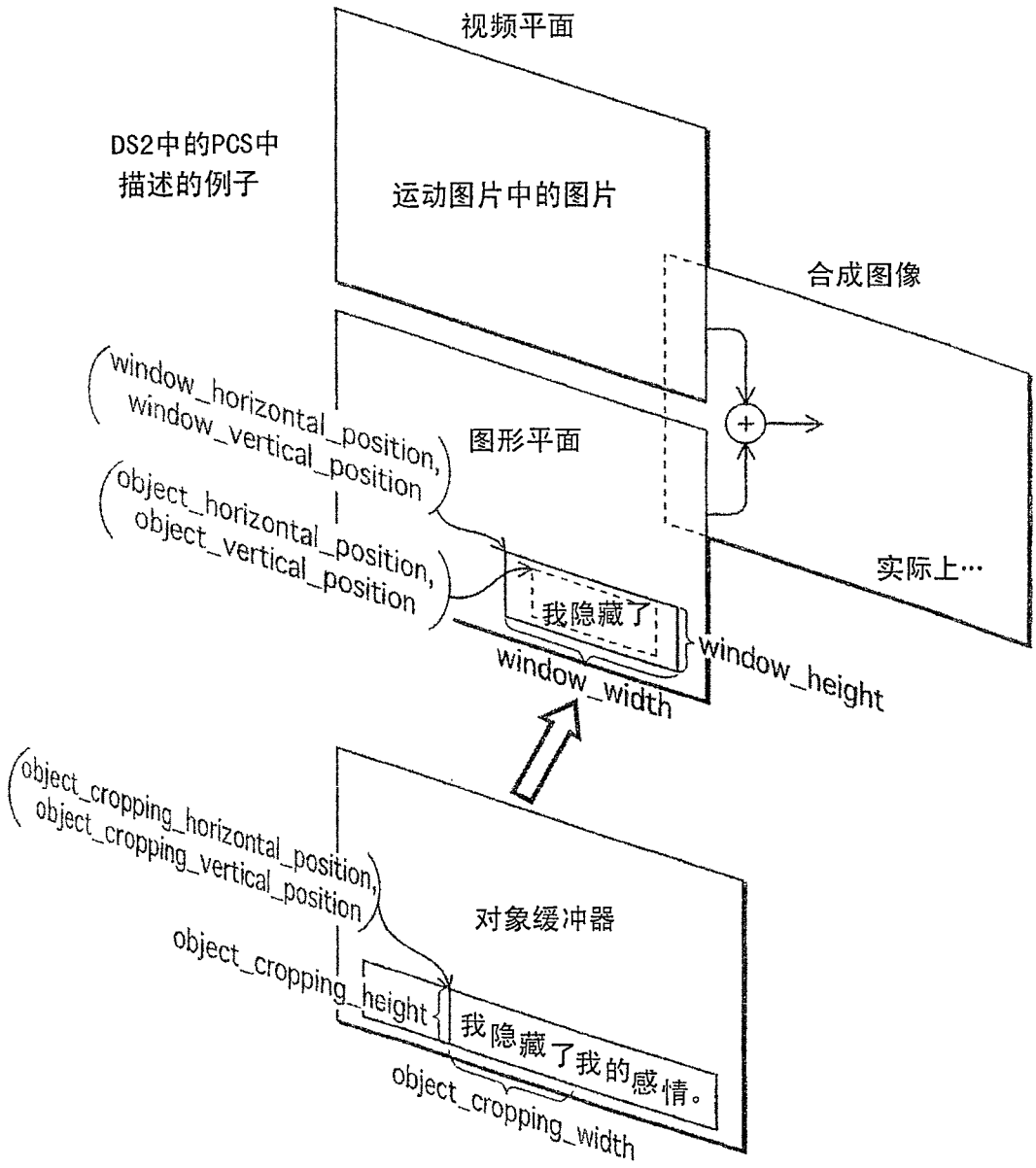

视频平面
DS2中的PCS中
描述的例子
运动图片中的图片
合成图像
window_horizontal_position,
window_vertical_position
图形平面
object_horizontal_position,
object_vertical_position
我隐藏了
window_width
window_height
实际上…
object_cropping_horizontal_position,
object_cropping_vertical_position
对象缓冲器
object_cropping_height
我隐藏了我的感情。
object_cropping_width

图11

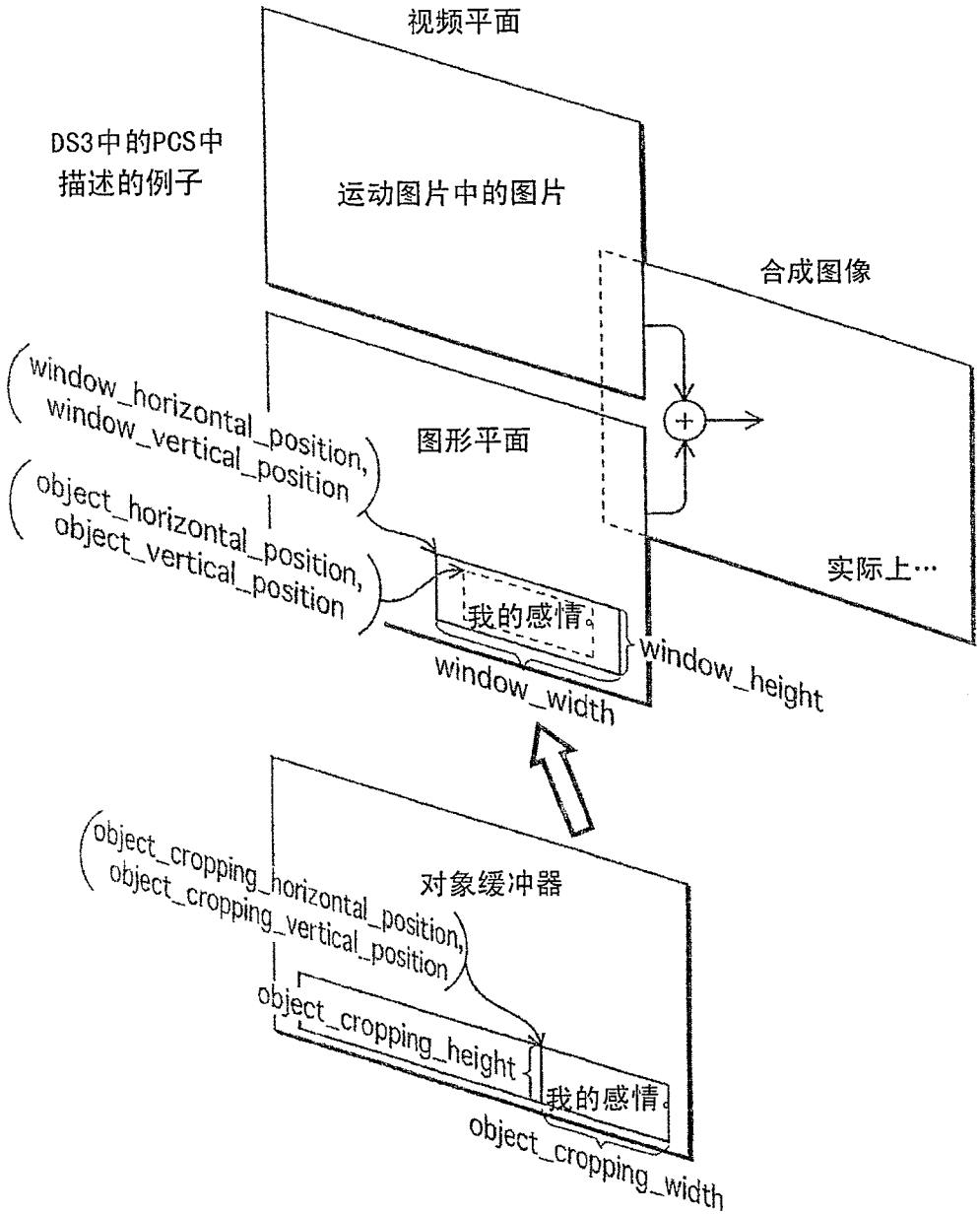

视频平面
DS3中的PCS中
描述的例子
运动图片中的图片
合成图像
window_horizontal_position,
window_vertical_position
图形平面
object_horizontal_position,
object_vertical_position
我的感情。
实际上…
window_height
window_width
对象缓冲器
object_cropping_horizontal_position,
object_cropping_vertical_position
object_cropping_height
我的感情。
object_cropping_width

图12

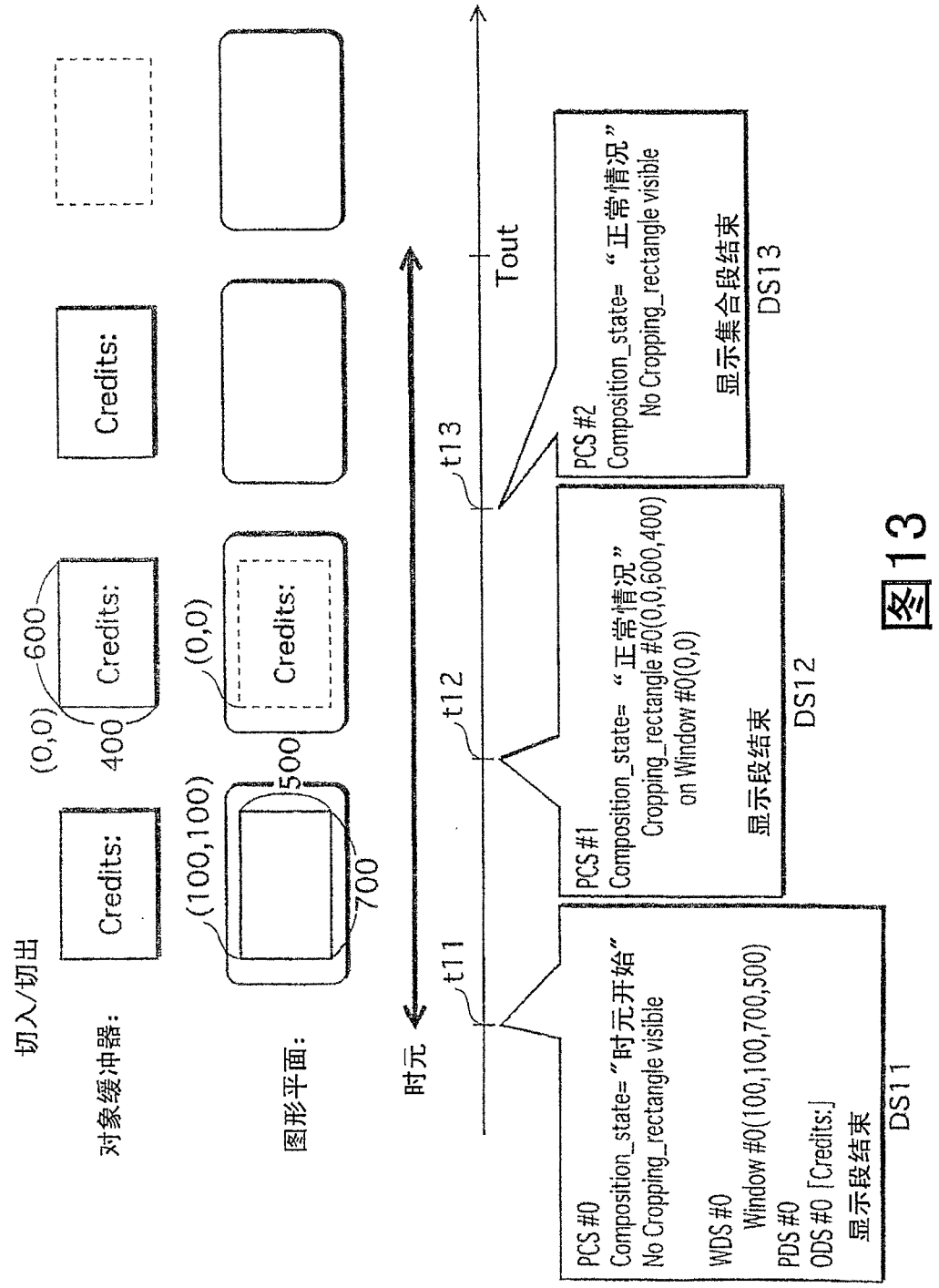

切入/切出
对象缓冲器:
Credits:
Credits:
Credits:
(0,0)
600
400
(100,100)
500
700
(0,0)
图形平面:
时元
t11
t12
t13
Tout
PCS #0
Composition_state= "时元开始"
No Cropping_rectangle visible
WDS #0
Window #0(100,100,700,500)
PDS #0
ODS #0 [Credits:]
显示段结束
DS11
PCS #1
Composition_state= "正常情况"
Cropping_rectangle #0(0,0,600,400)
on Window #0(0,0)
显示段结束
DS12
PCS #2
Composition_state= "正常情况"
No Cropping_rectangle visible
显示集合段结束
DS13

图13

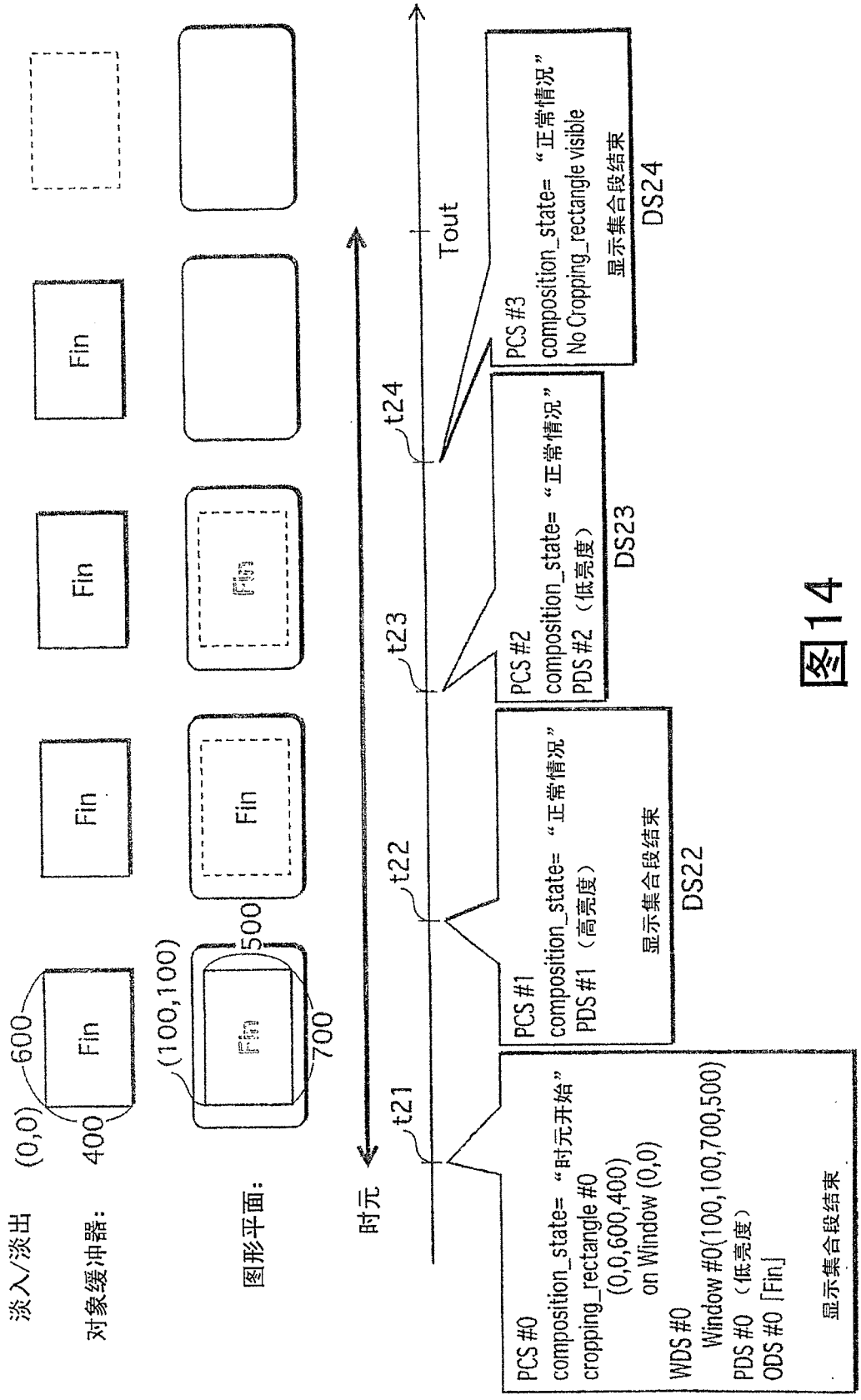

淡入/淡出
对象缓冲器:
图形平面:
(0,0)
600
400
(100,100)
500
700
Fin
时元
t21
t22
t23
t24
Tout
PCS #0
composition_state= “时元开始”
cropping_rectangle #0
(0,0,600,400)
on Window (0,0)
WDS #0
Window #0(100,100,700,500)
PDS #0 （低亮度）
ODS #0 「Fin」
显示集合段结束
DS21
PCS #1
composition_state= “正常情况”
PDS #1 （高亮度）
显示集合段结束
DS22
PCS #2
composition_state= “正常情况”
PDS #2 （低亮度）
DS23
PCS #3
composition_state= “正常情况”
No Cropping_rectangle visible
显示集合段结束
DS24

图14

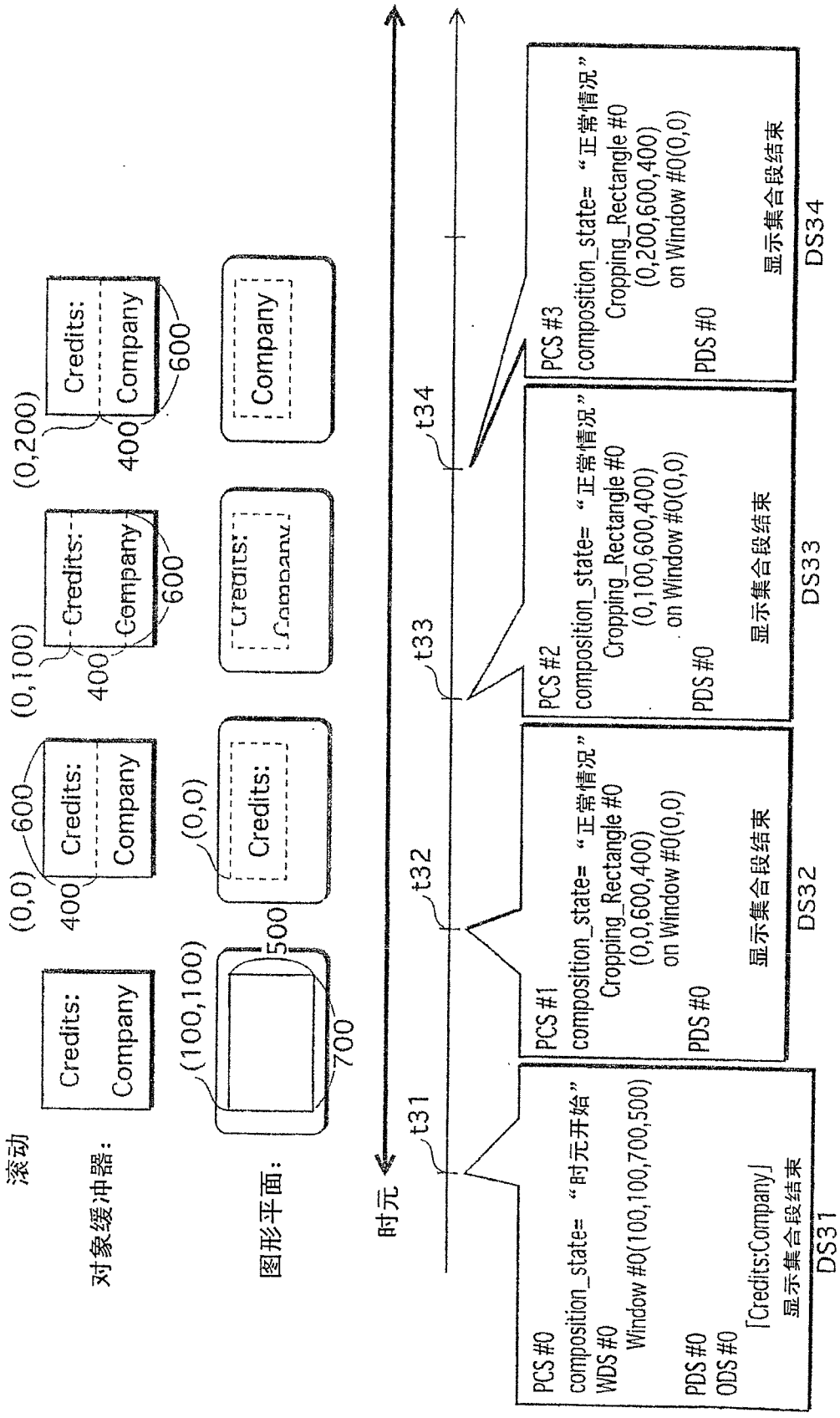

滚动
对象缓冲器:
图形平面:
时元
Credits:
Company
(0,0)
400
600
(100,100)
500
700
(0,100)
(0,200)
(0,0)
t31
t32
t33
t34
PCS #0
composition_state= "时元开始"
WDS #0
Window #0(100,100,700,500)
PDS #0
ODS #0
「Credits:Company」
显示集合段结束
DS31
PCS #1
composition_state= "正常情况"
Cropping_Rectangle #0
(0,0,600,400)
on Window #0(0,0)
PDS #0
显示集合段结束
DS32
PCS #2
composition_state= "正常情况"
Cropping_Rectangle #0
(0,100,600,400)
on Window #0(0,0)
PDS #0
显示集合段结束
DS33
PCS #3
composition_state= "正常情况"
Cropping_Rectangle #0
(0,200,600,400)
on Window #0(0,0)
PDS #0
显示集合段结束
DS34

图15

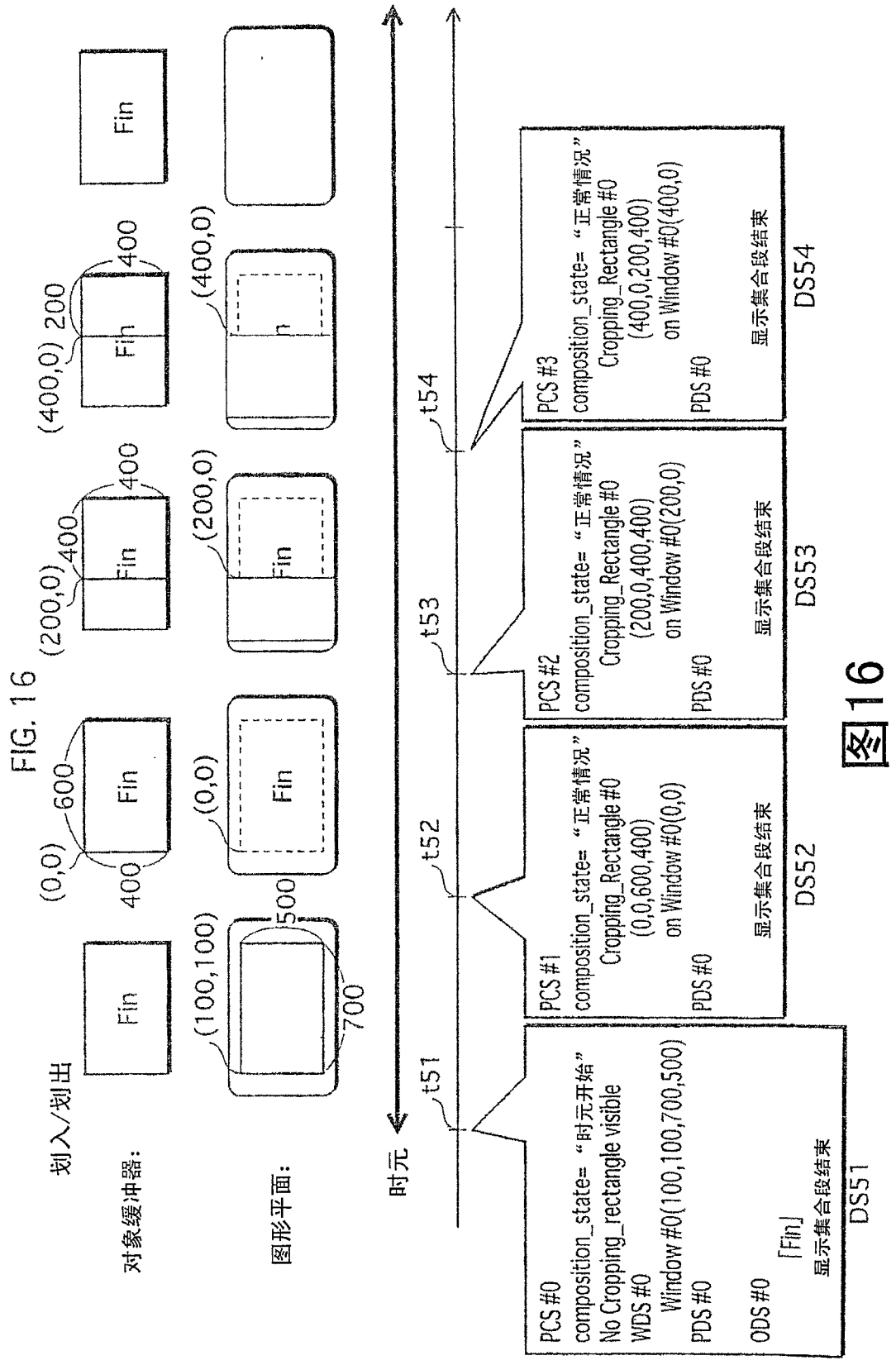

FIG. 16
划入/划出
对象缓冲器:
图形平面:
时元
t51
t52
t53
t54
(0,0)
600
400
Fin
(100,100)
500
700
(200,0)
400
(0,0)
(200,0)
(400,0)
(400,0) 200
Fin
n
PCS #0
composition_state= "时元开始"
No Cropping_rectangle visible
WDS #0
Window #0(100,100,700,500)
PDS #0
ODS #0
[Fin]
显示集合段结束
DS51
PCS #1
composition_state= "正常情况"
Cropping_Rectangle #0
(0,0,600,400)
on Window #0(0,0)
PDS #0
显示集合段结束
DS52
PCS #2
composition_state= "正常情况"
Cropping_Rectangle #0
(200,0,400,400)
on Window #0(200,0)
PDS #0
显示集合段结束
DS53
PCS #3
composition_state= "正常情况"
Cropping_Rectangle #0
(400,0,200,400)
on Window #0(400,0)
PDS #0
显示集合段结束
DS54

图16

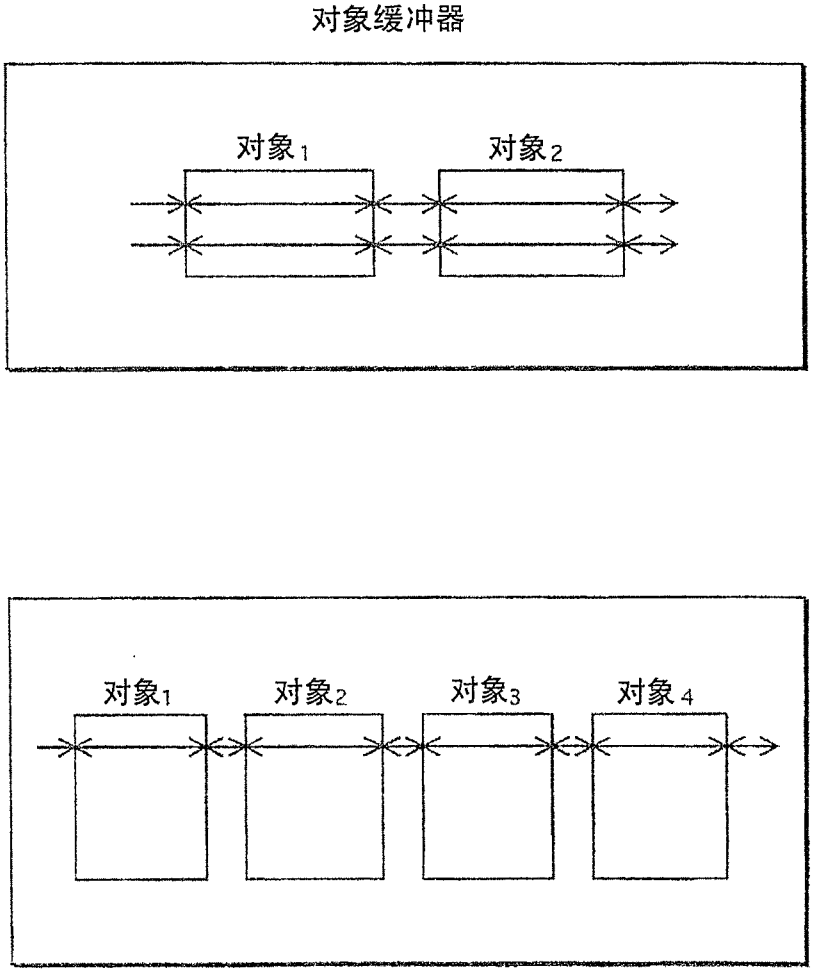


图17

```
        PTS( DSn[PCS)] )>=DTS( DSn[PCS] )+DECODEDURATION( DSn )
Where:
   • DECODEDURATION( DSn ) 被如下计算:
   decode_duration = 0 ;
   decode_duration += PLANEINITIALIZATIONTIME( DSn ) ;
   if( DSn. PCS. num_of_objects == 2 )
   {
      decode_duration += WAIT( DSn, DSn. PCS. OBJ[0], decode_duration ) ;
      if( DSn. PCS. OBJ[0]. window_id == DSn. PCS. OBJ[1]. window_id )
      {
            decode_duration += WAIT( DSn, DSn. PCS. OBJ[1], decode_duration ) ;
            decode_duration += 90000*( SIZE( DSn. PCS. OBJ[0]. window_id )//256*10^6 ) ;
      }
      else
      {
            decode_duration += 90000*( SIZE( DSn. PCS. OBJ[0]. window_id )//256*10^6 ) ;
            decode_duration += WAIT( DSn, DSn. PCS. OBJ[1], decode_duration );
            decode_duration += 90000*( SIZE( DSn. PCS. OBJ[1]. window_id )//256*10^6 ) ;
      }
   }
   else if( DSn. PCS. num_of_objects ==1 )
   {
      decode_duration += WAIT( DSn, DSn. PCS. OBJ[0], decode_duration ) ;
      decode_duration += 90000*( SIZE( DSn. PCS. OBJ[0]. window_id )//256*10^6 ) ;
   }
   return decode_duration ;

   • PLANEINITIALIZATIONTIME( DSn ) 被如下计算:
   initialize_duration=0 ;
   if( DSn. PCS. composition_state= = EPOCH_START )
   {
     initialize_duration = 90000*( 8*video_width*video_height//256*10^6) ;
   }

   else
   {
      for( i=0 ; i < WDS. num_windows ; i++ )
      {
            if( EMPTY(DSn.WDS.WIN[i],DSn ) )
                  initialize_duration += 90000*( SIZE( DSn. WDS. WIN[i] )//256*10^6) ;
      }
   }
   return initialize_duration ;

   • WAIT( DSn, OBJ, current_duration ) 被如下计算:
   wait_duration = 0 ;
   if( EXISTS( OBJ. object_id, DSn ) )
   {
      object_definition_ready_time = PTS( GET( OBJ. object_id. DSn ) ) ;
      current_time = DTS( DSn. PCS )+current_duration ;
      if( current_time < object_definition_ready_time )
            wait_duration += object_definition_ready_time - current_time ) ;
   }
   return wait_duration ;
```

图18

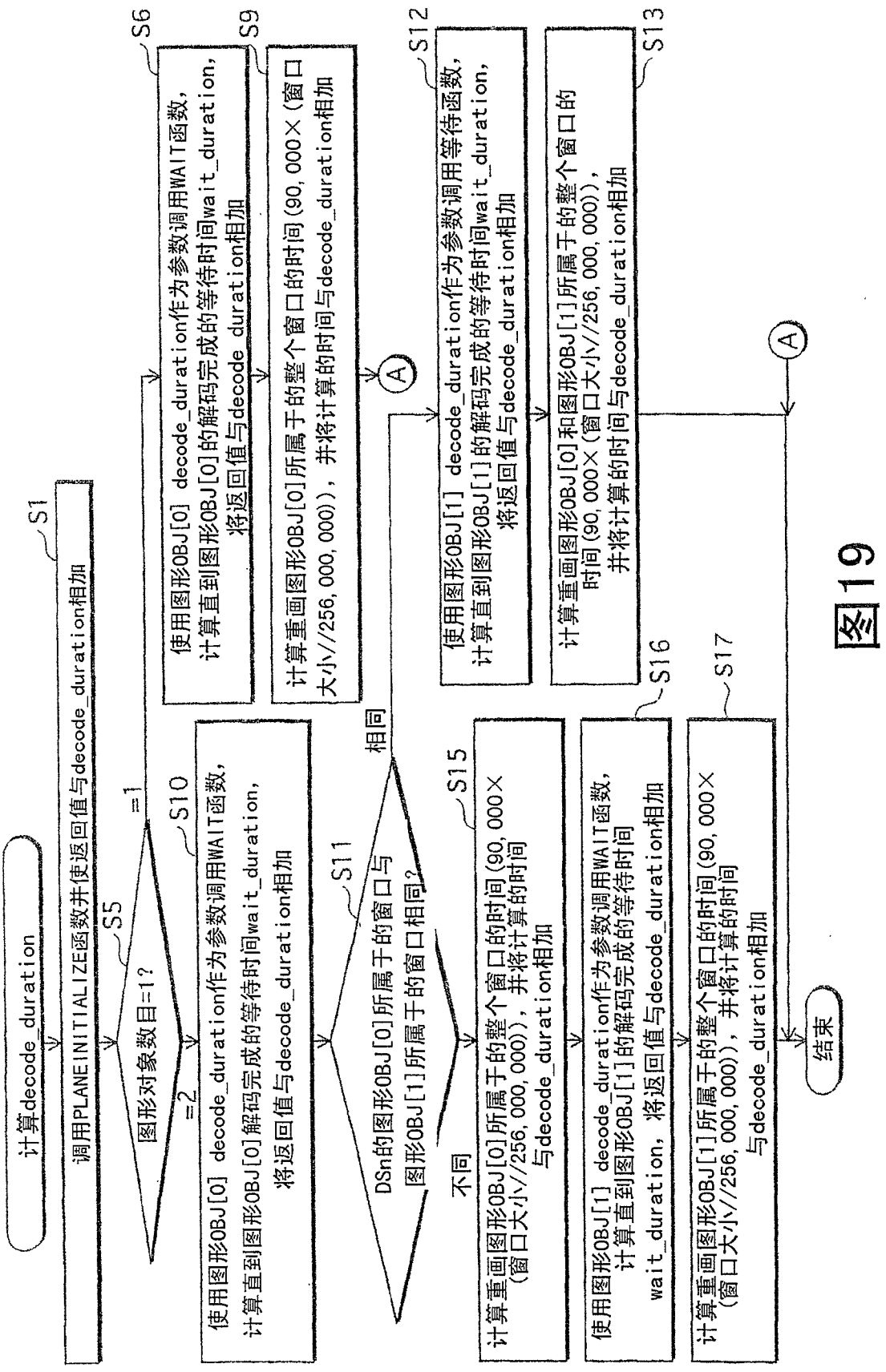

计算decode_duration
S1
调用PLANEINITIALIZE函数并使返回值与decode_duration相加
S5
图形对象数目=1?
=1
=2
S6
使用图形OBJ[0] decode_duration作为参数调用WAIT函数，
计算直到图形OBJ[0]的解码完成的等待时间wait_duration，
将返回值与decode_duration相加
S9
计算重画图形OBJ[0]所属于的整个窗口的时间(90,000×(窗口大小//256,000,000))，并将计算的时间与decode_duration相加
S10
使用图形OBJ[0] decode_duration作为参数调用WAIT函数，
计算直到图形OBJ[0]解码完成的等待时间wait_duration，
将返回值与decode_duration相加
S11
DSn的图形OBJ[0]所属于的窗口与图形OBJ[1]所属于的窗口相同?
相同
不同
A
S12
使用图形OBJ[1] decode_duration作为参数调用等待函数，
计算直到图形OBJ[1]的解码完成的等待时间wait_duration，
将返回值与decode_duration相加
S13
计算重画图形OBJ[0]和图形OBJ[1]所属于的整个窗口的时间(90,000×(窗口大小//256,000,000))，
并将计算的时间与decode_duration相加
S15
计算重画图形OBJ[0]所属于的整个窗口的时间(90,000×(窗口大小//256,000,000))，并将计算的时间与decode_duration相加
S16
使用图形OBJ[1] decode_duration作为参数调用WAIT函数，
计算直到图形OBJ[1]的解码完成的等待时间wait_duration，将返回值与decode_duration相加
S17
计算重画图形OBJ[1]所属于的整个窗口的时间(90,000×(窗口大小//256,000,000))，并将计算的时间与decode_duration相加
A
结束

图19

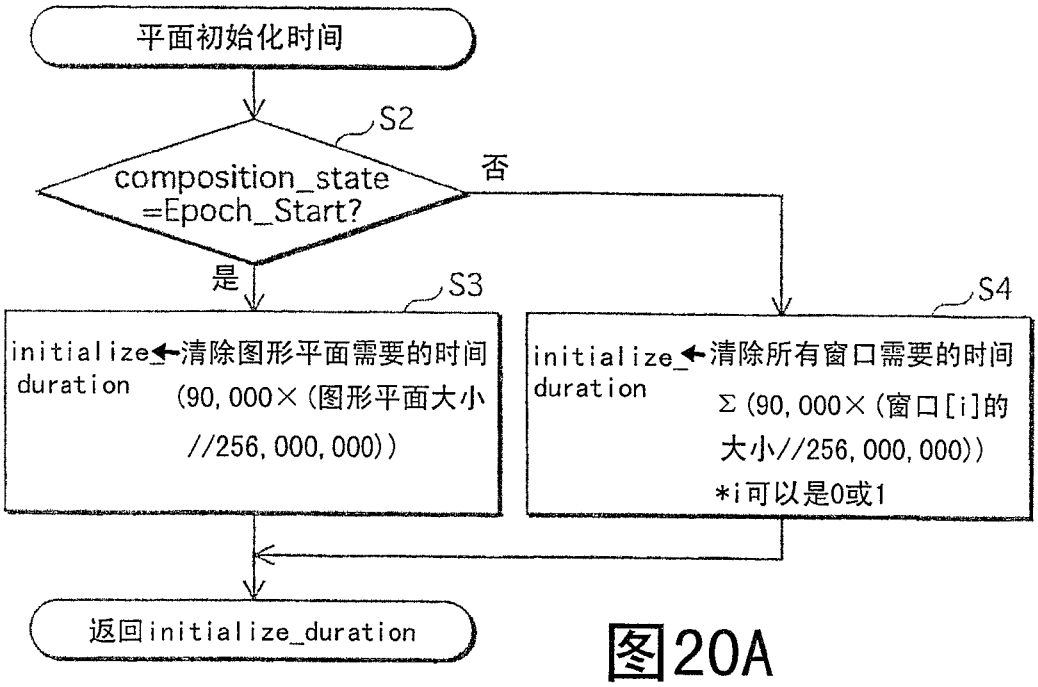

平面初始化时间
S2
composition_state
=Epoch_Start?
否
是
S3
initialize_duration ← 清除图形平面需要的时间
(90,000×(图形平面大小
//256,000,000))
S4
initialize_duration ← 清除所有窗口需要的时间
Σ(90,000×(窗口[i]的
大小//256,000,000))
*i可以是0或1
返回initialize_duration

图20A

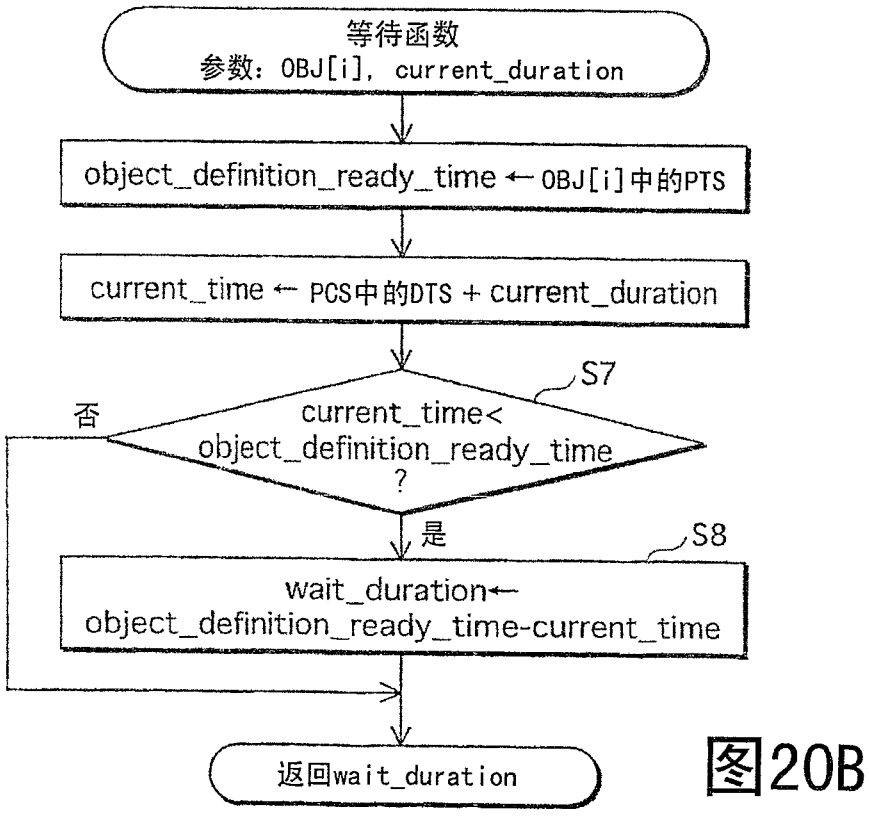

等待函数
参数：OBJ[i], current_duration
object_definition_ready_time ← OBJ[i]中的PTS
current_time ← PCS中的DTS + current_duration
S7
current_time<
object_definition_ready_time
?
否
是
S8
wait_duration ←
object_definition_ready_time-current_time
返回wait_duration

图20B

20

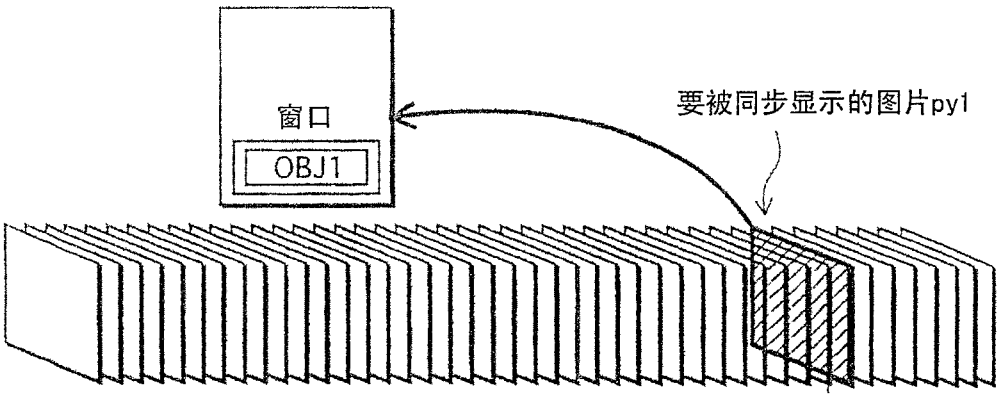


图21A

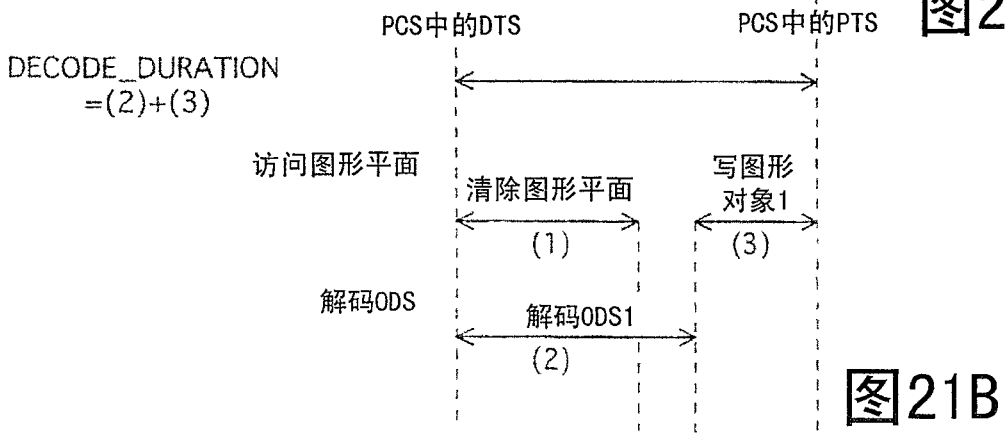


图21B

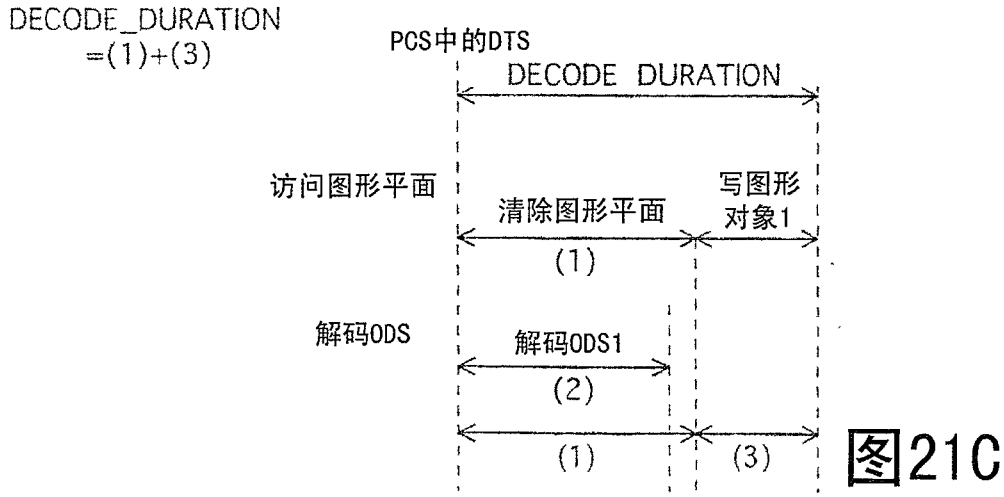


图21C

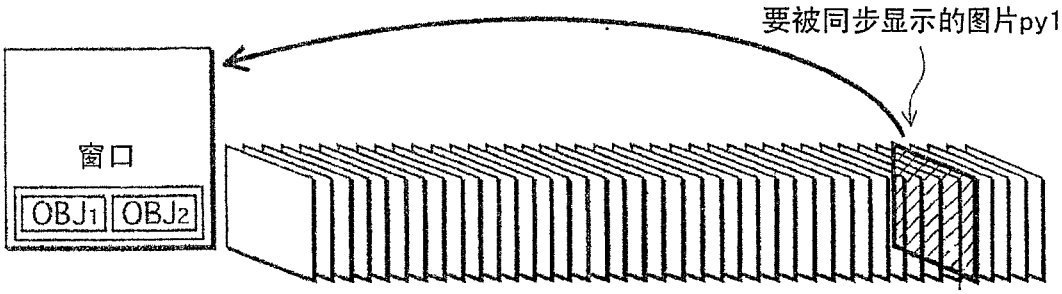


图22A

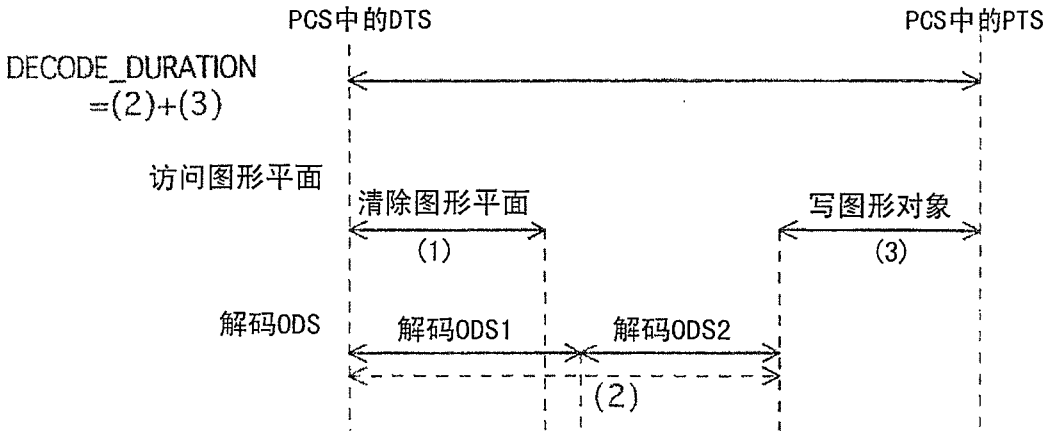


图22B

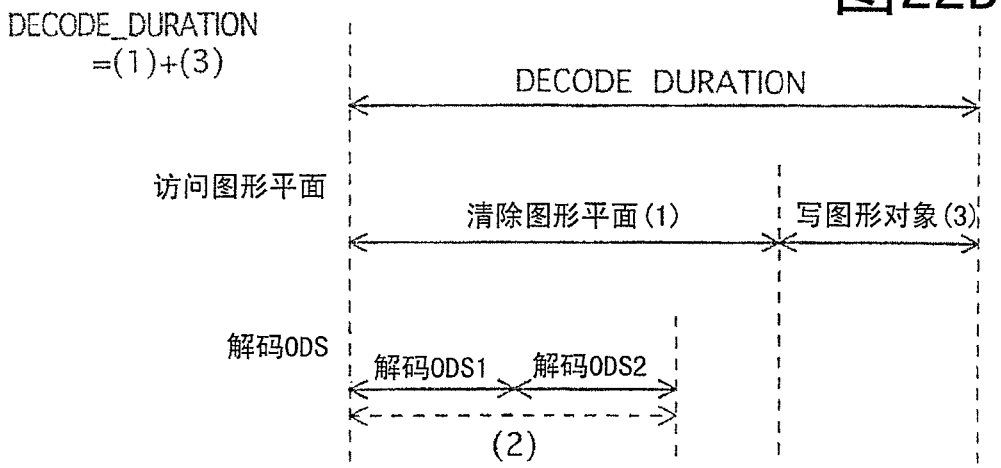


图22C

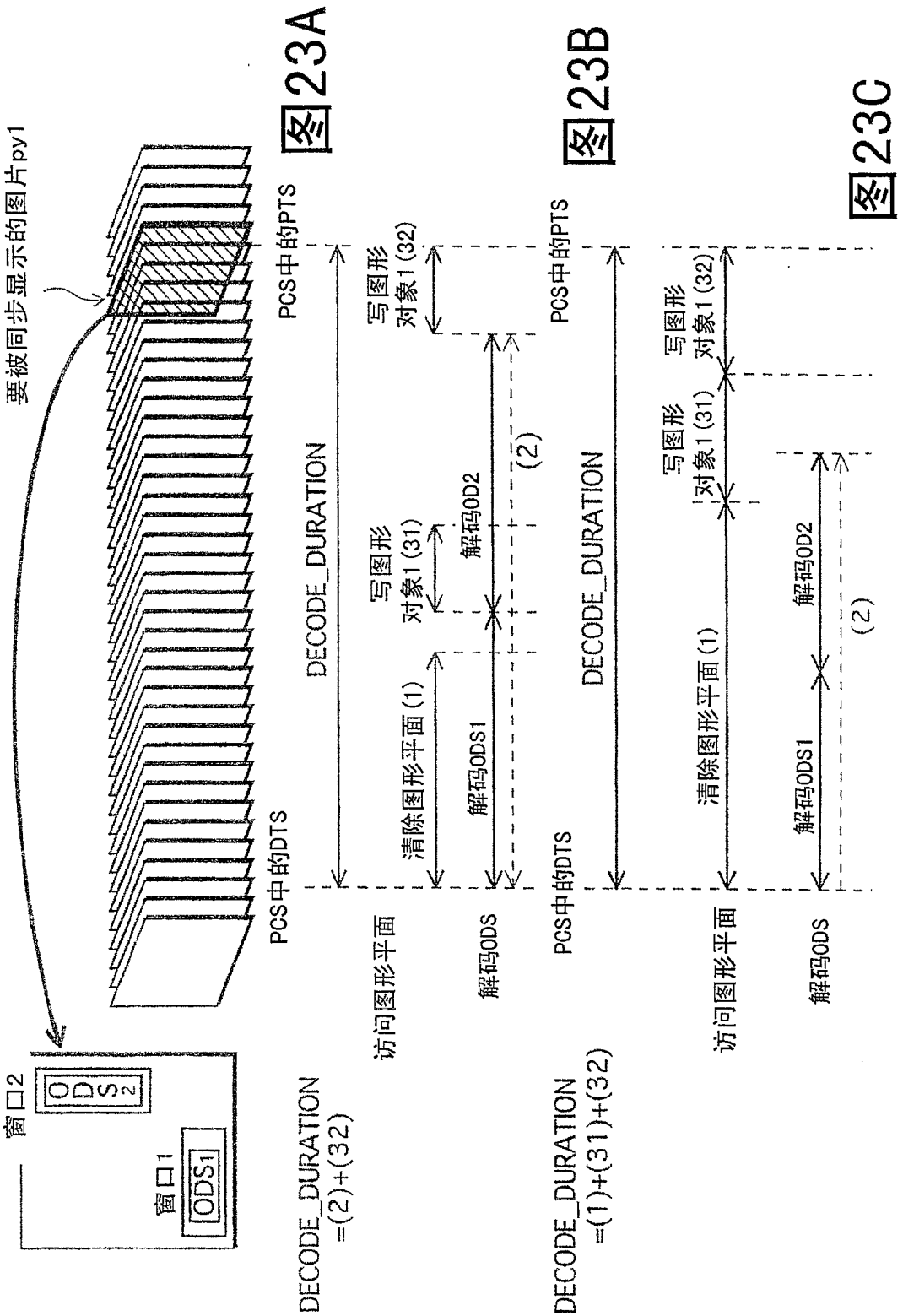


图23A

图23B

图23C

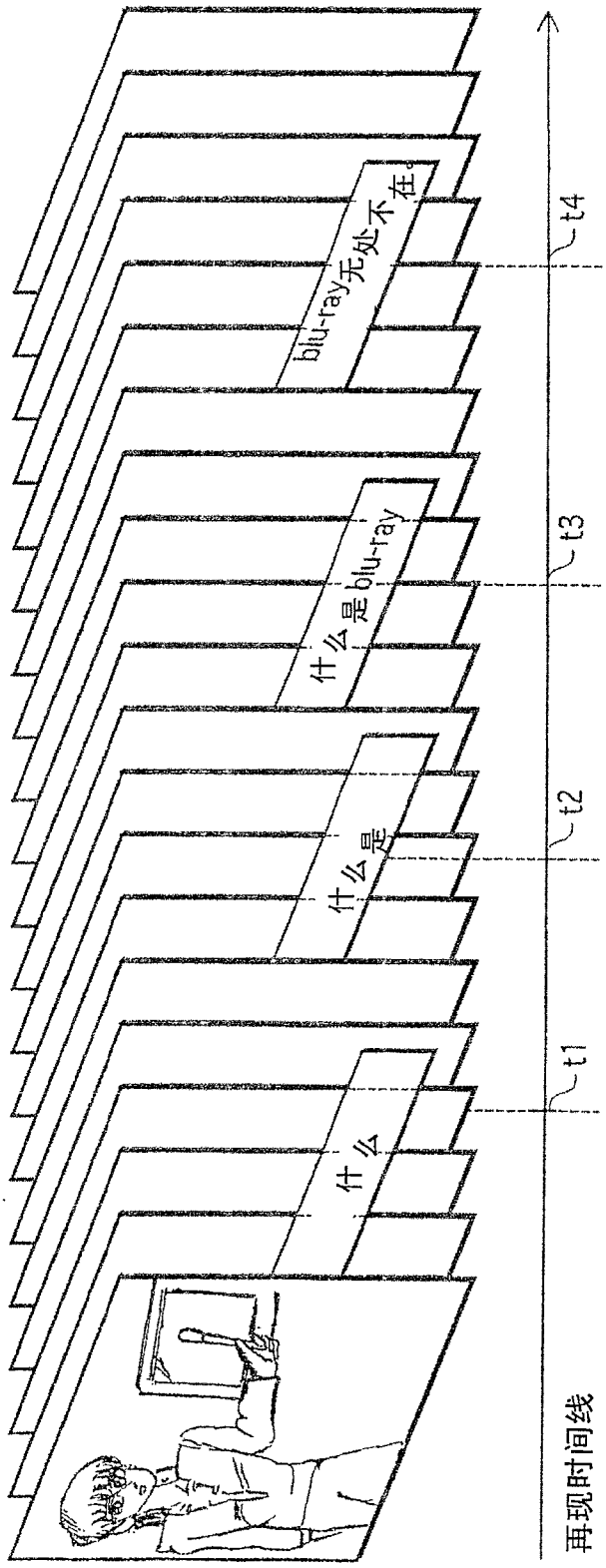


图24

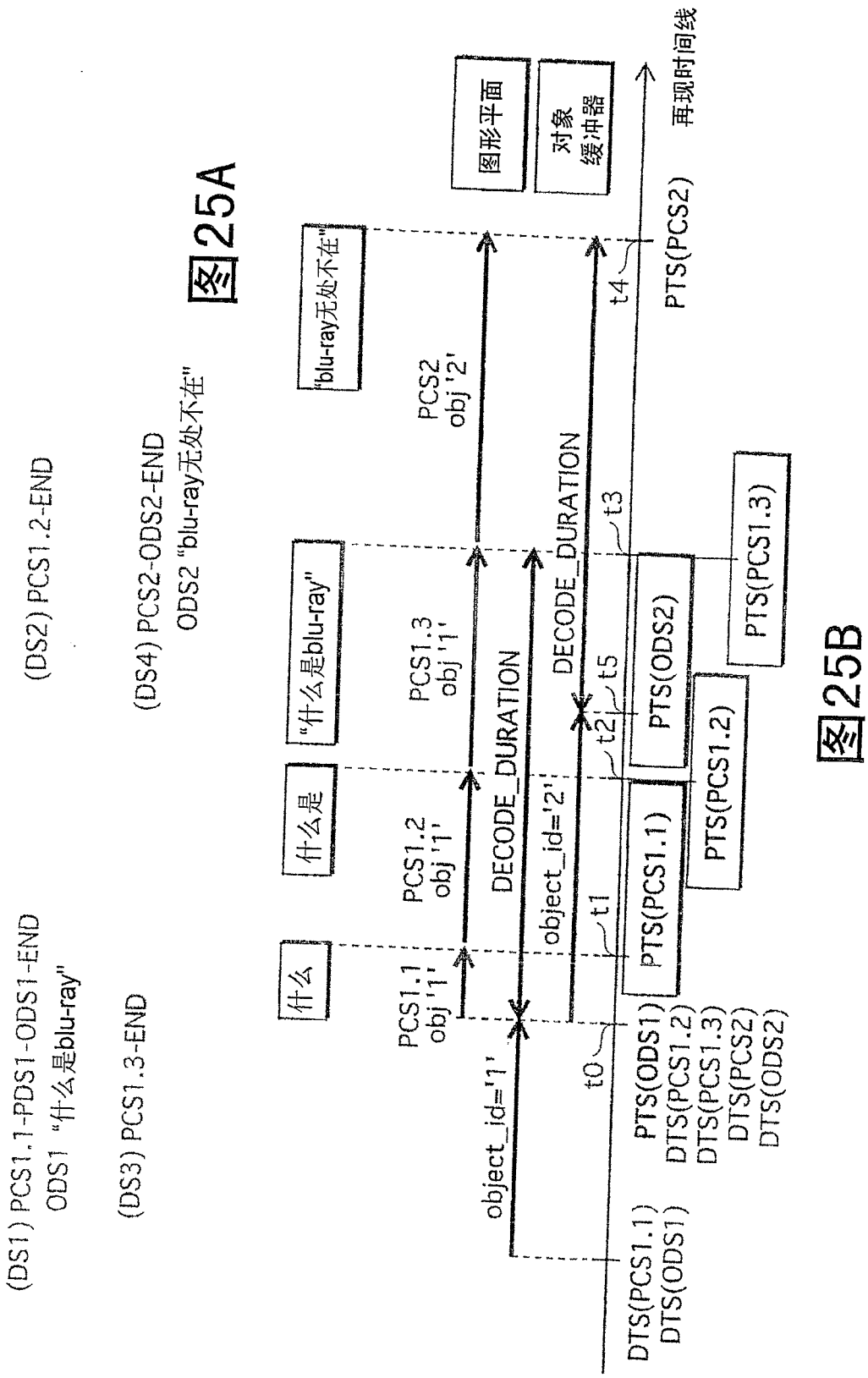

(DS1) PCS1.1-PDS1-ODS1-END
ODS1 "什么是blu-ray"
(DS3) PCS1.3-END
(DS2) PCS1.2-END
(DS4) PCS2-ODS2-END
ODS2 "blu-ray无处不在"
什么
什么是
"什么是blu-ray"
"blu-ray无处不在"
PCS1.1 obj '1'
PCS1.2 obj '1'
PCS1.3 obj '1'
PCS2 obj '2'
object_id='1'
DECODE_DURATION
object_id='2'
DECODE_DURATION
t0
t1
t2
t2'
t3
t4
t5
PTS(ODS1)
PTS(PCS1.1)
PTS(ODS2)
PTS(PCS1.2)
PTS(PCS1.3)
PTS(PCS2)
DTS(PCS1.1)
DTS(ODS1)
DTS(PCS1.2)
DTS(PCS1.3)
DTS(PCS2)
DTS(ODS2)
图形平面
对象缓冲器
再现时间线

图25A

图25B

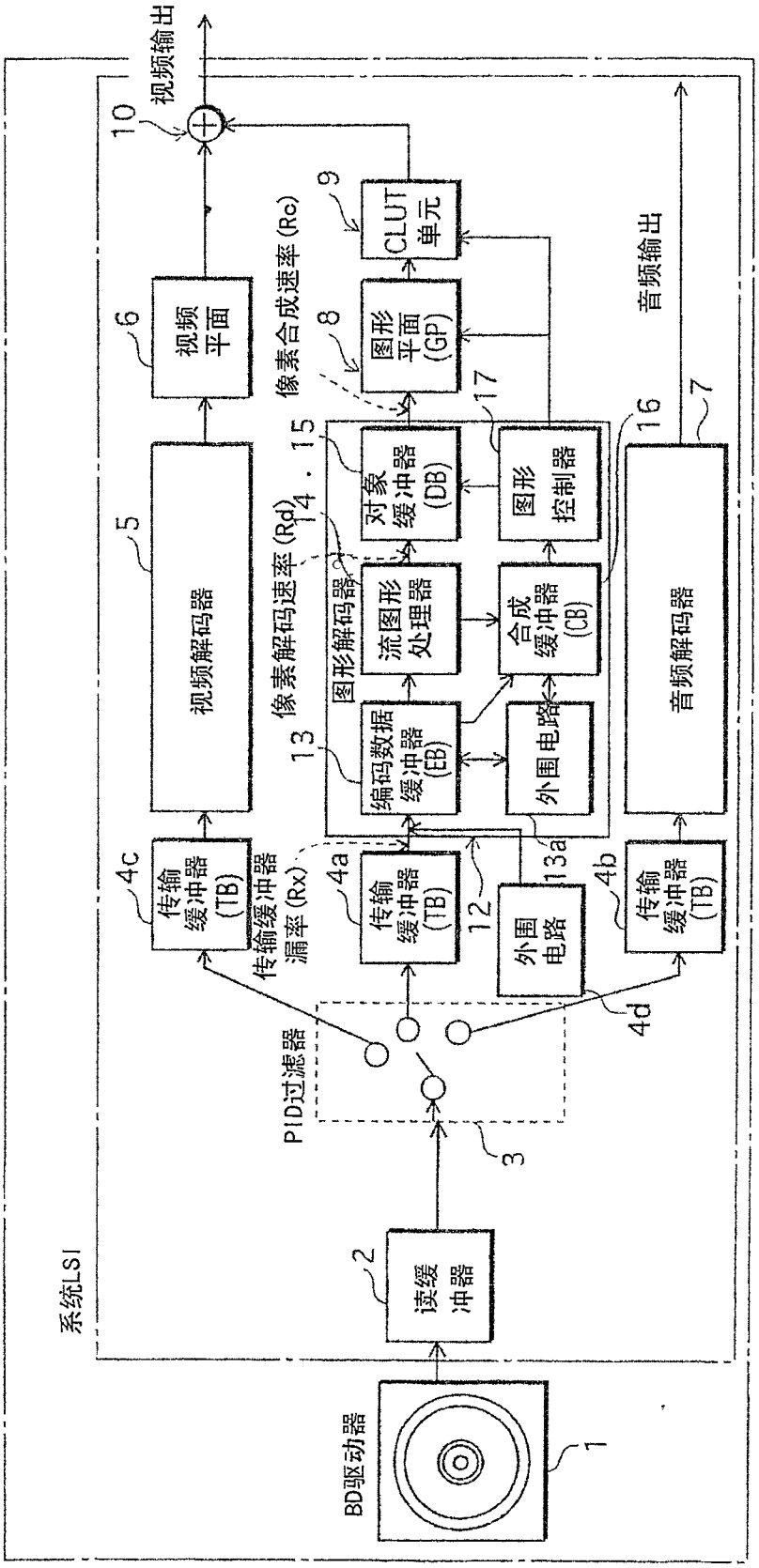

系统LSI
BD驱动器
1
读缓冲器
2
PID过滤器
3
传输缓冲器 (TB)
4a
传输缓冲器 (TB)
4b
传输缓冲器 (TB)
4c
外围电路
4d
外围电路
12
传输缓冲器漏率 (Rx)
13
编码数据缓冲器 (EB)
13a
图形解码器
14
像素解码速率 (Rd)
流图形处理器
合成缓冲器 (CB)
对象缓冲器 (DB)
15
图形控制器
16
音频解码器
7
音频输出
视频解码器
5
视频平面
6
像素合成速率 (Rc)
图形平面 (GP)
8
CLUT 单元
9
10
视频输出
17

图26

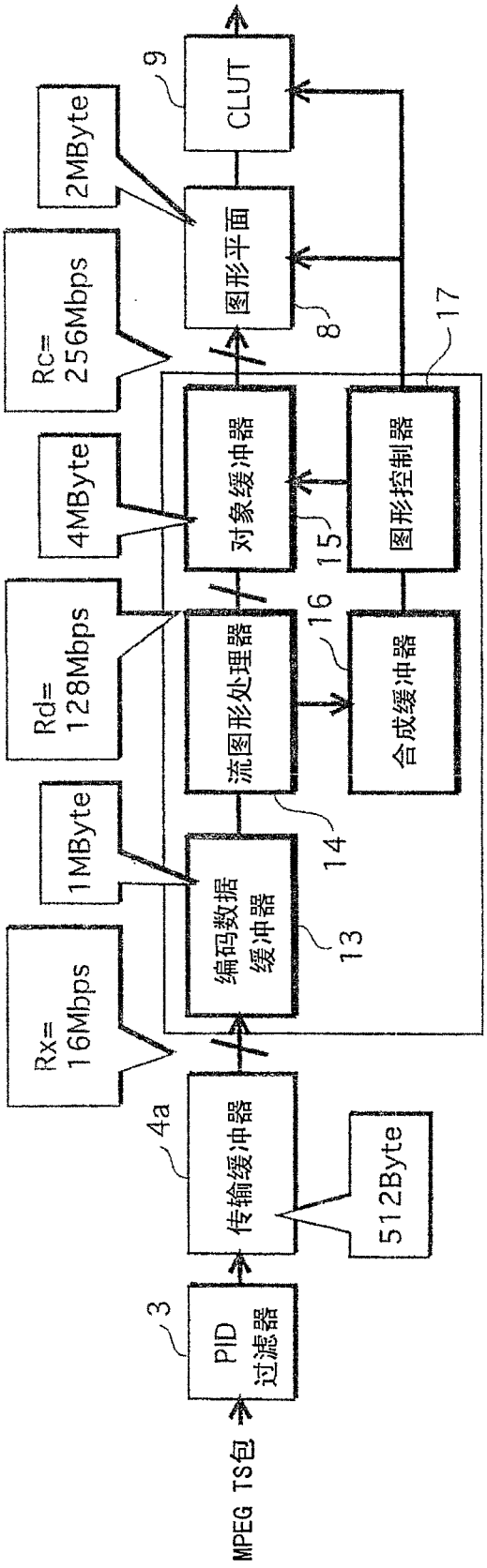

MPEG TS包
PID 过滤器
3
传输缓冲器
4a
512Byte
Rx= 16Mbps
编码数据 缓冲器
13
1MByte
流图形处理器
14
Rd= 128Mbps
合成缓冲器
16
对象缓冲器
15
4MByte
图形控制器
17
Rc= 256Mbps
图形平面
8
2MByte
CLUT
9

图27

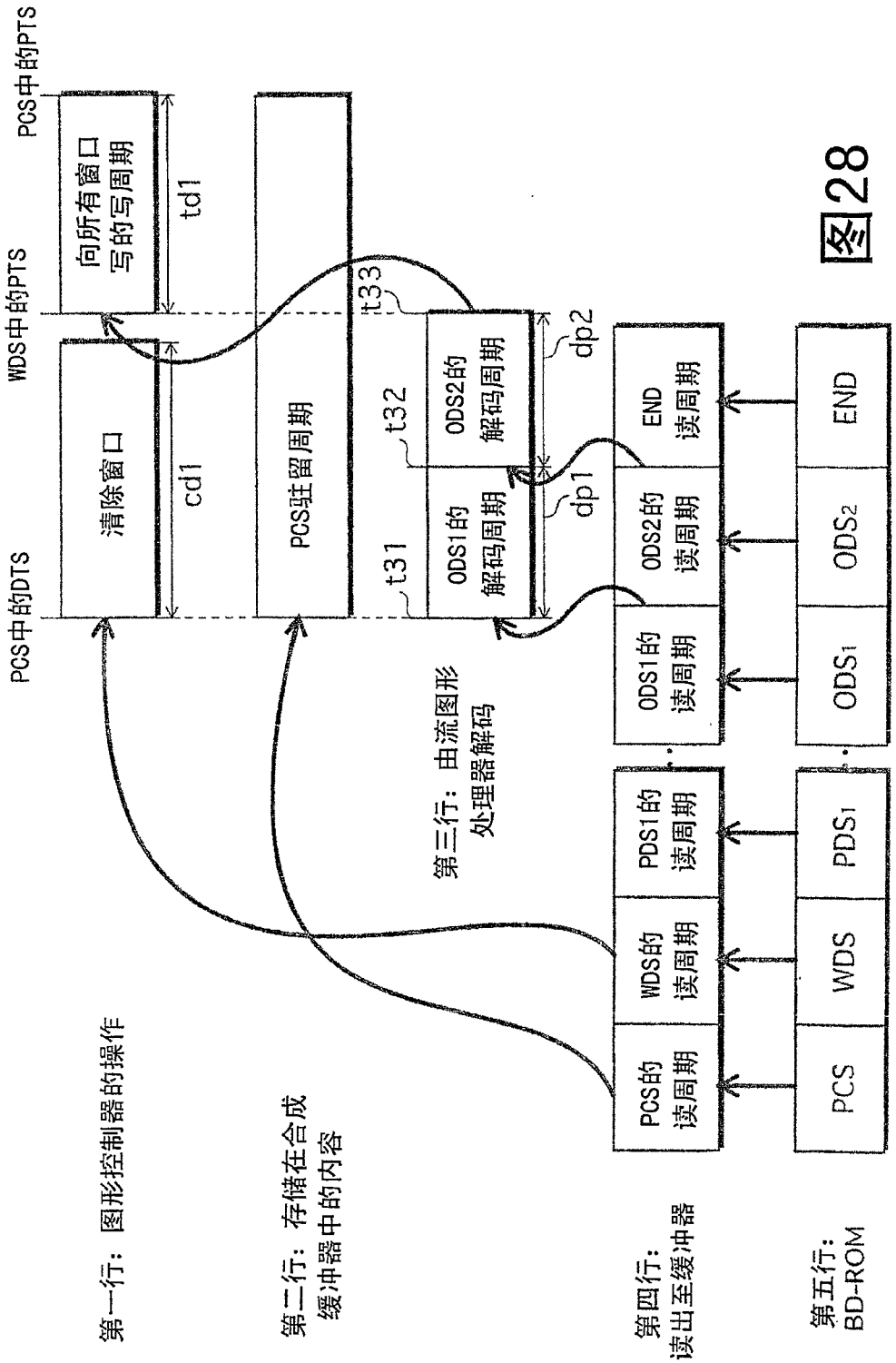

PCS中的DTS
WDS中的PTS
PCS中的PTS
清除窗口
向所有窗口
写的写周期
cd1
td1
PCS驻留周期
t31
t32
t33
ODS1的
解码周期
ODS2的
解码周期
dp1
dp2
第一行：图形控制器的操作
第二行：存储在合成
缓冲器中的内容
第三行：由流图形
处理器解码
第四行：
读出至缓冲器
第五行：
BD-ROM
PCS的
读周期
WDS的
读周期
PDS1的
读周期
ODS1的
读周期
ODS2的
读周期
END
读周期
PCS
WDS
PDS1
ODS1
ODS2
END

图28

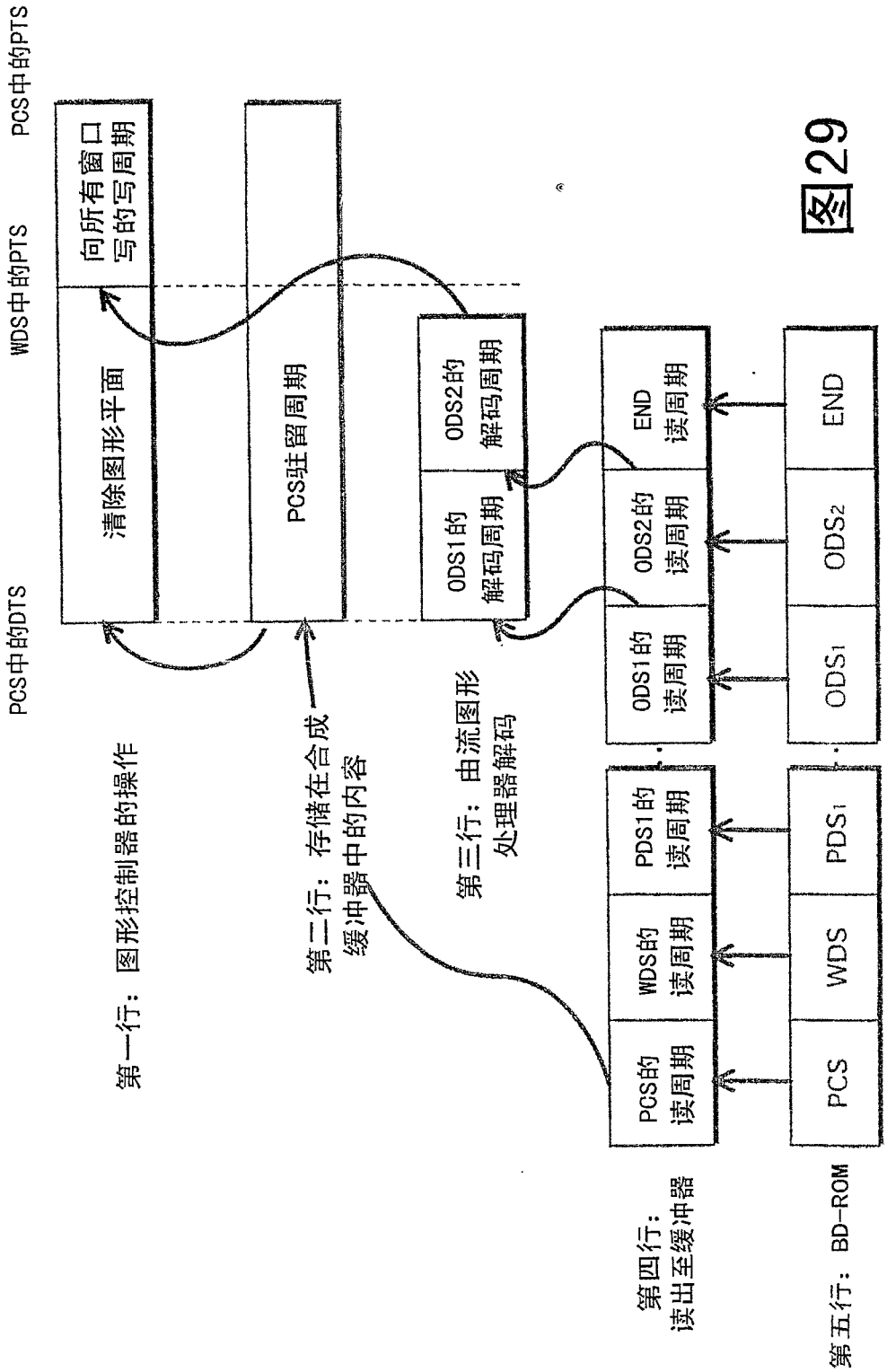

PCS中的DTS
WDS中的PTS
PCS中的PTS
第一行：图形控制器的操作
清除图形平面
向所有窗口写的写周期
PCS驻留周期
第二行：存储在合成缓冲器中的内容
ODS1的解码周期
ODS2的解码周期
第三行：由流图形处理器解码
PCS的读周期
WDS的读周期
PDS1的读周期
ODS1的读周期
ODS2的读周期
END读周期
第四行：读出至缓冲器
PCS
WDS
PDS1
ODS1
ODS2
END
第五行：BD-ROM

图29

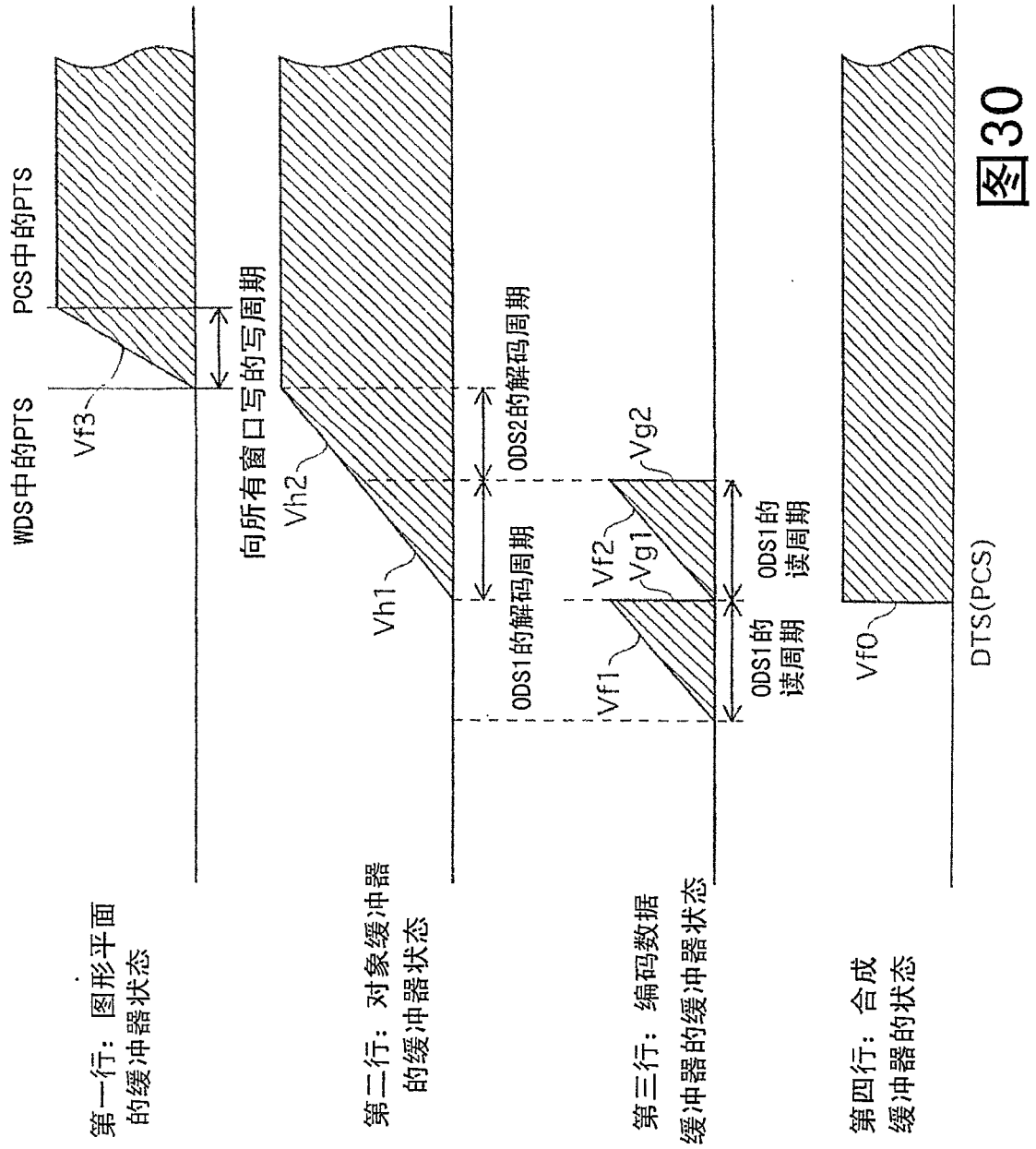

WDS中的PTS
PCS中的PTS
Vf3
向所有窗口写的写周期
第一行：图形平面
的缓冲器状态
Vh2
Vh1
ODS1的解码周期
ODS2的解码周期
第二行：对象缓冲器
的缓冲器状态
Vf2
Vg2
Vf1
Vg1
ODS1的
读周期
ODS1的
读周期
第三行：编码数据
缓冲器的缓冲器状态
VfO
DTS(PCS)
第四行：合成
缓冲器的状态

图30

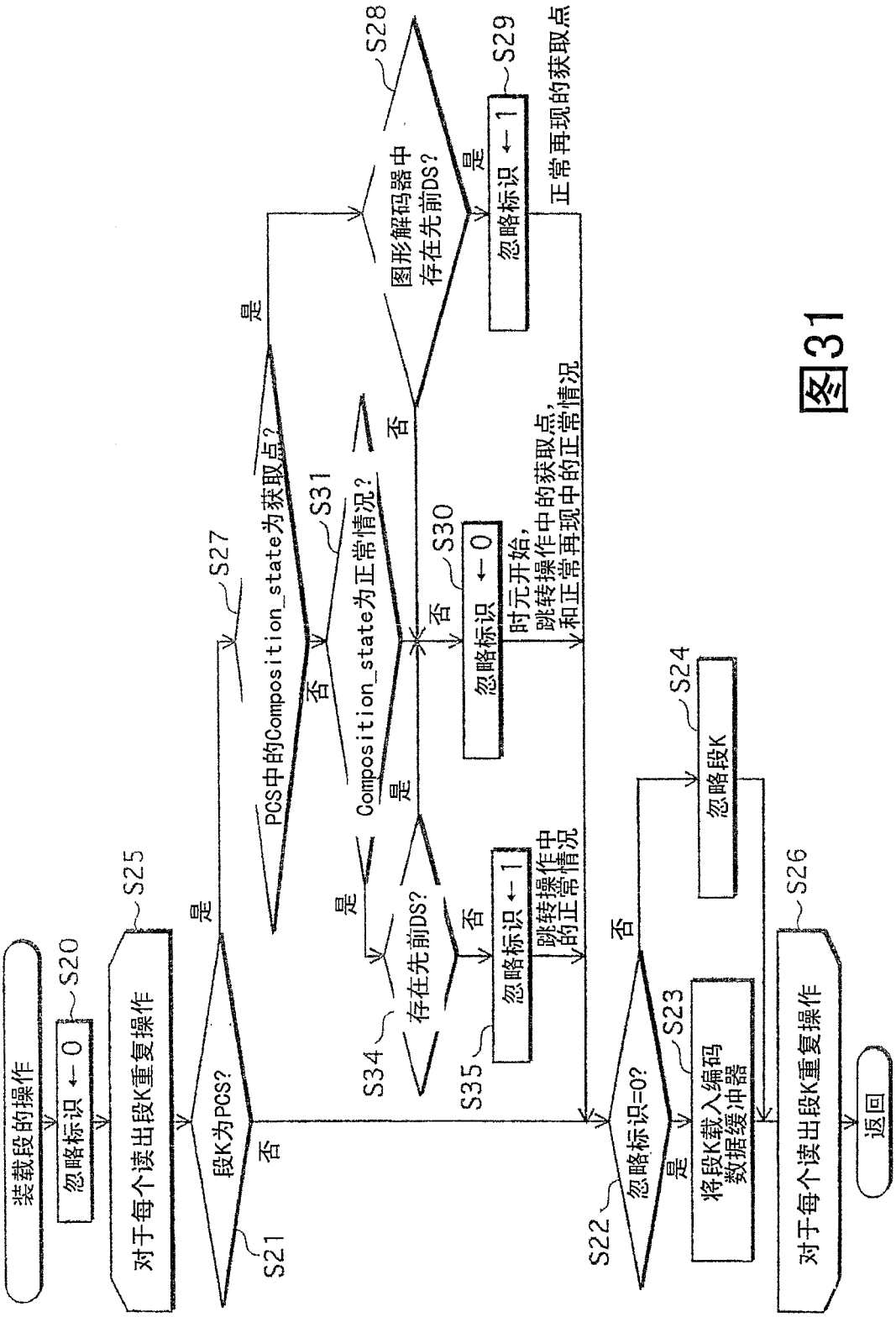

装载段的操作
忽略标识 ← 0
S20
对于每个读出段K重复操作
S25
段K为PCS?
是
否
S21
PCS中的Composition_state为获取点?
是
否
S27
Composition_state为正常情况?
是
否
S31
存在先前DS?
是
否
S34
忽略标识 ← 1
S35
跳转操作中的正常情况
图形解码器中存在先前DS?
是
否
S28
忽略标识 ← 1
S29
正常再现的获取点
忽略标识 ← 0
S30
时元开始，跳转操作中的获取点，和正常再现中的正常情况
忽略标识=0?
是
否
S22
将段K载入编码数据缓冲器
S23
忽略段K
S24
对于每个读出段K重复操作
S26
返回

图31

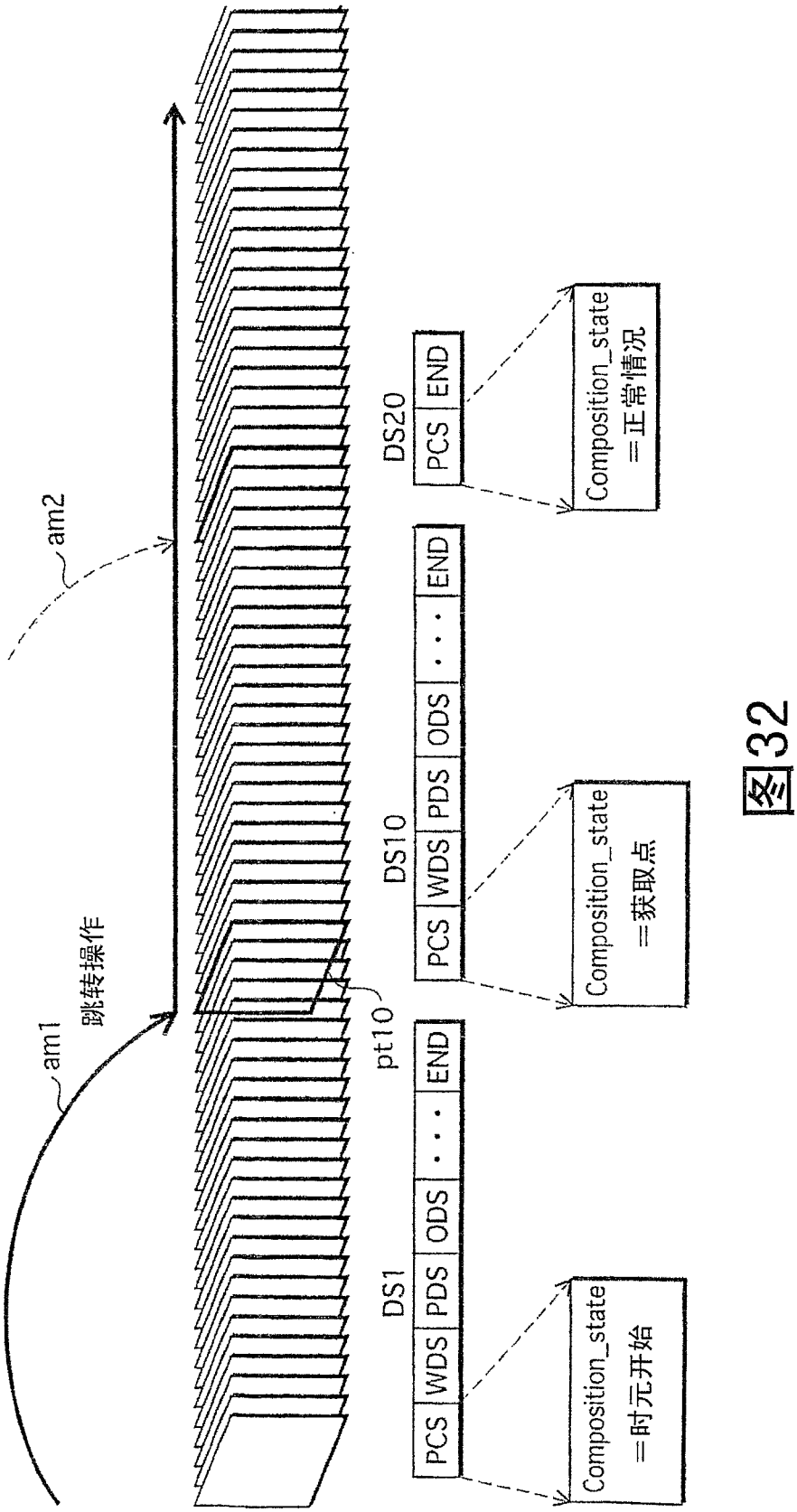

am1
跳转操作
am2
pt10
DS1
PCS
WDS
PDS
ODS
· · ·
END
Composition_state
=时元开始
DS10
PCS
WDS
PDS
ODS
· · ·
END
Composition_state
=获取点
DS20
PCS
END
Composition_state
=正常情况

图32

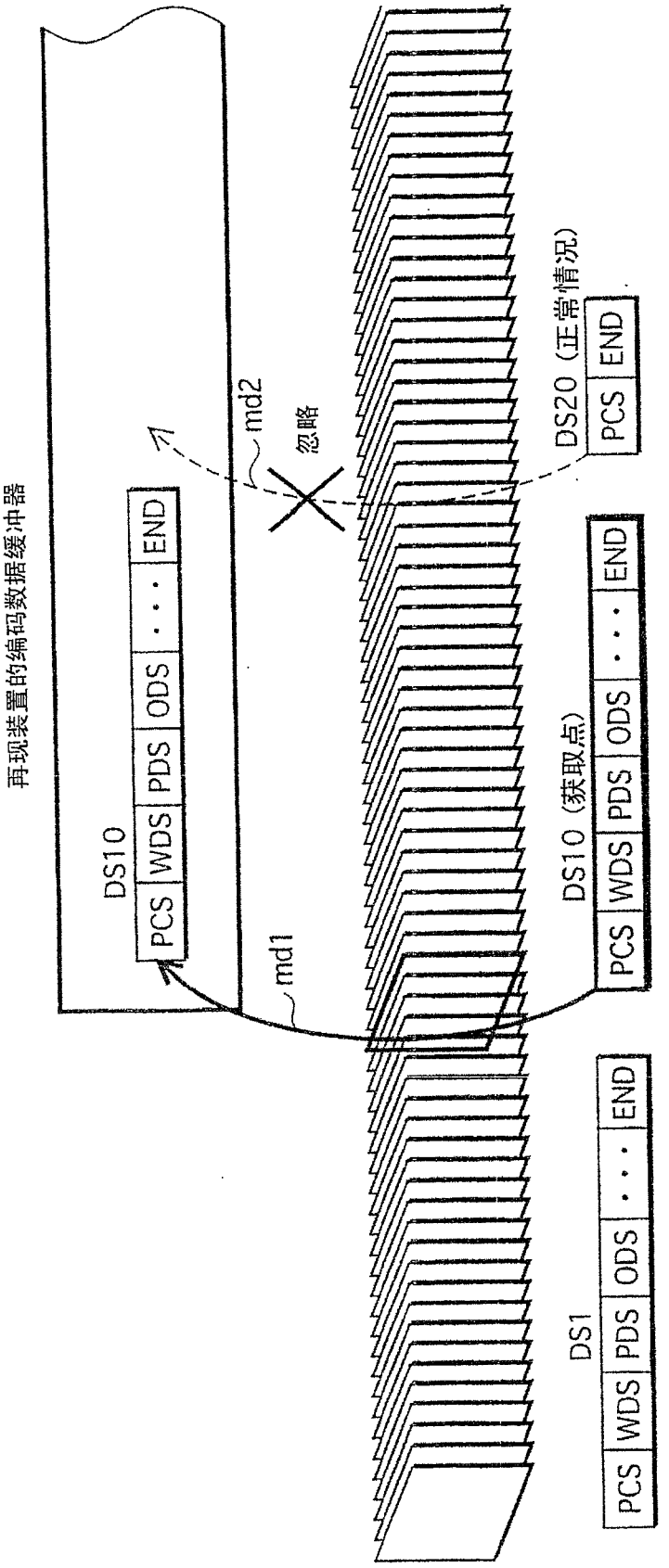

再现装置的编码数据缓冲器
DS10
PCS
WDS
PDS
ODS
· · ·
END
md1
md2
忽略
DS20（正常情况）
PCS
END
DS10（获取点）
PCS
WDS
PDS
ODS
· · ·
END
DS1
PCS
WDS
PDS
ODS
· · ·
END

图33

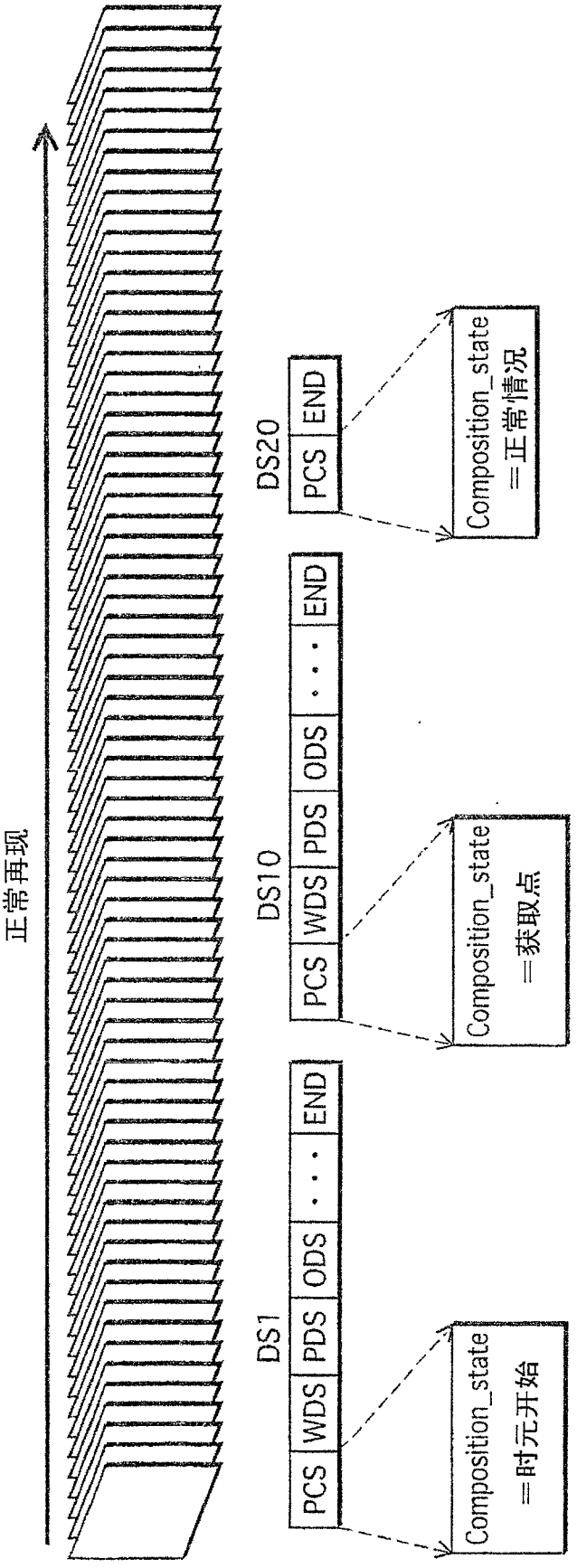

正常再现
DS1
PCS
WDS
PDS
ODS
· · ·
END
Composition_state
=时元开始
DS10
PCS
WDS
PDS
ODS
· · ·
END
Composition_state
=获取点
DS20
PCS
END
Composition_state
=正常情况

图34

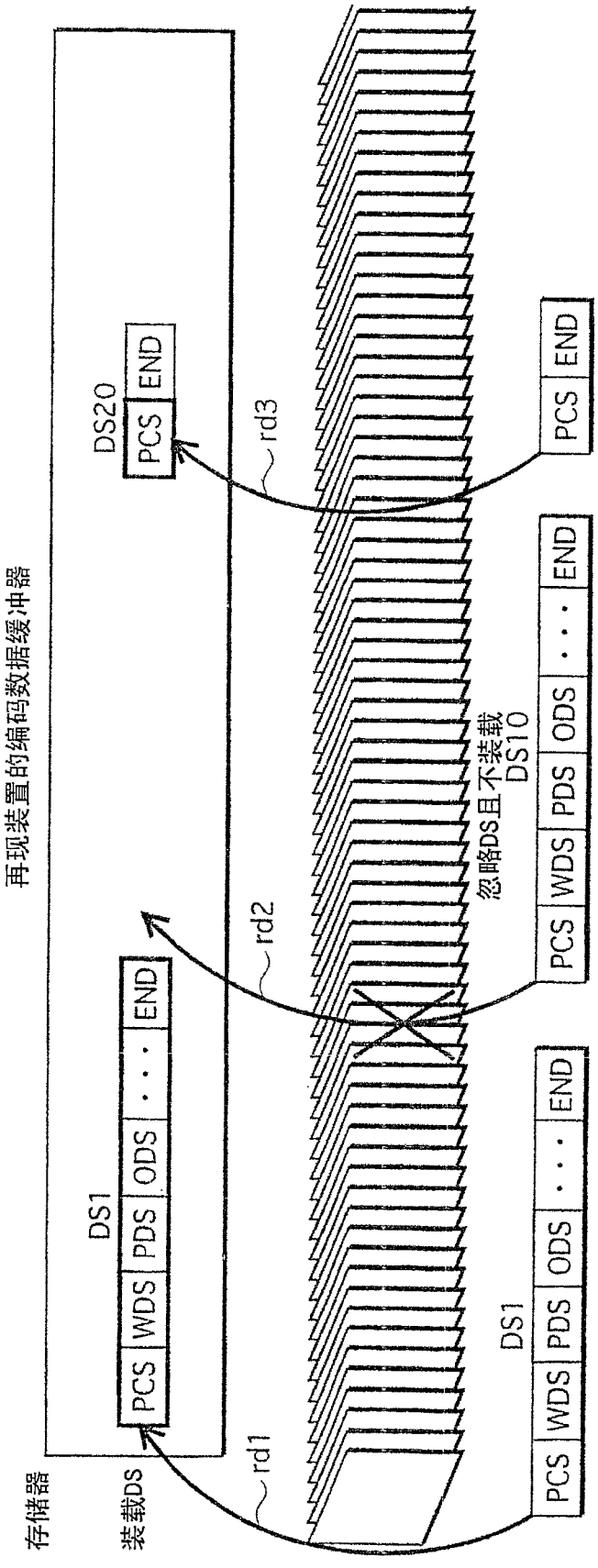

再现装置的编码数据缓冲器
存储器
装载DS
DS1
PCS
WDS
PDS
ODS
· · ·
END
DS20
PCS
END
rd1
rd2
rd3
忽略DS且不装载
DS10

图35

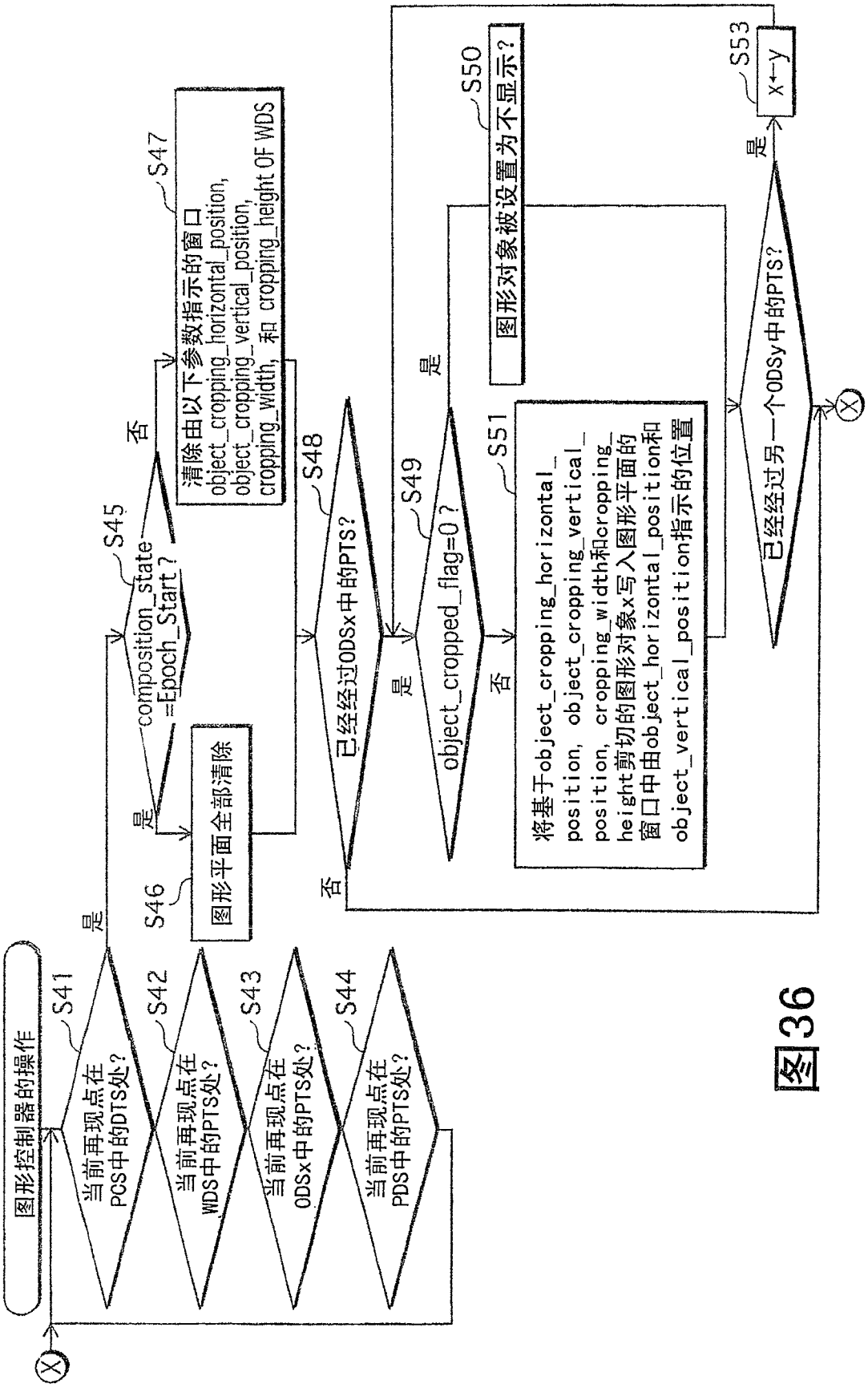

图形控制器的操作
S41
当前再现点在POS中的DTS处?
是
S42
当前再现点在WDS中的PTS处?
是
S43
当前再现点在ODSx中的PTS处?
是
S44
当前再现点在PDS中的PTS处?
是
S45
composition_state =Epoch_Start ?
否
S46
图形平面全部清除
S47
清除由以下参数指示的窗口
object_cropping_horizontal_position,
object_cropping_vertical_position,
cropping_width, 和 cropping_height OF WDS
S48
已经经过ODSx中的PTS?
否
S49
object_cropped_flag=0 ?
是
S50
图形对象被设置为不显示?
是
S51
否
将基于object_cropping_horizontal_position, object_cropping_vertical_position, cropping_width和cropping_height剪切的图形对象x写入图形平面的窗口中由object_horizontal_position和object_vertical_position指示的位置
已经经过另一个ODSy中的PTS?
否
S53
x←y
是

图36

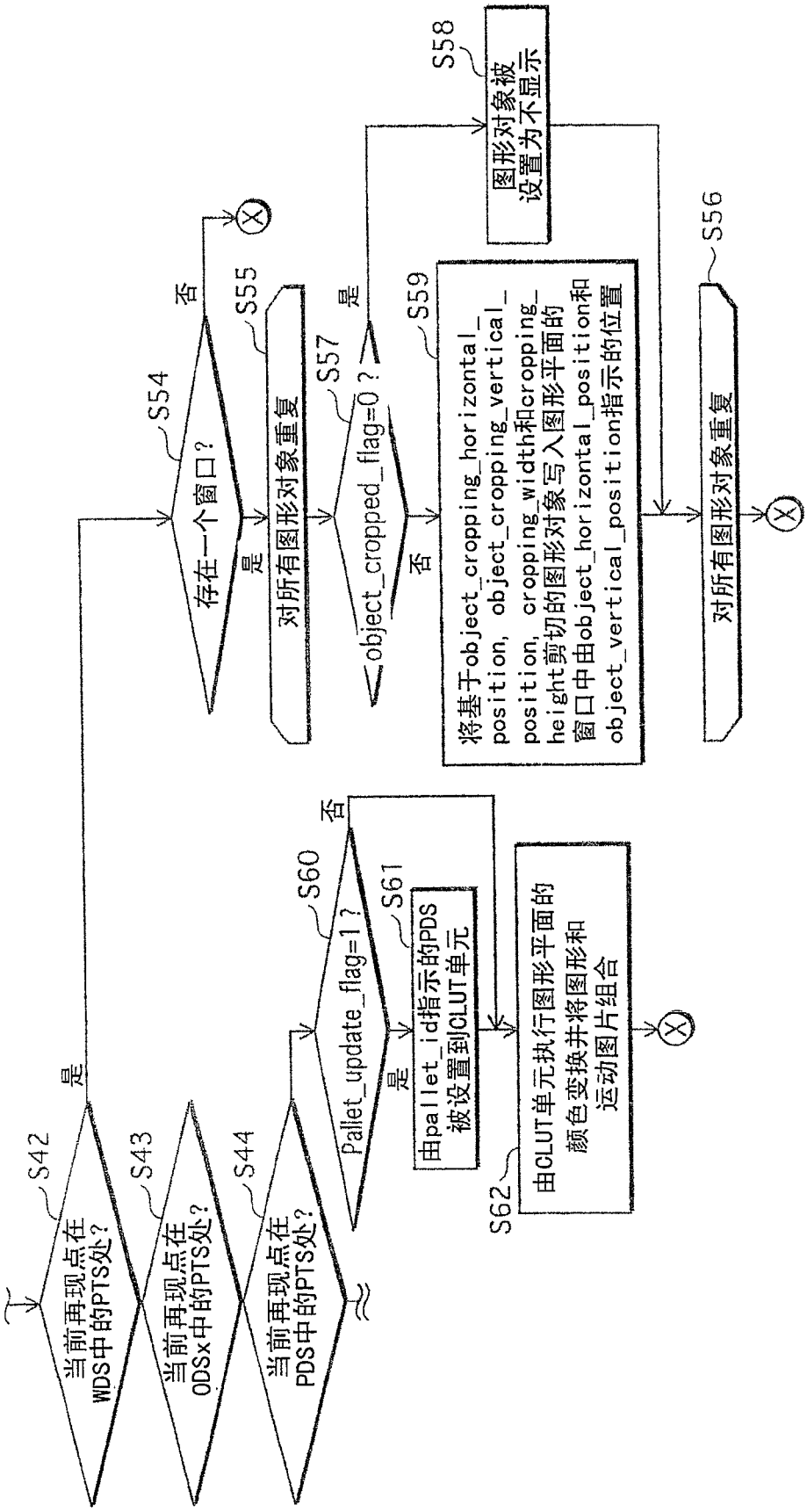

S42
当前再现点在WDS中的PTS处?
是
S43
当前再现点在ODSx中的PTS处?
S44
当前再现点在PDS中的PTS处?
S60
Pallet_update_flag=1 ?
否
是
S61
由pallet_id指示的PDS被设置到CLUT单元
S62
由CLUT单元执行图形平面的颜色变换并将图形和运动图片组合
S54
存在一个窗口?
否
是
S55
对所有图形对象重复
S57
object_cropped_flag=0 ?
否
是
S59
将基于object_cropping_horizontal_position, object_cropping_vertical_position, cropping_width和cropping_height剪切的图形对象写入图形平面的窗口中由object_horizontal_position和object_vertical_position指示的位置
S58
图形对象被设置为不显示
S56
对所有图形对象重复

图37

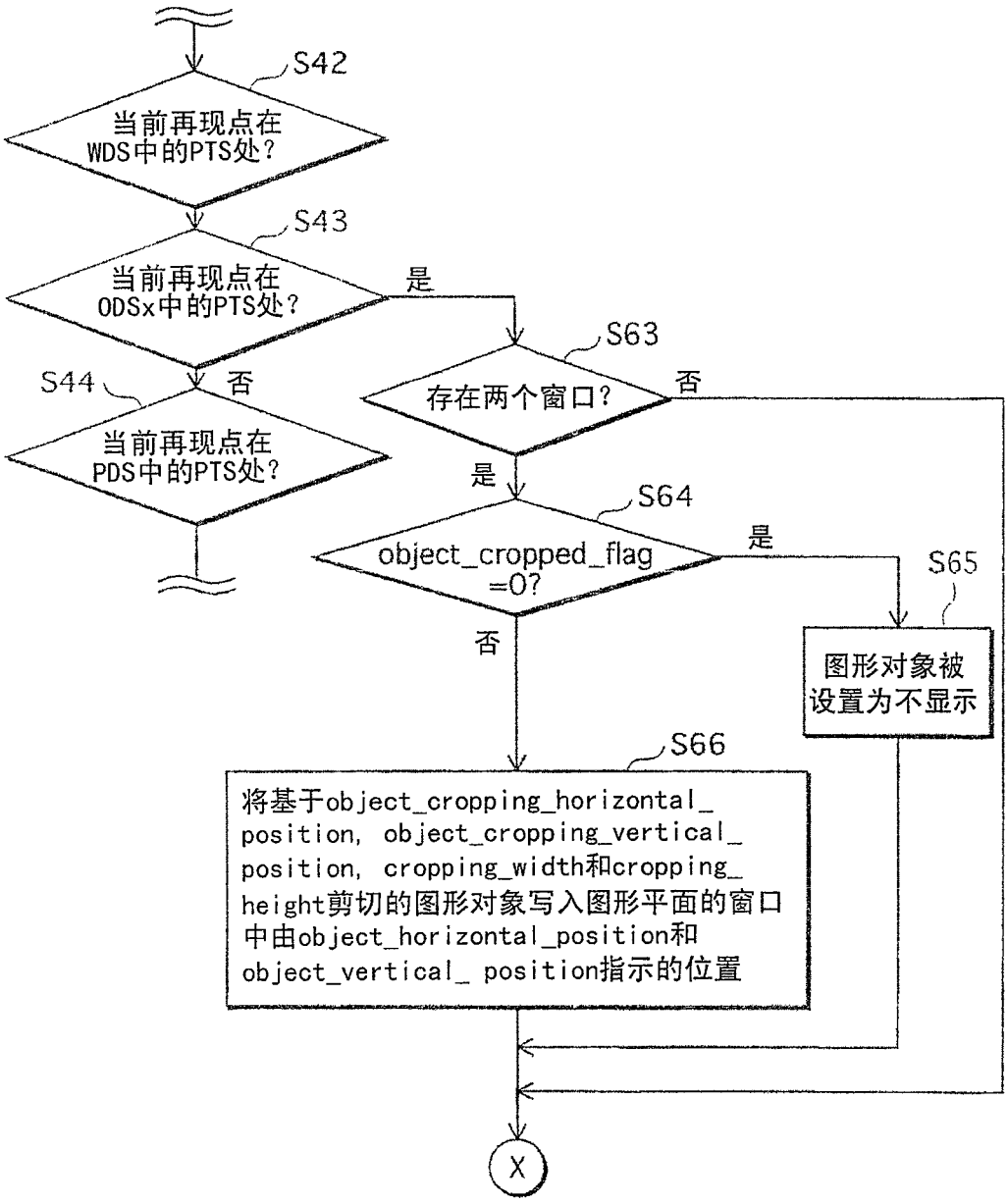

S42
当前再现点在WDS中的PTS处?
S43
当前再现点在ODSx中的PTS处?
是
否
S44
当前再现点在PDS中的PTS处?
S63
存在两个窗口?
否
是
S64
object_cropped_flag =0?
是
否
S65
图形对象被设置为不显示
S66
将基于object_cropping_horizontal_position, object_cropping_vertical_position, cropping_width和cropping_height剪切的图形对象写入图形平面的窗口中由object_horizontal_position和object_vertical_ position指示的位置
X

图38

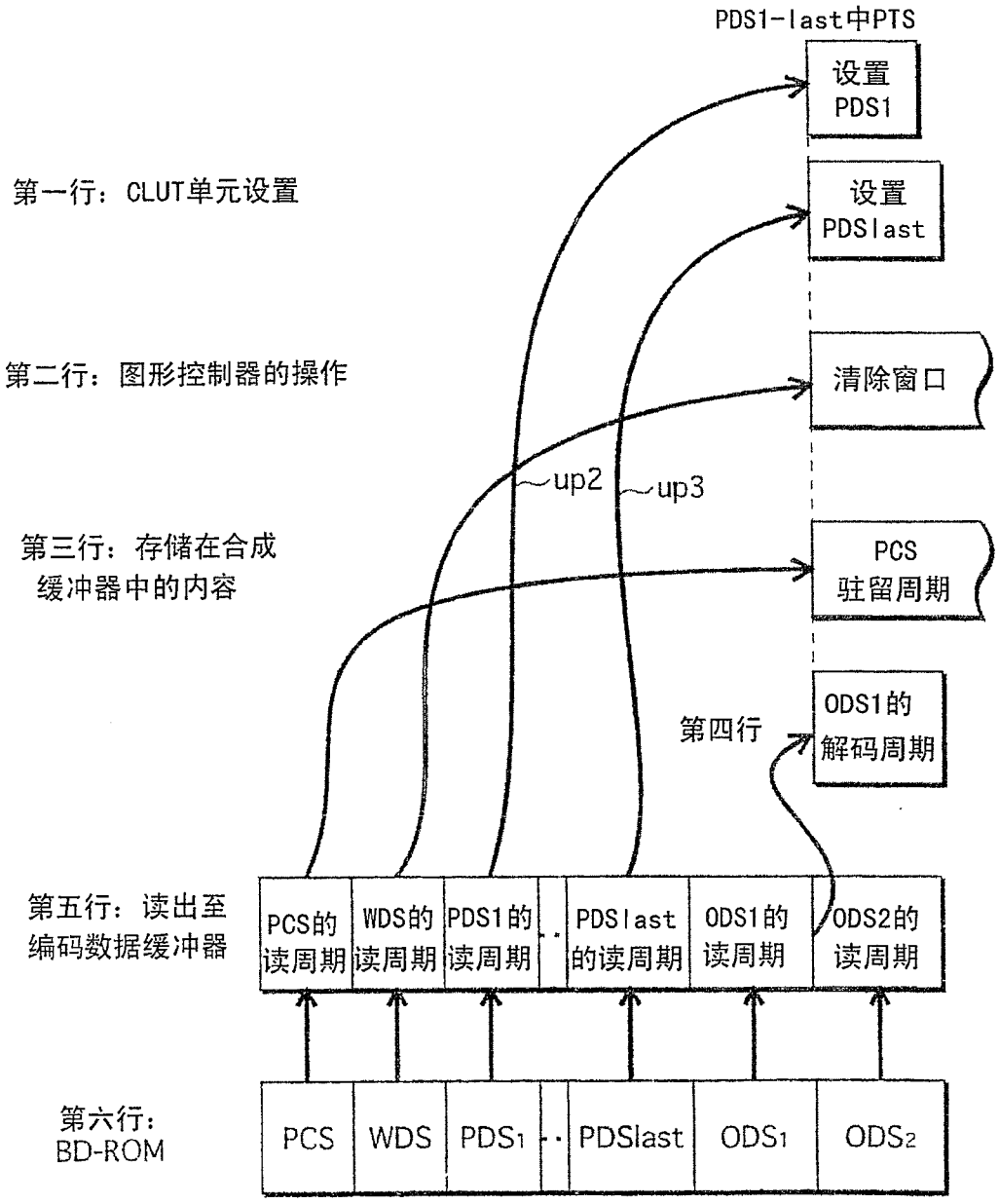

PDS1-last中PTS
设置 PDS1
设置 PDSlast
清除窗口
PCS 驻留周期
ODS1的 解码周期
第四行
up2
up3
第一行：CLUT单元设置
第二行：图形控制器的操作
第三行：存储在合成 缓冲器中的内容
第五行：读出至 编码数据缓冲器
第六行： BD-ROM
PCS的 读周期
WDS的 读周期
PDS1的 读周期
PDSlast 的读周期
ODS1的 读周期
ODS2的 读周期
PCS
WDS
PDS1
PDSlast
ODS1
ODS2

图39

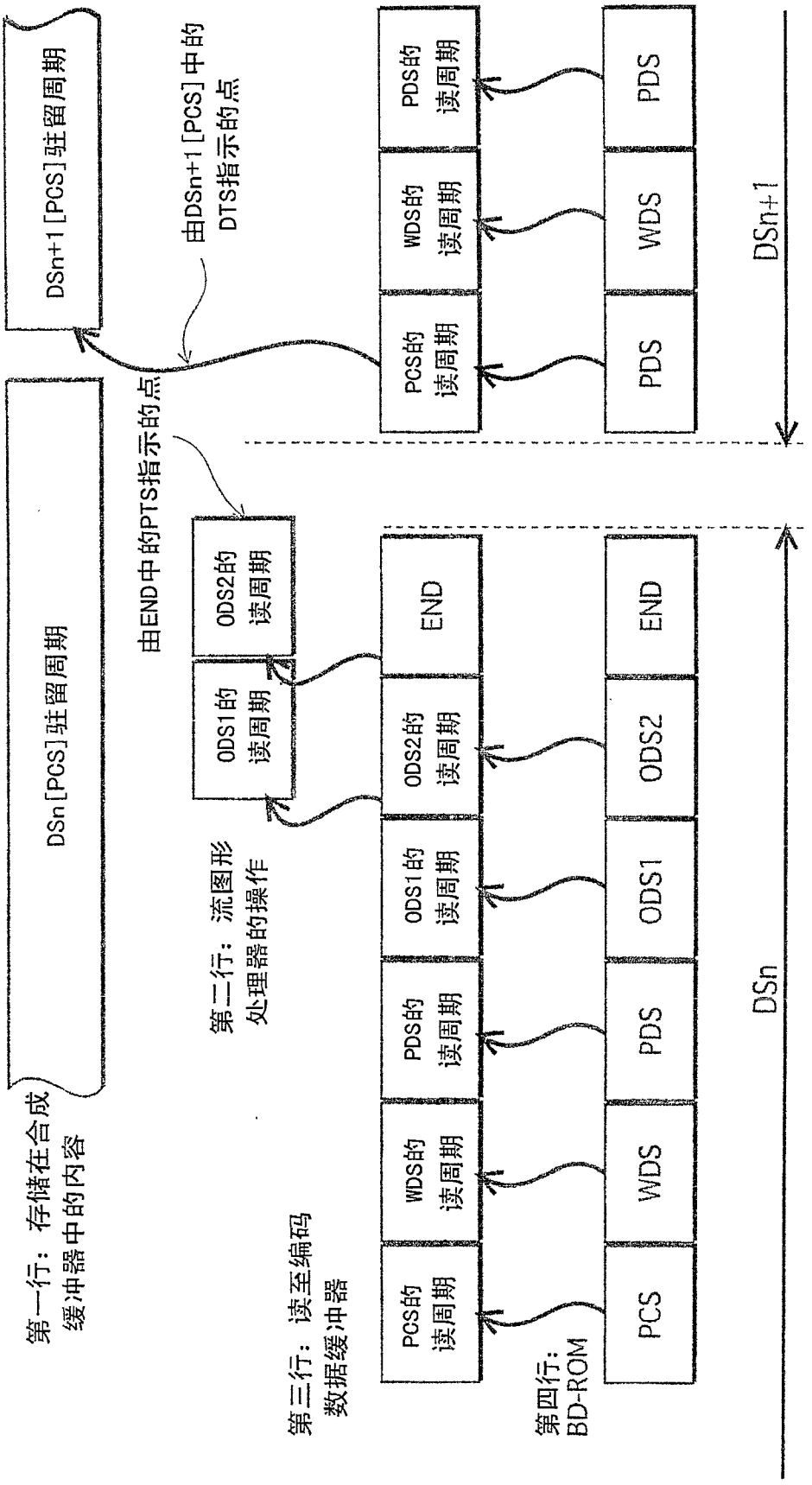

第一行：存储在合成缓冲器中的内容
DSn[PCS]驻留周期
DSn+1[PCS]驻留周期
由END中的PTS指示的点
由DSn+1[PCS]中的DTS指示的点
第二行：流图形处理器的操作
第三行：读至编码数据缓冲器
第四行：BD-ROM
PCS的读周期
WDS的读周期
PDS的读周期
ODS1的读周期
ODS2的读周期
END
PCS
WDS
PDS
ODS1
ODS2
END
DSn
DSn+1

图40

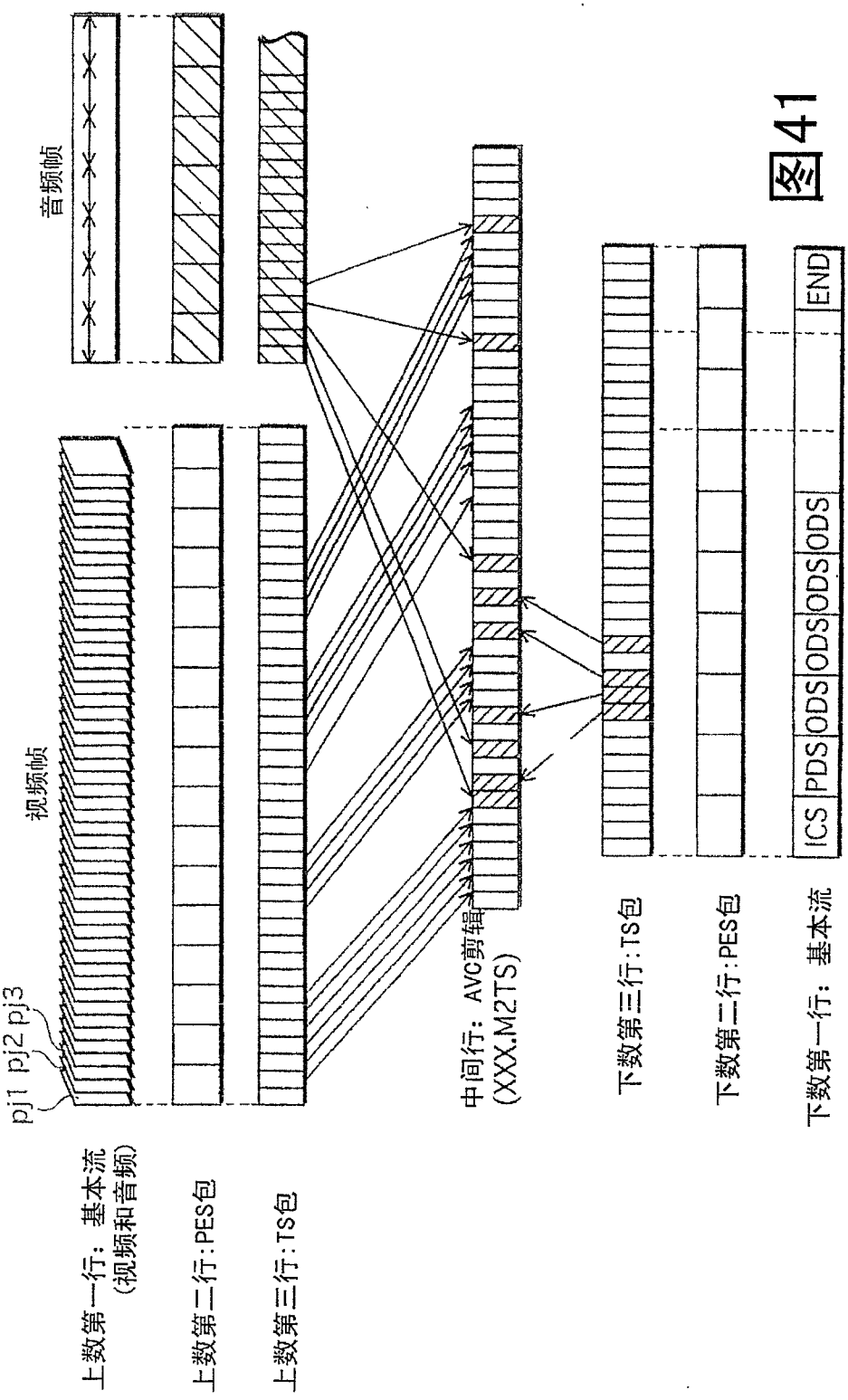

音频帧
视频帧
pj1 pj2 pj3
上数第一行：基本流
(视频和音频)
上数第二行:PES包
上数第三行:TS包
中间行：AVC剪辑
(XXX.M2TS)
下数第三行:TS包
下数第二行:PES包
下数第一行：基本流
ICS
PDS
ODS
ODS
ODS
ODS
END
交互图形流

图41

。

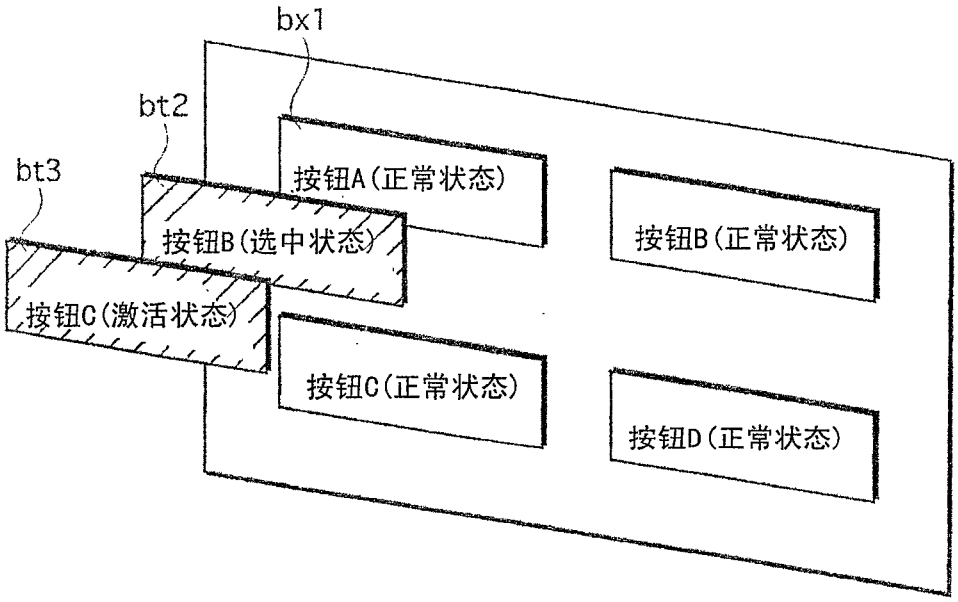

bx1
bt2
bt3
按钮A(正常状态)
按钮B(选中状态)
按钮C(激活状态)
按钮B(正常状态)
按钮C(正常状态)
按钮D(正常状态)

图42A

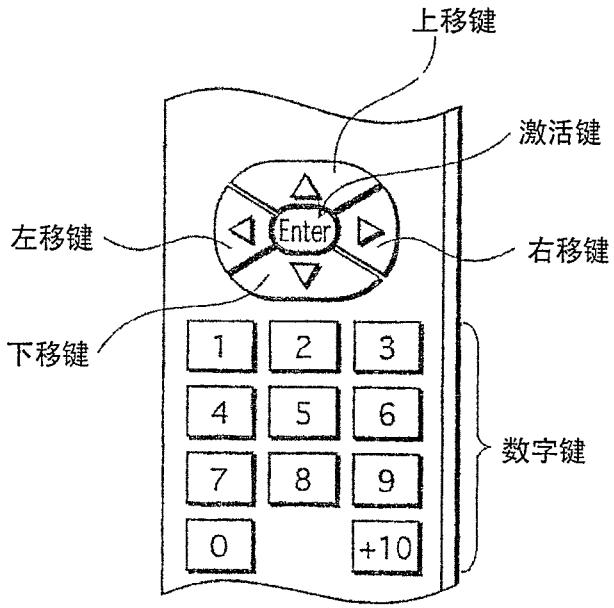

上移键
激活键
左移键
Enter
右移键
下移键
1
2
3
4
5
6
7
8
9
0
+10
数字键

图42B

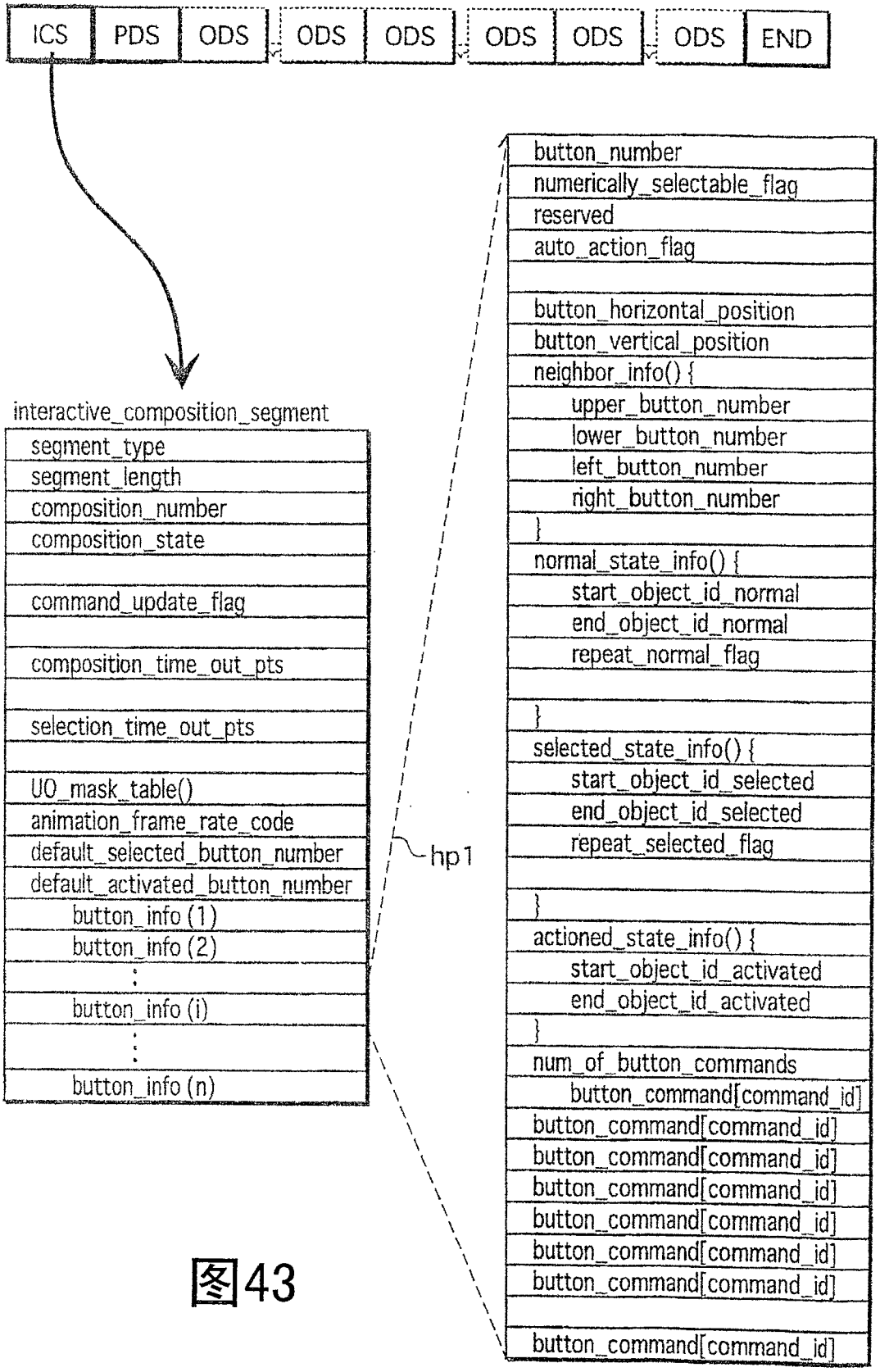

ICS
PDS
ODS
ODS
ODS
ODS
ODS
ODS
END
interactive_composition_segment
segment_type
segment_length
composition_number
composition_state
command_update_flag
composition_time_out_pts
selection_time_out_pts
UO_mask_table()
animation_frame_rate_code
default_selected_button_number
default_activated_button_number
button_info (1)
button_info (2)
button_info (i)
button_info (n)
hp1
button_number
numerically_selectable_flag
reserved
auto_action_flag
button_horizontal_position
button_vertical_position
neighbor_info() {
upper_button_number
lower_button_number
left_button_number
right_button_number
}
normal_state_info() {
start_object_id_normal
end_object_id_normal
repeat_normal_flag
}
selected_state_info() {
start_object_id_selected
end_object_id_selected
repeat_selected_flag
}
actioned_state_info() {
start_object_id_activated
end_object_id_activated
}
num_of_button_commands
button_command[command_id]
button_command[command_id]
button_command[command_id]
button_command[command_id]
button_command[command_id]
button_command[command_id]
button_command[command_id]
button_command[command_id]

图43

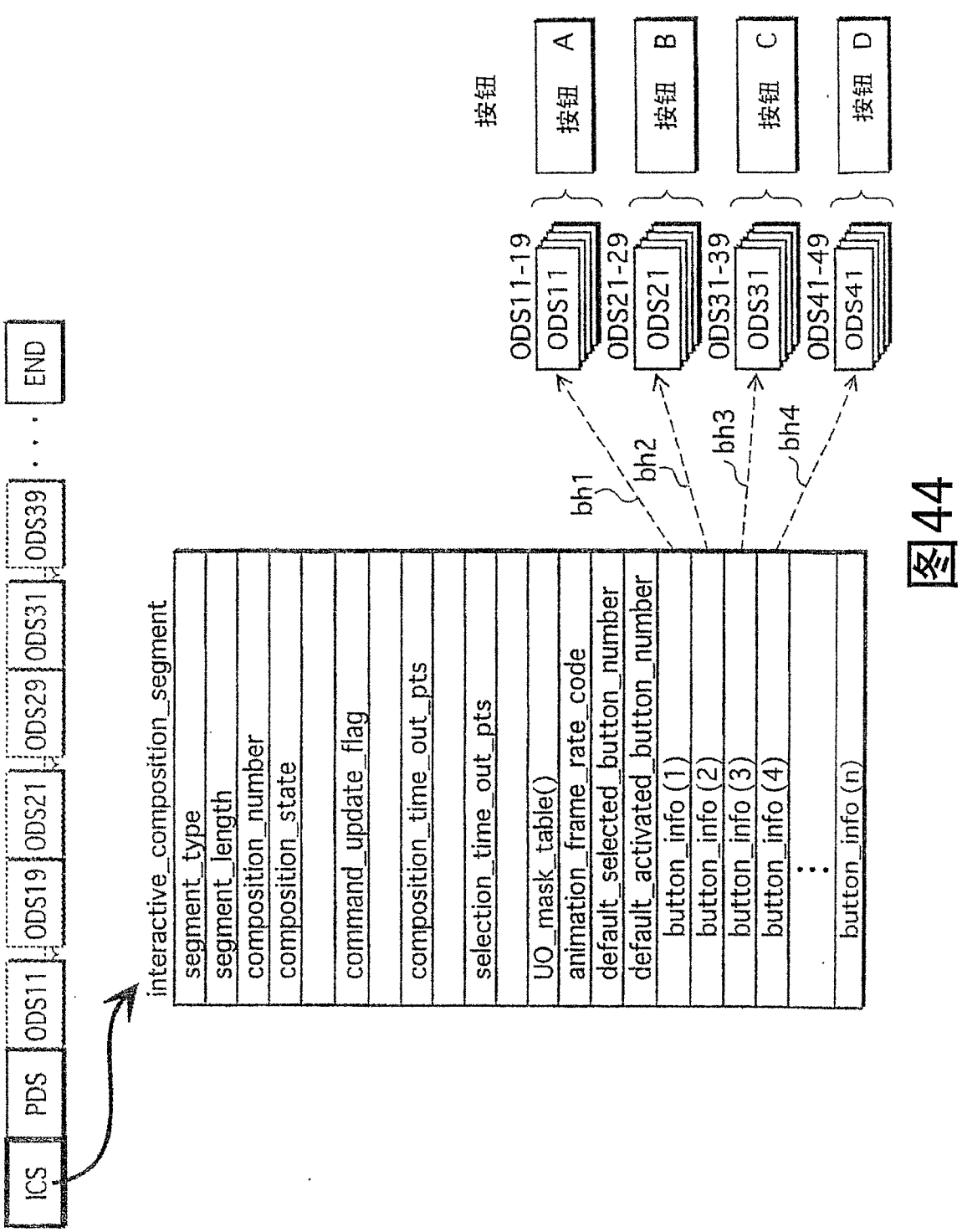

ICS
PDS
ODS11
ODS19
ODS21
ODS29
ODS31
ODS39
END
interactive_composition_segment
segment_type
segment_length
composition_number
composition_state
command_update_flag
composition_time_out_pts
selection_time_out_pts
UO_mask_table()
animation_frame_rate_code
default_selected_button_number
default_activated_button_number
button_info (1)
button_info (2)
button_info (3)
button_info (4)
button_info (n)
bh1
bh2
bh3
bh4
按钮
按钮 A
按钮 B
按钮 C
按钮 D
ODS11-19
ODS11
ODS21-29
ODS21
ODS31-39
ODS31
ODS41-49
ODS41

图44

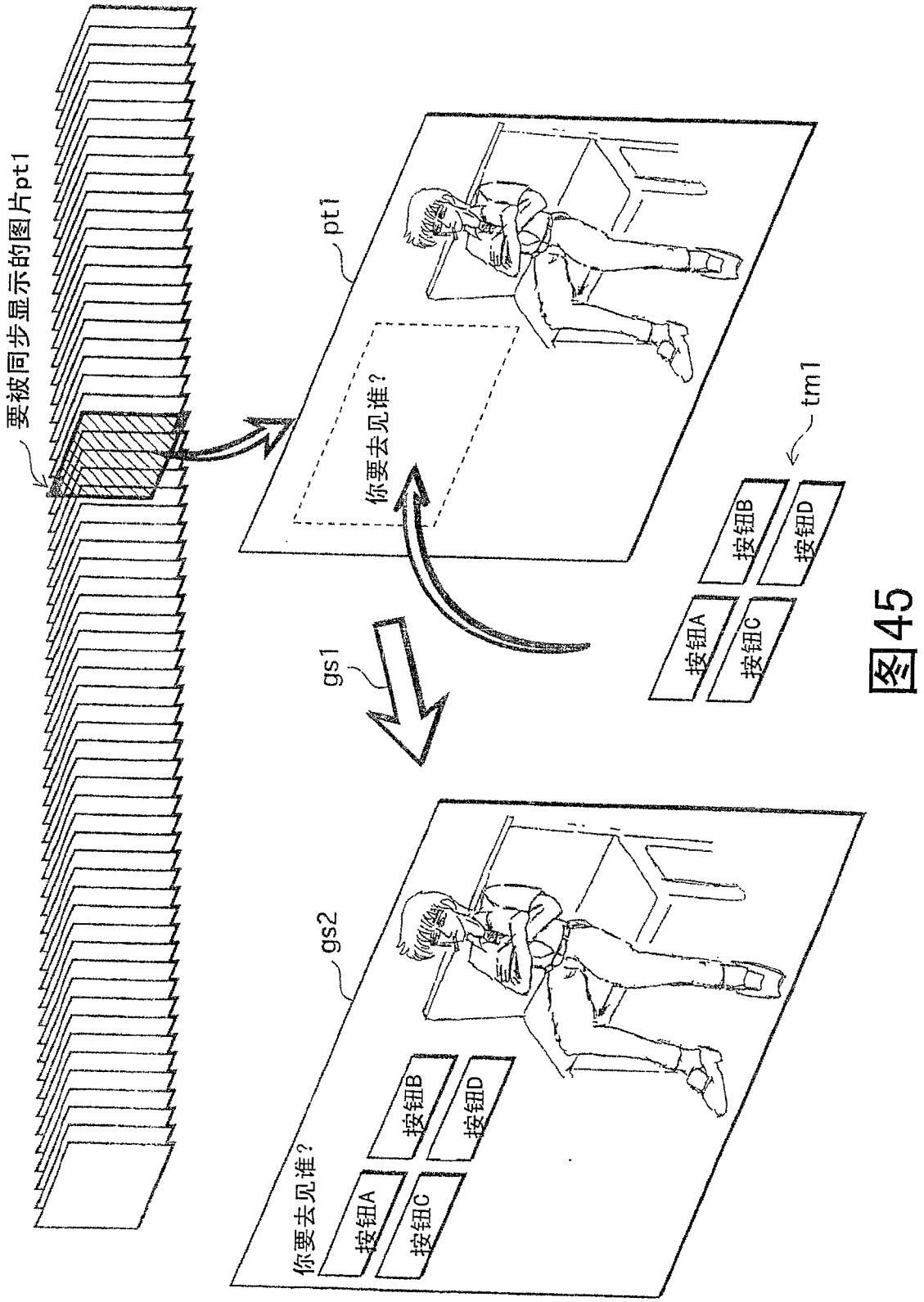

要被同步显示的图片pt1
pt1
你要去见谁？
tm1
按钮A
按钮B
按钮C
按钮D
gs1
gs2
你要去见谁？
按钮A
按钮B
按钮C
按钮D

图45

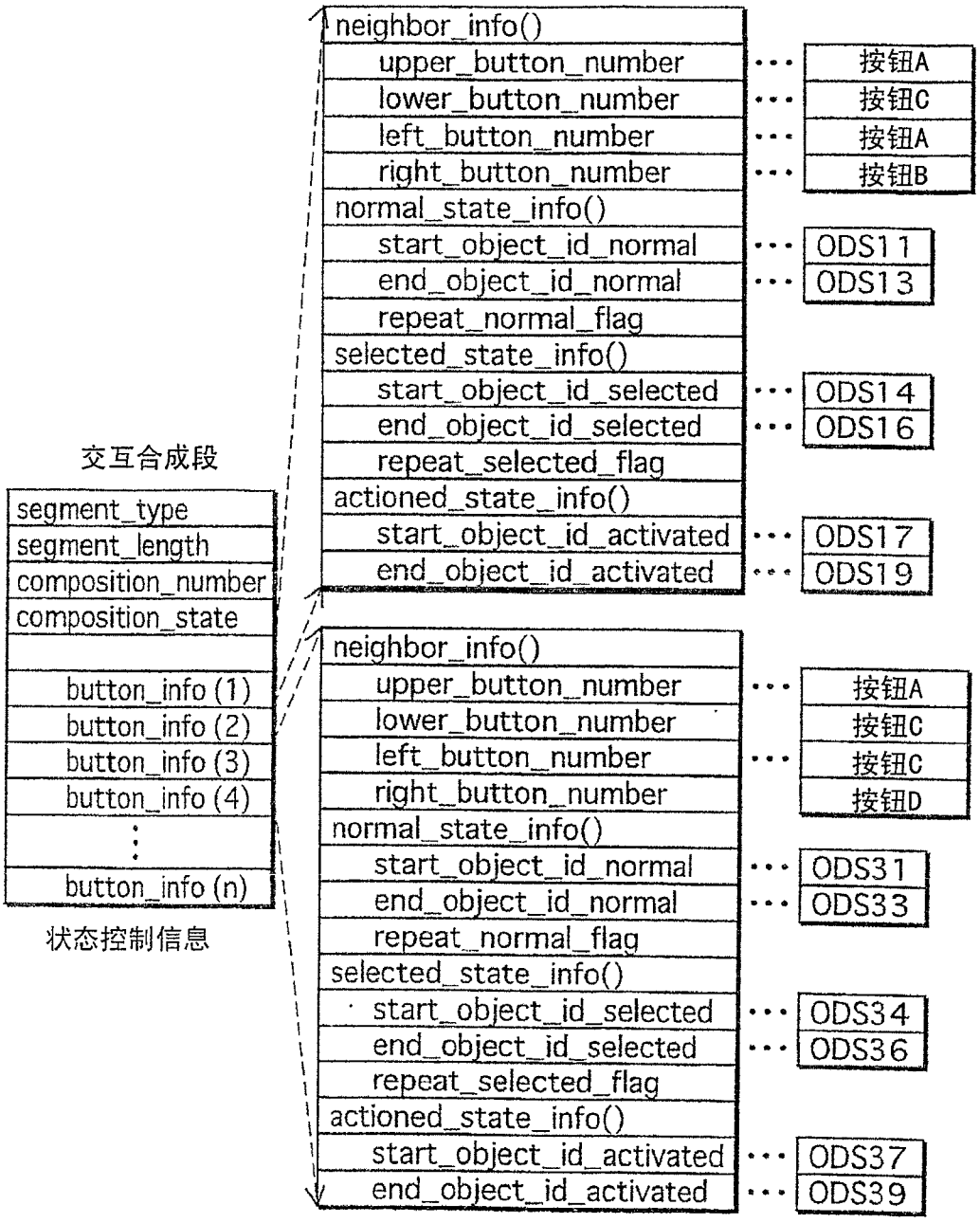

交互合成段
segment_type
segment_length
composition_number
composition_state
button_info (1)
button_info (2)
button_info (3)
button_info (4)
button_info (n)
状态控制信息
neighbor_info()
upper_button_number
lower_button_number
left_button_number
right_button_number
normal_state_info()
start_object_id_normal
end_object_id_normal
repeat_normal_flag
selected_state_info()
start_object_id_selected
end_object_id_selected
repeat_selected_flag
actioned_state_info()
start_object_id_activated
end_object_id_activated
按钮A
按钮C
按钮A
按钮B
ODS11
ODS13
ODS14
ODS16
ODS17
ODS19
neighbor_info()
upper_button_number
lower_button_number
left_button_number
right_button_number
normal_state_info()
start_object_id_normal
end_object_id_normal
repeat_normal_flag
selected_state_info()
start_object_id_selected
end_object_id_selected
repeat_selected_flag
actioned_state_info()
start_object_id_activated
end_object_id_activated
按钮A
按钮C
按钮C
按钮D
ODS31
ODS33
ODS34
ODS36
ODS37
ODS39

图46

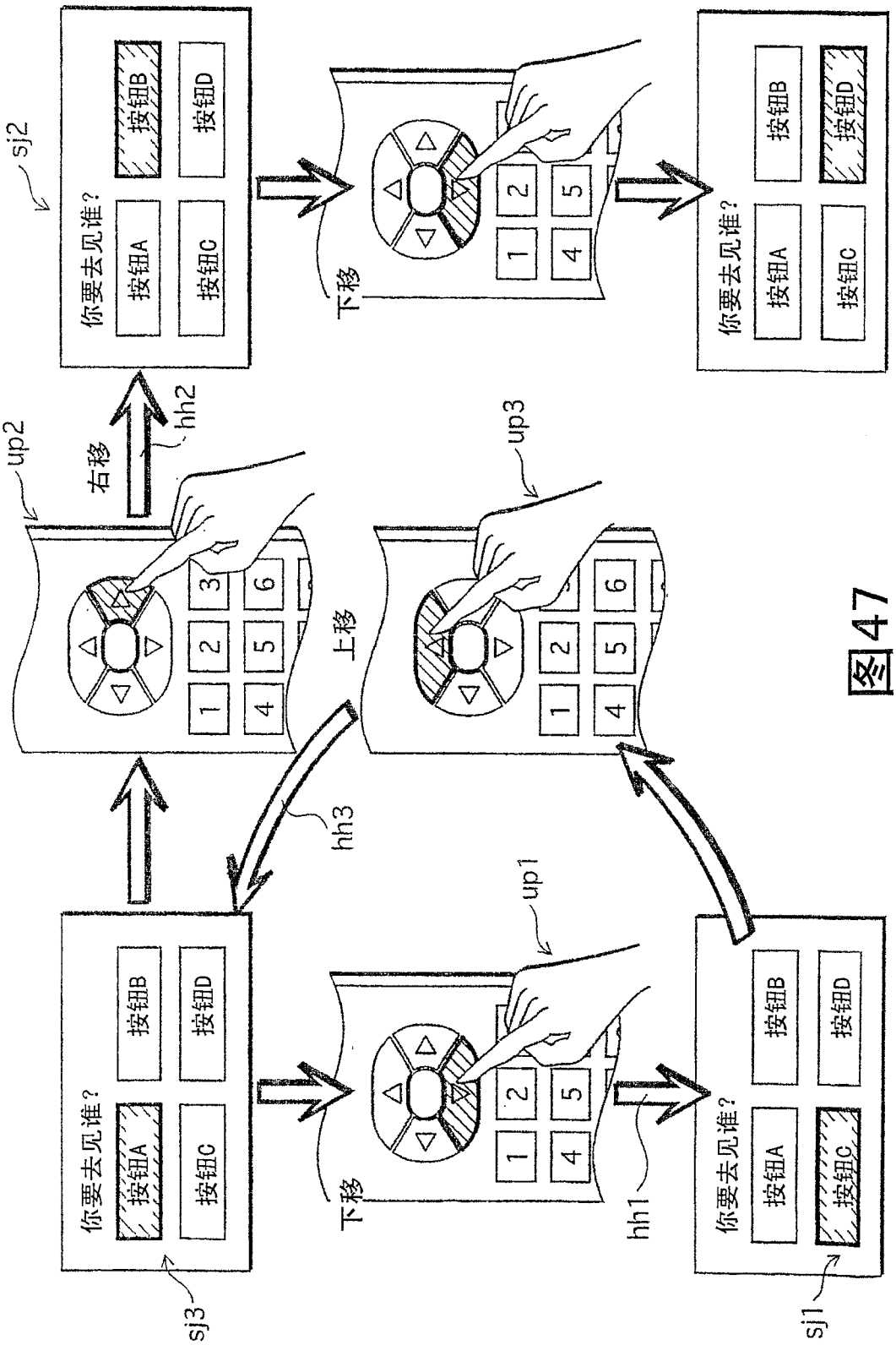

sj2
你要去见谁？
按钮A
按钮B
按钮C
按钮D
up2
右移
hh2
up3
下移
上移
hh3
up1
hh1
sj3
sj1
1
2
3
4
5
6

图47



图48

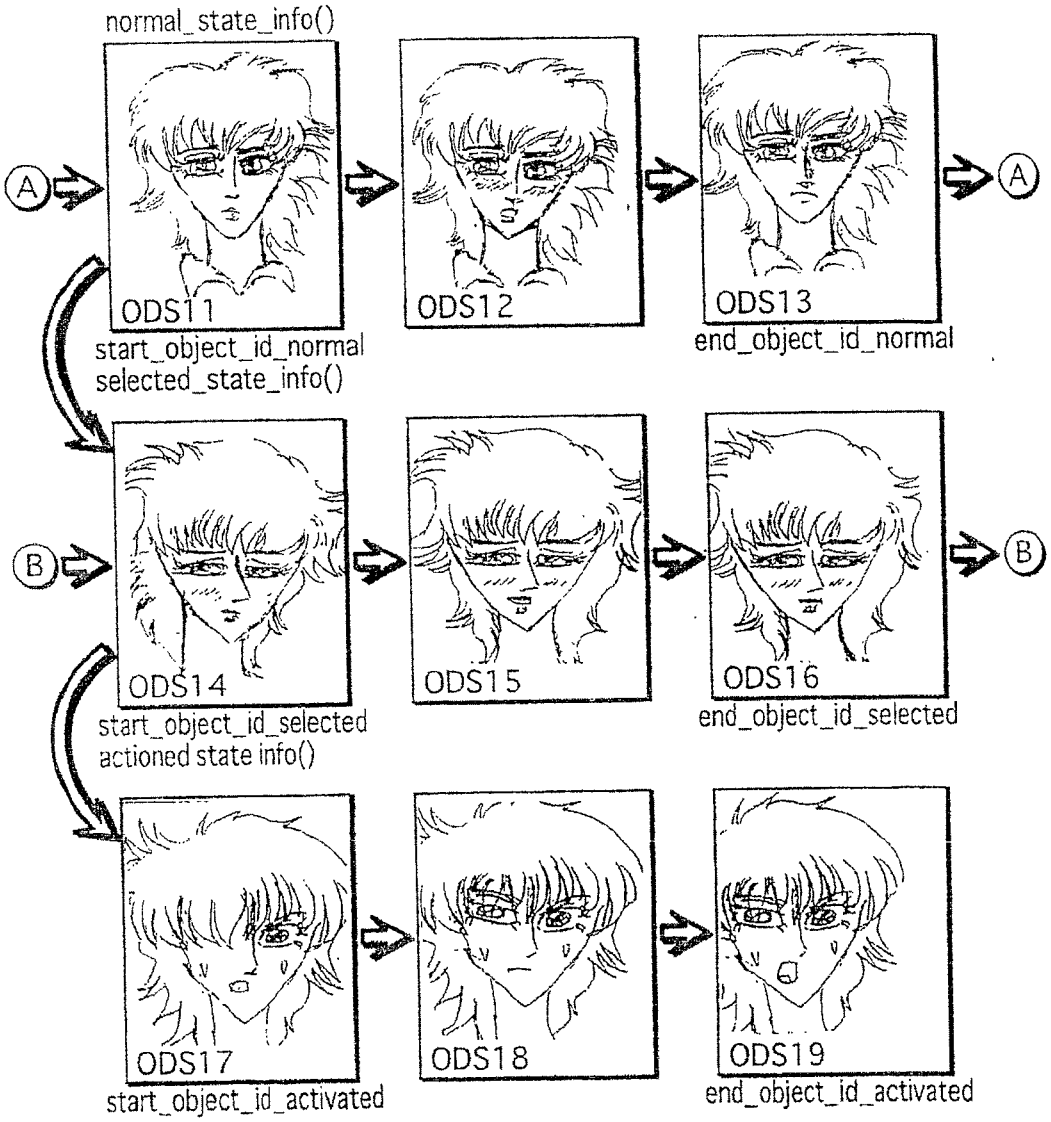

normal_state_info()
A
ODS11
ODS12
ODS13
A
start_object_id_normal
selected_state_info()
end_object_id_normal
B
ODS14
ODS15
ODS16
B
start_object_id_selected
actioned state info()
end_object_id_selected
ODS17
ODS18
ODS19
start_object_id_activated
end_object_id_activated

图49

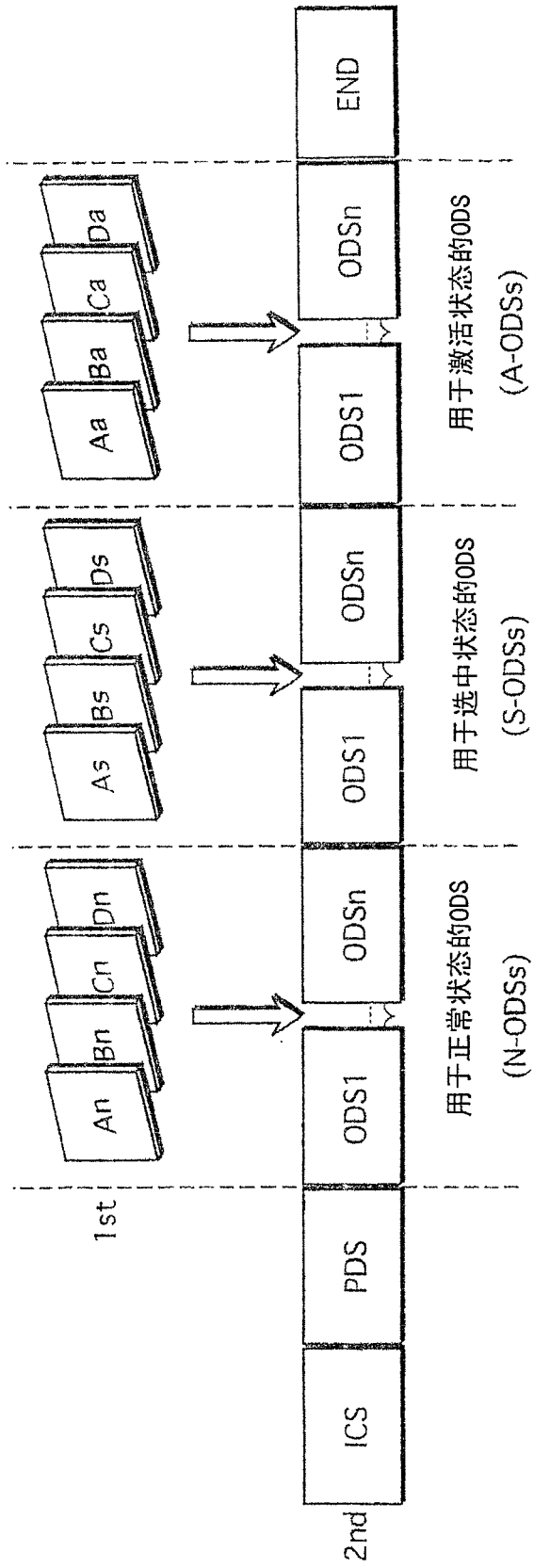

2nd
ICS
PDS
1st
ODS1
ODSn
END
An
Bn
Cn
Dn
As
Bs
Cs
Ds
Aa
Ba
Ca
Da
用于正常状态的ODS
(N-ODSs)
用于选中状态的ODS
(S-ODSs)
用于激活状态的ODS
(A-ODSs)

图50

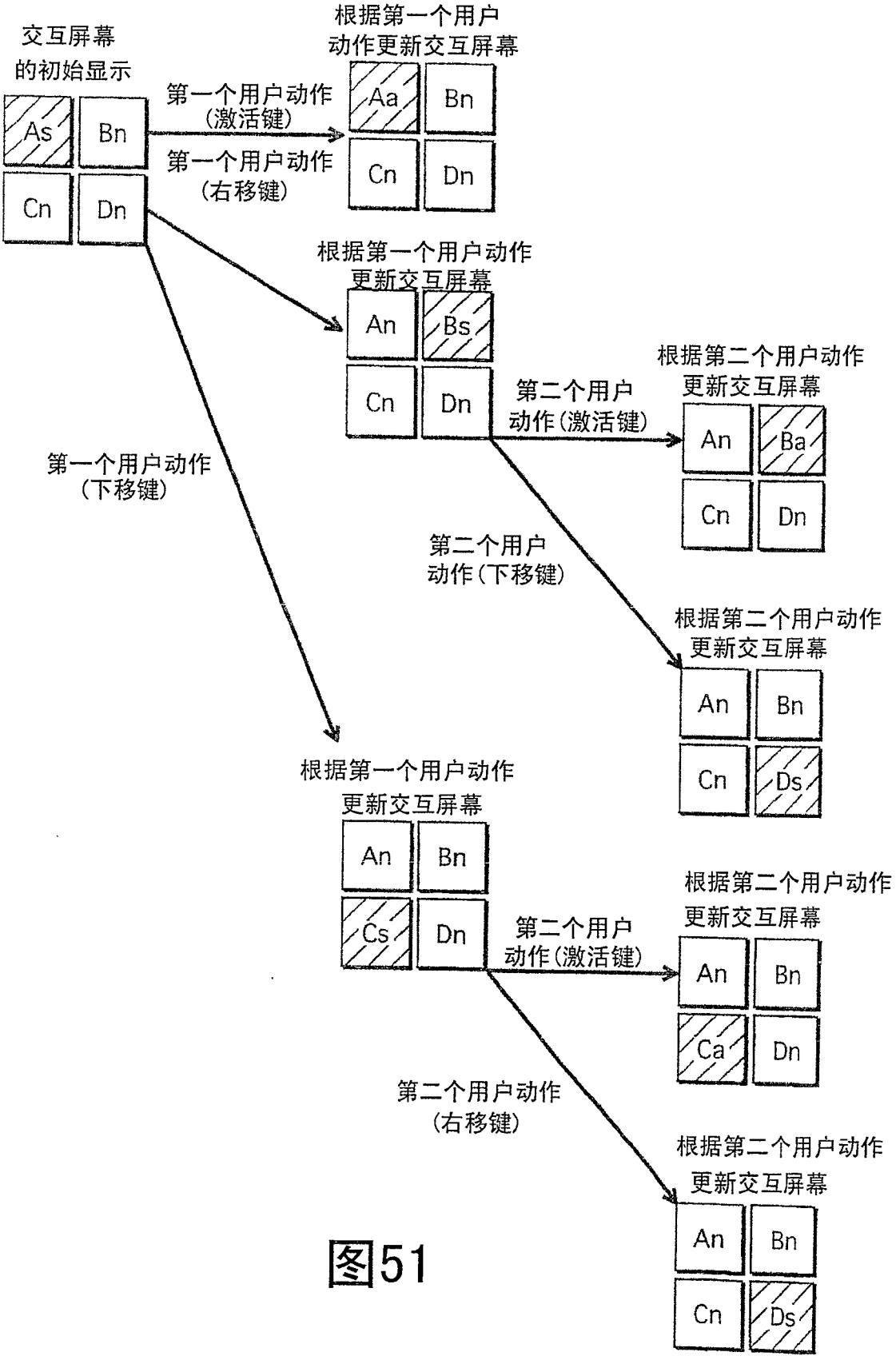

交互屏幕
的初始显示
As
Bn
Cn
Dn
第一个用户动作
(激活键)
第一个用户动作
(右移键)
根据第一个用户
动作更新交互屏幕
Aa
Bn
Cn
Dn
第一个用户动作
(下移键)
根据第一个用户动作
更新交互屏幕
An
Bs
Cn
Dn
第二个用户
动作(激活键)
根据第二个用户动作
更新交互屏幕
An
Ba
Cn
Dn
第二个用户
动作(下移键)
根据第二个用户动作
更新交互屏幕
An
Bn
Cn
Ds
根据第一个用户动作
更新交互屏幕
An
Bn
Cs
Dn
第二个用户
动作(激活键)
根据第二个用户动作
更新交互屏幕
An
Bn
Ca
Dn
第二个用户动作
(右移键)
根据第二个用户动作
更新交互屏幕
An
Bn
Cn
Ds

图51

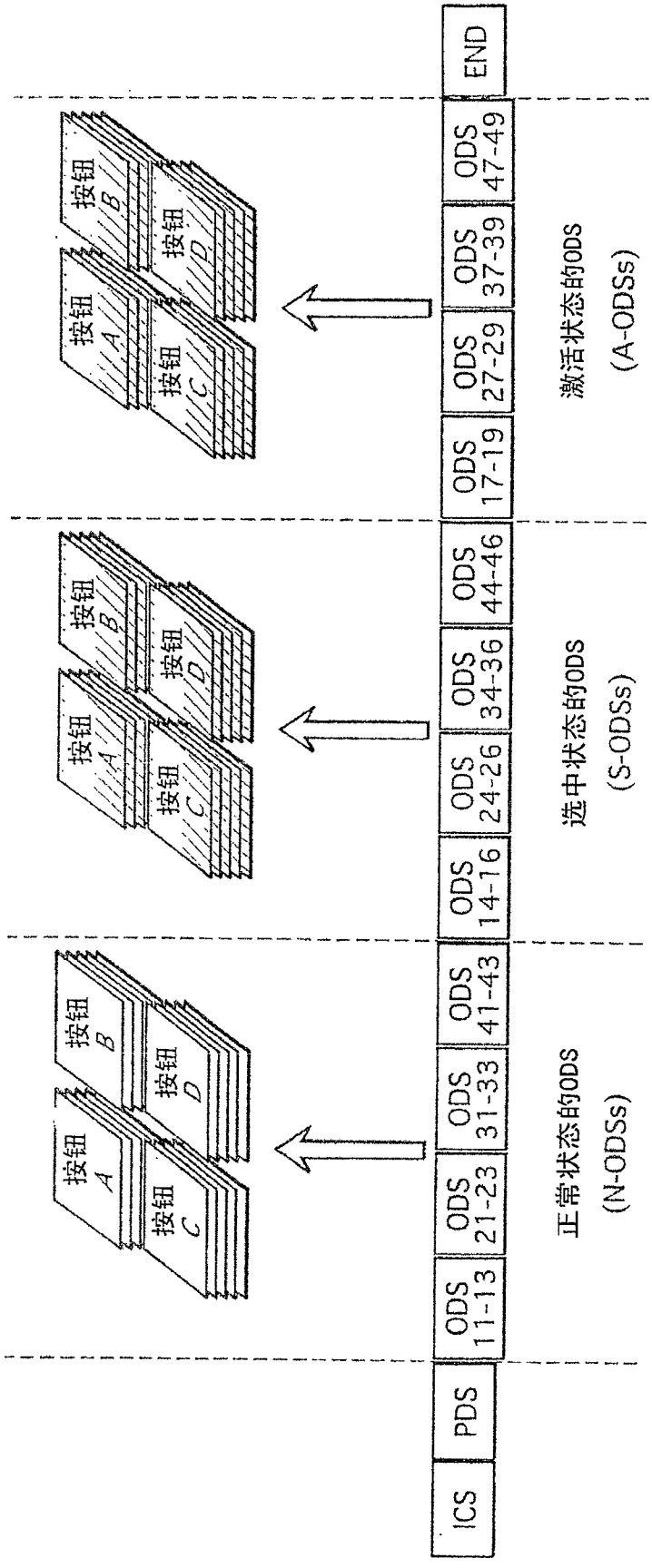

ICS
PDS
ODS 11-13
ODS 21-23
ODS 31-33
ODS 41-43
ODS 14-16
ODS 24-26
ODS 34-36
ODS 44-46
ODS 17-19
ODS 27-29
ODS 37-39
ODS 47-49
END
正常状态的ODS (N-ODSs)
选中状态的ODS (S-ODSs)
激活状态的ODS (A-ODSs)
按钮 A
按钮 B
按钮 C
按钮 D

图52

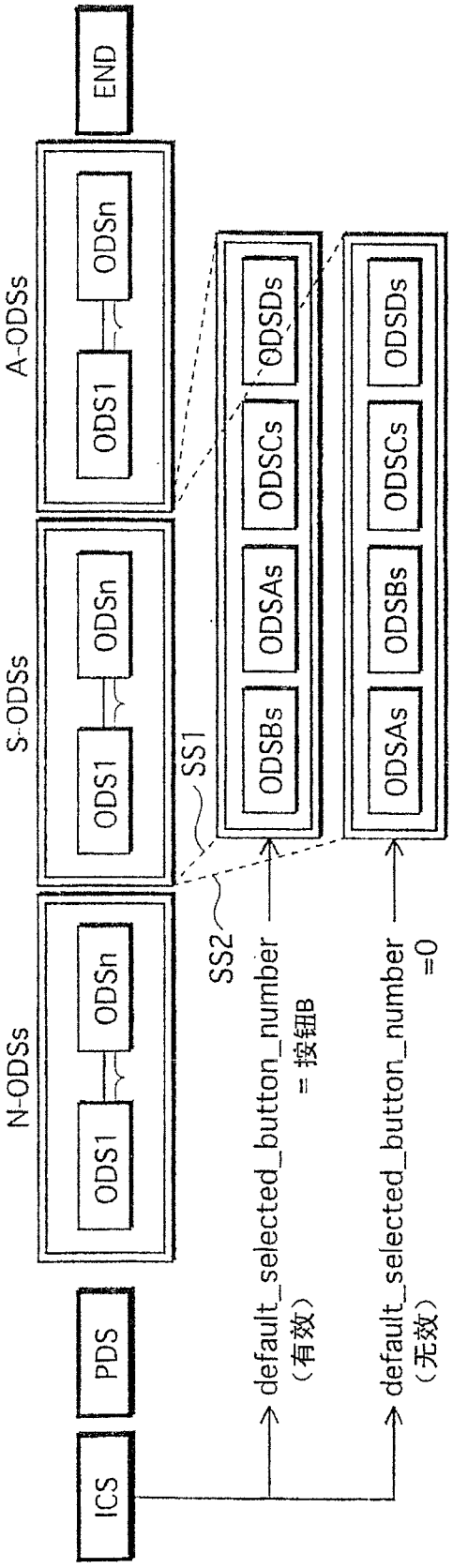

ICS
PDS
N-ODSs
ODS1
ODSn
S-ODSs
ODS1
ODSn
A-ODSs
ODS1
ODSn
END
SS1
SS2
ODSBs
ODSAs
ODSCs
ODSDs
default_selected_button_number
（有效）
= 按钮B
default_selected_button_number
（无效）
=0

图53

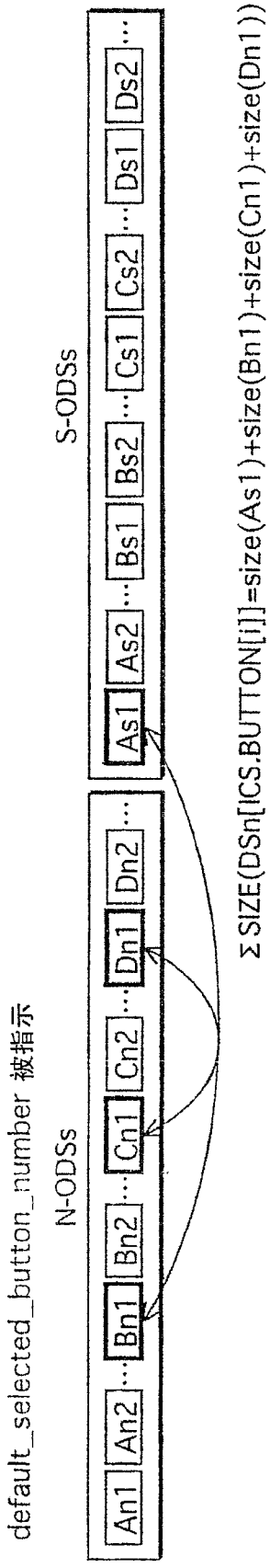


图54A

FIG. 54B

图54B

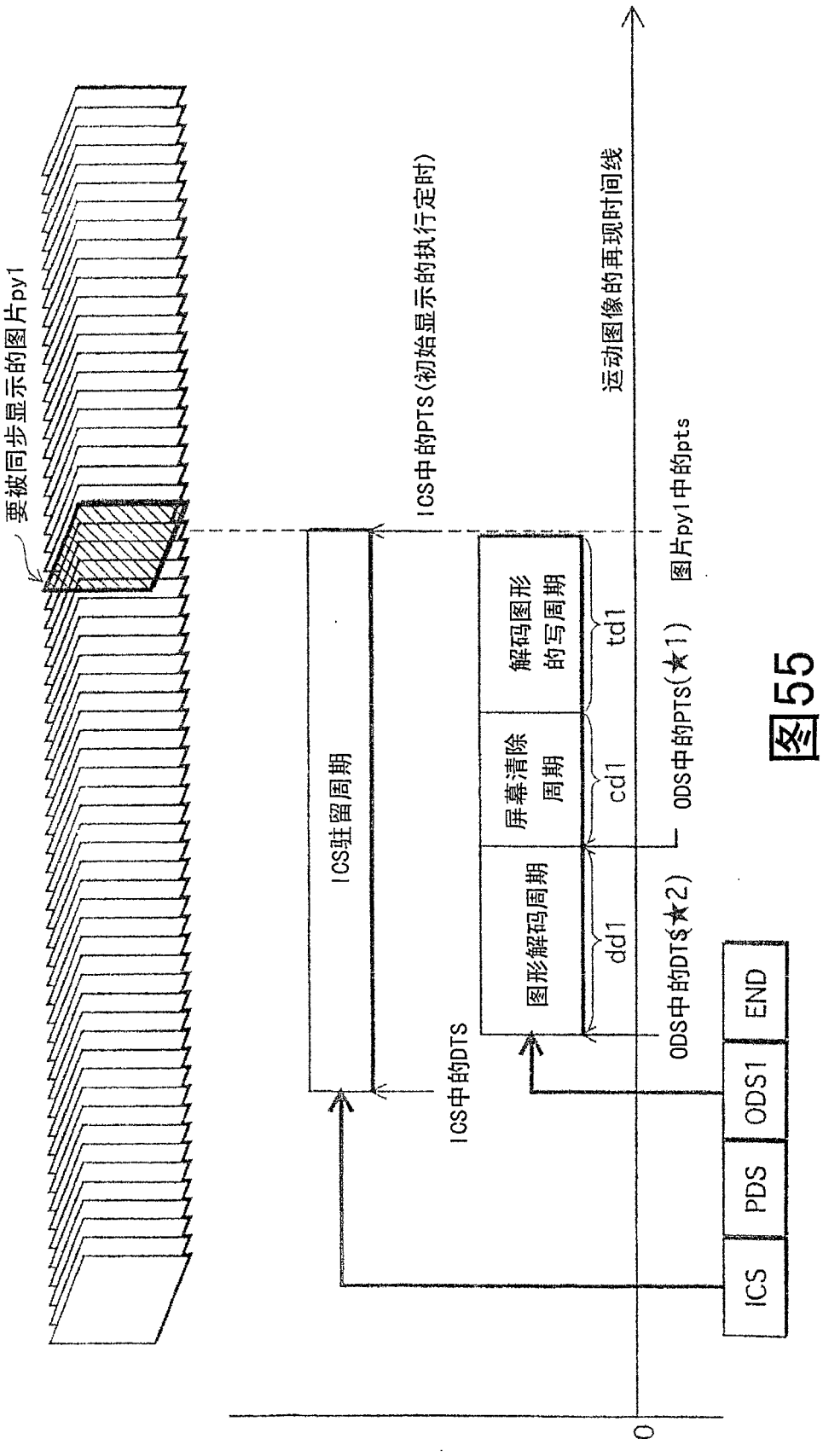

要被同步显示的图片py1
ICS中的PTS(初始显示的执行定时)
运动图像的再现时间线
ICS驻留周期
屏幕清除周期
解码图形的写周期
图形解码周期
td1
cd1
dd1
ODS中的PTS(★1)
图片py1中的pts
ODS中的DTS(★2)
ICS中的DTS
ICS
PDS
ODS1
END
0

图55

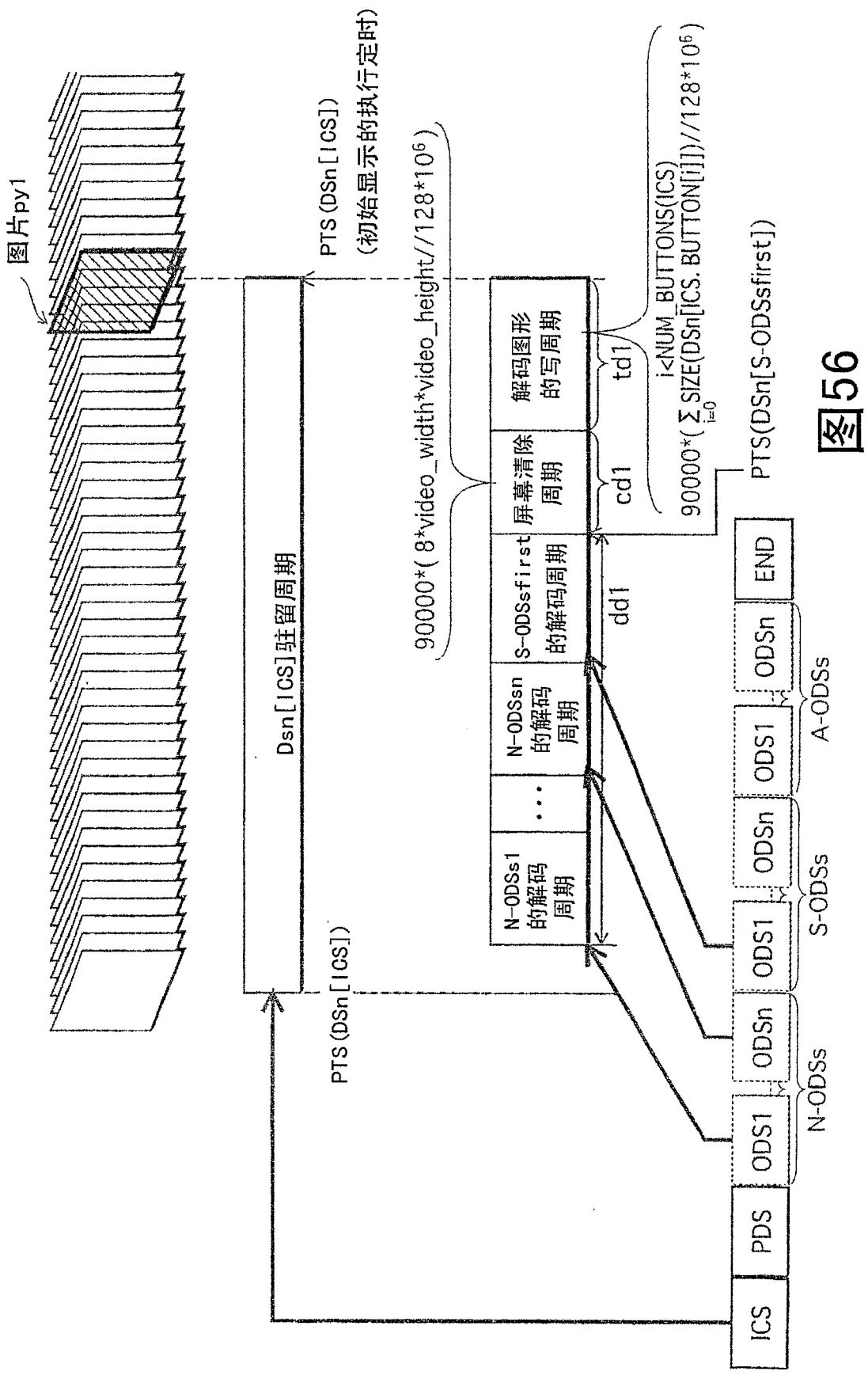

图片py1
PTS(DSn[ICS])
(初始显示的执行定时)
90000*(8*video_width*video_height//128*10^6)
Dsn[ICS]驻留周期
PTS(DSn[ICS])
N-ODSs1的解码周期
...
N-ODSsn的解码周期
S-ODSsfirst的解码周期
屏幕清除周期
解码图形的写周期
dd1
cd1
td1
i<NUM_BUTTONS(ICS)
90000*(Σ SIZE(DSn[ICS. BUTTON[i]])//128*10^6)
PTS(DSn[S-ODSsfirst])
ICS
PDS
ODS1
ODSn
ODS1
ODSn
ODS1
ODSn
END
N-ODSs
S-ODSs
A-ODSs

图56

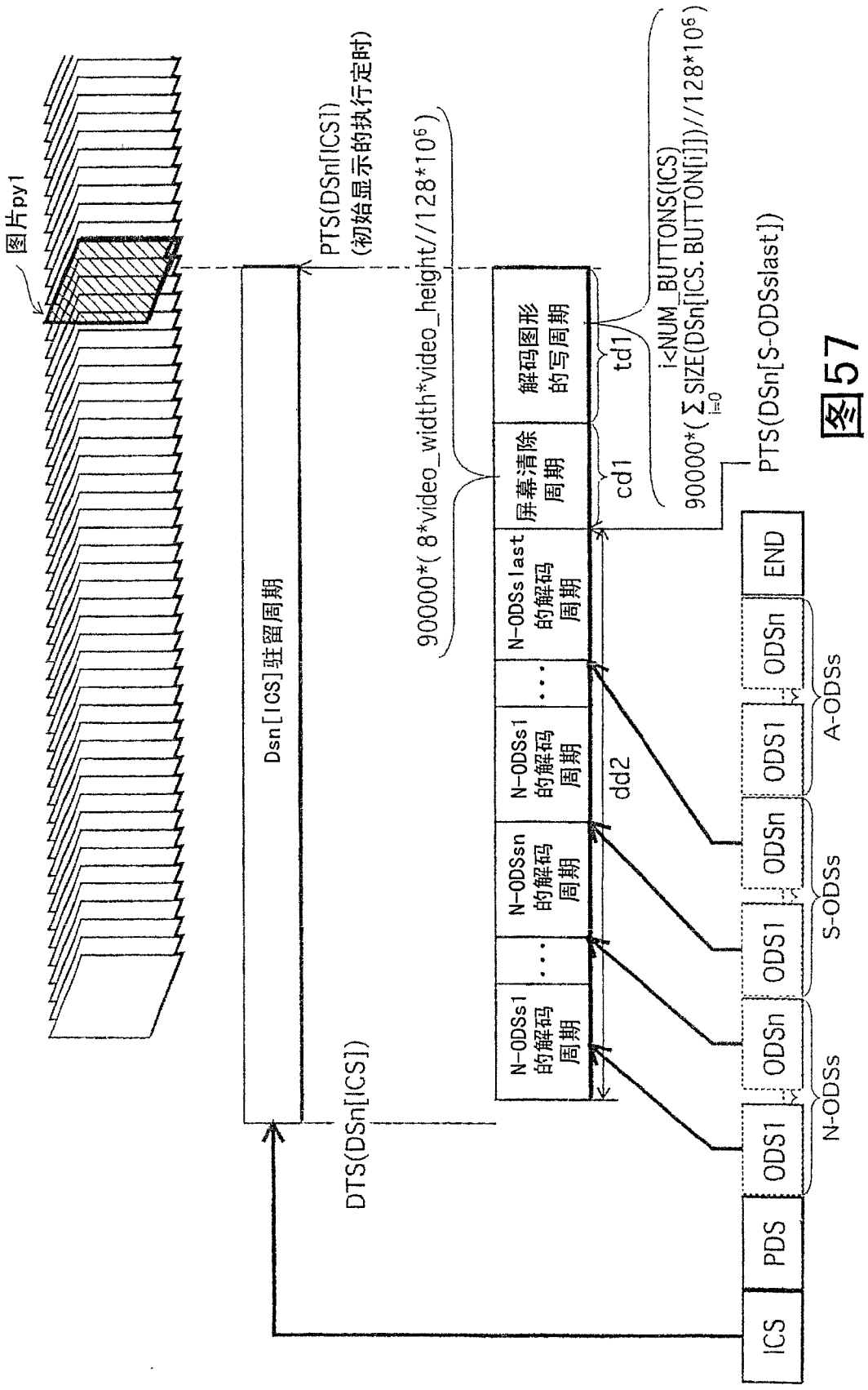

图片py1
PTS(DSn[ICS1])
(初始显示的执行定时)
90000*(8*video_width*video_height//128*10^6)
Dsn[ICS]驻留周期
DTS(DSn[ICS])
N-ODSs1 的解码周期
N-ODSsn 的解码周期
N-ODSs1 的解码周期
N-ODSslast 的解码周期
屏幕清除周期
解码图形的写周期
dd2
cd1
td1
i<NUM_BUTTONS(ICS)
90000*(Σ SIZE(DSn[ICS. BUTTON[i]])//128*10^6)
i=0
PTS(DSn[S-ODSslast])
ICS
PDS
ODS1
ODSn
ODS1
ODSn
ODS1
ODSn
END
N-ODSs
S-ODSs
A-ODSs

图57

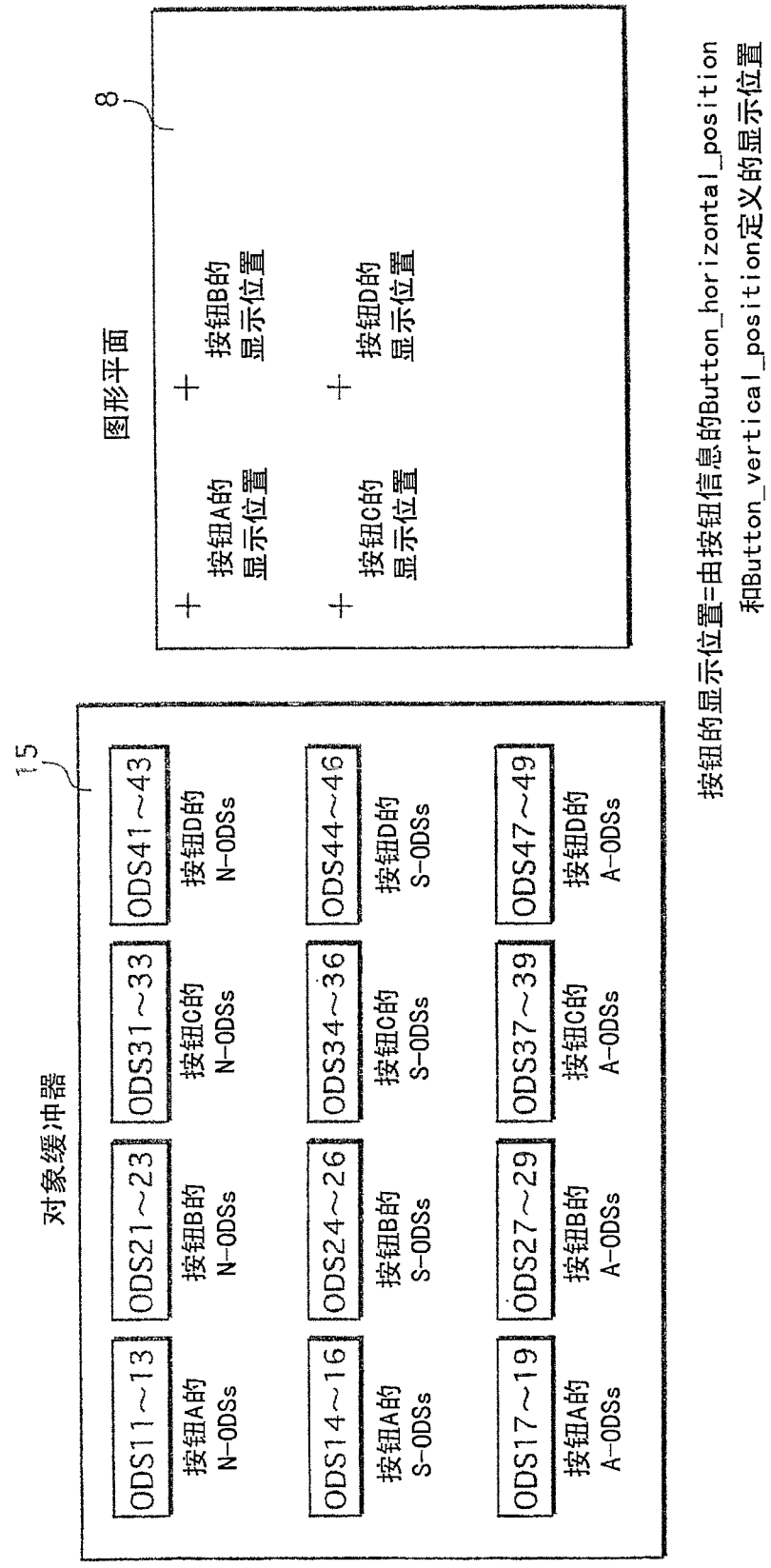

15
对象缓冲器
ODS11～13
按钮A的
N-ODSs
ODS21～23
按钮B的
N-ODSs
ODS31～33
按钮C的
N-ODSs
ODS41～43
按钮D的
N-ODSs
ODS14～16
按钮A的
S-ODSs
ODS24～26
按钮B的
S-ODSs
ODS34～36
按钮C的
S-ODSs
ODS44～46
按钮D的
S-ODSs
ODS17～19
按钮A的
A-ODSs
ODS27～29
按钮B的
A-ODSs
ODS37～39
按钮C的
A-ODSs
ODS47～49
按钮D的
A-ODSs
8
图形平面
+
按钮A的
显示位置
+
按钮B的
显示位置
+
按钮C的
显示位置
+
按钮D的
显示位置
按钮的显示位置=由按钮信息的Button_horizontal_position
和Button_vertical_position定义的显示位置

图58

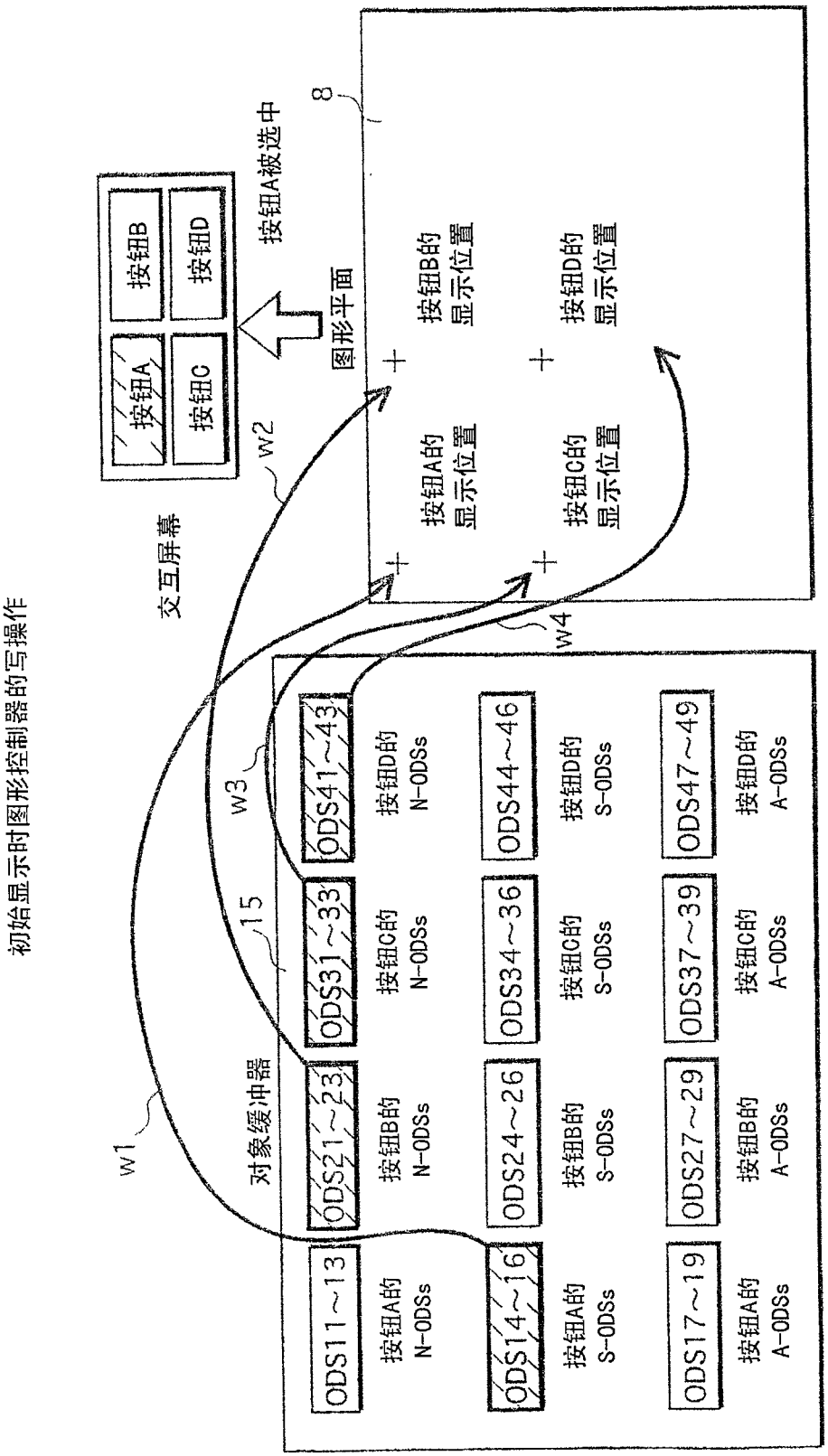

初始显示时图形控制器的写操作
交互屏幕
按钮A
按钮B
按钮C
按钮D
按钮A被选中
图形平面
8
w1
w2
w3
w4
15
对象缓冲器
ODS11～13
按钮A的 N-ODSs
ODS21～23
按钮B的 N-ODSs
ODS31～33
按钮C的 N-ODSs
ODS41～43
按钮D的 N-ODSs
ODS14～16
按钮A的 S-ODSs
ODS24～26
按钮B的 S-ODSs
ODS34～36
按钮C的 S-ODSs
ODS44～46
按钮D的 S-ODSs
ODS17～19
按钮A的 A-ODSs
ODS27～29
按钮B的 A-ODSs
ODS37～39
按钮C的 A-ODSs
ODS47～49
按钮D的 A-ODSs
按钮A的显示位置
按钮B的显示位置
按钮C的显示位置
按钮D的显示位置
按钮的显示位置=由按钮信息的Button_horizontal_position
和Button_vertical_position定义的显示位置

图59

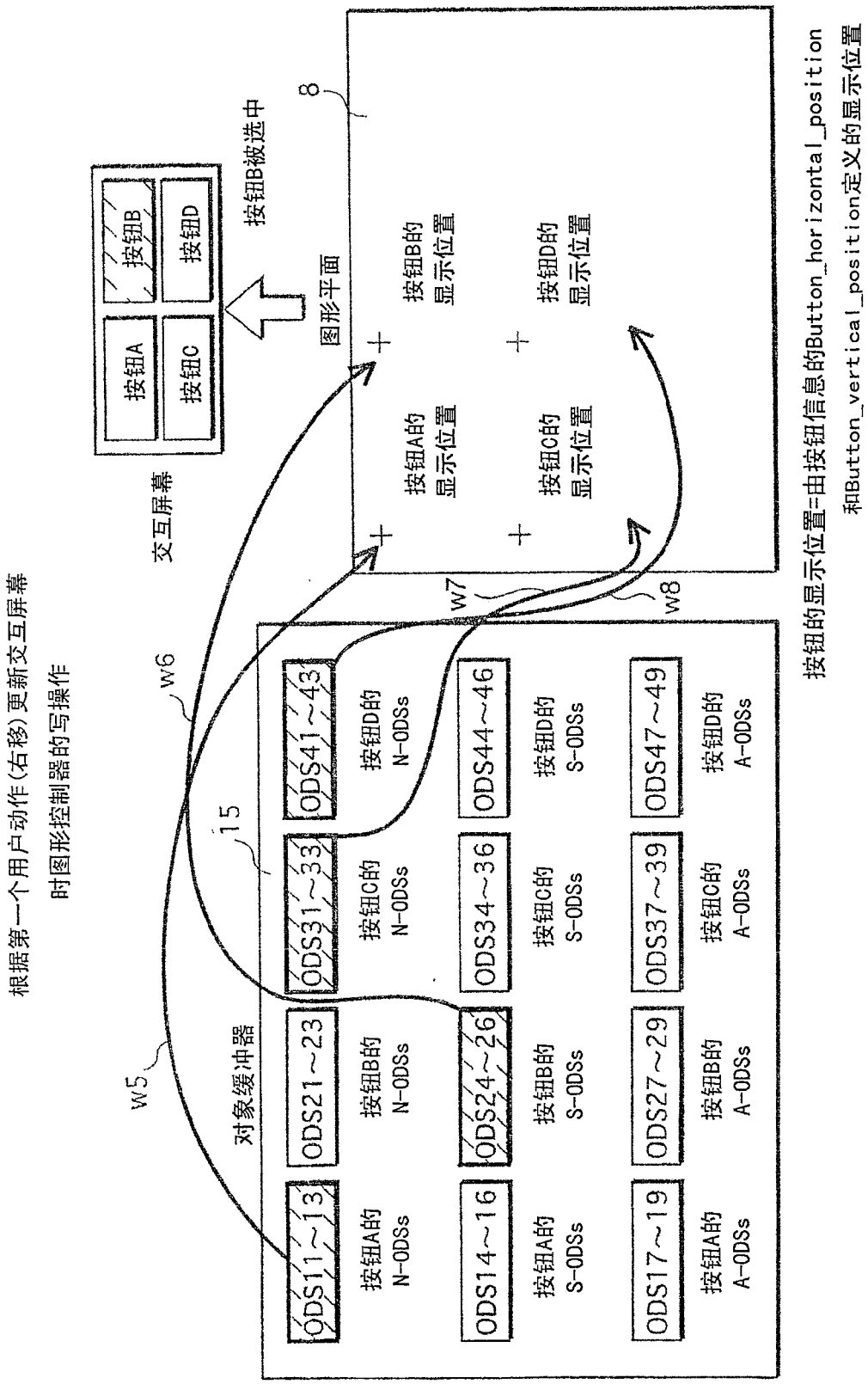

根据第一个用户动作(右移)更新交互屏幕
时图形控制器的写操作
交互屏幕
按钮A
按钮B
按钮C
按钮D
按钮B被选中
图形平面
8
按钮A的
显示位置
按钮B的
显示位置
按钮C的
显示位置
按钮D的
显示位置
w5
w6
w7
w8
15
对象缓冲器
ODS11～13
ODS21～23
ODS31～33
ODS41～43
ODS14～16
ODS24～26
ODS34～36
ODS44～46
ODS17～19
ODS27～29
ODS37～39
ODS47～49
按钮A的
N-ODSs
按钮B的
N-ODSs
按钮C的
N-ODSs
按钮D的
N-ODSs
按钮A的
S-ODSs
按钮B的
S-ODSs
按钮C的
S-ODSs
按钮D的
S-ODSs
按钮A的
A-ODSs
按钮B的
A-ODSs
按钮C的
A-ODSs
按钮D的
A-ODSs
按钮的显示位置=由按钮信息的Button_horizontal_position
和Button_vertical_position定义的显示位置

图60

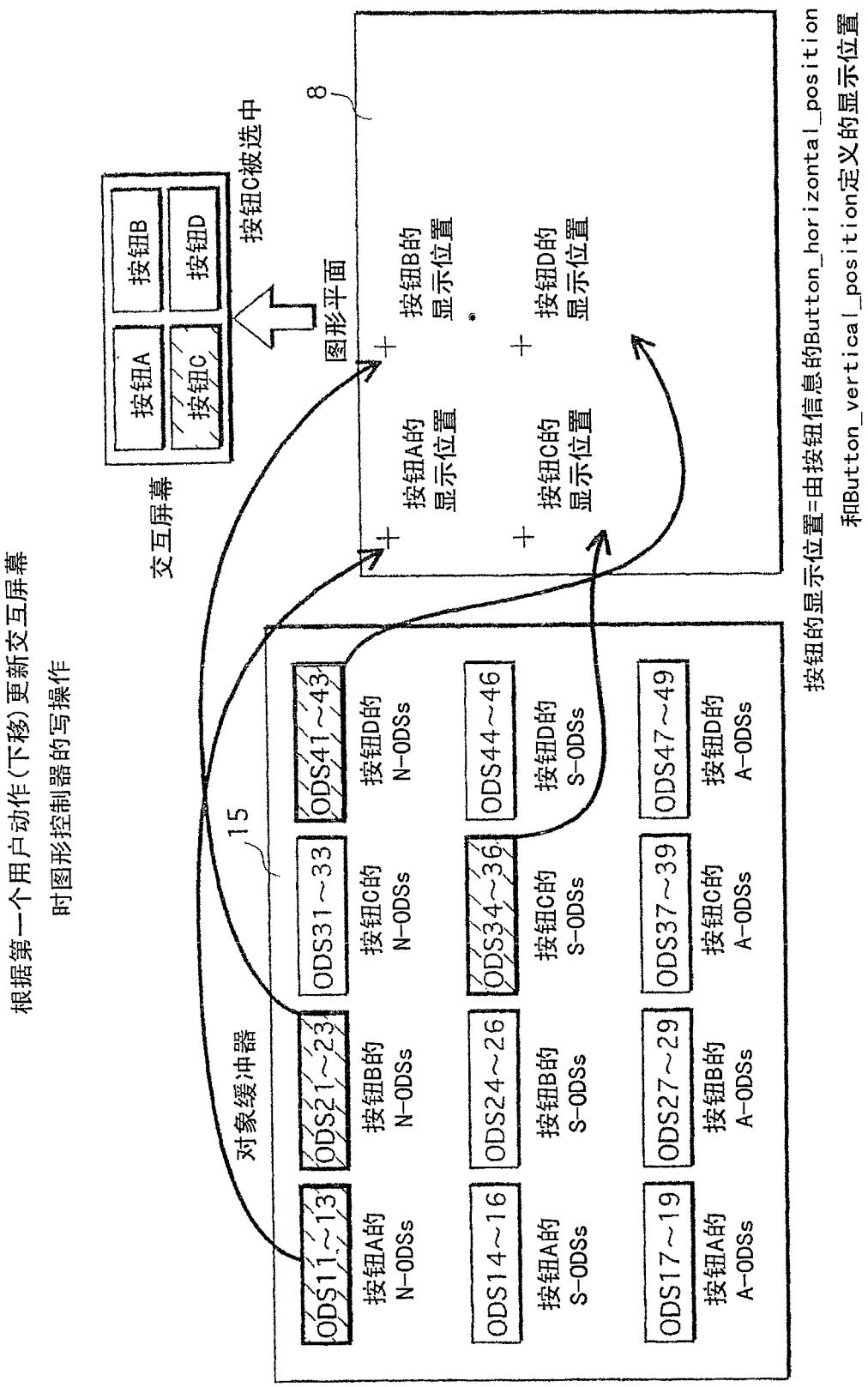

根据第一个用户动作(下移)更新交互屏幕
时图形控制器的写操作
交互屏幕
按钮A
按钮B
按钮C
按钮D
按钮C被选中
图形平面
8
按钮A的
显示位置
按钮B的
显示位置
按钮C的
显示位置
按钮D的
显示位置
15
对象缓冲器
ODS11～13
ODS21～23
ODS31～33
ODS41～43
ODS14～16
ODS24～26
ODS34～36
ODS44～46
ODS17～19
ODS27～29
ODS37～39
ODS47～49
按钮A的
N-ODSs
按钮B的
N-ODSs
按钮C的
N-ODSs
按钮D的
N-ODSs
按钮A的
S-ODSs
按钮B的
S-ODSs
按钮C的
S-ODSs
按钮D的
S-ODSs
按钮A的
A-ODSs
按钮B的
A-ODSs
按钮C的
A-ODSs
按钮D的
A-ODSs
按钮的显示位置=由按钮信息的Button_horizontal_position
和Button_vertical_position定义的显示位置

图61

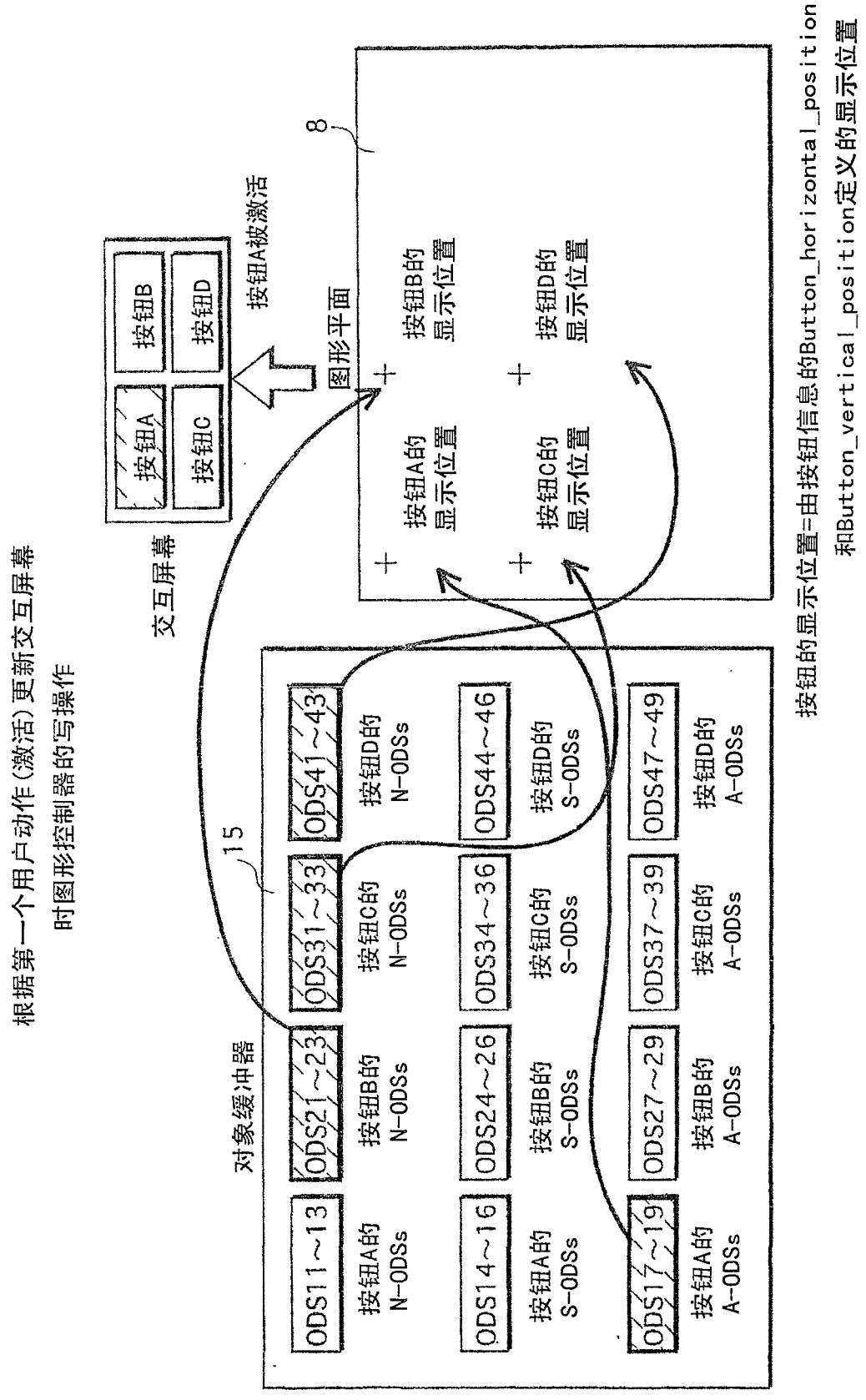

根据第一个用户动作(激活)更新交互屏幕
时图形控制器的写操作
交互屏幕
按钮A
按钮B
按钮C
按钮D
按钮A被激活
8
图形平面
按钮A的
显示位置
按钮B的
显示位置
按钮C的
显示位置
按钮D的
显示位置
15
对象缓冲器
ODS11～13
按钮A的
N-ODSs
ODS21～23
按钮B的
N-ODSs
ODS31～33
按钮C的
N-ODSs
ODS41～43
按钮D的
N-ODSs
ODS14～16
按钮A的
S-ODSs
ODS24～26
按钮B的
S-ODSs
ODS34～36
按钮C的
S-ODSs
ODS44～46
按钮D的
S-ODSs
ODS17～19
按钮A的
A-ODSs
ODS27～29
按钮B的
A-ODSs
ODS37～39
按钮C的
A-ODSs
ODS47～49
按钮D的
A-ODSs
按钮的显示位置=由按钮信息的Button_horizontal_position
和Button_vertical_position定义的显示位置

图62

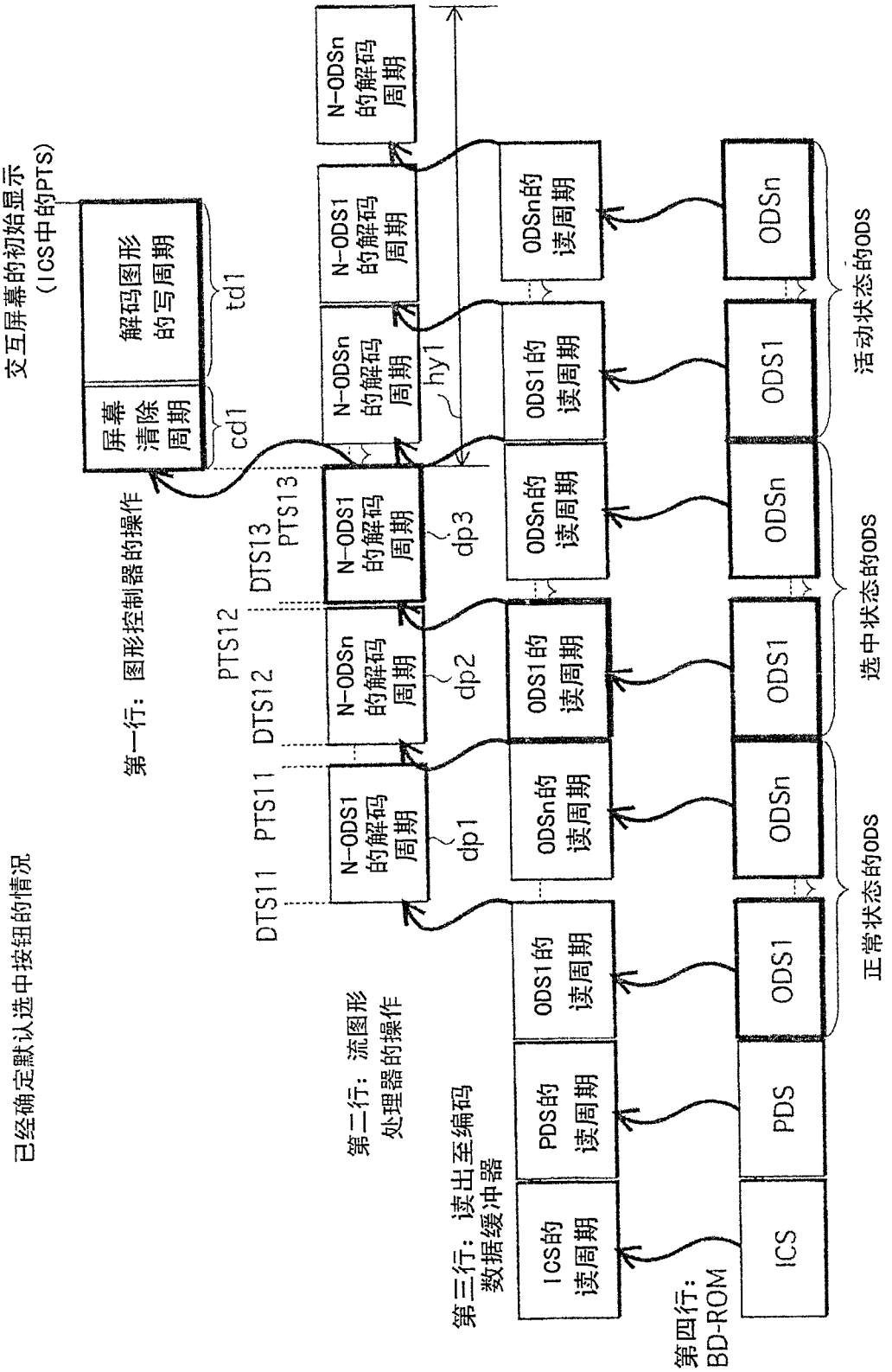

已经确定默认选中按钮的情况
交互屏幕的初始显示
(ICS中的PTS)
第一行：图形控制器的操作
屏幕清除周期
解码图形的写周期
cd1
td1
DTS11
PTS11
DTS12
PTS12
DTS13
PTS13
N-ODS1的解码周期
N-ODSn的解码周期
dp1
dp2
dp3
hy1
第二行：流图形处理器的操作
第三行：读出至编码数据缓冲器
第四行：BD-ROM
ICS的读周期
PDS的读周期
ODS1的读周期
ODSn的读周期
ICS
PDS
ODS1
ODSn
正常状态的ODS
选中状态的ODS
活动状态的ODS

图63

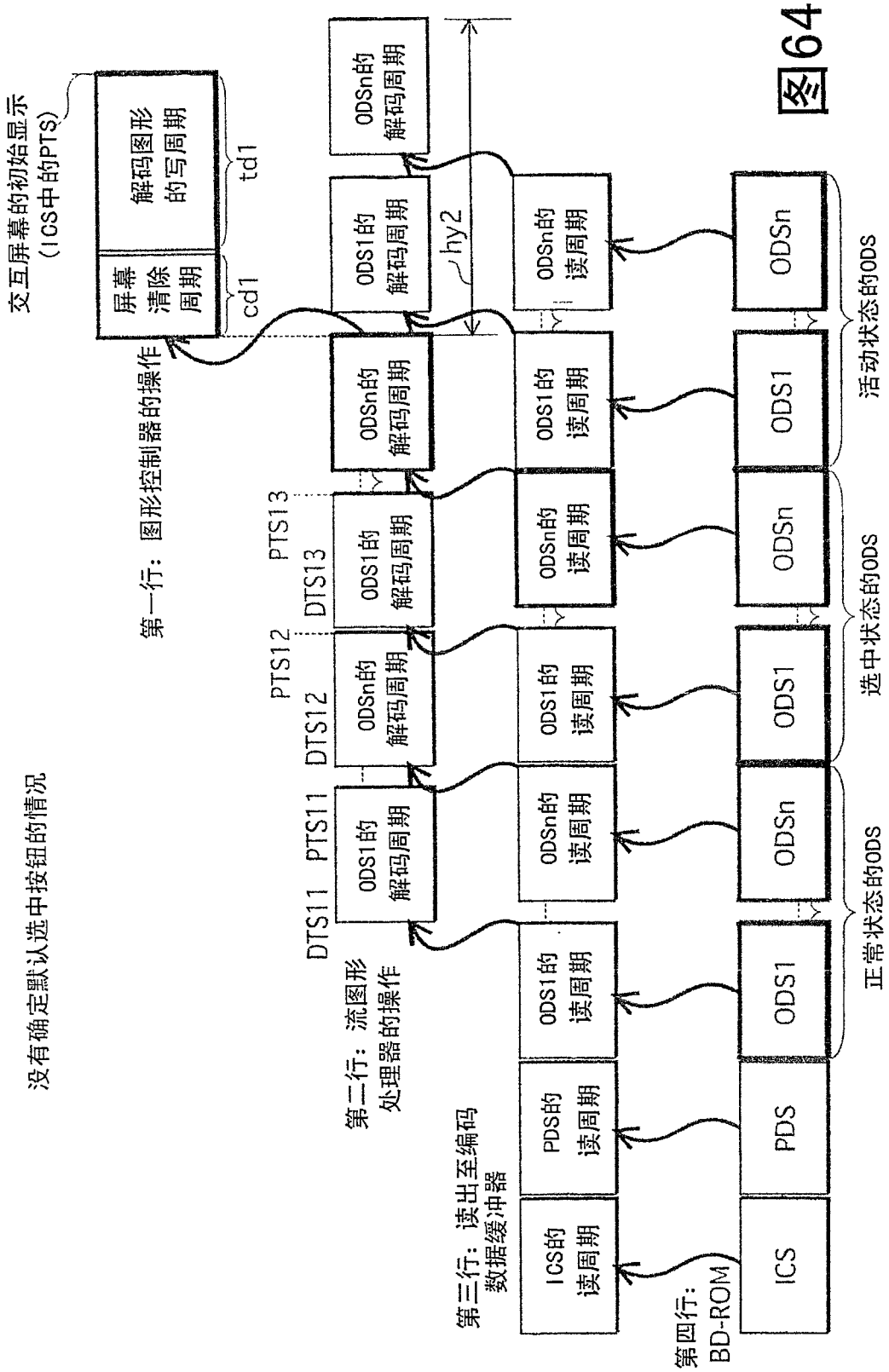

没有确定默认选中按钮的情况
交互屏幕的初始显示
(ICS中的PTS)
第一行：图形控制器的操作
第二行：流图形
处理器的操作
第三行：读出至编码
数据缓冲器
第四行：
BD-ROM
屏幕
清除
周期
解码图形
的写周期
cd1
td1
hy2
DTS11
PTS11
DTS12
PTS12
DTS13
PTS13
ODS1的
解码周期
ODSn的
解码周期
ODS1的
读周期
ODSn的
读周期
ICS的
读周期
PDS的
读周期
ICS
PDS
ODS1
ODSn
正常状态的ODS
选中状态的ODS
活动状态的ODS

图64

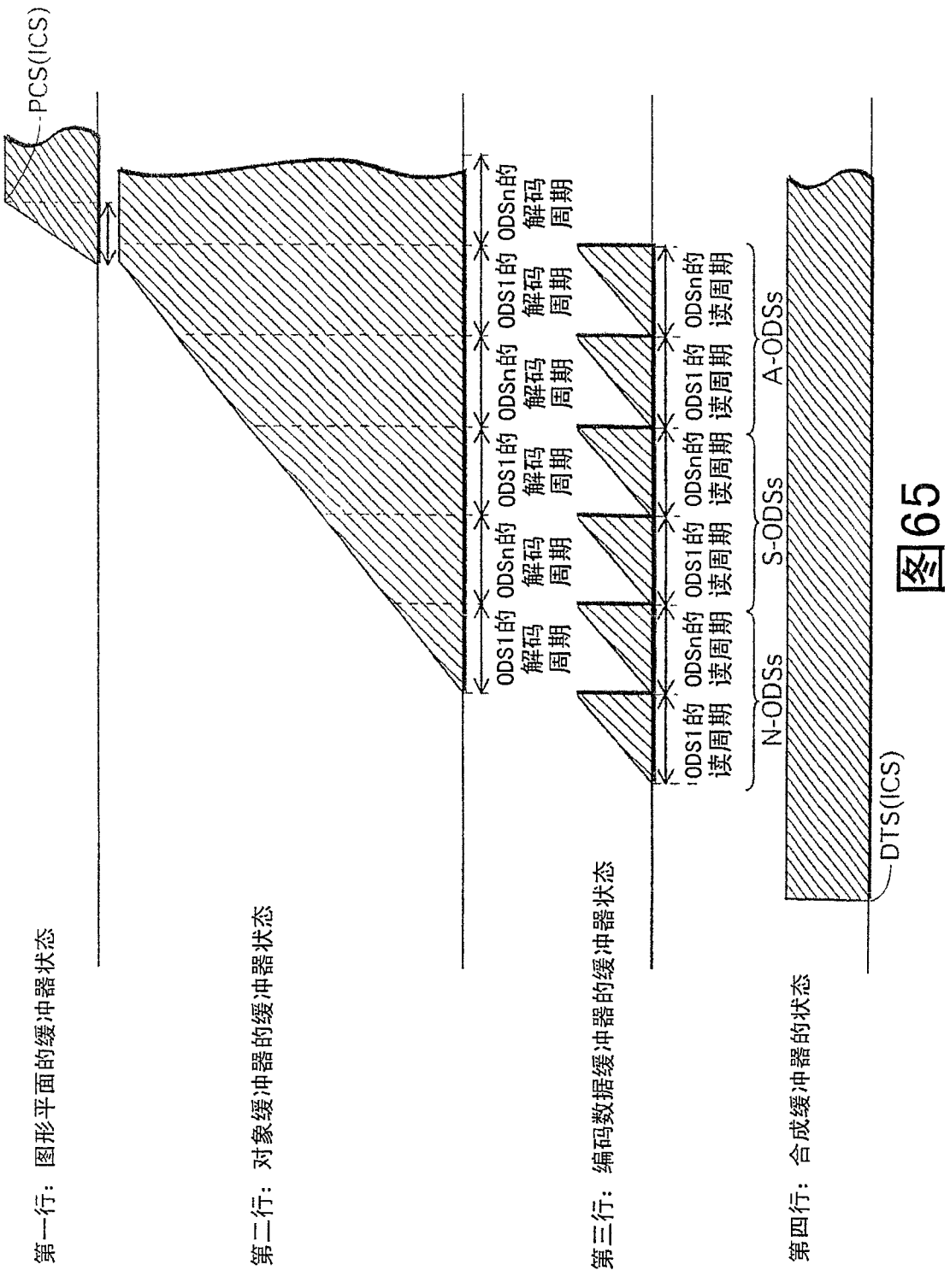

PCS(ICS)
第一行：图形平面的缓冲器状态
第二行：对象缓冲器的缓冲器状态
第三行：编码数据缓冲器的缓冲器状态
第四行：合成缓冲器的状态
ODS1的 解码 周期
ODSn的 解码 周期
ODS1的 读周期
ODSn的 读周期
N-ODSs
S-ODSs
A-ODSs
DTS(ICS)

图65

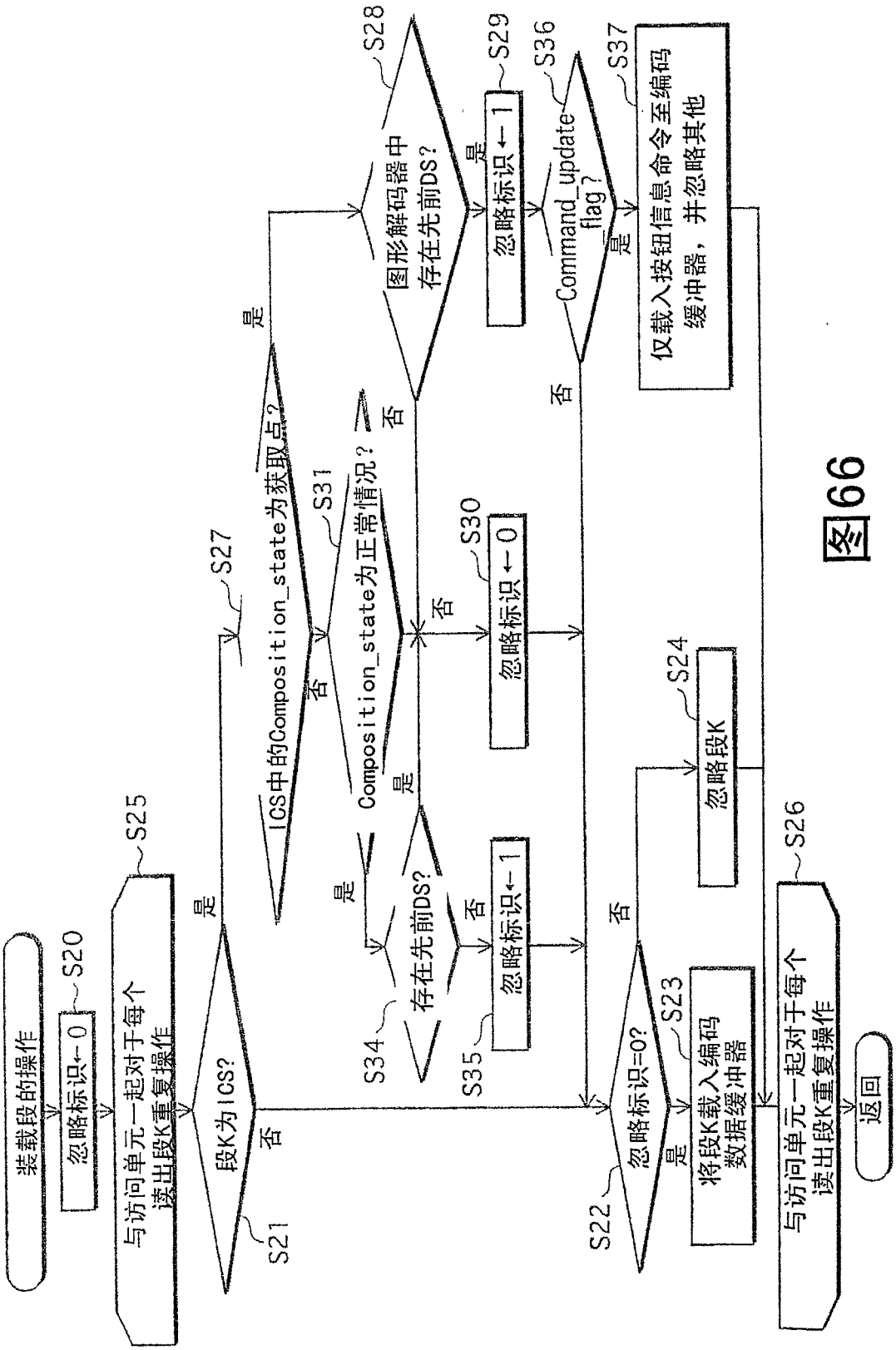

装载段的操作
忽略标识← 0
S20
与访问单元一起对于每个
读出段K重复操作
S25
段K为ICS?
S21
是
否
ICS中的Composition_state为获取点?
S27
是
否
Composition_state为正常情况?
S31
是
否
存在先前DS?
S34
是
否
忽略标识← 1
S35
忽略标识← 0
S30
图形解码器中
存在先前DS?
S28
是
否
忽略标识← 1
S29
Command_update
flag?
S36
是
否
仅载入按钮信息命令至编码
缓冲器，并忽略其他
S37
忽略标识=0?
S22
是
否
将段K载入编码
数据缓冲器
S23
忽略段K
S24
与访问单元一起对于每个
读出段K重复操作
S26
返回

图66

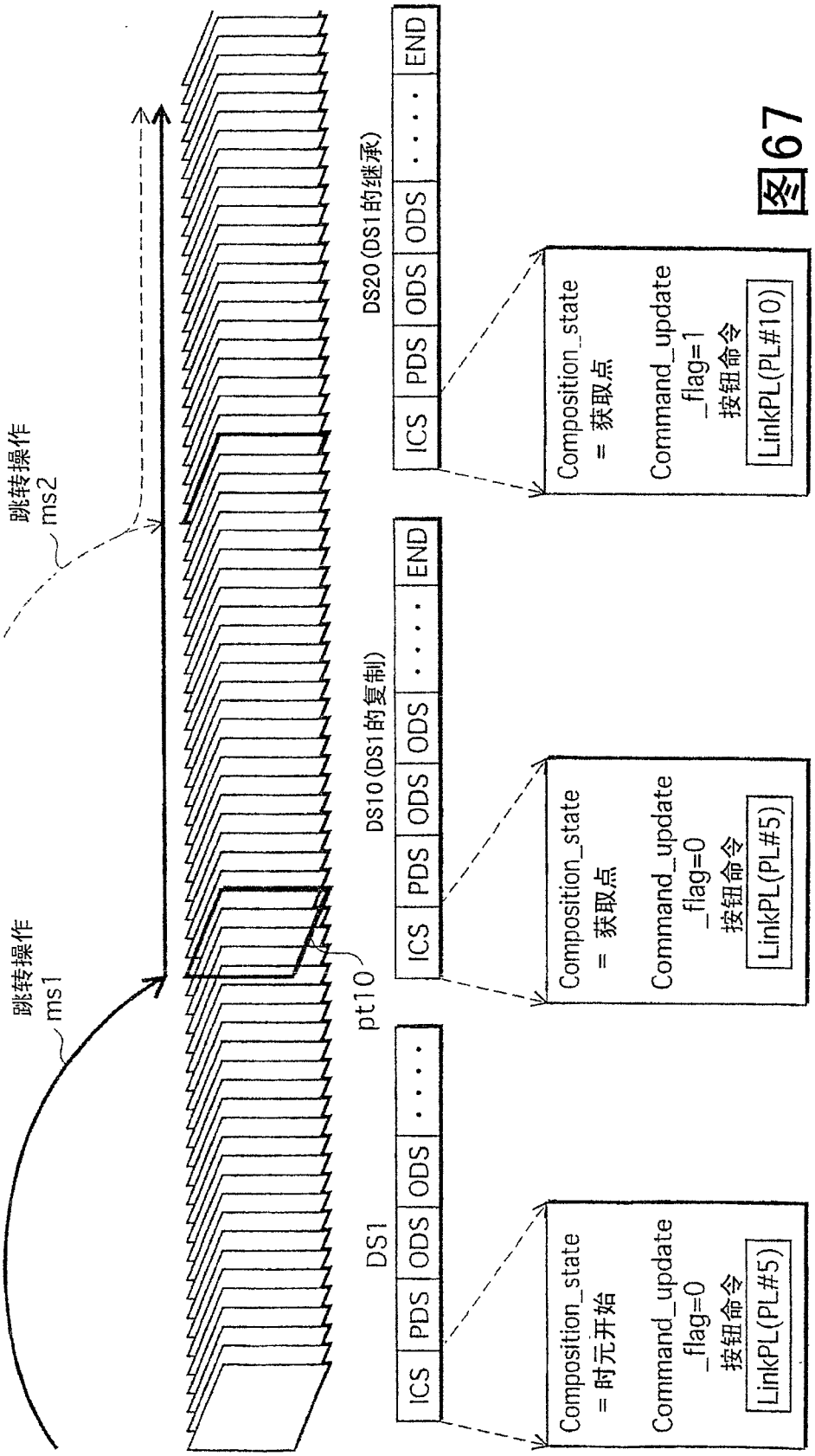

跳转操作
ms1
跳转操作
ms2
pt10
DS1
ICS
PDS
ODS
ODS
· · · ·
Composition_state
= 时元开始
Command_update
_flag=0
按钮命令
LinkPL(PL#5)
DS10 (DS1的复制)
ICS
PDS
ODS
ODS
· · · ·
END
Composition_state
= 获取点
Command_update
_flag=0
按钮命令
LinkPL(PL#5)
DS20 (DS1的继承)
ICS
PDS
ODS
ODS
· · · ·
END
Composition_state
= 获取点
Command_update
_flag=1
按钮命令
LinkPL(PL#10)

图67

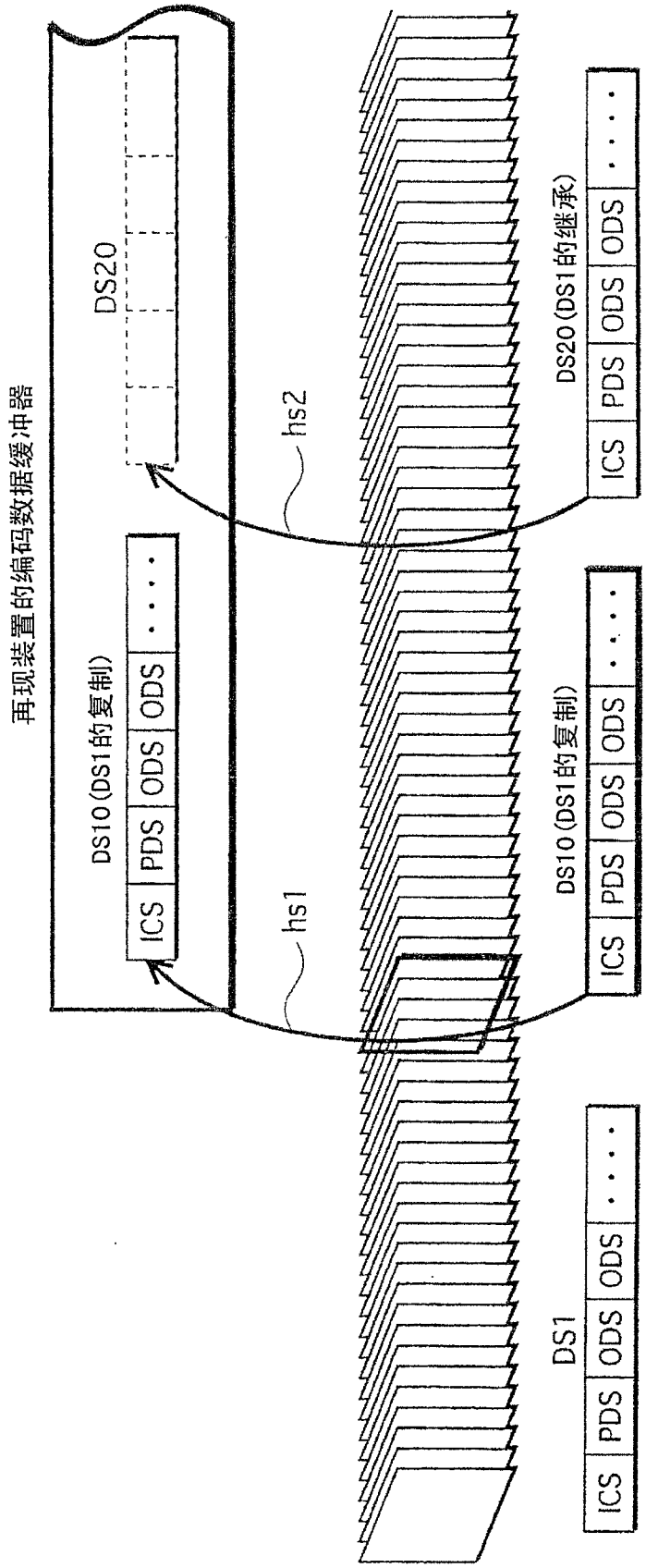

再现装置的编码数据缓冲器
DS20
DS10(DS1的复制)
ICS
PDS
ODS
ODS
· · · ·
hs1
hs2
DS20(DS1的继承)
DS10(DS1的复制)
DS1

图68

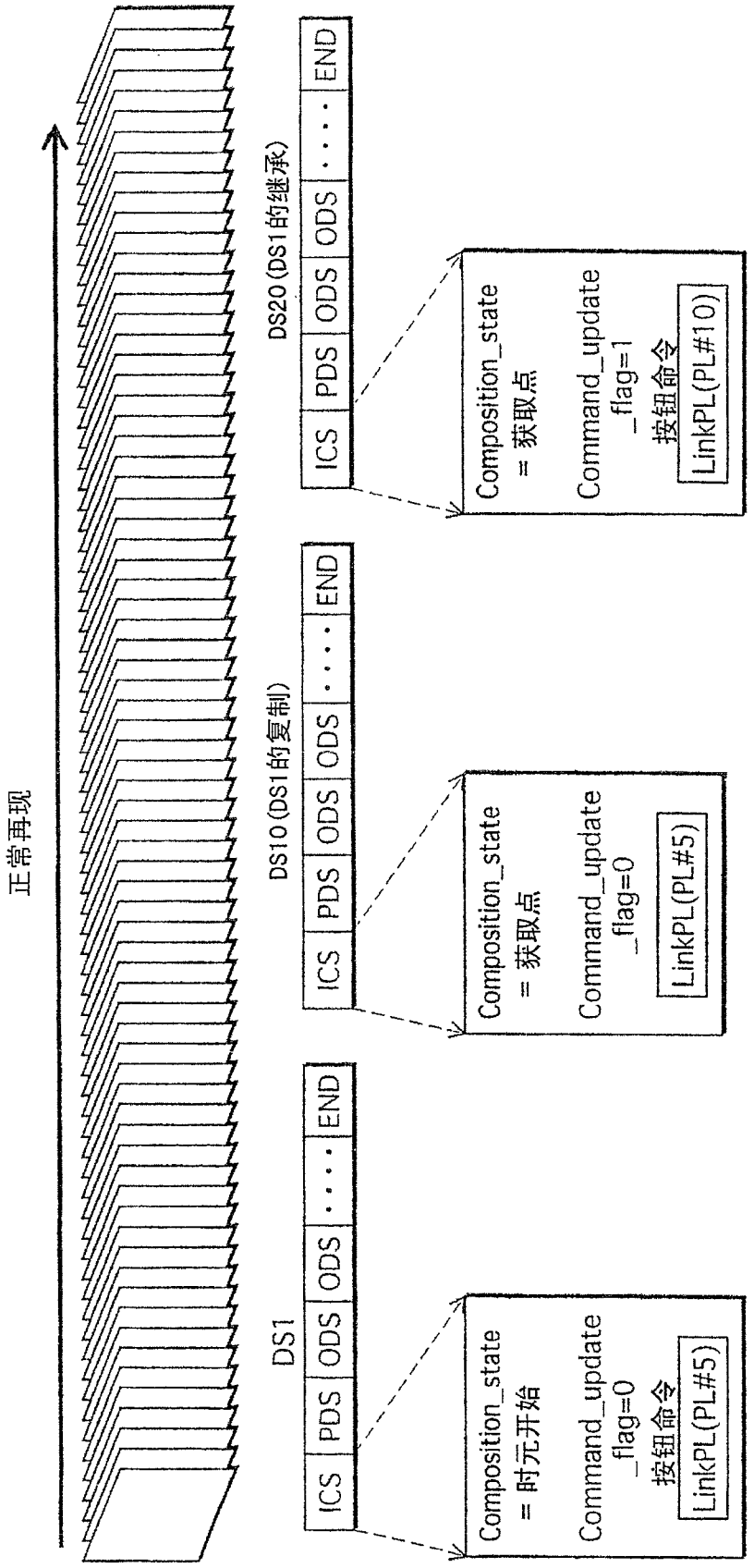

正常再现
DS1
ICS
PDS
ODS
ODS
· · · ·
END
Composition_state
= 时元开始
Command_update
_flag=0
按钮命令
LinkPL(PL#5)
DS10 (DS1的复制)
ICS
PDS
ODS
ODS
· · · ·
END
Composition_state
= 获取点
Command_update
_flag=0
LinkPL(PL#5)
DS20 (DS1的继承)
ICS
PDS
ODS
ODS
· · · ·
END
Composition_state
= 获取点
Command_update
_flag=1
按钮命令
LinkPL(PL#10)

图69

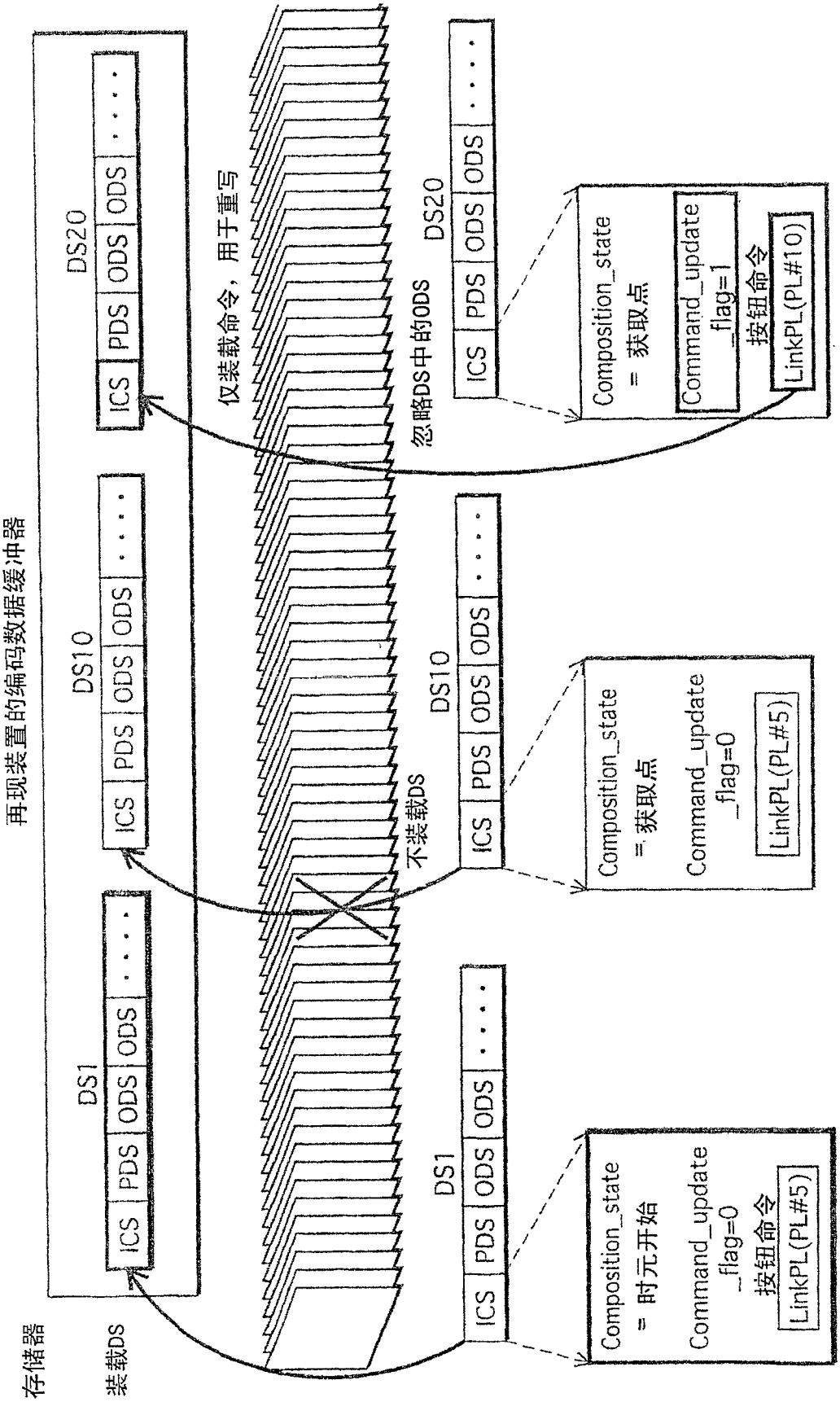

再现装置的编码数据缓冲器
存储器
装载DS
DS1
ICS
PDS
ODS
ODS
DS10
ICS
PDS
ODS
ODS
DS20
ICS
PDS
ODS
ODS
仅装载命令，用于重写
不装载DS
忽略DS中的ODS
Composition_state
= 时元开始
Command_update
_flag=0
按钮命令
LinkPL(PL#5)
Composition_state
= 获取点
Command_update
_flag=0
LinkPL(PL#5)
Composition_state
= 获取点
Command_update
_flag=1
按钮命令
LinkPL(PL#10)

图70

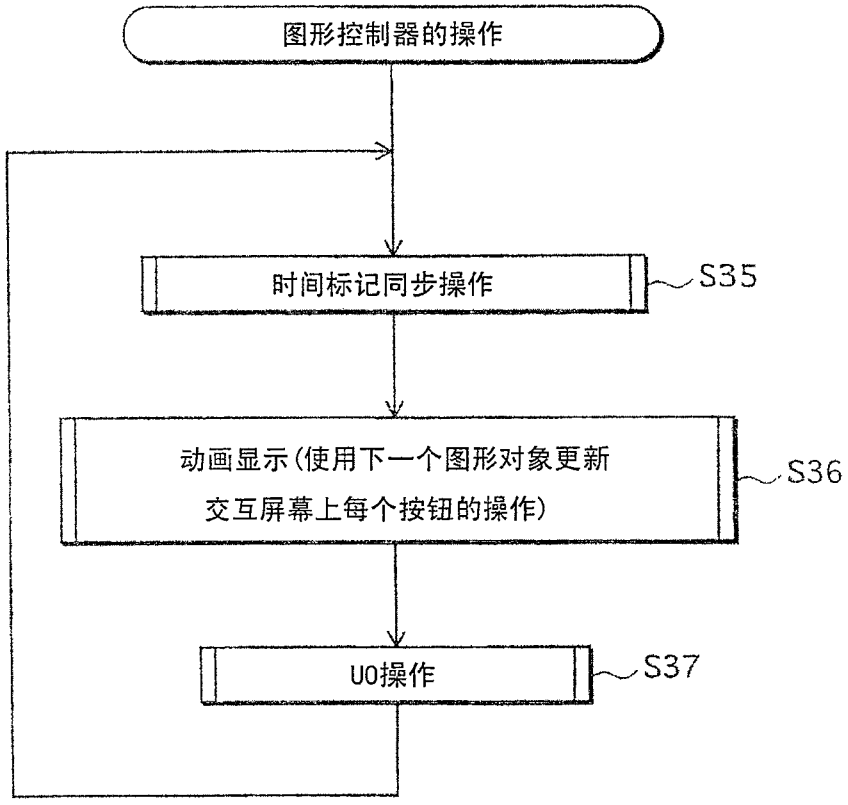

图形控制器的操作
时间标记同步操作
S35
动画显示(使用下一个图形对象更新
交互屏幕上每个按钮的操作)
S36
UO操作
S37

图71

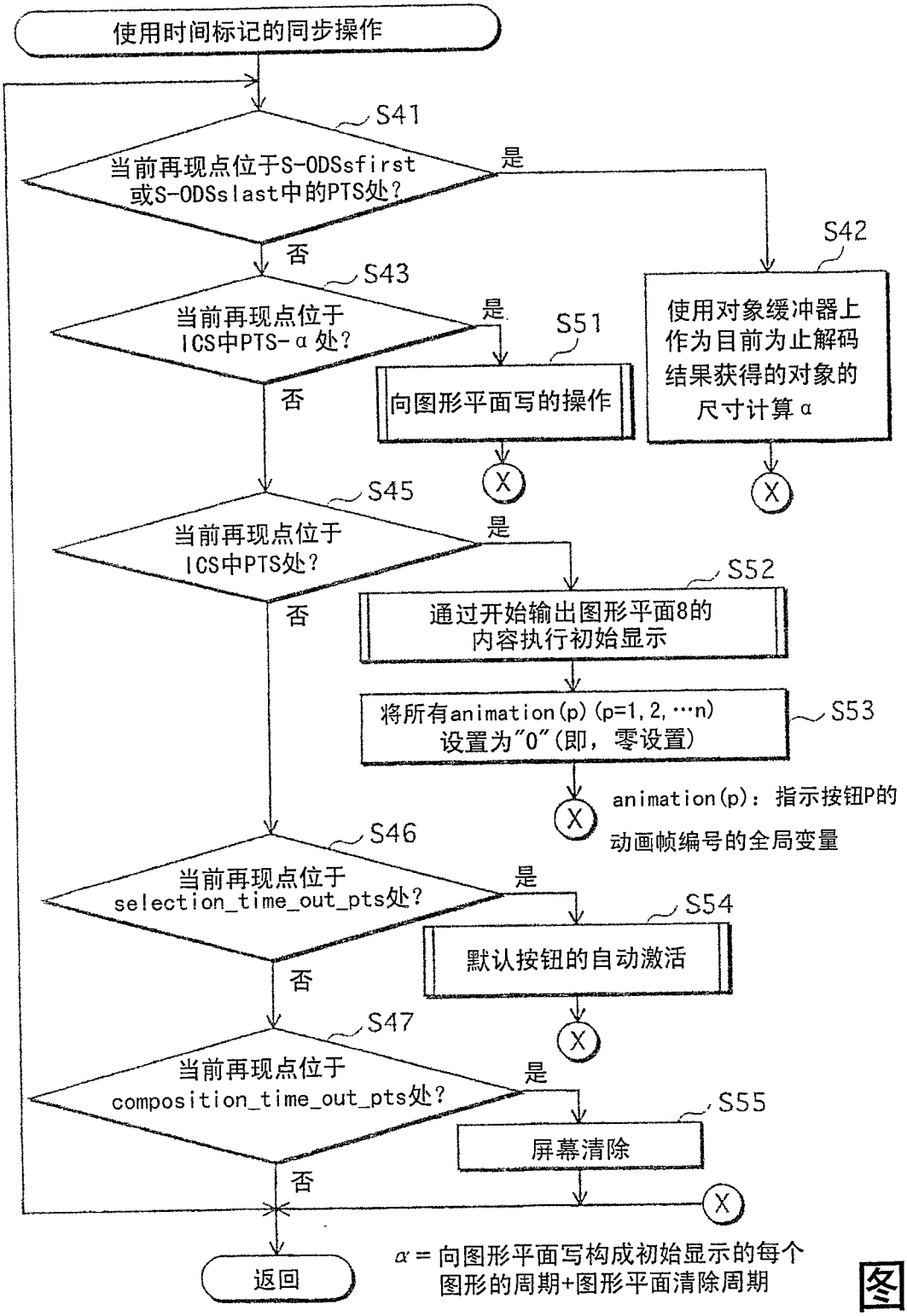

使用时间标记的同步操作
S41
当前再现点位于S-ODSsfirst或S-ODSslast中的PTS处?
是
否
S42
使用对象缓冲器上作为目前为止解码结果获得的对象的尺寸计算α
S43
当前再现点位于ICS中PTS-α处?
是
否
S51
向图形平面写的操作
X
X
S45
当前再现点位于ICS中PTS处?
是
否
S52
通过开始输出图形平面8的内容执行初始显示
将所有animation(p)(p=1,2,…n)设置为"0"(即，零设置)
S53
X
animation(p)：指示按钮P的动画帧编号的全局变量
S46
当前再现点位于selection_time_out_pts处?
是
否
S54
默认按钮的自动激活
X
S47
当前再现点位于composition_time_out_pts处?
是
否
S55
屏幕清除
X
返回
α＝向图形平面写构成初始显示的每个图形的周期+图形平面清除周期

图72

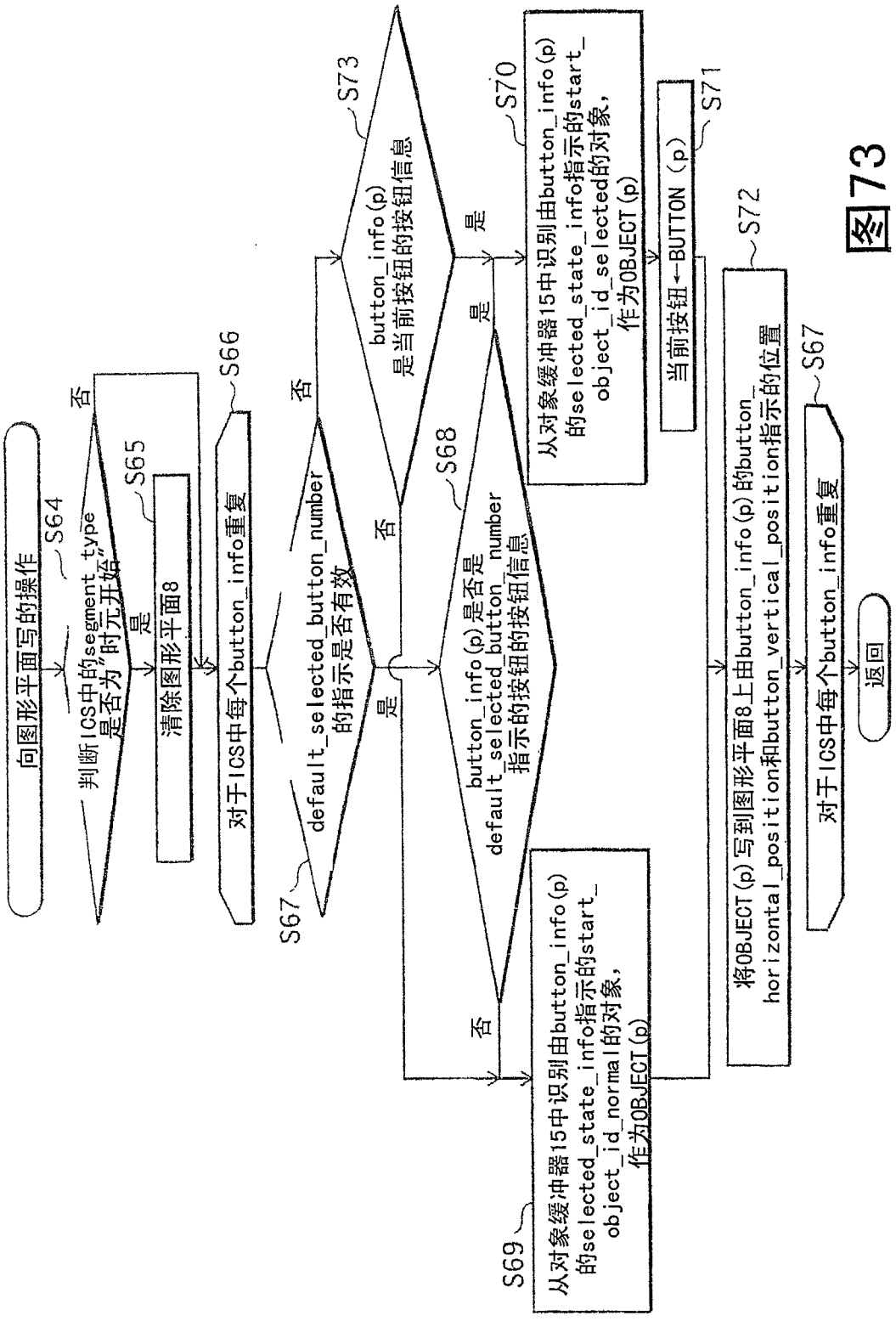

向图形平面写的操作
S64
判断ICS中的segment_type是否为"时元开始"
否
是
S65
清除图形平面8
S66
对于ICS中每个button_info重复
S67
default_selected_button_number的指示是否有效
否
是
S73
button_info(p)是当前按钮的按钮信息
否
是
S68
button_info(p)是否是default_selected_button_number指示的按钮信息
否
是
S70
从对象缓冲器15中识别由button_info(p)的selected_state_info指示的start_object_id_selected的对象，作为OBJECT(p)
S71
当前按钮←BUTTON（p）
S72
将OBJECT(p)写到图形平面8上由button_info(p)的button_horizontal_position和button_vertical_position指示的位置
S69
从对象缓冲器15中识别由button_info(p)的selected_state_info指示的start_object_id_normal的对象，作为OBJECT(p)
S67
对于ICS中每个button_info重复
返回

图73

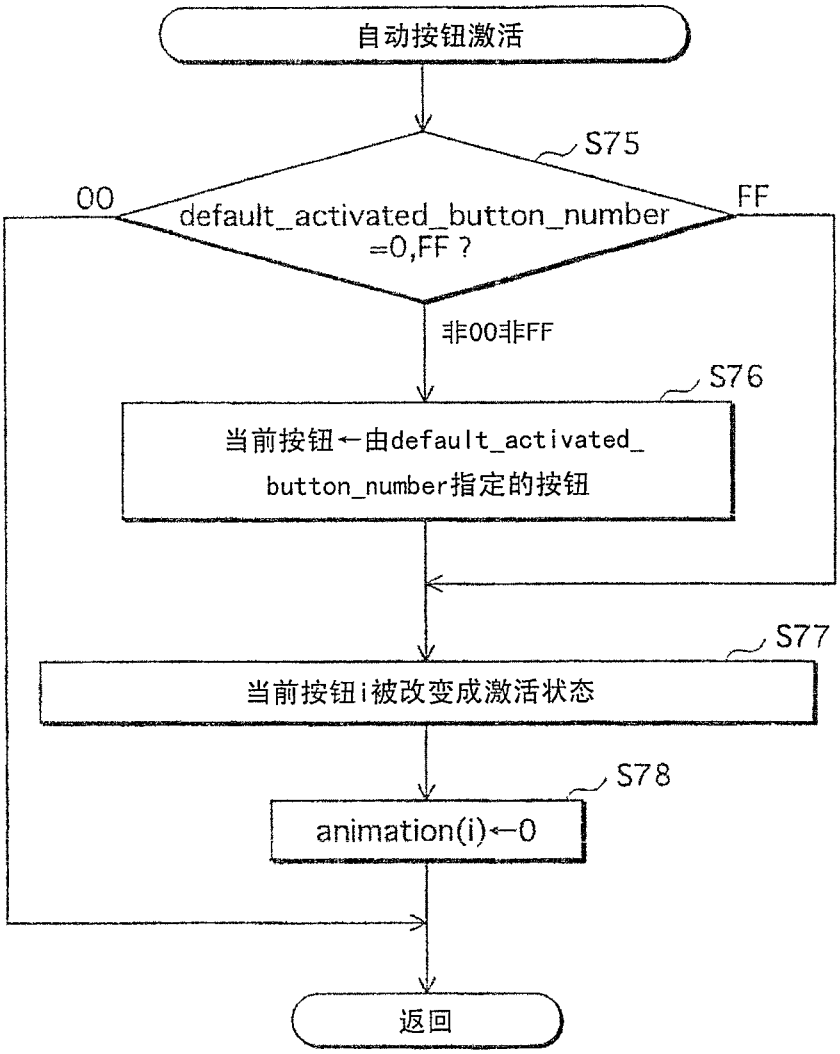

自动按钮激活
S75
00
default_activated_button_number
=0,FF ?
FF
非00非FF
S76
当前按钮←由default_activated_
button_number指定的按钮
S77
当前按钮i被改变成激活状态
S78
animation(i)←0
返回

图74

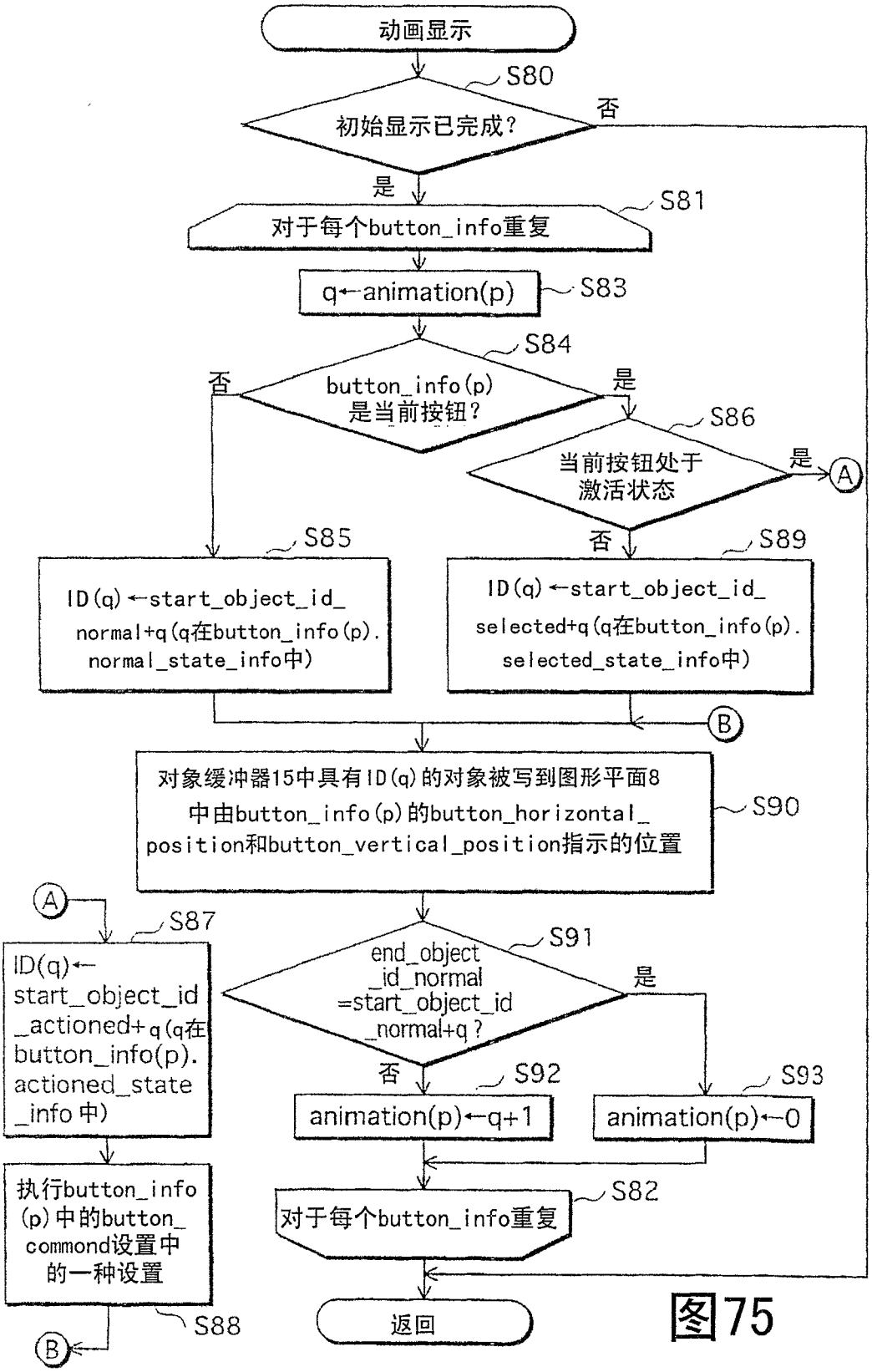

动画显示
S80
初始显示已完成?
否
是
S81
对于每个button_info重复
q←animation(p)
S83
S84
button_info(p)
是当前按钮?
否
是
S86
当前按钮处于
激活状态
是
A
否
S85
ID(q)←start_object_id_
normal+q(q在button_info(p).
normal_state_info中)
S89
ID(q)←start_object_id_
selected+q(q在button_info(p).
selected_state_info中)
B
对象缓冲器15中具有ID(q)的对象被写到图形平面8
中由button_info(p)的button_horizontal_
position和button_vertical_position指示的位置
S90
A
S87
ID(q)←
start_object_id
_actioned+q(q在
button_info(p).
actioned_state
_info中)
S91
end_object
_id_normal
=start_object_id
_normal+q?
是
否
S92
S93
animation(p)←q+1
animation(p)←0
执行button_info
(p)中的button_
commond设置中
的一种设置
S82
对于每个button_info重复
S88
B
返回

图75

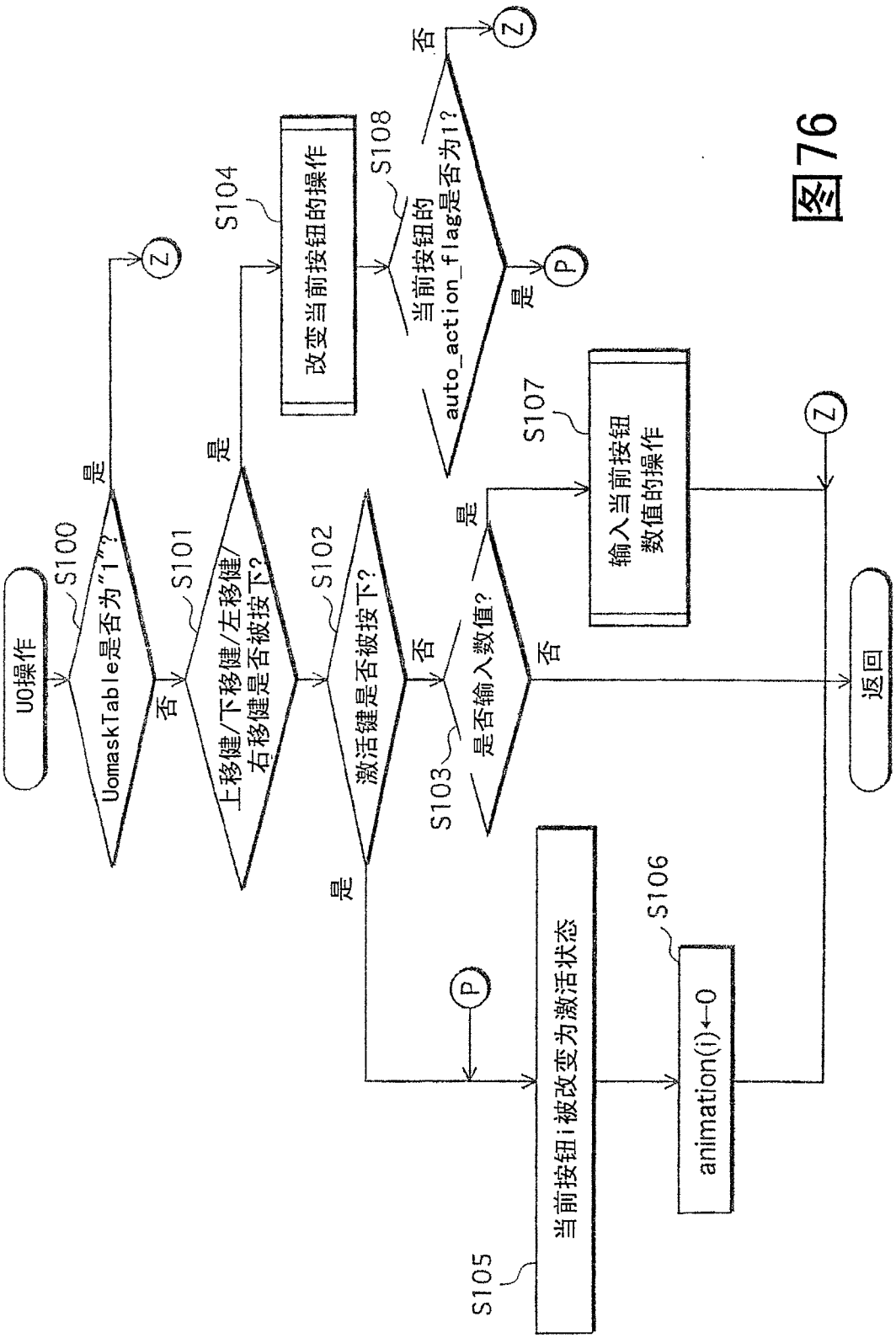

UO操作
S100
UomaskTable是否为"1"?
是
否
S101
上移健/下移健/左移健/右移健是否被按下?
是
否
S102
激活键是否被按下?
是
否
S103
是否输入数值?
是
否
S104
改变当前按钮的操作
S105
当前按钮i被改变为激活状态
S106
animation(i)←0
S107
输入当前按钮数值的操作
S108
当前按钮的auto_action_flag是否为1?
是
否
返回
Z
P

图76

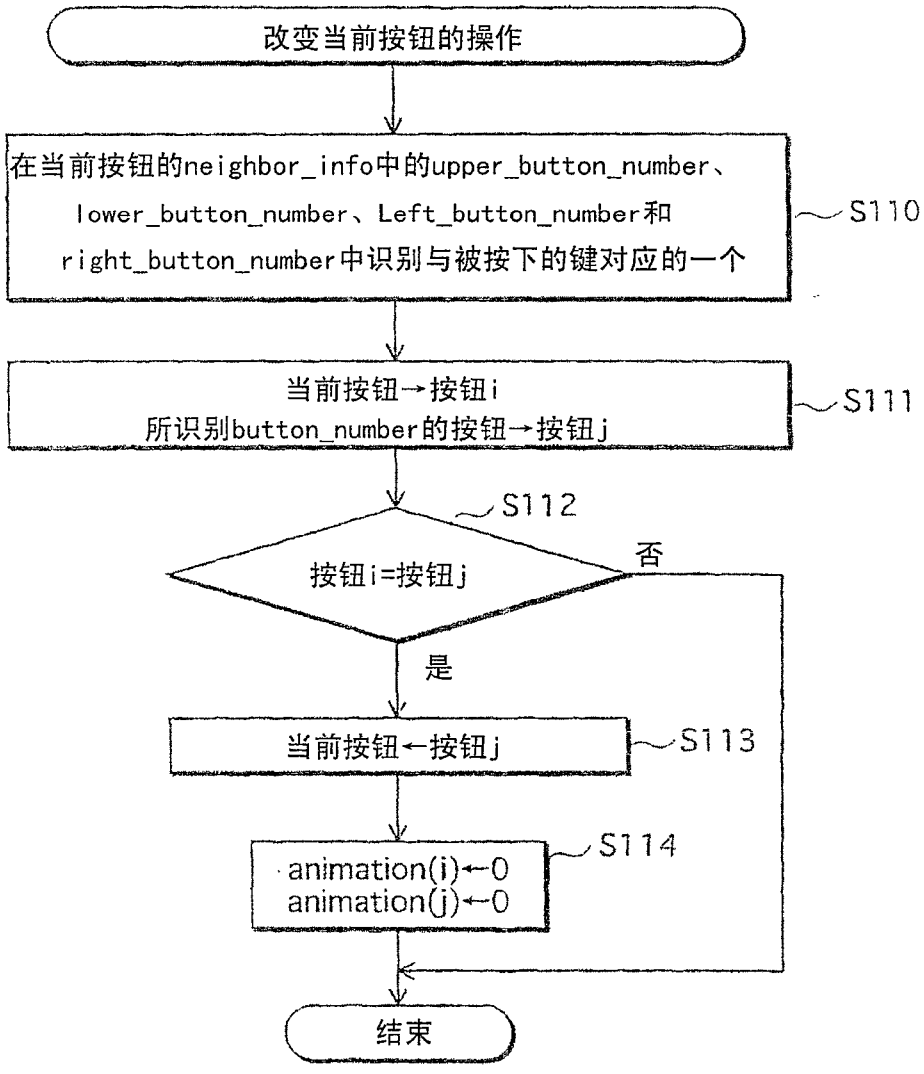

改变当前按钮的操作
在当前按钮的neighbor_info中的upper_button_number、
lower_button_number、Left_button_number和
right_button_number中识别与被按下的键对应的一个
S110
当前按钮→按钮i
所识别button_number的按钮→按钮j
S111
S112
按钮i=按钮j
否
是
当前按钮←按钮j
S113
animation(i)←0
animation(j)←0
S114
结束

图77

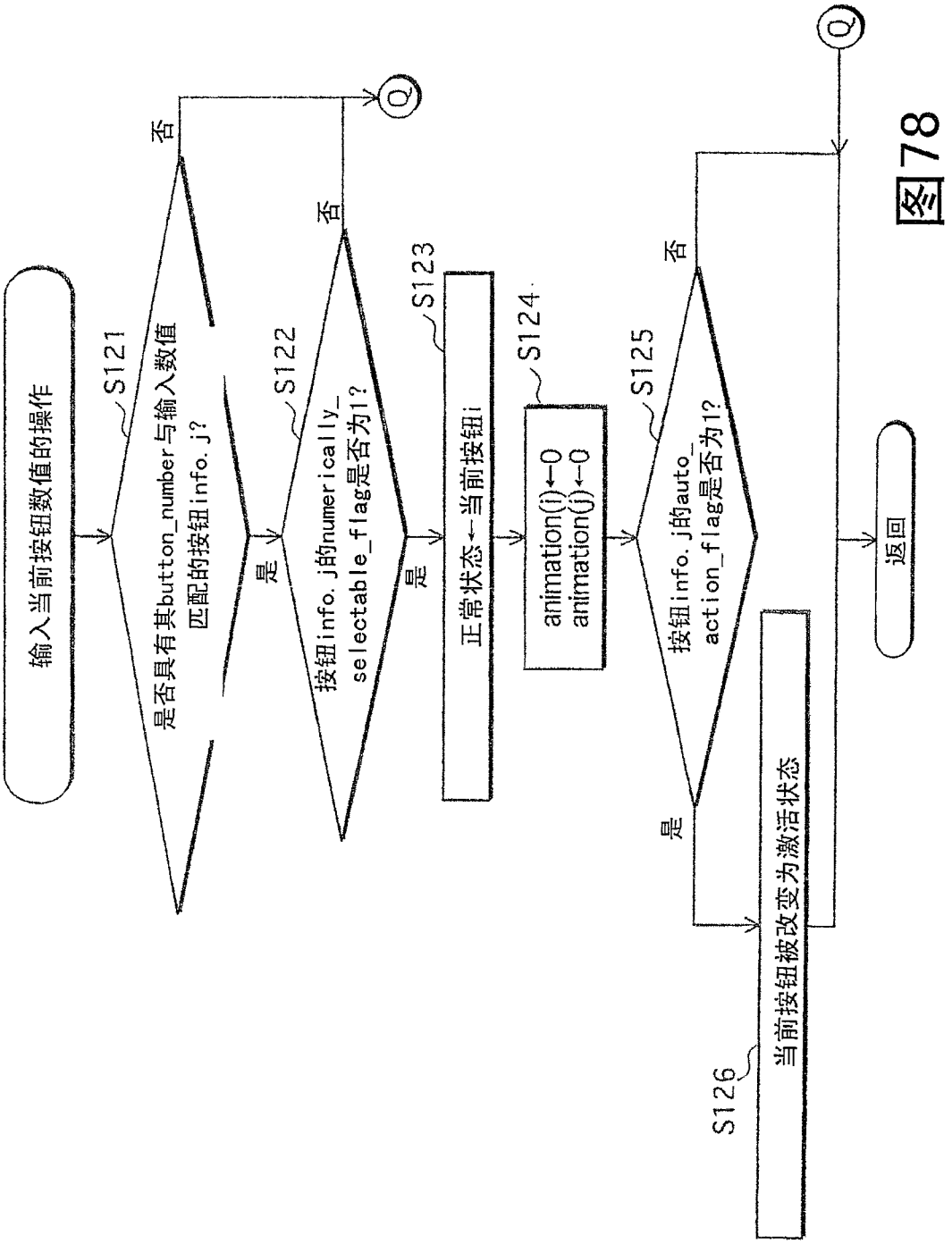

输入当前按钮数值的操作
S121
是否具有其button_number与输入数值匹配的按钮info.j?
否
是
S122
按钮info.j的numerically_selectable_flag是否为1?
否
是
Q
S123
正常状态←当前按钮i
S124
animation(i)←0
animation(j)←0
S125
按钮info.j的auto_action_flag是否为1?
否
是
S126
当前按钮被改变为激活状态
Q
返回

图78

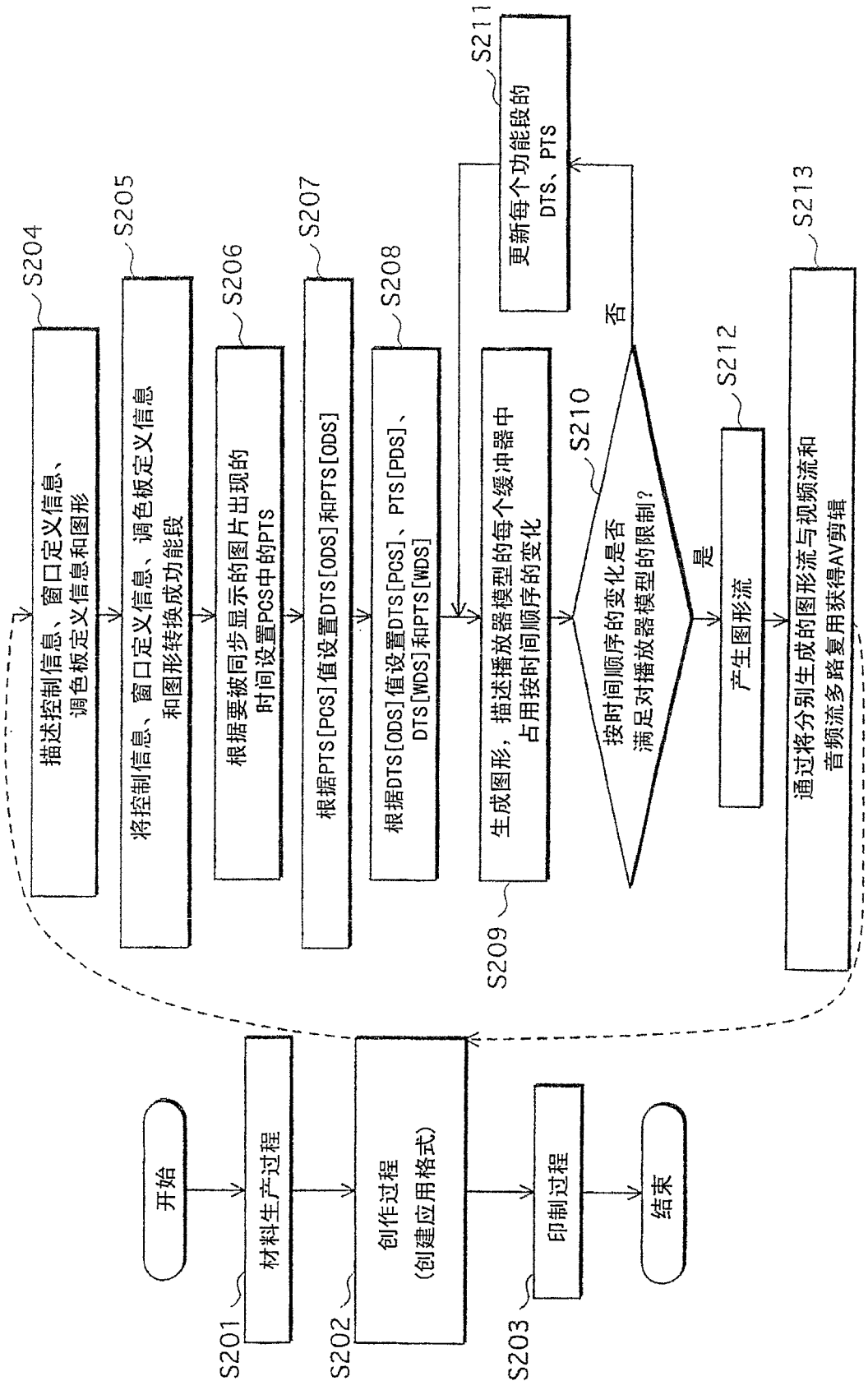

开始
S201
材料生产过程
S202
创作过程
(创建应用格式)
S203
印制过程
结束
S204
描述控制信息、窗口定义信息、
调色板定义信息和图形
S205
将控制信息、窗口定义信息、调色板定义信息
和图形转换成功能段
S206
根据要被同步显示的图片出现的
时间设置PCS中的PTS
S207
根据PTS[PCS]值设置DTS[ODS]和PTS[ODS]
S208
根据DTS[ODS]值设置DTS[PCS]、PTS[PDS]、
DTS[WDS]和PTS[WDS]
S209
生成图形，描述播放器模型的每个缓冲器中
占用按时间顺序的变化
S210
按时间顺序的变化是否
满足对播放器模型的限制？
否
是
S211
更新每个功能段的
DTS、PTS
S212
产生图形流
S213
通过将分别生成的图形流与视频流和
音频流多路复用获得AV剪辑

图79

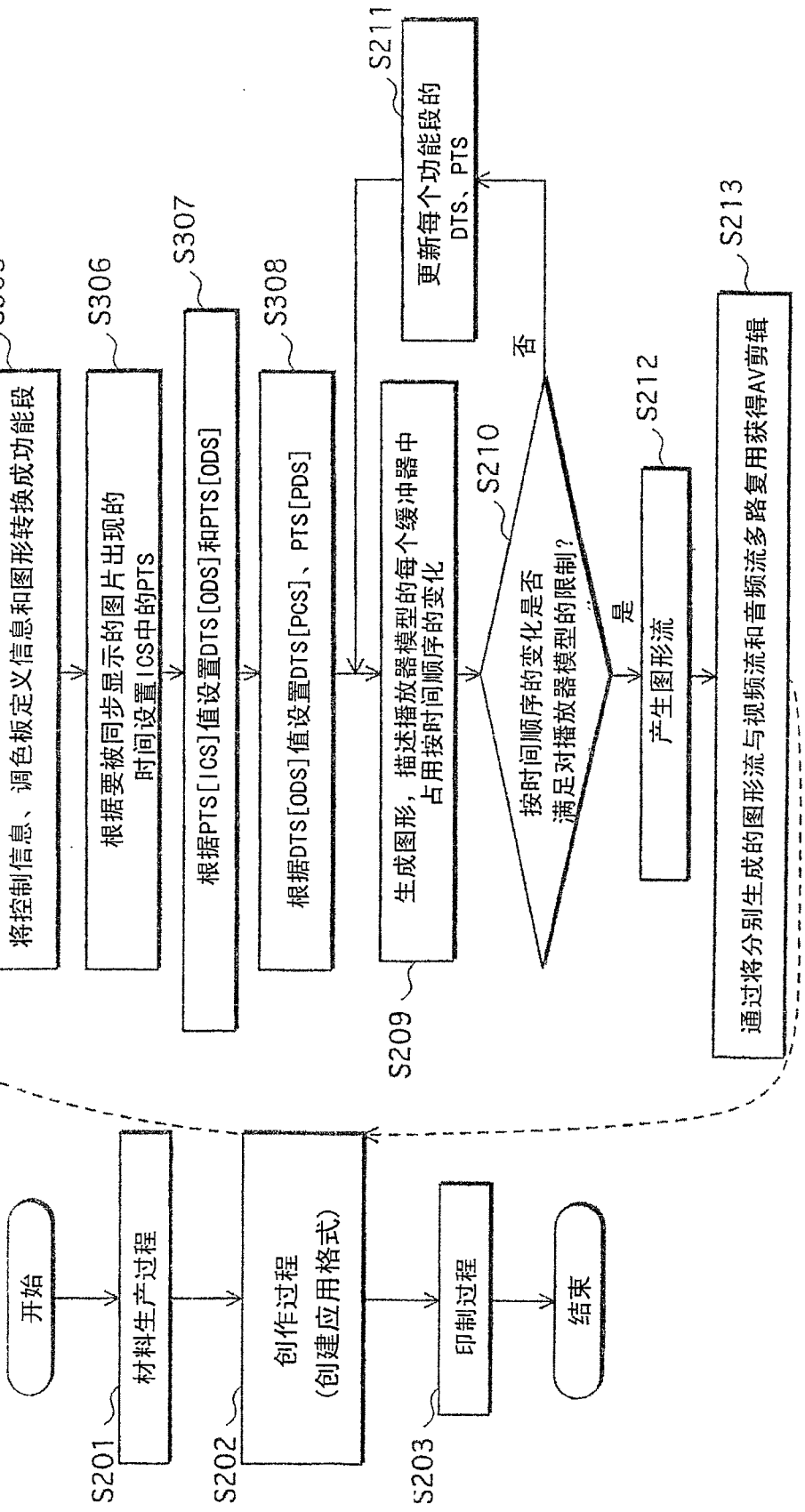

开始
S201
材料生产过程
S202
创作过程
(创建应用格式)
S203
印制过程
结束
S209
描述控制信息、调色板定义信息和图形
S304
将控制信息、调色板定义信息和图形转换成功能段
S305
根据要被同步显示的图片出现的
时间设置ICS中的PTS
S306
根据PTS[ICS]值设置DTS[ODS]和PTS[ODS]
S307
根据DTS[ODS]值设置DTS[PCS]、PTS[PDS]
S308
生成图形，描述播放器模型的每个缓冲器中
占用按时间顺序的变化
S210
按时间顺序的变化是否
满足对播放器模型的限制？
否
是
更新每个功能段的
DTS、PTS
S211
产生图形流
S212
通过将分别生成的图形流与视频流和音频流多路复用获得AV剪辑
S213

图80