

[19] 中华人民共和国国家知识产权局



[12] 发明专利说明书

专利号 ZL 200480018704.6

[51] Int. Cl.

H04N 5/91 (2006.01)

H04N 5/93 (2006.01)

G09G 5/00 (2006.01)

G09G 5/36 (2006.01)

G11B 27/34 (2006.01)

[45] 授权公告日 2009 年 12 月 30 日

[11] 授权公告号 CN 100576898C

[22] 申请日 2004.6.29

[21] 申请号 200480018704.6

[30] 优先权

[32] 2003.6.30 [33] US [31] 60/483,228

[86] 国际申请 PCT/JP2004/009517 2004.6.29

[87] 国际公布 WO2005/002220 日 2005.1.6

[85] 进入国家阶段日期 2005.12.30

[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 矢羽田洋 约瑟夫·麦克罗森

冈田智之 池田航

[56] 参考文献

CN1193444A 1998.9.16

CN1206533A 1999.1.27

CN1126404A 1996.7.10

JP2002-101386A 2002.4.5

US6469718B1 2002.10.22

US5854873A 1998.12.29

EP0787404B1 2002.2.6

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

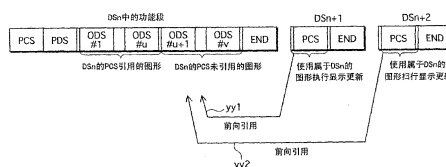
权利要求书 3 页 说明书 43 页 附图 41 页

[54] 发明名称

记录方法、再现装置和方法

[57] 摘要

BD-ROM 上记录了 AV 剪辑记录, 所述 AV 剪辑是通过复用视频流和图形流而获得的。所述图形流是 PES 数据包的序列。所述 PES 数据包可以是包含图形数据的 ODS(对象定义段), 也可以是包含控制信息的 PCS(呈现控制段)。PCS 表示: 在预定时机, 将位于该 PCS 之前的 ODS 中包含的图形数据与运动图像流进行组合并显示。PCS 和 ODS 属于不同的显示集合。ODS 可以是未被相同显示集合中的 PCS 引用的无引用图形数据。



1、一种用于再现数字流的再现装置，所述数字流是通过复用视频流和图形流而生成的，所述再现装置包括：

视频解码器，用于对所述视频流进行解码，以生成运动图像，并把构成所述运动图像的图片存入视频平面中；

图形解码器，用于对所述图形流进行解码，以生成图形，并把所生成的图形存入图形平面中；

加法器，用于将所述视频平面中存储的图片和所述图形平面中存储的图形的相应像素相加，从而重叠所述图形和所述运动图像；

其中，

从记录介质中新读取控制数据之后，所述图形解码器便基于所述控制数据，将已经通过对所述图形流中位于所述控制数据之前的图形数据进行解码而生成的图形传送给所述图形平面。

2、如权利要求1所述的再现装置，其中：

所述图形流包括多个显示集合，所述多个显示集合都用于产生图形显示；以及

读取所述图形数据所属的显示集合之后，所述图形解码器便对所述图形数据进行解码，以生成所述图形，并将所生成的图形存储到对象缓冲器中。

3、如权利要求2所述的再现装置，其中：

所述多个显示集合包括时元开始显示集合，后者包括窗口定义数据，用于定义呈现所述图形数据的区域；

所述窗口定义数据指定所述图形平面中用于覆盖所述运动图像的窗口的位置、高度和宽度；以及

所述图形解码器通过清除所述窗口定义数据指定的所述窗口并将所述图形写入所述窗口，来显示所述图形。

4、一种向记录介质上进行记录的方法，包括下列步骤：

生成图形流；

复用视频流和所述图形流；

将通过所述复用而获得的数字流记录到所述记录介质上；

其中，生成图形流的步骤包括以下子步骤：

将图形数据和控制数据分别存储在不同的数据包中；

按序排列用于存储图形数据的数据包和用于存储控制数据的数据包，以获得图形流；

其中，所述控制数据指示：在预定时间，将数据包序列中位于用来存储控制数据的数据包之前的数据包里的图形数据以覆盖到所述视频流之上的状态进行显示。

5、如权利要求4所述的记录方法，其中：

当将所述图形数据和所述控制数据存储在不同的数据包中时，每一个数据包都属于多个显示集合中的一个显示集合，所述多个显示集合都用于产生图形显示；以及

所述图形数据和所述控制数据属于不同的显示集合。

6、如权利要求5所述的记录方法，其中：

所述多个显示集合包括时元开始显示集合；

所述时元开始显示集合包括窗口定义数据，以用于定义呈现所述图形数据的区域；

所述窗口定义数据指定图形平面中用于覆盖运动图像的窗口的位置、高度和宽度，其中所述图形平面用于存储通过解码所述图形流而生成的图形；以及

所述控制数据指示：所述图形数据要位于所述窗口之内。

7、一种用于再现数字流的方法，所述数字流是通过复用视频流和图形流而生成的，所述方法包括下列步骤：

对所述视频流进行解码，以生成运动图像，并把构成所述运动图

像的图片存入视频平面中；

对所述图形流进行解码，以生成图形，并把所生成的图形存入图形平面中；

把所述视频平面中存储的图片和所述图形平面中存储的图形的相应像素相加，从而重叠所述图形和所述运动图像；

其中，

从记录介质中新读取出控制数据之后，对所述图形流进行解码的步骤便基于所述控制数据，将已经通过对所述图形流中位于所述控制数据之前的图形数据进行解码而生成的图形传送给所述图形平面。

记录方法、再现装置和方法

发明领域

本发明涉及一种上面记录了数字流的记录介质以及一种用于再现数字流的再现装置,所述数字流是通过复用视频流和图像流而生成的。具体而言,在本发明所涉及的技术中,再现装置对视频流和图形流独立地进行解码,并重叠所得的视频和图形,从而产生再现图像。

技术背景

采用上述视频/图形覆盖技术,可以根据再现装置的语言设置或显示设置,选择图形和显示选中的图形。此外,采用上述视频/图形覆盖技术,还可以选择是否根据需要来显示图形。由于能够提供这些选择,所以,DVD-Video 和 ETSI EN 300 743 的播放机模型中使用了视频/图形覆盖技术。在这些标准中,图形流是 PES 数据包的序列。PES 数据包可分为两种类型:承载控制信息的 PES 数据包和承载用于描绘图形的图形数据的 PES 数据包。控制信息设置在图形数据之前,一对控制信息和图形数据产生一个图形显示。更详细地说,再现装置顺序地读取控制信息和图形数据,对控制信息进行解码,再对图形数据进行解码,从而生成解压缩的图形。再现装置根据控制信息的解码结果,在预期的显示时机,显示解压缩的图形。

根据该技术,对于每一次图形显示,都需要执行图形数据的解码。因此,从一次图形显示到下一次图形显示之间的时间间隔取决于图形数据的解码处理所需的时间。如果图形具有较高的分辨率并且图形数据的解码处理要花费较长的时间,则图形更新的时间间隔就会变长。

对于电影中的字幕等,其中更新图形的时间间隔是 2 至 3 秒,所以,即使图形具有较高的分辨率并且图形数据的解码需要较长的时间,也可以产生合适的图形显示。但是,对于图形专用于娱乐的情况,就需要缩短图形更新时间间隔。

例如，在用于娱乐目的的图形的应用中，需要移动表示字符对话的图形，以吸引浏览者的注意力，或需要改变其颜色，以给浏览者带来惊喜。诸如杂耍表演之类内容的制造商尤其渴望采用这样的图形显示。

但是，为了实现图形的流畅运动，需要在短短的一帧周期（NTSC中是 1/29.97 秒）内执行对图形数据进行解码和呈现图形的操作。在每个帧周期内重复对图形数据进行解码和呈现图形的操作需要很大的负载，并且，所实现的再现装置的硬件规格很难以低成本来提供。

通过提前覆盖每个视频帧上的图形以使得图形成为每帧图像图案的一部分，可以实现前述图形运动和颜色变化。但是，提前覆盖每个帧上的图形的方法灵活性不够高，无法根据再现装置的语言设置或显示设置来有选择地显示图形，因此未来的潜力不大。

发明内容

本发明的目的在于提供一种记录介质，它使得图形能够象运动图像一样流畅地运动。

一种用于存储数据的记录介质可以实现上述目的，该记录介质包括通过复用视频流和图形流而生成的数字流，其中：所述图形流是多个数据包的序列，所述数据包序列包括一个包含控制信息的数据包；所述控制信息表示，在预定时间，将所述序列中前一数据包中包含的图形数据以覆盖到所述视频流之上的状态进行显示。

控制信息表示：使用位于该控制信息之前的图形数据，产生显示。因此，只需向再现装置输入表示图形新坐标的控制信息，就可以改变图形的显示位置。同样，只需向再现装置输入适当的控制信息，就可以改变图形的颜色。

由于只使用控制信息就可更新图形显示，所以，很容易使图形与视频同步。

通常，为了改变图形的显示位置，不需要改变图形本身的图像图案。这是因为，人眼无法如此清楚地察觉到运动对象。所以，本发明提供的技术适于显示相同的图形，同时高速地改变其位置。

附图说明

- 图 1 是本发明所涉及的记录介质的示例性应用；
- 图 2 示出了图 1 所示的 BD-ROM 的结构；
- 图 3 示出了 AV 剪辑的结构；
- 图 4A 示出了呈现图形流的结构；
- 图 4B 示出了包含功能段的 PES 数据包；
- 图 5 示出了由多种功能段构成的逻辑结构；
- 图 6 示出了字幕显示位置和时元之间的关系；
- 图 7A 示出了 ODS 的数据结构；
- 图 7B 示出了 PDS 的数据结构；
- 图 8A 示出了 WDS 的数据结构；
- 图 8B 示出了 PCS 的数据结构；
- 图 9 示出了用于显示字幕的 DS 的示例性描述；
- 图 10 示出了 DS1 中的 PCS 和 WDS 的示例性描述；
- 图 11 示出了 DS2 中的 PCS 的示例性描述；
- 图 12 示出了 DS3 中的 PCS 的示例性描述；
- 图 13 示出了当执行如图 10 至 12 所示的图形更新时对象缓冲器中的存储空间；
- 图 14 示出了用于计算 DECODEDURATION 的示例性算法；
- 图 15 是图 14 所示算法的流程图；
- 图 16A 和 16B 是图 14 所示算法的流程图；
- 图 17A 示出了一个窗口中存在一个图形对象的情形；
- 图 17B 和 17C 是图 14 所示算法中所用的参数的时序图；
- 图 18A 示出了一个窗口中存在两个图形对象的情形；
- 图 18B 和 18C 是图 14 所示算法中所用的参数的时序图；
- 图 19A 示出了两个图形对象分别存在于两个窗口中的情形；
- 图 19B 是当解码时间 (2) 大于清除时间 (1) 与写入时间 (31) 之和时的时序图；
- 图 19C 是当清除时间 (1) 与写入时间 (31) 之和大于解码时间

(2) 时的时序图;

图 20 示出了前向引用 PCS;

图 21A 示出了用于在屏幕上产生运动图形显示效果的时元;

图 21B 示出了 DS1 至 DS8 中的 PCS 的内容和 PTS;

图 22A 示出了 DS0 中的 ODS;

图 22B 示出了由窗口定义的坐标系中的坐标 $(x1,y1)$ 、 $(x2,y2)$ 、 $(x3,y3)$ …… $(x8,y8)$ 的位置;

图 23 示出了每个 DS 中的功能段的时戳的设置;

图 24 示出了连续图形更新的一个具体例子;

图 25 示出了用于显示仅调色板显示更新的一系列 DS;

图 26A 示出了 DS0 中的 PDS;

图 26B 示出了 DS0 至 DS3 中的 PCS;

图 27 示出了馈入四个 DS 所达到的显示效果;

图 28 示出了本发明的实施例所涉及的再现装置的内部构造;

图 29 示出了图 28 中所示的图形平面、编码数据缓冲器和对象缓冲器的传输速率 R_x 、 R_c 、 R_d 和尺寸;

图 30 是再现装置中的流水线处理的时序图;

图 31 是当 ODS 解码在清除图形平面之前结束情况下的流水线处理的时序图;

图 32 是合成缓冲器、对象缓冲器、编码数据缓冲器和图形平面的占用率变化的时序图;

图 33 是加载功能段的操作的流程图;

图 34 示出了执行跳节操作的情形;

图 35 示出了当执行如图 34 所示的跳节操作时向编码数据缓冲器中加载 DS10 的情形;

图 36 示出了当执行普通再现时的情形;

图 37 示出了当执行如图 36 所示的普通再现时向编码数据缓冲器中加载 DS1 和 DS20 的情形;

图 38 是图 28 所示的图形控制器的操作流程;

图 39 是图形控制器的操作流程;

图 40 是图形控制器的操作流程圖；

图 41 示出了 BD-ROM 的制造步骤。

具体实施方式

（第一个实施例）

下面描述本发明第一个实施例所涉及的记录介质。下面首先介绍该记录介质的使用。图 1 示出了该记录介质的示例性应用。在图中，该记录介质为 BD-ROM 100。BD-ROM 100 用于在家庭影院系统中提供电影，该家庭影院系统包括再现装置 200、电视 300 和遥控 400。

接着介绍记录介质的制造。通过改进 BD-ROM 的应用层，可以实现该记录介质。图 2 示出了 BD-ROM 100 的示例性结构。

在图中，第四级示出了 BD-ROM 100，第三级示出了 BD-ROM 100 上的轨道。图中所示的轨道向外延伸成直线，但实际上，轨道从 BD-ROM 100 的中心向外螺旋延伸。轨道包括引入区、容量区和引出区。容量区的层模型是物理层、文件系统层和应用层。第一级以目录结构的形式示出了 BD-ROM 100 的应用层格式（应用格式）。如图所示，BD-ROM 100 在根目录下有 BDMV 目录。BDMV 目录包含一个用于存储 AV 剪辑的文件（XXX.M2TS）、一个用于存储 AV 剪辑的管理信息的文件（XXX.CLPI）和一个用于定义 AV 剪辑的逻辑播放路径（播放列表）的文件（YYY.MPLS）。通过生成这样的应用格式，可以实现 BD-ROM 100。如果上述各文件类型都有一个以上的文件，则在 BDMV 目录下设置三个名为 STREAM、CLIPNF 和 PLAYLIST 的目录，分别用来存储与 XXX.M2TS 类型相同的文件、与 XXX.CLPI 类型相同的文件以及与 YYY.MPLS 类型相同的文件。

下面说明该应用格式中的 AV 剪辑（XXX.M2TS）。

AV 剪辑（XXX.M2TS）是 MPEG-TS（传输流）格式的数字流，它是通过复用—个视频流、至少一个音频流和—个呈现图形流而获得的。视频流代表电影的运动图像，音频流代表电影的音频，呈现图形流代表电影的字幕。图 3 示出了 AV 剪辑（XXX.M2TS）的结构。

在图中，中间级示出了 AV 剪辑。可以如下创建该 AV 剪辑。将

上方第一级中由多个视频帧（图像 pj1、pj2、pj3……）构成的视频流和由多个音频帧构成的音频流转换成上方第二级中的 PES 数据包，然后再转换成上方第三级中的 TS 数据包。同样，将下方第一级中的呈现图形流转换成下方第二级中的 PES 数据包，然后再转换成下方第三级中的 TS 数据包。将视频、音频和呈现图形流的这些 TS 数据包进行复用，从而形成 AV 剪辑。

在图 3 所示的例子中，只有一个呈现图形流被复用到 AV 剪辑中。如果 BD-ROM 100 支持多种语言，则可以将每种语言的呈现图形流都复用到 AV 剪辑中。以上述方式生成的 AV 剪辑按照与计算机文件相同的方式被分成多个内容，并存储到 BD-ROM 100 上。

下面说明呈现图形流。图 4A 示出了呈现图形流的结构。在图中，第一级显示的是构成该 AV 剪辑的 TS 数据包序列。第二级显示的是构成呈现图形流 PES 数据包序列。该 PES 数据包序列是通过将第一级的 TS 数据包序列中具有预定 PID 的 TS 数据包的有效负载进行连接而形成的。

第三级示出了呈现图形流的结构。呈现图形流是由多个功能段构成的，这些功能段包括 PCS（呈现合成段）、WDS（窗口定义段）、PDS（调色板定义段）、ODS（对象定义段）和 END（显示集合段的末尾）。在这些功能段中，PCS 是画面合成段，WDS、PDS 和 ODS 是定义段。一个功能段对应于一个 PES 数据包或多个 PES 数据包。也就是说，一个功能段被转换成一个 PES 数据包并记录到 BD-ROM 100 上，或者，被分成多个片断，再转换成 PES 数据包，并记录到 BD-ROM 100 上。

图 4B 示出了包含功能段的 PES 数据包。如图所示，每个 PES 数据包由包头和有效负载构成。有效负载携带功能段，包头携带与该功能段相关联的 DTS 和 PTS。后面，将包含一个功能段的 PES 数据包的包头中的 DTS 和 PTS 视为该功能段的 DTS 和 PTS。

这些多种类型的功能段形成如图 5 所示的逻辑结构。在图中，第三级示出了功能段，第二级示出了 DS（显示集合），第一级示出了时元。

第二级中的 DS 是呈现图形流中的一组功能段，用于创建一个图形画面。虚线 h2 表明第三级中的功能段属于哪个 DS。从图中可以看出，功能段序列 PCS-WDS-PDS-ODS-END 组成一个 DS。再现装置 200 从 BD-ROM 100 中读取构成该 DS 的这些功能段，以产生一个图形画面。

第一级中的时元指的是 AV 剪辑的再现时间轴上的连续存储器管理的一个时间单元以及分配给该时间单元的一组数据。这里所说的存储器包括：图形平面，用于存储一个图形画面；对象缓冲器，用于存储解压缩的图形数据。连续存储器管理意味着：在该时元内，不刷新图形平面和对象缓冲器，只在图形平面的预定矩形区域中执行图形的删除和呈现（刷新意味着清除整个图形平面或整个对象缓冲器）。在该时元内，该矩形区域的大小和位置是固定的。只要在图形平面的该固定矩形区域中执行图形的删除和呈现，就可以保证视频和图形的同步。换言之，时元是 AV 剪辑的再现时间轴上的一个时间单位，其中，可以保证视频和图形的同步。为了改变图形平面中的图形删除/呈现区域，需要定义再现时间轴上的变化点以及从该点向前设置一个新的时元。在这两个时元之间的边界中，视频和图形的同步无法得到保证。

对于字幕，时元是再现时间轴上的一个时间段，其中，字幕出现在平面上固定矩形区域内。图 6 示出了字幕显示位置和时元之间的关系。在图中，字幕显示位置的改变取决于图像的图案。更详细地说，三个字幕“Actually”、“I lied to you”和“Sorry”位于屏幕的底部，而两个字幕“三年已经过去了”和“从那以后。”位于屏幕的顶部。因此，为了增加可视性，字幕显示位置从一个边缘部分变化到另一边缘部分。在这种情况下，在 AV 剪辑的再现时间轴上，在屏幕底部显示字幕的时间段是时元 1，在屏幕顶部显示字幕的时间段是时元 2。这两个时元各有自己的字幕呈现区域。在时元 1 中，字幕呈现区域是 Window1，对应于屏幕的底部边缘部分。在时元 2 中，字幕呈现区域是 Window2，对应于屏幕的顶部边缘部分。在时元 1 和时元 2 中，对象缓冲器和图形平面的存储器管理是连续的，所以，字幕在该屏幕的相应边缘部分中无缝地显示。对时元的解释到此结束。

下面解释 DS。

在图 5 中，虚线 hk1 表示第二级中的 DS 属于哪个时元。如图所示，DS 序列(Epoch Start DS、Acquisition Point DS 和 Normal Case DS)构成第一级中的一个时元。在这里，Epoch Start (时元开始)、Acquisition Point (采集点)和 Normal Case (正常情况)是 DS 的类型。在图 5 中，Acquisition Point DS 位于 Normal Case DS 之前，但它们的排列顺序也可以反过来。

Epoch Start DS 提供显示效果“新显示”，并表示一个新时元的开始。Epoch Start DS 包含下一画面合成所需的全部功能段。提供 Epoch Start DS 的位置是跳节操作的目的地，如电影中一章的开始。

Acquisition Point DS 提供显示效果“显示刷新”，并且与前面的 Epoch Start DS 相同。Acquisition Point DS 不是时元的开始，但却包含下一画面合成所需的全部功能段。因此，当从 Acquisition Point DS 开始再现时，能够可靠地显示图形。也就是说，通过 Acquisition Point DS，可以从时元的中间点进行画面合成。

提供 Acquisition Point DS 的位置可以是跳节操作的目的地，例如可以通过时间搜索来指定的位置。时间搜索是一种操作，用于对与用户的时间输入（单位为分/秒）相对应的再现点进行定位。时间输入的单位较大，如 10 分或 10 秒。因此，提供 Acquisition Point DS 的位置可以通过单位是 10 分或 10 秒的时间搜索来指定。通过在可由时间搜索指定的位置中提供 Acquisition Point DS，当执行时间搜索时，可以流畅地再现图形流。

Normal Case DS 提供显示效果“显示更新”，并且仅包含与前一画面合成之间的差值。例如，如果 DS_v 与它之前的 DS_u 具有相同的字幕，但却具有不同的画面合成，则 DS_v 是仅包含一个 PCS 和一个 END 的 Normal Case DS。这样，就不必在 DSs 中提供覆盖的 ODS，从而可以减少 BD-ROM 100 上所存储的数据量。由于 Normal Case DS 只包含差值，所以，不能单用 Normal Case DS 来显示图形。

下面说明 ODS、WDS 和 PDS (定义段)。

ODS 是用于定义图形对象的功能段。BD-ROM 上记录的 AV 剪

辑的图像质量与高清晰度电视一样高。因此，以 1920×1080 的高分辨率来设置图形对象。由于该高分辨率，可以在 BD-ROM 上逼真地再现影院屏幕风格的字幕，即，雅致的手写体字幕。图形对象由多个游程数据构成。游程数据表示使用像素编码的像素串，是像素值和该像素值的连续长度。像素编码有 8 个比特，表示从 1 到 255 的值。通过使用该像素编码，游程数据设置全部颜色（16777216 色）中任意 256 个像素颜色。应当注意的是，为了将图形对象显示为字幕，需要将字符串放置在透明色的背景上。

ODS 定义根据图 7A 所示的数据结构的图形对象。如图所示，ODS 包括：segment_type 字段，表示段类型“ODS”；segment_length 字段，表示 ODS 的数据长度；object_id 字段，标识时元中的图形对象；object_version_number 字段，表示时元中的 ODS 的版本；last_in_sequence_flag 字段；object_data_fragment 字段，承载与图形对象的部分或全部相对应的连续字节序列。

更详细地讲，object_id 是一个标识符，用于标识图形对象，以及当对 ODS 进行解码和将图形对象缓冲到对象缓冲器中时该图形对象在对象缓冲器中所占用的存储区域。因此，当对象缓冲器中有一个或多个图形对象时，用 object_id 字段值标识对象缓冲器中的各存储区域。假设将一个 object_id 分配给两个或更多个 ODS。在这种情况下，在将与一个 ODS 相对应的图形对象存储到对象缓冲器之后，该图形对象会被与随后具有相同 object_id 的 ODS 相对应的图形对象覆盖。这样的更新意在防止对象缓冲器中出现很多小的自由空间和防止该对象缓冲器中的图形对象的分散。当显示图形时，对象缓冲器中的图形对象不断地被传输到图形平面。因此，如果对象缓冲器中存在很多小的自由空间或者一个图形对象分散在对象缓冲器中，则读取图形对象的开销导致从对象缓冲器到图形平面的传输效率降低。传输效率的这种降低会影响图形和视频的同步显示。为了防止这种问题，对象缓冲器中的现有图形对象被具有相同 object_id 的一个新图形对象覆盖。

这里，覆盖现有图形对象的新图形对象需要在尺寸上等于现有图形对象，也就是说，新图形对象既不能小于现有图形对象，也不能大

于现有图形对象。因此，在创作时，创作者需要使这些图形对象尺寸相等。该尺寸约束条件，即具有相同 `object_id` 的图形对象应该宽度和高度相等，只适用于一个时元内。具有相同 `object_id` 的图形对象不必尺寸相等，如果它们属于不同时元的话。

接下来解释 `last_in_sequence_flag` 字段和 `object_data_fragment` 字段。由于 PES 数据包的有效负载的约束条件，构成一个字幕的解压缩图形可能无法包含在一个 ODS 中。在这种情况下，将图形分割成多个片断，将其中一个片断承载在 `object_data_fragment` 中。当将图形对象跨过多个 ODS 进行存储时，除最后片断之外的每个片断都具有相同的尺寸。也就是说，最后的片断小于或等于前面片断的尺寸。承载图形对象的这些片断的 ODS 顺序地出现在 DS 中。`last_in_sequence_flag` 字段表示图形对象的结束。尽管上述 ODS 数据结构所基于的方法将片断不留空隙地存储在连续 PES 数据包中，但是，也可以通过在 PES 数据包中留出一些空隙的方式将片断存储在 PES 数据包中。

PDS 是功能段，定义用于颜色转换的调色板。调色板是表示 1 至 255 的像素代码和像素值的组合的数据。这里所指的像素值是由红色差值分量（Cr 值）、蓝色差值分量（Cb 值）、亮度分量（Y 值）和透明色（T 值）构成的。将每个游程数据的像素代码代入调色板上的像素值中，从而产生颜色。图 7B 示出了 PDS 的数据结构。如图所示，PDS 包括：`segment_type` 字段，表示段类型“PDS”；`segment_length` 字段，表示 PDS 的数据长度；`palette_id` 字段，唯一地标识该调色板；`palette_version_number` 字段，表示该时元内的 PDS 的版本；以及 `palette_entry` 字段，承载每个条目的信息。`palette_entry` 字段表示每个条目的红色差值分量（Cr 值）、蓝色差值分量（Cb 值）和透明色（T 值）

WDS 是功能段，用于定义图形平面上的矩形区域。如前所述，在图形平面上的固定矩形区域中执行清除和呈现的时元内，存储器管理是连续的。图形平面上的该矩形区域被称为窗口，它是用 WDS 来定义的。图 8A 示出了 WDS 的数据结构。如图所示，WDS 包括：

`window_id` 字段，唯一地标识图形平面上的该窗口；
`window_horizontal_position` 字段，指定图形平面上该窗口的左上方像素的水平位置；`window_vertical_position` 字段，指定图形平面上该窗口的左上方像素的垂直位置；`window_width` 字段，指定图形平面上该窗口的长度；`window_height` 字段，指定图形平面上该窗口的高度。

`window_horizontal_position` 字段、`window_vertical_position` 字段、`window_width` 字段和 `window_height` 字段可以取以下值。图形平面作为这些字段值的坐标系。图形平面具有由参数 `window_width` 和 `window_height` 定义的二维尺寸。

`window_horizontal_position` 字段指定图形平面上该窗口的左上方像素的水平位置，因此取值范围是 0 至 $(\text{video_width})-1$ 。
`window_vertical_position` 字段指定图形平面上该窗口的左上方像素的垂直位置，因此取值范围是 0 至 $(\text{video_height})-1$ 。

`window_width` 字段指定图形平面上该窗口的长度，因此取值范围是 1 至 $(\text{video_width})-(\text{window_horizontal_position})$ 。`window_height` 字段指定图形平面上该窗口的高度，因此取值范围是 1 至 $(\text{video_height})-(\text{window_vertical_position})$ 。对于每个时元，使用 WDS 中的 `window_horizontal_position`、`window_vertical_position`、`window_width` 和 `window_height` 字段，可以定义窗口的位置和尺寸。因此，在创作时，创作者可以调整窗口，使其出现在时元中的每个图像的预期边缘部分中，从而不干扰该图像的图案。可以清楚地观看按照这种方式显示的字幕的图形。对于每个时元，可以定义 WDS。因此，当图像图案随时间而变化时，可以基于这样的变化，移动图形，从而不降低可视度。这将电影质量增强到与将字幕集成到运动图像中相同的等级。

下面解释 END。END 是用于表示 DS 传输结束的功能段。END 位于 DS 中最后的 ODS 之后。END 包括：`segment_type` 字段，表示段类型“END”；`segment_length`，表示 END 的数据长度。这些字段不是本发明的主要特色，因此这里不再赘述。

下面解释 PCS（合成段）。

PCS 是功能段，用于合成可与运动图像同步的画面。图 8B 示出了 PCS 的数据结构。如图所示，PCS 包括：segment_type 字段；segment_length 字段；composition_number 字段；composition_state 字段；palette_update_flag 字段；palette_id_ref 字段；composition_object(1) 至 composition_object(m) 字段。

composition_number 字段用从 0 至 15 的数字来唯一地标识 DS 中的图形更新。更详细地讲，对于每个图形更新，从时元的开始到 PCS，composition_number 字段加 1。

composition_state 字段表示 DS 是 Normal Case DS、Acquisition Point DS、还是时元 Start DS。

palette_update_flag 字段表示 PCS 是否描述仅调色板显示更新。仅调色板显示更新指的是只用新调色板替换先前调色板的更新。为了表示仅调色板显示更新，将 palette_update_flag 字段设置为 1。

palette_id_ref 字段指定在该 DS 中使用的调色板。

composition_object(1) 至 composition_object(m) 字段各包含用于实现该 DS 中的画面合成的控制信息。在图 8B 中，作为示例，虚线 wd1 表示 composition_object(i) 的内部结构。如图所示，composition_object(i) 包括：object_id_ref 字段；window_id_ref 字段；object_cropped_flag 字段；object_horizontal_position 字段；object_vertical_position 字段；cropping_rectangle 信息(1) 至 cropping_rectangle 信息(n)。

object_id_ref 字段表示图形对象的标识符 (object_id) 的引用值。该引用值表示为了产生与 composition_object(i) 相对应的画面合成而要使用的图形对象。

window_id_ref 字段表示窗口标识符 (window_id) 的引用值。该引用值表示为了产生与 composition_object(i) 相对应的画面合成而要显示图形对象的窗口。

object_cropped_flag 字段表示是否要显示对象缓冲器中剪裁的图形对象。当将 object_cropped_flag 字段设为 1 时，显示对象缓冲器中剪裁的图形对象。当将 object_cropped_flag 字段设为 0 时，不显示对象缓冲器中剪裁的图形对象。

`object_horizontal_position` 字段表示图形对象在图形平面的左上方像素的水平位置。

`object_vertical_position` 字段表示图形对象在图形平面的左上方像素的垂直位置。

当 `object_cropped_flag` 字段为 1 时, `cropping_rectangle` 信息(1) `cropping_rectangle` 信息(n)是有效的。作为示例, 虚线 `wd2` 指示了 `cropping_rectangle` 信息(i)的内部结构。如图所示, `cropping_rectangle` 信息 (i) 包括: `object_cropping_horizontal_position` 字段; `object_cropping_vertical_position` 字段; `object_cropping_width` 字段; `object_cropping_height` 字段。

`object_cropping_horizontal_position` 字段表示剪裁矩形在图形对象中左上角的水平位置。剪裁矩形用于取出图形对象的一部分, 其对应于 ETSI EN 300 743 中的“区域”。

`object_cropping_vertical_position` 字段表示剪裁矩形在图形对象中左上角的垂直位置。

`object_cropping_width` 字段表示剪裁矩形在图形对象中的水平长度。

`object_cropping_height` 字段表示剪裁矩形在图形对象中的垂直长度。

下面具体描述 PCS, 在所使用的例子中, 当运动图像的再现进行时, 通过三个向图形平面写入的操作, 顺序地显示图 6 所示的三个字幕 “Actually”、“I lied to you.” 和 “Sorry.”。图 9 示出了实现该字幕显示的示例性描述。在图中, 一个时元具有 DS1 (Epoch Start DS)、DS2 (Normal Case DS)和 DS3 (Normal Case DS)。DS1 包括: 一个 WDS, 用于定义显示字幕的窗口; 一个 ODS, 表示行 “Actually I lied to you. Sorry.”; 一个 PCS。DS2 包括一个 PCS。DS3 包括一个 PCS。

下面描述各个 PCS。图 10 至 12 示出了属于 DS1 和 DS3 的 WDS 和 PCS。

图 10 示出了 DS1 中的 PCS 和 WDS。在图中, WDS 中的 `window_horizontal_position` 字段值和 `window_vertical_position` 字段值

指定图形平面上的窗口的左上方坐标 LP1, WDS 中的 window_width 字段值和 window_height 字段值指定窗口的宽度和高度。

PCS 中 cropping_rectangle 信息中的 object_cropping_horizontal_position 字段值和 object_cropping_vertical_position 字段值指定剪裁矩形在坐标系中左上角坐标 ST1, 该剪裁矩形的原点是对象缓冲器中图形对象的左上方坐标。剪裁矩形是一个区域, 由从左上角坐标 ST1 的 object_cropping_width 字段值和 object_cropping_height 字段值进行定义。剪裁过的图形对象位于区域 cp1 (被虚线框包围) 中, 所以, 剪裁过的图形对象的左上角位于一个由坐标系中该图形平面的 object_horizontal_position 字段和 object_vertical_position 字段指定的像素。这样, 将字幕 “Actually I lied to you. Sorry.” 中的 “Actually” 写入图形平面上的窗口中。字幕 “Actually” 覆盖在一张图片之上, 然后, 显示所得的图像。

图 11 示出了 DS2 中的 PCS。由于图中的 WDS 的描述与图 10 相同, 所以这里不再赘述。同时, cropping_rectangle 信息的描述与图 10 不同。在图 11 中, cropping_rectangle 信息中的 object_cropping_horizontal_position 字段值和 object_cropping_vertical_position 字段值指定与对象缓冲器中的字幕 “I lied to you.” 相对应的剪裁矩形的左上方坐标, 而 object_cropping_width 字段值和 object_cropping_height 字段值指定剪裁矩形的宽度和高度。从而, 将字幕 “I lied to you.” 写入图形平面上的窗口中。字幕 “I lied to you.” 覆盖在一张图片上, 并显示所得的图像。

图 12 示出了 DS3 中的 PCS。由于图中的 WDS 的描述与图 10 相同, 所以这里不再赘述。同时, cropping_rectangle 信息的描述与图 10 不同。在图 12 中, object_cropping_horizontal_position 字段值和 object_cropping_vertical_position 字段值表示对象缓冲器中与字幕 “Sorry.” 相对应的剪裁矩形的左上角坐标, 而 object_cropping_width 字段值和 object_cropping_height 字段值表示剪裁矩形的宽度和高度。

从而，将字幕“Sorry.”写入图形平面上的窗口中。字幕“Sorry.”覆盖在一张图片上，并显示所得图像。图 13 示出了当执行如图 10 至 12 的图形更新时对象缓冲器的存储器空间。如图所示，对象缓冲器有四个存储区域 A 至 D，它们各有固定的高度和宽度以及固定的位置。在存储器区域 A 至 D 中，图 10 所示的字幕存储在存储区域 A 中。存储区域 A 至 D 中的每一个用与该存储区域中要存储的图形对象相对应的 object_id 标识。详细地讲，存储区域 A 用 object_id=1 标识，存储区域 B 用 object_id=2 标识，存储区域 C 用 object_id=3 标识，存储区域 D 用 object_id=4 标识。为了维持从对象缓冲器到图形平面的传输效率，存储区域 A 至 D 中的每一个区域的高度和宽度都是固定的。因此，当解码之后获得具有相同 object_id 的图形对象时，将该图形对象写入由该 object_id 标识的存储区域中，覆盖现有的图形对象。例如，为了在与图 10 至 12 中显示的字幕相同的位置用相同的尺寸显示字幕，需要在后续的 DS 中提供与 DS1 中的 ODS 具有相同 object_id 的 ODS。通过如此添加相同的 object_id，对象缓冲器中的图形对象被新的图形对象覆盖，新的图形对象的显示位置以及尺寸与被覆盖的图形对象相同。

下面描述实现显示效果的约束条件。为了流畅地显示字幕，在窗口上需要执行清除和呈现。当以视频帧的帧速率执行窗口清除和窗口呈现时，需要下面的从对象缓冲器到图形平面的传输速率。

首先，检查对窗口尺寸的约束条件。假设 R_c 是从对象缓冲器到图形平面的传输速率。在最糟糕的情况下，需要以 R_c 执行窗口清除和窗口呈现。换言之，窗口清除和窗口呈现各需要以 R_c 的一半($R_c/2$)执行。

为了一个视频帧同步窗口清除和窗口呈现，需要满足：

$$(\text{窗口尺寸}) \times (\text{帧速率}) \approx R_c/2。$$

如果帧速率为 29.97，则

$$R_c = (\text{窗口尺寸}) \times 2 \times 29.97$$

为了显示字幕，窗口尺寸至少需要是整个图形平面的 25%至 33%。如果图形平面的像素总数量是 1920×1080 ，每个像素的索引的

比特长度为 8 比特，则图形平面的总容量是 2M 字节 ($\approx 500\text{K}$ 字节 $\times 2 \times 29.97$)。

假定窗口尺寸是图形平面的 1/4，即 500 K 字节 ($= 2\text{ M 字节}/4$)。将其代入上式，则得到 $R_c = 256\text{ Mbps}$ ($500\text{ K 字节} \times 2 \times 29.97$)。

因此，如果窗口尺寸是图形平面的大约 25%至 33%，那么，只要用 $R_c = 256\text{ Mbps}$ 来显示字幕，就可以实现字幕的显示效果，而不会与运动图像失去同步。

如果以视频帧速率的 1/2 或 1/4 执行窗口清除和窗口呈现，则在 R_c 相同的情况下窗口尺寸可以为两倍或四倍。下面描述窗口的位置和范围。如前所述，窗口的位置和范围在一个时元内是固定的，原因如下。

如果窗口的位置或范围在时元中变化，则需要改变到图形平面的写地址。这会产生开销，从而导致从对象缓冲器到图形平面的传输速率 R_c 降低。

当向图形平面传输解码后的图形对象时，为了降低开销，在一个窗口中可同时显示的图形对象的数量是有限的。当设置图形对象的边缘部分的地址时，会产生这里所说的开销发生。如果边缘部分的数量较大，则该开销会增加。

如果对一个窗口中可显示的图形对象的数量没有限制，那么，当向图形平面传输图形对象时，开销会无限地发生，这会增加传输负载的变化。另一方面，如果一个窗口中的图形对象的数量被限制为 2，那么，基于最糟糕情形下开销数量为 4 的假设，可以设置传输速率 R_c 。因此，可以很容易地确定传输速率 R_c 的最低标准。对窗口的解释到此结束。

下面说明如何将承载诸如 PCS 和 ODS 之类功能段的 DS 分配到 AV 剪辑的再现时间轴上。时元是再现时间轴上的一个时间段，其中，存储器管理是连续的，并且，时元由一个或多个 DS 构成。因此，将 DS 有效地分配到 AV 剪辑的再现时间轴上是很重要的。这里所说的 AV 剪辑的再现时间轴是一个时间轴，用于定义构成 AV 剪辑中复用的视频流的各图片的解码时间和呈现时间。再现时间轴上的解码时间

和呈现时间用 90 KHz 的时间精度来表示。DS 中的 PCS 和 ODS 的 DTS 和 PTS 表示该再现时间轴上的同步控制的时机。换言之，通过使用 PCS 和 ODS 的 DTS 和 PTS 来实施同步控制，将 DS 分配到再现时间轴上。

首先介绍用 PCS 和 ODS 的 DTS 和 PTS 来实施同步控制。

DTS 表示 ODS 的解码处理的开始时间，具有的精度是 90 KHz。
PTS 表示 ODS 的解码处理的结束时间，具有的精度是 90 KHz。

解码处理包括：对 ODS 进行解码；然后，将通过解码而生成的解压缩图形对象传输到对象缓冲器。该解码处理并不是瞬间结束的，而是需要一定的时间量。ODS 的 DTS 和 PTS 分别表示 ODS 的解码开始时间和解码结束时间，以指定解码处理的开始和结束。

由于 PTS 所示的时间是最终期限，所以，需要在 PTS 所示时间之前对 ODS 进行解码并将解压缩的图形对象存储到对象缓冲器中。

DS_n 中任一 ODS_j 的解码开始时间用 DTS(DS_n[ODS_j])表示，其精度是 90 KHz。因此，DS_n 中 ODS_j 的解码结束时间（即 PTS(DS_n[ODS_j])）是 DTS(DS_n[ODS_j])与解码处理所需最长时间之和。

假设 SIZE(DS_n[ODS_j])表示 ODS_j 的尺寸，R_d 表示 ODS 解码速率。那么，解码处理所需的最长时间（以秒为单位）是 SIZE(DS_n[ODS_j])/R_d。符号“//”表示一种将小数部分取整的除法运算符。

通过将该最长时间转换为 90 KHz 的精度并将该结果加上 ODS_j 的 DTS，则可以计算出 PTS 指定的 ODS_j 的解码结束时间，精度为 90 KHz。

DS_n 中 ODS_j 的该 PTS 可用下面的公式表示：

$$\text{PTS}(\text{DS}_n[\text{ODS}_j]) = \text{DTS}(\text{DS}_n[\text{ODS}_j]) + 90000 \times (\text{SIZE}(\text{DS}_n[\text{ODS}_j]) // R_d)$$

此外，DS_n 中两个相邻的 ODS（ODS_j 和 ODS_{j+1}）需要满足下面的关系：

$$\text{PTS}(\text{DS}_n[\text{ODS}_j]) \leq \text{DTS}(\text{DS}_n[\text{ODS}_{j+1}])$$

DS_n 中的 END 表示 DS_n 的结束。因此，END 表示 DS_n 中最后

一个 ODS (ODSlast) 的解码结束时间。ODSlast 的 PTS (PTS(DSn[ODSlast])) 表示 ODSlast 的解码结束时间, 所以, 如下设置 END 的 PTS:

$$\text{PTS}(\text{DSn}[\text{END}]) = \text{PTS}(\text{DSn}[\text{ODSlast}])$$

同时, 如下设置 DSn 中的 PCS 的 DTS 和 PTS。

PCS 的 DTS 表示 DSn 中顶部 ODS (ODS1) 的解码开始时间或比它更早的时间。这是因为, 需要将 PCS 加载到再现装置 200 的缓冲器中, 与 ODS1 的解码开始时间 (DTS(DSn[ODS1])) 和 DSn 中的顶部 PDS(PDS1)变为无效时 (PTS(DSn[PDS1])) 相同或更早。也就是说, PCS 的 DTS 需要满足下面的公式:

$$\text{DTS}(\text{DSn}[\text{PCS}]) \leq \text{DTS}(\text{DSn}[\text{ODS1}])$$

$$\text{DTS}(\text{DSn}[\text{PCS}]) \leq \text{PTS}(\text{DSn}[\text{ODS1}])$$

另一方面, 如下计算 PCS 的 PTS:

$$\text{PTS}(\text{DSn}[\text{PCS}]) \geq \text{DTS}(\text{DSn}[\text{PCS}]) + \text{DECODEDURATION}(\text{DSn})$$

在这里, DECODEDURATION(DSn)表示 DSn 中的 PCS 所述的更新所用的全部图形对象的呈现和解码所需的时间。尽管 DECODEDURATION(DSn)不是固定值, 它也不会受诸如再现装置实现和状态差异之类因素的影响。当 DSn.PCS.OBJ[j]表示 DSn 中的 PCS 描述的画面合成所用的图形对象时, 可以通过以下方面来改变 DECODEDURATION(DSn): (i) 窗口清除所需的时间; (ii) 对 DSn.PCS.OBJ[j]进行解码所需的时间; (iii) 将 DSn.PCS.OBJ[j]写到图形平面上所需的时间。因此, 只要 Rd 和 Rc 是预先确定的, 那么, DECODEDURATION(DSn)就是相同的, 而不管再现装置的实现方式。因此, 在合成时, 计算上述各时间段的长度, 从而指定 PCS 的 PTS。

基于图 14 所示的程序, 执行 DECODEDURATION(DSn)的计算。图 15 以及图 16A 和 16B 的流程图示出了该程序的算法。下面结合这些附图解释 DECODEDURATION(DSn)的计算过程。在图 15 中, 调用函数 PLANEINITIALIZATIONTIME, 然后将返回值加上 decode_duration(S1)。函数 PLANEINITIALIZATIONTIME (图 16A) 是用于计算初始化图形平面从而产生 DSn 的显示所需时间的函数。

在步骤 S1 中, 用 DS_n、DS_n.PCS.OBJ[0]和 decode_duration 作为参数, 调用该 PLANEINITIALIZATIONTIME 函数。

图 16A 示出了 PLANEINITIALIZATIONTIME 函数的过程。在图中, initialize_duration 是一个变量, 用于表示 PLANEINITIALIZATIONTIME 函数的返回值。

步骤 S2 判断 DS_n 中的 PCS 的 composition_state 字段是不是 Epoch Start。如果 composition_state 字段是 Epoch Start (S2: 是, 图 14 中 DS_n.PCS.composition_state==EPOCH_START), 则将清除图形平面所需的时间设置为 initialize_duration (S3)。

如上所述, 假设从对象缓冲器到图形平面的传输速率 R_c 是 256000000, 图形平面的总尺寸是(video_width)*(video_height)。那么, 清除图形平面所需的时间 (单位是秒) 是 (video_width)*(video_height)//256000000。将其乘以 90000 Hz, 用 PTS 精度表示。因此, 清除图形平面所需的时间是 90000 × (video_width)*(video_height)//256000000。将其加上 intialize_duration, 作为返回值返回。

如果 composition_state 字段不是 Epoch Start (S2: 否), 则对所有窗口 Window[i], 执行将清除 Window[i]所需的时间加上 initialize_duration 的操作 (S4)。如前所述, 假设从对象缓冲器到图形平面的传输速率 R_c 是 256000000, 并且, Window[i]的总尺寸是 $\sum \text{SIZE}(\text{WDS.WIN}[i])$ 。那么, 清除全部 Window[i]所需的时间 (单位是秒) 是 $\sum \text{SIZE}(\text{WDS.WIN}[i])//256000000$ 。将其乘以 90000 Hz, 用 PTS 精度表示。因此, 清除所有 Window[i]所需的时间是 $90000 \times \sum \text{SIZE}(\text{WDS.WIN}[i])//256000000$ 。将其加上 intialize_duration, 作为返回值返回。对 PLANEINITIALIZATIONTIME 函数的描述到此结束。

在图 15 中, 步骤 S5 判断 DS_n 中的图形对象的数量是 1 还是 2 (图 14 中的 if(DS_n.PCS.num_of_objects==2)、if(DS_n.PCS.num_of_objects==1))。如果 DS_n 中的图形对象的数量是 1 (S5:=1), 则将用于等待该图形对象解码结束的等待时间加上

decode_duration (S6)。通过调用 WAIT 函数，计算该等待时间（在图 14 中，
 (decode_duration+=WAIT(DSn,DSn.PCS.OBJ[0],decode_duration))）。
 使用 DSn、DSn.PCS.OBJ[0]和 decode_duration 作为参数，调用 WAIT 函数，并将表示等待时间的 wait_duration 作为返回值返回。

图 16B 示出了 WAIT 函数的过程。

在 WAIT 函数中，current_duration 是一个变量，将 decode_duration 设置为该参数，object_definition_ready_time 是表示 DSn 中的图形对象 OBJ[i]的 PTS 的变量。

此外，current_time 是一个变量，表示 current_duration 与 DSn 中的 PCS 的 DTS 之和。如果 object_definition_ready_time 大于 current_time（S7：是，图 14 中 if(current_time<object_definition_ready_time)），则将 object_definition_ready_time 和 current_time 之间的差值设为 wait_duration，然后作为返回值返回（S8，图 14 中 wait_duration+=object_definition_ready_time-current_time）。WAIT 函数到此结束。

再回到图 15 中，WAIT 函数的返回值与用于在 OBJ[0]所属窗口上进行呈现所需的时间（ $90000 \times \sum \text{SIZE}(\text{DSn.WDS.WIN}[0]) / 256000000$ ）被设置为 decode_duration (S9)。

上述过程涉及 DSn 中图形对象数量为 1 的情况。如果图形对象的数量是 2（S5:=2，图 14 中的 if(DSn.PCS.num_of_objects==2)），则使用 DSn、DSn.PCS.OBJ[0]和 decode_duration 作为参数，调用 WAIT 函数，然后将 WAIT 函数的返回值加到 decode_duration 中（S10）。

步骤 S11 判断 OBJ[0]所属的窗口与 OBJ[1]所属的窗口是否相同（在图 14 中，if(DSn.PCS.OBJ[0].window_id==DSn.PCS.OBJ[1].window_id)）。如果判断结果是肯定的（S11：是），则用 DSn、DSn.PCS.OBJ[1]和 decode_duration 作为参数，调用 WAIT 函数，然后将 WAIT 函数的返回值加上 decode_duration (S12)。此外，将在 OBJ[0]和 OBJ[1]所属窗口上进行呈现所需的时间

($90000 \times \sum \text{SIZE}(\text{DSn.WDS.WIN}[0].\text{window_id}) // 256000000$) 加上 decode_duration (S13)。

另一方面, 如果判断结果是否定的 (S11: 否), 则将在 OBJ[0] 所属窗口上进行呈现所需的时间 ($90000 \times \sum \text{SIZE}(\text{DSn.WDS.OBJ}[0].\text{window_id}) // 256000000$) 加上 decode_duration (S15)。此后, 用 DS_n、DS_n.PCS.OBJ[1] 和 decode_duration 作为参数, 调用 WAIT 函数, 然后将 WAIT 函数的返回值加上 decode_duration (S16)。此外, 将在 OBJ[1] 所属窗口上进行呈现所需的时间 ($90000 \times \sum \text{SIZE}(\text{DSn.WDS.OBJ}[1].\text{window_id}) // 256000000$) 加上 decode_duration (S17)。这样, 计算出 $\text{DECODEDURATION}(\text{DSn})$ 。

下面通过具体的例子, 说明如何设置一个 DS 中的 PCS 的 PTS。

在图 17A 示出的情形中, 与一个 ODS (ODS1) 相对应的一个 OBJ (OBJ1) 属于一个窗口。图 17B 和 17C 示出了图 14 中使用的参数之间的关系时序图。这些时序图都有三级。在这三级中, “图形平面访问” 级和 “ODS 解码” 级表示当再现 ODS 时并行执行的两个过程。上述算法基于这两个过程并行执行的假设。

图形平面访问由清除时间 (1) 和写入时间 (3) 构成。清除时间 (1) 表示用于清除整个图形平面所需的时间 ($90000 \times ((\text{图形平面尺寸}) // 256000000)$), 或用于清除图形平面上所有窗口所需的时间 ($\sum (90000 \times ((\text{窗口}[i]\text{的尺寸}) // 256000000))$)。

写入时间 (3) 表示在整个窗口上进行呈现所需的时间 ($90000 \times ((\text{窗口尺寸}) // 256000000)$)。

ODS 解码由解码时间 (2) 构成。解码时间 (2) 表示从 ODS1 的 DTS 到 PTS 的时间周期。

清除时间 (1)、解码时间 (2) 和写入时间 (3) 的变化取决于要清除的范围、要解码的 ODS 的尺寸和要写入图形平面的图形对象的尺寸。在图 17 中, 为简单起见, 假设解码时间 (2) 的开始与清除时间 (1) 的开始相同。

图 17B 所示的情形中, 解码时间 (2) 比清除时间 (1) 要长。

在这种情况下，`decode_duration` 是解码时间（2）与写入时间（3）之和。

图 17C 所示的情形中，清除时间（1）比解码时间（2）要长。在这种情况下，`decode_duration` 是清除时间（1）与写入时间（3）之和。

在图 18A 至 18C 所示情形中，与两个 ODS（ODS1 和 ODS2）相对应的两个 OBJ（OBJ1 和 OBJ2）属于一个窗口。在图 18B 和 18C 中，解码时间（2）表示对 ODS1 和 ODS2 进行解码所需的总时间。同样，写入时间（3）表示将 OBJ1 和 OBJ2 写入图形平面所需的总时间。尽管 ODS 的数量是 2，但是可以采用与图 17 相同的方式计算 `decode_duration`。详细地讲，如果 ODS1 和 ODS2 的解码时间（2）比清除时间（1）长，则 `decode_duration` 是解码时间（2）与写入时间（3）之和，如图 18B 所示。

如果清除时间（1）比解码时间（2）长，则 `decode_duration` 是清除时间（1）与写入时间（3）之和，如图 18C 所示。

图 19A 至 19C 所示的情形中，OBJ1 属于 Window1，而 OBJ2 属于 Window2。在这种情形中，如果清除时间（1）比 ODS1 和 ODS2 的解码时间（2）长，则 `decode_duration` 是清除时间（1）与写入时间（3）之和。另一方面，如果清除时间（1）比解码时间（2）短，则可以将 OBJ1 写入 Window1，而不必等待解码时间（2）的结束。在这种情况下，`decode_duration` 不仅仅是解码时间（2）与写入时间（3）之和。假设写入时间（31）表示将 OBJ1 写入 Window1 所需的时间，写入时间（32）表示将 OBJ2 写入 Window2 所需的时间。在图 19B 所示情形中，解码时间（2）比清除时间（1）与写入时间（31）之和要长。在这种情况下，`decode_duration` 是解码时间（2）与写入时间（32）之和。

图 19C 所示的情形中，清除时间（1）与写入时间（31）之和比解码时间（2）长。在这种情况下，`decode_duration` 是清除时间（1）、写入时间（31）以及写入时间（32）之和。

根据播放机模型，图形平面的尺寸是固定的。窗口和 ODS 的尺

寸和数量也是在合成时预先设置的。因此，计算出来的 `decode_duration` 可能是以下之一：清除时间（1）与写入时间（3）之和；解码时间（2）与写入时间（3）之和；解码时间（2）与写入时间（32）之和；清除时间（1）、写入时间（31）与写入时间（32）之和。通过基于如此计算出的 `decode_duration` 来设置 PCS 的 PTS，可以使图形与图像数据同步，且具有高精度。如此准确的同步控制是通过定义窗口和限制窗口中的清除和呈现操作来实现的。因此，在创作中，对概念“窗口”的介绍是非常重要的。

下面介绍如何设置 DS_n 中 WDS 的 DTS 和 PTS。设置 WDS 的 DTS，以满足下面的公式：

$$DTS(DS_n[WDS]) \geq DTS(DS_n[PCS])$$

WDS 的 PTS 表示启动向图形平面写入的最后期限。由于向图形平面写入只限于一个窗口，所以，通过从 PCS 的 PTS 所示时间减去在所有窗口上进行呈现所需的时间，可以确定启动向图形平面写入的时间。假设 $\sum SIZE(WDS.WIN[i])$ 是 Windows[i] 的总尺寸。那么，用于在所有 Windows[i] 上进行清除和呈现所需的时间是 $\sum SIZE(WDS.WIN[i])/256000000$ 。用 90000 KHz 的精度表达该时间，得到 $90000 \times \sum SIZE(WDS.WIN[i])/256000000$ 。

因此，可以如下计算 WDS 的 PTS：

$$PTS(DS_n[WDS]) = PTS(DS_n[PCS]) - 90000 \times \sum SIZE(WDS.WIN[i])/256000000$$

由于 WDS 的 PTS 表示最后期限，所以，可以在该 PTS 所示的时间之前启动向图形平面的写入。也就是说，在属于这两个窗口之一的一个 ODS 解码结束之后，可以将通过解码获得的图形对象立即写入图 19 所示的窗口中。

因此，使用 WDS 的 DTS 和 PTS，可以将窗口分配给 AV 剪辑的再现时间轴上的预期点。对 DS_n 中 PCS 和 WDS 的 DTS 和 PTS 的介绍到此结束。

下面解释 PCS 的前向引用。前向引用是一种再现控制，其引用已经存储在再现装置 200 中的图形对象。图 20 示出了前向引用 PCS。

图中示出了三个 DS，即 DS_n 、 DS_{n+1} 和 DS_{n+2} 。 DS_{n+1} 和 DS_{n+2} 的 PCS 是前向引用 PCS。箭头 yy_1 和 yy_2 表示由这些 PCS 做出的前向引用。这些 PCS 前向引用 DS_n 中 $ODS\#1$ 至 $ODS\#v$ 中定义的图形对象。在这里， DS_n 包含多个图形对象，即 $ODS\#1$ 至 $ODS\#u$ 和 $ODS\#u+1$ 至 $ODS\#v$ 。在这些图形对象中， DS_n 的 PCS 引用 $ODS\#1$ 至 $ODS\#u$ ，但 DS_n 的 PCS 未引用 $ODS\#u+1$ 至 $ODS\#v$ 。 DS_{n+1} 和 DS_{n+2} 前向引用未被 DS_n 的 PCS 引用的这些图形对象。

一个 DS 中的 PCS 前向引用的图形对象已经存储在再现装置 200 中，所以，不必再等待 ODS 的解码。因此，只要向再现装置 200 输入 PCS，就可以执行基于该 PCS 的显示控制。由于当向再现装置 200 输入 PCS 时立即执行显示控制，所以，可以在很短的时间间隔内，执行用新图形替换现有图形的图形更新。当运动图像的再现进行时，通过重复如此短间隔的图形更新，就可以流畅地实现运动图形的显示控制。下面描述两个前向引用 PCS 的示例。

首先描述用于在屏幕上实现运动图形显示效果的前向引用 PCS。图 21A 示出了一个具有此显示效果的时元。该时元由九个 DS 构成 (DS_0 至 DS_8)。该时元的第一个 DS，即 DS_0 ，包括 PCS、PDS 和 ODS。 DS_0 中的 ODS 定义如图 22A 所示的字幕“My heart is fluttering”，且具有 `object_id=1`。同时， DS_0 后面的 DS_1 至 DS_8 都仅包括一个 PCS。

图 21B 示出了 DS_1 至 DS_8 中的 PCS 的 PTS 和 `composition_object` 字段。如图所示， DS_1 至 DS_8 中的 PCS 指明在由 `object_horizontal_position` 和 `object_vertical_position` 表示的相应坐标 (x_1, y_1) 、 (x_2, y_2) 、 (x_3, y_3) 、……、 (x_8, y_8) 上显示由 `object_id_ref=1` 表示的图形对象。图 22B 示出了由窗口定义的坐标系中的坐标 (x_1, y_1) 、 (x_2, y_2) 、 (x_3, y_3) 、……、 (x_8, y_8) 的位置。从图中可以看出，坐标 (x_1, y_1) 、 (x_2, y_2) 、 (x_3, y_3) 、……、 (x_8, y_8) 位于一条波浪线上。设置坐标 (x_1, y_1) 、 (x_2, y_2) 、 (x_3, y_3) 、……、 (x_8, y_8) ，以使得图形对象位于 WDS 定义的窗口之内。这是因为，如果图形对象不包含在窗口之内，则无法建立图形流的图形显示和视频流的视频显示之间的同步。

在图 21B 中， DS_1 至 DS_8 中的 PCS 的 PTS 指定视频流的再现时

间轴上的时间 t_1 、 t_2 、 t_3 、…… t_8 ，在这些时间，在相应的坐标 (x_1, y_1) 、 (x_2, y_2) 、 (x_3, y_3) 、……、 (x_8, y_8) 上显示图形对象。

图 23 示出了向再现装置 200 顺序地输入具有上述 PCS 的 DS1 至 DS8 时的显示效果。在视频流的再现时间轴上的 t_1 处，向再现装置输入 DS1，从而，在窗口内的坐标 (x_1, y_1) 处显示图形。在 t_4 处，向再现装置输入 DS4，从而，在窗口内的坐标 (x_4, y_4) 处显示图形。在 t_6 处，向再现装置输入 DS6，从而，在窗口内的坐标 (x_6, y_6) 处显示图形。在 t_8 处，向再现装置输入 DS8，从而，在窗口内的坐标 (x_8, y_8) 处显示图形。由于坐标 (x_1, y_1) 、 (x_4, y_4) 、 (x_6, y_6) 和 (x_8, y_8) 位于一条波浪线上，所以，图形会沿着波浪线，如图 24 所示。

因此，当向再现装置 200 顺序地输入包括 PCS（各具有指定的 `object_horizontal_position` 和 `object_vertical_position`）的这些 DS 时，再现装置 200 根据这些 PCS，显示对象缓冲器中已经存储的图形对象。由于只需通过向再现装置 200 输入控制信息就可以改变图形的位置，所以，可以在屏幕上实现图形的快速运动。

当向再现装置 200 输入这些 DS 时，新的 PCS 可能会覆盖现有的 PCS。这使得再现装置 200 只保存最新的 PCS，即便有数十个或数百个 PCS 给出流畅运动的图形的显示坐标。这样，可以降低再现装置 200 中的存储器占用率。

作为前向引用 PCS 的另一示例，下面描述仅调色板显示更新。图 25 示出了用于实现仅调色板显示更新的 DS 序列。在图中，第一级表示仅调色板显示更新的 DS 序列（DS0、DS1、DS2 和 DS3）。DS0 有一个 PCS、四个 PDS（PDS0、PDS1、PDS2 和 PDS3）和一个 ODS。DS0 后面的 DS1、DS2 和 DS3 各仅有一个 PCS。

DS0 中的 ODS 定义字幕 “I will never forget”，如指导线 hv1 所示。单词 “I”、“will”、“never” 和 “forget” 各由多个游程数据（图 25 所示的矩形表示）构成。构成 “I” 的游程数据具有像素代码 0，构成 “will” 的游程数据具有像素代码 1，构成 “never” 的游程数据具有像素代码 2，构成 “forget” 的游程数据具有像素代码 3。

图 26A 示出了 DS0 中的四个 PDS。

PDS0 是 DS0 中的第一个 PDS (在第一级中), 具有 `palette_id=0`, 它为像素代码 0 分配红色, 为像素代码 1 至 3 分配白色。

PDS1 是 DS0 中的第二个 PDS (在第二级中), 具有 `palette_id=1`, 它为像素代码 0 和 1 分配红色, 为像素代码 2 和 3 分配白色。

PDS2 是 DS0 中的第三个 PDS (在第三级中), 具有 `palette_id=2`, 它为像素代码 0、1 和 2 分配红色, 为像素代码 3 分配白色。

PDS3 是 DS0 中的第四个 PDS (在第四级中), 具有 `palette_id=3`, 它为像素代码 0 至 3 分配红色。

图 26B 示出了 DS0 至 DS3 中的 PCS。

DS0 中的 PCS (在第一级中) 用 `object_id_ref=1` 表示的图形对象指定要执行的显示控制。该 PCS 具有 `palette_update_flag=0` 和 `palette_id_ref=0`。因此, 当向再现装置 200 输入 DS0 时, 用 `palette_id=0` 表示的 PDS 显示 DS0 中 ODS 表示的图形对象。

DS1 中的 PCS (在第二级中) 用 `object_id_ref=1` 表示的图形对象指定要执行的显示控制。该 PCS 具有 `palette_update_flag=1` 和 `palette_id_ref=1`。因此, 当向再现装置 200 输入 DS1 时, 用 `palette_id=1` 表示的 PDS 更新图形对象的显示。

DS2 中的 PCS (在第三级中) 用 `object_id_ref=1` 表示的图形对象指定要执行的显示控制。该 PCS 具有 `palette_update_flag=1` 和 `palette_id_ref=2`。因此, 当向再现装置 200 输入 DS2 时, 用 `palette_id=2` 表示的 PDS 更新图形对象的显示。

DS3 中的 PCS (在第四级中) 用 `object_id_ref=1` 表示的图形对象指定要执行的显示控制。该 PCS 具有 `palette_update_flag=1` 和 `palette_id_ref=3`。因此, 当向再现装置 200 输入 DS3 时, 用 `palette_id=3` 表示的 PDS 更新图形对象的显示。通过如此设置 DS0 至 DS3 中的 PCS, 可以用不同的调色板数据重复地更新属于 DS0 的图形对象。

图 27 示出了当向再现装置 200 顺序地输入这四个 DS 时所达到的显示效果。当向再现装置输入 DS0 时, 显示字幕 “I will never forget”, 其中的单词 “I” 用红色显示。当向再现装置输入 DS1 时, 单词 “will” 变红。当向再现装置输入 DS2 时, 单词 “never” 变红。

当向再现装置输入 DS3 时, 单词 “never” 变红, 因此, 整个字幕 “I will never forget” 以红色显示。所以, 当从 BD-ROM 100 向再现装置 200 顺序地输入 DS 时, 构成字幕 “I will never forget” 的单词从最左边的单词开始, 逐个从白色变成红色。当音频流的再现进行时, 这产生字幕颜色变化的显示效果。

应当注意的是, 上述 DS 的数据结构 (PCS、WDS、PDS 和 ODS) 是用程序语言编写的类结构的例子。创作者根据蓝光只读光盘格式中定义的句法编写这些类结构, 从而在 BD-ROM 100 上创建这些数据结构。对根据本发明第一个实施例的记录介质的描述到此结束。下面描述根据本发明第一个实施例的再现装置。图 28 示出了再现装置 200 的内部构造。再现装置 200 是基于该内部构造制造的。再现装置 200 主要包括三个部分: 系统 LSI; 驱动装置; 微处理器系统。可以将这些部件装配到该装置的壳体和衬底上, 从而制造再现装置 200。系统 LSI 是集成电路, 包括各种处理单元, 用于实现再现装置 200 的功能。再现装置 200 包括: BD 驱动器 1; 读取缓冲器 2; PID 滤波器 3; 传输缓冲器 4a、4b 和 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。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; 图形控制器 17。

BD 驱动器 1 执行 BD-ROM 100 的加载、读取和弹出。BD 驱动器 1 访问 BD-ROM 100。

读取缓冲器 2 是 FIFO (先进先出) 存储器。因此, 从 BD-ROM 100 读取的 TS 数据包在读取缓冲器 2 中消除的次序与它们的到达次序相同。

PID 滤波器 3 对从读取缓冲器 2 输出的 TS 数据包执行滤波。更详细地讲, PID 滤波器 3 只把具有预定 PID 的 TS 数据包传递给传输缓冲器 4a、4b 和 4c。PID 滤波器 3 中没有缓冲机制。因此, 进入 PID 滤波器 3 的 TS 数据包立即被写入传输缓冲器 4a、4b 和 4c。

传输缓冲器 4a、4b 和 4c 是 FIFO 存储器, 用于存储从 PID 滤波器 3 输出的 TS 数据包。传输速度 Rx 表示从传输缓冲器 4a 读取 TS

数据包的速度。

外围电路 4d 具有连线逻辑，用于把从传输缓冲器 4a 读取的 TS 数据包转换成功能段。然后，将功能段存储在编码数据缓冲器 13 中。

视频解码器 5 对从 PID 滤波器 3 输出的 TS 数据包进行解码，从而获得解压缩的图片，并将其写入视频平面 6。

视频平面 6 是用于存储运动图像的平面存储器。

音频解码器 7 对从 PID 滤波器 3 输出的 TS 数据包进行解码，并输出解压缩的音频数据。

图形平面 8 是一个平面存储器，它具有一幅画面的存储区域，能够存储一幅画面的解压缩图形。

CLUT 电路 9 基于 PDS 中所示的 Y、Cr 和 Cb 值，对图形平面 8 上的解压缩图形的索引色（index color）进行转换。

加法器 10 将 CLUT 电路 9 转换过的解压缩图形乘以 PDS 中所示的 T 值（透明色）。然后，加法器 10 对所得的解压缩图形和视频平面 6 上的解压缩图片数据中的相应像素执行加法，并输出所得的图像。

图形解码器 12 对图形流进行解码，从而获得解压缩的图形，并将解压缩的图形作为图形对象写入图形平面 8 中。对图形流进行解码之后，字幕和菜单就出现在屏幕上。

图形解码器 12 从对象缓冲器 15 中读取属于 DS_n 的图形对象，同时将属于 DS_{n+1} 的图形对象写入对象缓冲器 15，从而执行流水线处理。

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；图形控制器 17。

编码数据缓冲器 13 用于将功能段与 DTS 和 PTS 一起存储。通过将传输缓冲器 4a 中存储的各 TS 数据包去除 TS 包头和 PES 包头以及将剩余的有效负载顺序地进行排列，得到这些功能段。去除的 TS 包头和 PES 包头中包含的 DTS 和 PTS 存储在编码数据缓冲器 13 中，与这些功能段相对应。

外围电路 13a 具有连线逻辑，用于将来自编码数据缓冲器 13 的数据传输到流图形处理器 14，并将来自编码数据缓冲器 13 的数据传

输到合成缓冲器 16。更详细地讲，如果当前时间到达 ODS 的 DTS，则外围电路 13a 将 ODS 从编码数据缓冲器 13 传输到流图形处理器 14。此外，如果当前时间到达 PCS 或 PDS 的 DTS，则外围电路 13a 将 PCS 或 PDS 从编码数据缓冲器 13 传输到合成缓冲器 16。

流图形处理器 14 对 ODS 进行解码，从而获得具有索引色的解压缩图形，并将解压缩图形作为图形对象传输到对象缓冲器 15。流图形处理器 14 的解码是瞬时的，并且，通过解码获得的图形对象暂时存储在流图形处理器 14 中。尽管流图形处理器 14 的解码是瞬时的，但图形对象从流图形处理器 14 到对象缓冲器 15 的传输不是瞬时的。这是因为，对于蓝光只读光盘格式的播放机模型，到对象缓冲器 15 的传输是以 128 Mbps 的传输速率执行的。DS 中的 END 的 PTS 给出了属于 DS 的所有图形对象到对象缓冲器 15 的传输的结束。因此，在 END 的 PTS 所示的时间之前，不会启动下一 DS 的处理。通过对各 ODS 进行解码而得到的图形对象向对象缓冲器 15 的传输开始于 ODS 的 DTS 所示的时间，结束于 ODS 的 PTS 所示的时间。

如果 DS_n 的图形对象和 DS_{n+1} 的图形对象具有不同的 Object_id，那么，流图形处理器 14 将这两个图形对象写入对象缓冲器 15 中不同的存储区域。这样，就可以实现图形对象的流水线呈现，并且，DS_n 的图形对象不会被 DS_{n+1} 的图形对象覆盖。另一方面，如果 DS_n 的图形对象和 DS_{n+1} 的图形对象具有相同的 Object_id，那么，流图形存储器 14 将 DS_{n+1} 的图形对象写入对象缓冲器 15 中也存储了 DS_n 的图形对象的存储区域，从而覆盖 DS_n 的图形对象。在这种情况下，不执行流水线处理。DS 也可以包括由该 DS 的 PCS 引用的 ODS 以及 PCS 未引用的 ODS。流图形处理器 14 不仅顺序地对该 PCS 引用的 ODS 进行解码，而且还对 PCS 未引用的 ODS 进行解码，然后，将解码所得的图形存储到对象缓冲器 15 中。

对象缓冲器 15 对应于 ETSI EN 300 743 中的像素缓冲器。流图形处理器 14 解码的图形对象存储在对象缓冲器 15 中。对象缓冲器 15 的尺寸应当是图形平面 8 的尺寸的两倍或四倍。这是因为，为了实现滚读 (scrolling)，对象缓冲器 15 需要能够存储图形平面 18 两倍

或四倍的图形。

合成缓冲器 16 用于存储 PCS 和 PDS。当 DS_n 和 DS_{n+1} 中的 PCS 的激活时间重叠时，组合缓冲器 16 存储 DS_n 中的 PCS 和 DS_{n+1} 中的 PCS。

图形控制器 17 对合成缓冲器 16 中的 PCS 进行解码。基于解码结果，图形控制器 17 将 DS_{n+1} 的图形对象写入对象缓冲器 15，同时从对象缓冲器 15 中读取 DS_n 的图形对象，并将其呈现以供显示。在 DS_n 中的 PCS 的 PTS 所示的时间，执行图形控制器 17 的呈现。

为了用 PCS 产生画面合成，图形控制器 17 执行如下处理。图形控制器 17 从 PCS 中读取 object_id_ref，object_id_ref 是图形对象的标识符的引用值。图形控制器 17 从图形对象中取出由 object_cropping_horizontal_position、object_cropping_vertical_position、object_cropping_width 和 object_cropping_height 定义的部分。图形控制器 17 将取出的部分写在图形平面 8 上由 object_horizontal_position、object_vertical_position 表示的位置处。如果该引用值指定的图形对象属于与该 PCS 相同的 DS，则在对该 DS 中相应的 ODS 进行解码结束之前，不能执行上述的向图形平面 8 的写入。

另一方面，当 PCS 的 composition_state 是 Normal Case 时，图形对象是已经解码的，并存储在对象缓冲器 15 中。因此，可以执行上述的向图形平面 8 的写入，而不必等待 ODS 的解码。

为了实现仅调色板显示更新，图形控制器 17 执行以下处理。从 BD-ROM 100 中读出 DS 后，图形控制器 17 判断该 DS 中的 PCS 的 palette_update_flag 是否为“1”。如果 palette_update_flag 为“1”，则图形控制器 17 指示 CLUT 单元 9 使用该 PCS 的 palette_id 表示的调色板数据来执行像素值转换。然后，图形控制器 17 从对象缓冲器 15 中读取图形对象，并将其写入图形平面 8。因此，仅仅通过设置 CLUT 单元 9 中的调色板数据，就可以执行图形更新。

下面给出建议的传输速率以及 PID 滤波器 3、传输缓冲器 4a、4b 和 4c、图形平面 8、CLUT 单元 9、编码数据缓冲器 13、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 的缓冲器尺

寸。图 29 示出了传输速率 R_x 、 R_c 和 R_d 以及图形平面 8、传输缓冲器 4a、编码数据缓冲器 13 和对象缓冲器 15 的尺寸。

从对象缓冲器 15 到图形平面 8 的传输速率 R_c （像素合成速率）是再现装置 200 中最高的传输速率，根据窗口大小和帧速率，将其计算为 256 Mbps（ $=500 \text{ K 字节} \times 29.97 \times 2$ ）。

与 R_c 不同，从流图形处理器 14 到对象缓冲器 15 的传输速率 R_d （像素解码速率）不必与帧速率一致，它可以是 R_c 的 1/2 或 1/4。因此，传输速率 R_d 是 128 Mbps 或 64 Mbps。

从传输缓冲器 4a 到编码数据缓冲器 13 的传输速率 R_x （传输缓冲器泄漏速率）ODS 处于压缩状态下的传输速率。因此，通过用 ODS 的压缩率乘以 R_d ，可以计算出传输速率 R_c 。例如，如果压缩率为 25%，则传输速率 R_x 是 16 Mbps（ $=64 \text{ Mbps} \times 25\%$ ）。

给出的这些传输速率和缓冲器尺寸只是最低标准，传输速率和缓冲器大小也可以大于图 29 所示情况。

在如上构造的再现装置 200 中，组成部件以流水线方式执行处理。

图 30 的时序图示出了再现装置 200 中执行的流水线处理。在图中，第五级示出了 BD-ROM 100 上的 DS。第四级示出了 PCS、WDS、PDS、ODS 和 END 向编码数据缓冲器 13 的写入时间。第三级示出了流图形处理器 14 对 ODS 的解码时间。第二级示出了合成缓冲器 16 的存储内容。第一级示出了图形控制器 17 的操作。

ODS1 和 ODS2 的 DTS 分别给出了 t_{31} 和 t_{32} 。因此，在 t_{31} 和 t_{32} 之前，需要将 ODS1 和 ODS2 缓冲在编码数据缓冲器 13 中。因此，在 t_{31} 之前，ODS1 向编码数据缓冲器 13 的写入结束，在 t_{31} ，解码时间 $dp1$ 开始，在 t_{31} 之前，ODS2 向编码数据缓冲器 13 的写入结束，在 t_{31} ，解码时间 $dp2$ 开始。

同时，ODS1 和 ODS2 的 PTS 分别给出了 t_{32} 和 t_{33} 。因此，在 t_{32} 之前，流图形处理器 14 对 ODS1 的解码结束，在 t_{33} 之前，流图形处理器 14 对 ODS2 的解码结束。因此，在 ODS 的 DTS 给出的时间之前，ODS 缓存在编码数据缓冲器 13 中，在 ODS 的 PTS 所示的

时间之前,对所缓冲的 ODS 进行解码并将其传输到对象缓冲器 15。

在第一级中,cd1 表示图形控制器 17 清除图形平面 8 所需的时间,td1 表示图形控制器 17 将对象缓冲器 15 中包含的图形写入图形平面 8 所需的时间。WDS 的 PTS 表示启动写入图形的最后期限。PCS 的 PTS 表示向图形平面 8 中写入图形结束并呈现所写入图形以供显示的时间。因此,在 PCS 的 PTS 所示的时间处,获得图形平面 8 上的一幅画面的解压缩图形。CLUT 单元 9 对解压缩的图形执行颜色转换,加法器 10 将该图形覆盖在视频平面 6 上存储的解压缩图片上。从而产生所得的图像。

在图形解码器 12 中,当图形控制器 17 清除图形平面 8 时,流图形处理器 14 继续解码。该流水线处理的结果是,可以快速地显示图形。

图 30 示出了在 ODS 解码之前图形平面 8 的清除结束的例子。另一方面,图 31 的时序图示出了 ODS 的解码在清除图形平面 8 之前结束的流水线处理。在这种情况下,ODS 解码结束之后,还不能把通过解码获得的图形写入图形平面 8。只有图形平面 8 的清除结束之后,才可以将该图形写入图形平面 8 中。

图 32 的时序图示出了再现装置 200 中的缓冲器占用率的变化。在图中,第一至第四级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 的占用率。这些变化是用线图的形式表示的,其中,水平轴表示时间,垂直轴表示占用率。

第四级表示合成缓冲器 16 的占用率的变化。如图所示,合成缓冲器 16 的占用率的变化包括单调增加 Vf0,单调增加 Vf0 反映将从编码数据缓冲器 13 输出的 PCS 进行存储。

第三级显示编码数据缓冲器 13 的占用率的变化。如图所示,数据缓冲器 13 的占用率的变化包括单调增加 Vf1 和 Vf2 以及单调减少 Vg1 和 Vg2,单调增加 Vf1 和 Vf2 反映 ODS1 和 ODS2 的存储,单调减少 Vg1 和 Vg2 反映流图形处理器 14 顺序地读取 ODS1 和 ODS2。单调增长 Vf1 和 Vf2 的斜率基于从传输缓冲器 4a 到编码数据缓冲器 13 的传输速率 Rx,而单调减少 Vg1 和 Vg2 是瞬时的,因为流图形处

理器 14 的解码是瞬时执行的。也就是说，流图形处理器 14 瞬时地对每个 ODS 进行解码，并保存通过解码获得的解压缩图形。由于从流图形处理器 14 到对象缓冲器 15 的传输速率 R_d 是 128 Mbps，所以，对象缓冲器 15 的占用率以 128 Mbps 增加。

第二级给出了对象缓冲器 15 的占用率的变化。如图所示，对象缓冲器 15 的占用率的变化包括单调增加 V_{h1} 和 V_{h2} ，单调增加 V_{h1} 和 V_{h2} 反映将从流图形处理器 14 输出的 ODS1 和 ODS2 的图形对象进行存储。单调增加 V_{h1} 和 V_{h2} 的斜率基于从流图形处理器 14 到对象缓冲器 15 的传输速率 R_d 。ODS1 和 ODS2 的解码时间分别对应于第三级中出现单调减少和第二级中出现单调增加的时间段。ODS 的 DTS 表示解码时间的开始，而 ODS 的 PTS 表示解码时间的结束。如果在 ODS 的 PTS 显示的时间之前将解压缩图形对象传输到对象缓冲器 15，则 ODS 的解码结束。在 ODS 的 PTS 所示时间之前将解压缩图形对象存储在对象缓冲器 15 中是很重要的。只要满足这一点，解码时间中的单调减少和单调增加不限于图 32 所示情形。

第一级示出了图形平面 8 的占用率的变化。如图所示，图形平面 8 的占用率的变化包括单调增加 V_{f3} ，单调增加 V_{f3} 反映将从对象缓冲器 15 输出的图形对象进行存储。单调增加 V_{f3} 的斜率基于从对象缓冲器 15 到图形平面 8 的传输速率 R_c 。PCS 的 PTS 给出了单调增加 V_{f3} 的结束。

通过使用 ODS 的 DTS 和 PTS、PCS 的 DTS 和 PTS 以及图 29 所示的缓冲器尺寸和传输速率，可以创建图 32 所示的图。该图使得当再现 BD-ROM 100 上的 AV 剪辑时创作者能够了解缓冲器状态如何改变。

通过重写 DTS 和 PTS，可以调整缓冲器状态的这些变化。因此，创作者可以防止解码负载超过再现装置的解码器的规格，或者，防止再现期间的缓冲器溢出。这使得当开发再现装置 200 时硬件和软件的实现更容易。对再现装置 200 的内部构造的解释到此结束。

下面说明如何实现图形解码器 12。通过让一个通用 CPU 执行用于实现图 33 所示操作的程序，可以实现图形解码器 12。下面结合图

33, 描述图形解码器 12 的操作。

图 33 的流程图示出了功能段的加载操作。在图中, SegmentK 是一个变量, 表示一个属于 DS 并且在再现 AV 剪辑期间被读取的 Segment (PCS、WDS、PDS 或 ODS), 忽略标志表示是忽略 SegmentK 还是加载 SegmentK。在该流程图中, 在将忽略标志复位为 0 之后 (S20), 对于每个 SegmentK, 执行步骤 S21 到 S24 和 S27 到 S31 的循环 (S25 和 S26)。

步骤 S21 判断 SegmentK 是不是 PCS。如果 SegmentK 是 PCS, 则操作转入步骤 S27。

步骤 S22 判断忽略标志是 0 还是 1。如果忽略标志是 0, 则操作转入步骤 S23。如果忽略标志是 1, 则操作转入步骤 S24。在步骤 S23 中, 将 SegmentK 加载到编码数据缓冲器 13 中。

如果忽略标志是 1 (S22: 否), 则忽略 SegmentK (S24)。这导致在步骤 S22 中对属于该 DS 的所有功能段的否定判断, 因此, 忽略 DS 的所有功能段。

因此, 忽略标志指示要忽略或加载 SegmentK。执行步骤 S27 至 S31 和 S34 至 S35, 从而设置该忽略标志。

步骤 S27 判断 PCS 的 composition_state 字段是不是 Acquisition Point。如果 composition_state 字段是 Acquisition Point, 则操作转入步骤 S28。如果 composition_state 字段是 Epoch Start 或 Normal Case, 则操作转入步骤 S31。

步骤 S28 判断前一 DS 是否存在图形解码器 12 的某一缓冲器中 (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)。如果执行跳节操作的话, 则前面的 DS 不存在于图形解码器 12 中。在这种情况下, 需要从 Acquisition Point Ds 开始显示, 从而使操作转入步骤 S30 (S28: 否)。

在步骤 S30 中, 将忽略标志设置为 0, 于是操作转入步骤 S22。

另一方面, 如果执行正常再现的话, 则前面的 DS 存在图形解码器 12 中。在这种情况下, 操作转入步骤 S29 (S28: 是)。在步骤 S29 中, 将忽略标志设置为 1, 于是操作转入步骤 S22。

步骤 S31 判断 composition_state 字段是否为 Normal Case。如果 composition_state 字段是 Normal Case，则操作转入步骤 S34。如果 composition_state 字段为 Epoch Start，则操作转入步骤 S30，其中，将忽略标志设置为 0。步骤 S34 与步骤 S28 相同，判断前面的 DS 是否存在图形解码器 12 中。如果前面的 DS 是否存在，则将忽略标志设置为 0 (S30)。否则，则将忽略标志设置为 1，因为无法获得足以合成一幅画面图形的功能段 (S35)。这样，当前面的 DS 不存在于图形解码器 12 中时，忽略 Normal Case DS 的功能段。

下面结合图 34 给出加载 DS 的具体例子。在图 34 中，将三个 DS (DS0、DS1 和 DS2) 与视频进行复用。DS1 的 composition_state 字段是 Epoch Start，DS10 的 composition_state 字段是 Acquisition Point，DS20 的 composition_state 字段是 Normal Case。

假设对复用了这三个 DS 和视频的 AV 剪辑中的图片数据 pt10 执行跳节操作，如箭头 am1 所示。在这种情况下，最接近 pt10 的 DS10 受到图 33 所示操作的影响。DS10 的 composition_state 字段是 Acquisition Point (S27: 是)，但是，前面的 DS (DS1) 不存在于编码数据缓冲器 13 中 (S28: 否)。因此，将忽略标志设置为 0 (S30)。从而，将 DS10 加载到编码数据缓冲器 13 中，如图 35 中的箭头 md1 所示。另一方面，假设对 DS10 之后的图片数据执行跳节操作，如图 34 中的箭头 am2 所示。在这种情况下，DS20 是 Normal Case DS，之前的 DS (DS10) 不存在于编码数据缓冲器 13 中。因此，忽略 DS20，如图 35 中的箭头 md2 所示。

图 37 示出了当执行如图 36 所示的正常再现时如何加载 DS1、DS10 和 DS20。在这三个 DS 中，DS1 是一个 Epoch Start DS，被加载到编码数据缓冲器 13 中，如箭头 rd1 所示 (S23)。但是，对于作为 Acquisition Point DS 的 DS10，忽略标志被设置成 1 (S29)，所以，DS10 的功能段不会被加载到编码数据缓冲器 13 中，而是被忽略，如箭头 rd2 所示 (S24)。同时，作为 Normal Case DS 的 DS20 被加载到编码数据缓冲器 13 中，如箭头 rd3 所示 (S23)。

下面解释图形控制器 17 的操作。图 38 至 40 的流程图示出了图

形控制器 17 的操作。

步骤 S41 至 S44 构成一个主程序，其中，等待由步骤 S41 到 S44 中任意之一所表示的事件。

在图 38 中，步骤 S41 判断当前再现时间是否为 PCS 的 DTS。如果是，则执行步骤 S45 至 S53。

步骤 S45 判断 PCS 的 `composition_state` 字段是不是 Epoch Start。如果是，则在步骤 S46 中清除整个图形平面 8。否则，在步骤 S47 中清除由 WDS 的 `window_horizontal_position` 字段、`window_vertical_position` 字段、`window_width` 字段和 `window_height` 字段表示的窗口。

在步骤 S46 至 S47 之后执行步骤 S48，判断是否已经超过任一 ODSx 的 PTS。清除整个图形平面 8 需要花费很长时间，所以，在清除整个图形平面 8 之前，ODSx 的解码可能已经结束。步骤 S48 检查这种可能性。如果没有超过 ODSx 的 PTS，则操作返回到主程序。如果超过了 ODSx 的 PTS，则执行步骤 S49 至 S51。步骤 S49 判断 `object_cropped_flag` 字段是否为 0。如果是，则将与 ODSx 对应的图形对象设置为不显示 (S50)。

如果 `object_cropped_flag` 字段是 1，则将基于 `object_cropping_horizontal_position` 字段、`object_cropping_vertical_position` 字段、`cropping_width` 字段和 `cropping_height` 字段剪裁的图形对象写入图形平面 8 上的窗口中，所在位置用 `object_horizontal_position` 字段和 `object_vertical_position` 字段表示 (S51)。这样，就将图形对象写入了窗口。

步骤 S52 判断是否超过另一 ODS (ODSy) 的 PTS。如果在将 ODSx 的图形对象写入图形平面 8 的期间完成了 ODSy 的解码，则将 ODSy 设置为 ODSx (S53)，然后，操作返回步骤 S49。因此，对 ODSy 执行步骤 S49 至 S51。

在图 39 中，步骤 S42 判断当前再现时间是否为 WDS 的 PTS。如果是，则操作转入步骤 S54，判断窗口的数量是否为 1。如果窗口的数量为 2，则操作返回主程序。如果窗口的数量为 1，则执行步骤

S55 至 S59 的循环。在该循环中，对于窗口中要显示的最多两个图形对象中的每一个，执行步骤 S57 至 S59。步骤 S57 判断 `object_cropped_flag` 是否为 0。如果是，则将图形对象设置为不显示（S58）。

如果 `object_cropped_flag` 是 1，则将基于 `object_cropping_horizontal_position` 字段、`object_cropping_vertical_position` 字段、`cropping_width` 字段和 `cropping_height` 字段剪裁的图形对象写入图形平面 8 上的窗口中，所在位置用 `object_horizontal_position` 字段和 `object_vertical_position` 字段表示（S59）。该循环之后，一个或多个图形对象被写入窗口中。

步骤 S44 判断当前再现时间是否为 PCS 的 PTS。如果是，则操作转入步骤 S60，以判断 `palette_update_flag` 字段是否为 1。如果是，则将 `palette_id` 字段标识的调色板发送到 CLUT 单元 9（S61）。如果 `palette_update_flag` 字段为 0，则跳过步骤 S61。

此后，CLUT 单元对图形平面 8 上的图形执行颜色转换。然后，将图形覆盖到视频上（S62）。

在图 40 中，步骤 S43 判断当前时间是否为 ODS 的 PTS。如果是，则操作转入步骤 S63，以判断窗口数量是否为 2。如果窗口数量为 1，则操作返回主程序。

这里，步骤 S43 和 S63 中做出的判断具有以下意思。如果窗口数量为 2，则分别在两个窗口中显示两个图形对象。在这种情况下，每次 ODS 解码结束时，需要将通过解码得到的图形对象写入图形平面 8（参见图 19B）。因此，如果当前时间是 ODS 的 PTS 并且窗口数量是 2，则执行步骤 S64 至 S66，以将每个图形对象写入图形平面 8 中。步骤 S64 判断 `object_cropped_flag` 字段是否为 0。如果是，则将图形对象设置为不显示（S65）。

如果 `object_cropped_flag` 字段是 1，则将基于 `object_cropping_horizontal_position` 字段、`object_cropping_vertical_position` 字段、`cropping_width` 字段和 `cropping_height` 字段剪裁的图形对象写入图形平面 8 上的窗口中，所

在位置用 `object_horizontal_position` 字段和 `object_vertical_position` 字段表示 (S66)。重复该处理, 从而将两个图形对象分别写入这两个窗口。

(第二个实施例)

本发明的第二个实施例涉及第一个实施例中介绍的 BD-ROM 100 的制造过程。图 41 是 BD-ROM 100 的制造过程的流程图。

该制造过程包括: 材料生产步骤, 用于记录视频、声音等 (S201); 创作步骤, 使用创作设备创建应用格式 (S202); 印制 (pressing) 步骤, 用于创建 BD-ROM 100 的原始主盘 (original master), 并执行压制 (stamping) 和粘合 (bonding), 以完成 BD-ROM 100 (S203)。

在该制造过程中, 创作步骤包括步骤 S204 至 S213。

在步骤 S204 中, 生成控制信息、窗口定义信息、调色板定义信息和图形。在步骤 S205 中, 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。在步骤 S206 中, 基于要同步图片的时间, 设置每个 PCS 的 PTS。在步骤 S207 中, 基于 PTS[PCS] 设置 DTS[ODS] 和 PTS[ODS]。在步骤 S208 中, 基于 DTS[ODS], 设置 DTS[PCS]、PTS[PDS]、DTS[WDS] 和 PTS[WDS]。在步骤 S209 中, 用图表示播放机模型中每个缓冲器的占用率的变化。在步骤 S210 中, 判断图示的变化是否满足播放机模型的约束条件。如果判断结果是否定的, 则在步骤 S211 中, 重写每个功能段的 DTS 和 PTS。如果判断结果是肯定的, 则在步骤 S212 中生成图形流, 并在步骤 S213 中将图形流与视频流和音频流进行复用, 以形成 AV 剪辑。此后, 将 AV 剪辑改编为蓝光只读光盘格式, 从而完成该应用格式。

(修改)

尽管通过上面的实施例对本发明进行了描述, 但本发明不限于此。也可以用下面的修改 (A) 至 (P) 中任意之一来实现本发明。本申请的每项权利要求的发明都包括上述实施例的扩展和概括以及下面的修改。扩展和概括程度取决于做出本发明时本发明技术领域的

技术发展水平。

(A) 在上述实施例描述的情形中, 将 BD-ROM 用作记录介质。但是, 本发明的主要特征在于记录介质上记录的图形流, 而这并不依赖于 BD-ROM 的物理特性。因此, 本发明同样适用于能够记录图形流的任何记录介质。这种记录介质的示例包括: 光盘, 如 DVD-ROM、DVD-RAM、DVD-RW、DVD-R、DVD+RW、DVD+R、CD-R 或 CD-RW; 磁光盘, 如 PD 和 MO; 半导体存储卡, 如 CF 卡、Smartmedia 卡、记忆棒、多媒体卡或 PCM-CIA 卡; 磁记录盘, 如软盘、SuperDisk、Zip 和 Click!; 可移动硬盘, 如 ORB、Jaz、SparQ、SyJet、EZFley 和 Microdrive; 以及不可移动硬盘。

(B) 在上述实施例描述的情形中, 再现装置对 BD-ROM 上的 AV 剪辑进行解码, 并将解码后的 AV 剪辑输出到电视。当然, 再现装置也可以只有 BD 驱动器, 其他的组成部件由电视来提供。在这种情况下, 可以将再现装置和电视集成到通过 IEEE 1394 连接器相连的家庭网络中。在上述实施例描述的情形中, 再现装置连接到电视, 但是, 再现装置也可以与显示装置集成在一起。此外, 再现装置也可以只包括系统 LSI (集成电路), 其构成处理的实质性部分。该再现装置和该集成电路都是本说明书所描述的发明。因此, 不管涉及再现装置还是涉及集成电路, 基于第一个实施例中描述的再现装置的内部构造来制造再现装置的行为都是实施本发明的一种行为。此外, 有偿转让 (即销售) 或无偿转让 (即, 作为礼物)、出租和进口该再现装置也是实施本发明的行为。同样, 使用橱窗展示、商品目录或宣传册来许诺转让或出租该再现装置的行为也是实施本发明的行为。

(C) 使用上述流程图所示的程序的信息处理实际上是用硬件资源实现的。因此, 描述这些流程图中所示操作过程的程序本身也是一项发明。在上述实施例所描述的情形中, 程序是集成到再现装置中的, 但是, 这些程序的使用也可以独立于再现装置。实施这些程序的行为包括: (1) 制造行为; (2) 有偿或无偿转让的行为; (3) 出租行为; (4) 进口行为; (5) 经由双向电子通信网络提供给公众的行为; (6) 使用橱窗展示、商品目录或宣传册许诺销售或出租的行为。

(D) 在每个流程图中以时间序列执行的步骤的时间单元可被视为本发明的必要单元。因此,这些流程图所示的再现方法是一项发明。如果执行每个流程图中所示的处理,通过以时间顺序执行这些步骤,从而实现预期的目的和预期的效果,那么,这也是实施本发明的记录方法的行为。

(E) 当把 AV 剪辑记录到 BD-ROM 上时,可以向 AV 剪辑中的各个 TS 包添加扩展头。该扩展头被称为 TP_extra_header,其包括 arrival_time_stamp 和 copy_permission_indicator,并且数据长度为 4 个字节。具有 TP_extra_header 的 TS 数据包(下文称之为“EX TS 包”)以 32 个数据包为单位进行分组,并且将每组写入 3 个扇区中。由 32 个 EX TS 数据包构成的一组具有 6144 个字节(=32x192),这相当于 3 个扇区的尺寸(6144 个字节=(2048x3))。包含在这 3 个扇区中的这 32 个 EX TS 数据包被称为一个对准单元(aligned unit)。

在通过 IEEE1394 连接器相连的家庭网络中,该再现装置以下面的方式发送对准单元。该再现装置将对准单元中的 32 个 EX TS 数据包中的每一个都删除 TP_extra_header,并且根据 DTCP 规范加密各 TS 数据包的主体,然后输出加密的 TS 数据包。当输出 TS 数据包时,该再现装置将一个同步包插入相邻的 TS 数据包之间。根据 TP_extra_header 的 arrival_time_stamp 所示的时间来确定插入同步包的位置。该再现装置输出 DTCP_descriptor 以及 TS 包。DTCP_descriptor 对应于 TP_extra_header 中的 copy_permission_indicator。通过提供表明“禁止复制”的 DTCP_descriptor,当在 IEEE 连接器连接的家庭网络中使用 TS 数据包时,可以防止将 TS 数据包记录到其他设备上。

(F) 在上述实施例所描述的情形中,将 BD-ROM 格式的 AV 剪辑用作数字流,但本发明还可以用 DVD 视频格式或 DVD 视频记录格式的 VOB(视频对象)来实现。VOB 是符合 ISO/IEC13818-1 标准的节目流,它是通过复用视频流和音频流而获得的。同样,AV 剪辑中的视频流也可以是 MPEG4 视频流或 WMV 视频流。此外,AV 剪

辑中的音频流可以是线性 PCM 音频流、Dolby AC-3 音频流、MP3 音频流、MPEG-AAC 音频流或 dts 音频流。

(G) 上述实施例中的电影可以是通过对经过模拟广播传输的模拟图像信号进行编码而获得的。此外，该电影也可以是经过数字广播传输的传输流构成的流数据。

或者，还可以通过对记录在录像带中的模拟/数字图像信号进行编码来获得内容。此外，还可以通过对从视频照相机中直接装载的模拟或数字图像信号进行编码来获得内容。这些内容也可以是通过分发服务器分发的数字作品。

(H) 上述实施例中描述的图形对象是游程编码的栅格数据。游程编码用于压缩和编码图形对象，因为游程编码适于压缩和解压缩字幕。字幕的属性在于，水平方向上的相同像素值的连续长度较长。因而，通过使用游程编码进行压缩，可以获得高压缩率。此外，游程编码降低了解压缩的负担，因此适于用软件实现解密。但是，对图形对象使用游程编码不是限制本发明。例如，图形对象可以是 PNG 数据。此外，图形对象不必是栅格数据，也可以是矢量数据。此外，图形对象可以是透明图案。

(I) 根据再现装置中的语言设置选择的字幕图形可以实现 PCS 的显示效果。因此，可以通过根据该再现装置的语言设置显示的字幕图形来实现通过使用在常规 DVD 中视频主体内包含的字符而达到的显示效果。这有助于提高实用性。

此外，根据该再现装置的显示设置而选择的字幕图形可以实现 PCS 的显示效果。例如，BD-ROM 上记录了各种显示模式的图形，如宽屏幕、平移与扫描(Pan and Scan, 即 4:3 模式)和邮箱(letterbox, 即 16:9 模式)，再现装置根据与再现装置连接的电视的显示设置来选择一种显示模式，并显示对应的图形。由于 PCS 的显示效果适用于这些图形，所以可视性得到了增强因此，用根据显示设置显示的字幕图形可以实现用传统 DVD 中视频主体内包含的字符达到的显示效果。这有助于提高实用性。

(J) 在第一个实施例描述的情形中，设置从对象缓冲器到图形平

面的传输速率 R_c ，从而在一个视频帧内，清除图形平面和在窗口上呈现图形，窗口尺寸是图形平面的 25%。但是，可以将 R_c 设置成在垂直消隐期间完成清除和呈现。假定垂直消隐周期是 1/29.93 秒的 25%。那么， R_c 是 1Gbps。如此设置 R_c ，可以更流畅地显示图形。

此外，可以与垂直消隐期间的写操作一起，使用与行扫描同步的写操作。这样，可以 $R_c=256$ Mbps 流畅地显示字幕。

(K) 在上述实施例所描述的情形中，再现装置包括图形平面上。但是，还可以将用于储存一行解压缩像素的线缓冲器代替图形平面。因为对于每行执行到图像信号的转换，所以，同样可以用线缓冲器执行到图像信号的转换。

(L) 在上述实施例所描述的情形中，图形是表示电影中的对话的字符串，但是，本发明同样适用于与视频精确同步显示的任何图形。这样的图形示例包括插图、图案、卡通人物和字符标记。这些图形的示例还包括构成商标的图案、字符和颜色的组合、国家饰章、国旗、国徽、国家政府使用的公共标记或图章、国际组织的饰章、旗帜或徽章，或者特殊项目的原始标记。

(M) 在第一个实施例所描述的情形中，基于字幕在屏幕顶部或底部上水平地显示的假设，将窗口设置在屏幕的顶部或底部。但是，也可以将窗口设置在屏幕的左侧或右侧，从而在屏幕的左侧和右侧上垂直地显示字幕。这样，就可以垂直地显示日文字幕。

(O) 当 DS_n 和 DS_{n+1} 属于图形流中同一时元时，图形解码器对 DS_n 和 DS_{n+1} 执行流水线处理。另一方面，当 DS_n 和 DS_{n+1} 属于不同的时元时，在 DS_n 的图形显示开始之后，图形解码器开始处理 DS_{n+1} 。

此外，有两种图形流，即：呈现图形流，主要用于与视频进行同步；交互图形流，主要用于实现交互显示。当该图形流是呈现图形流时，图形解码器对 DS_n 和 DS_{n+1} 执行流水线处理；当该图形流是交互图形流时，不执行流水线处理。

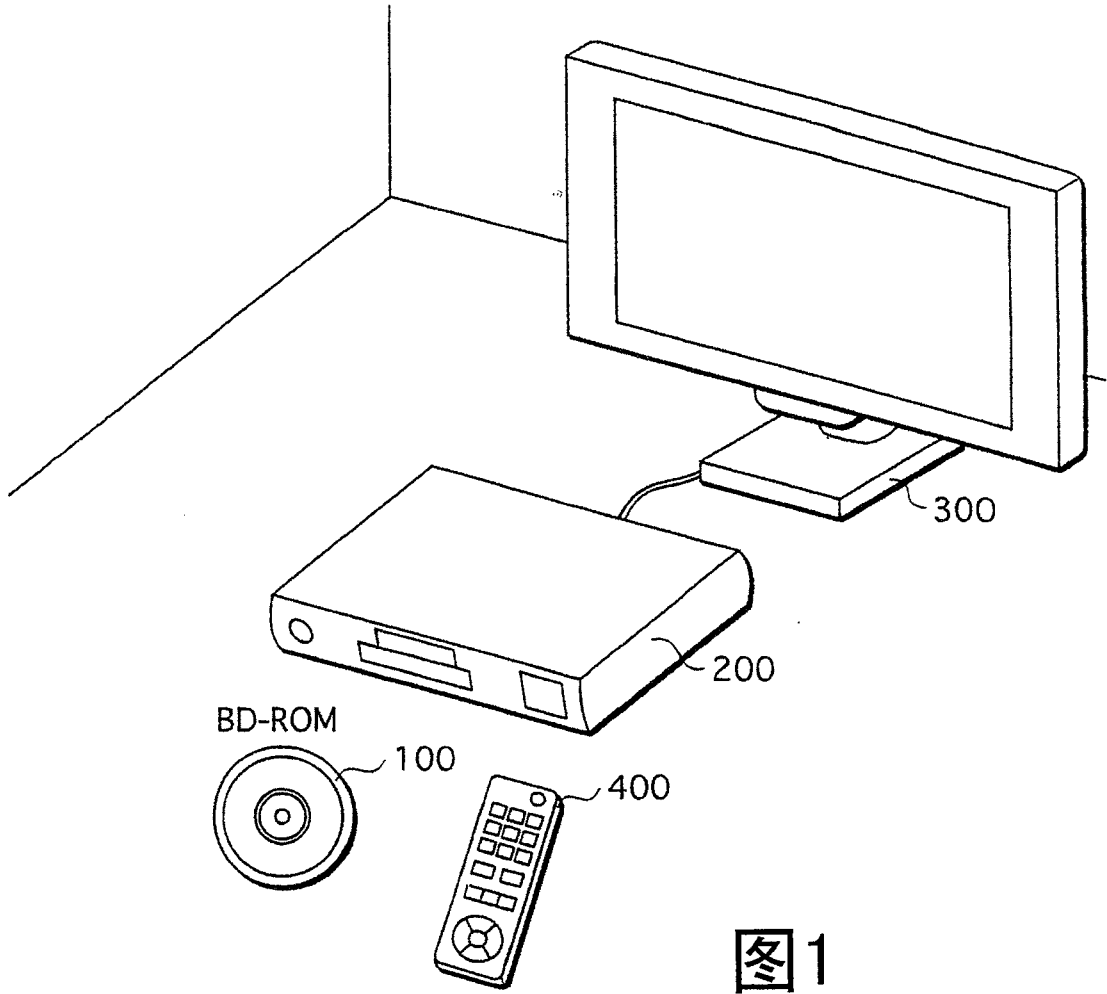
(P) 在图 20 所示的例子中，DS 中的 PCS 引用的图形数据设置在未被该 DS 中的 PCS 引用的图形数据之前。但是，多组引用图形数

据和多组无引用图形数据也可以在DS中按照它们object_id表示的次序排列。或者，多组引用图形数据可以按照扫描线次序排列。这里的扫描线次序指的是执行扫描的次序。通常，从屏幕的左上方到右下方执行扫描。因此，将显示坐标位于屏幕左上方的图形数据设置在显示坐标位于平面右下方的图形数据之前。这可以加速图形数据的呈现。

可以如上所述修改本发明。但是，本申请的每项权利要求的发明都反映了解决传统技术所遇到的技术问题的手段，所以，根据权利要求书的本发明的技术范围不会超过本领域技术人员认识该技术问题的技术范围。因此，根据权利要求书的本发明基本上对应于本申请中的说明书部分。

工业应用性

上述实施例披露了本发明所涉及的记录介质和再现装置的内部构造，可以基于所披露的内部构造，批量制造记录介质和再现装置。换言之，该记录介质和再现装置能够在工业上制造出来。因此，该记录介质和再现装置具有工业应用性。



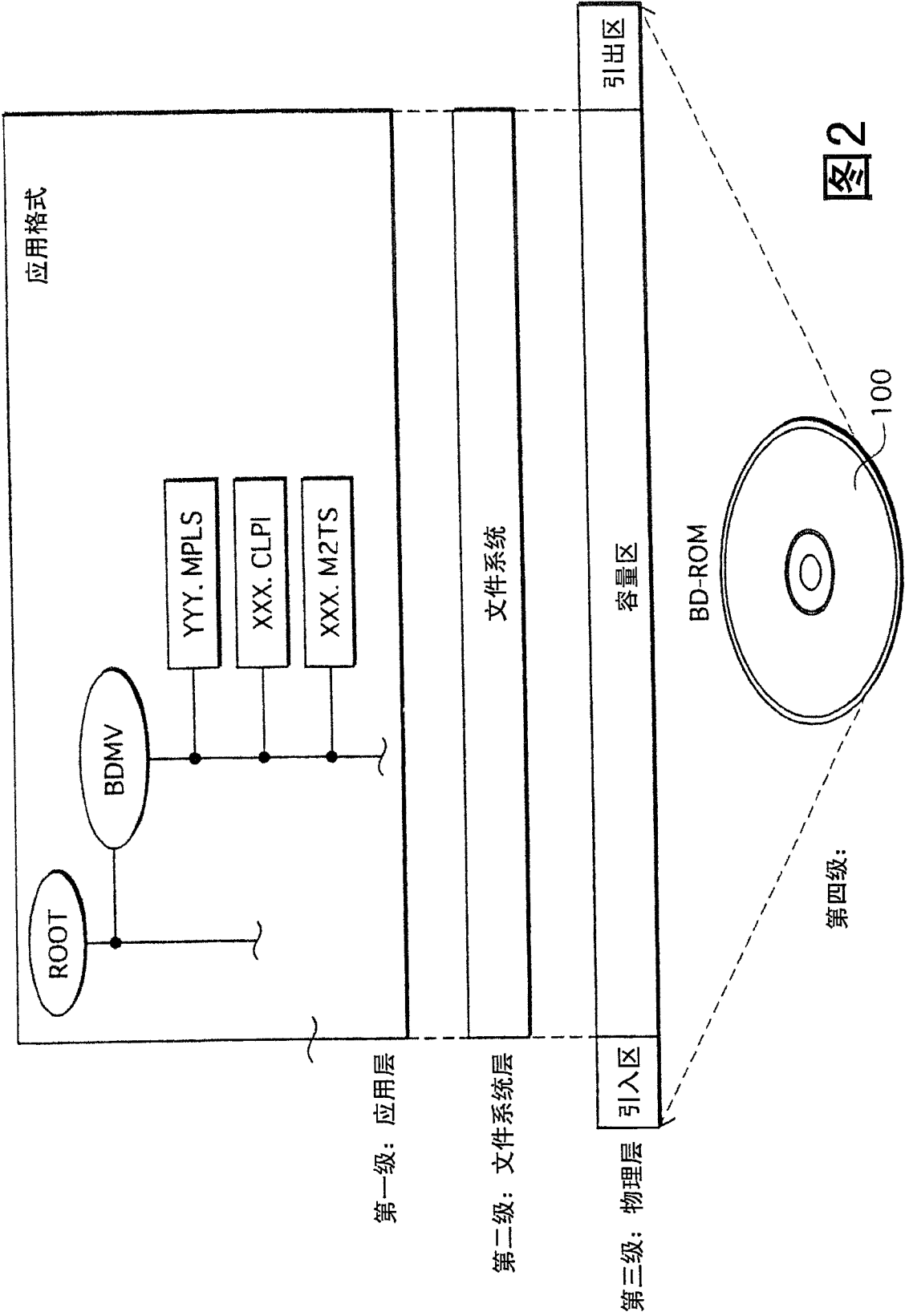


图2

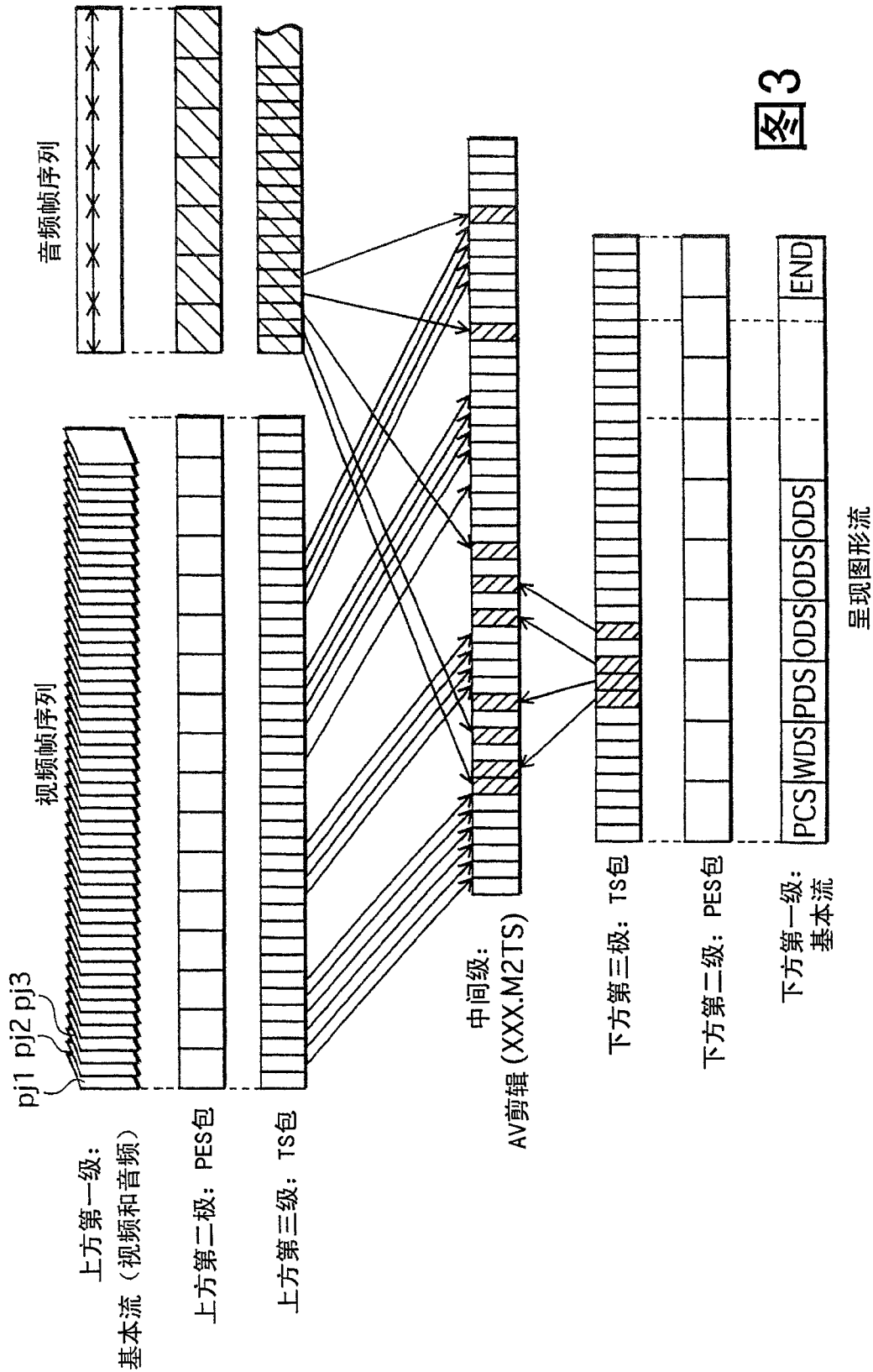


图3

呈现图形流

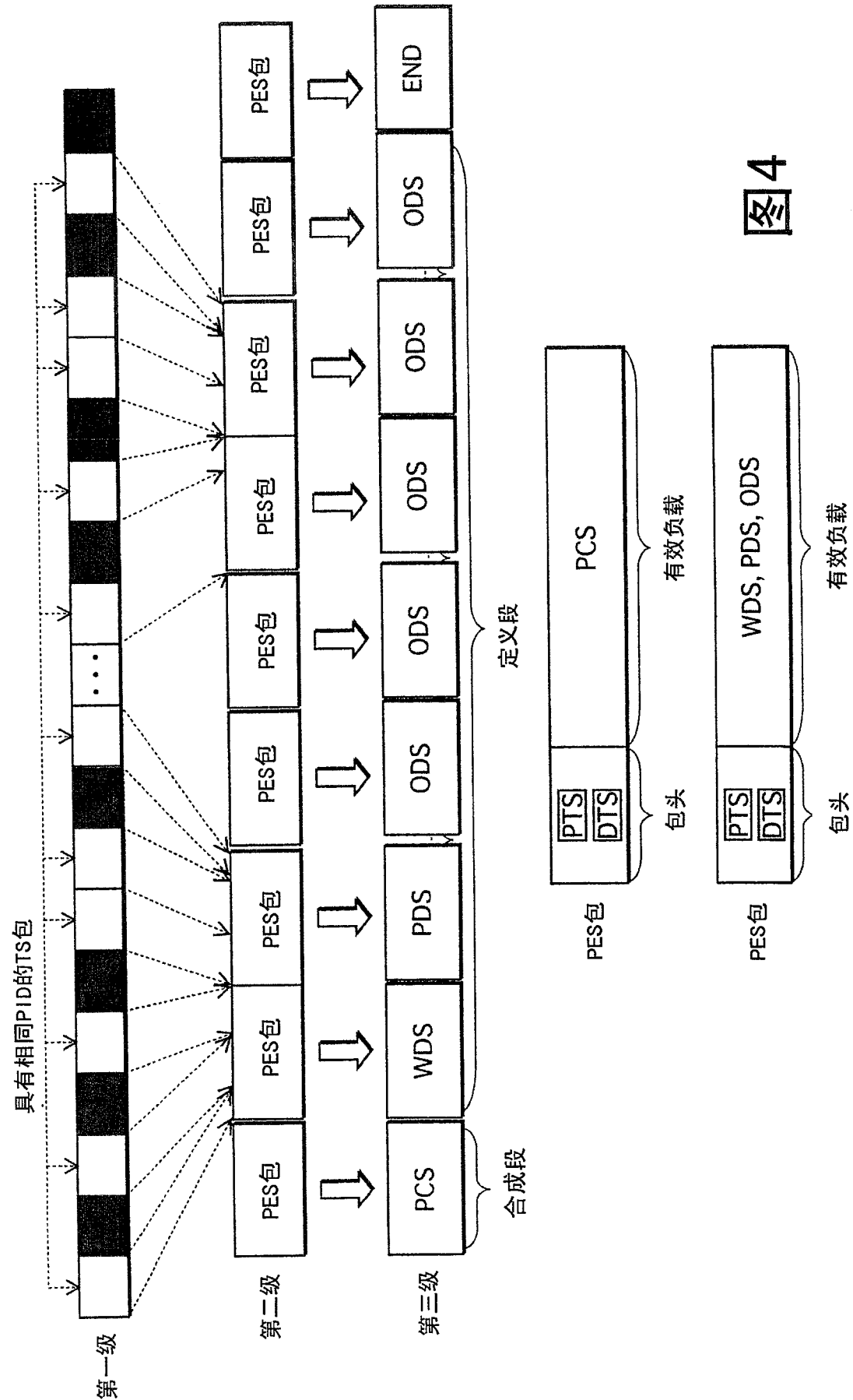


图4

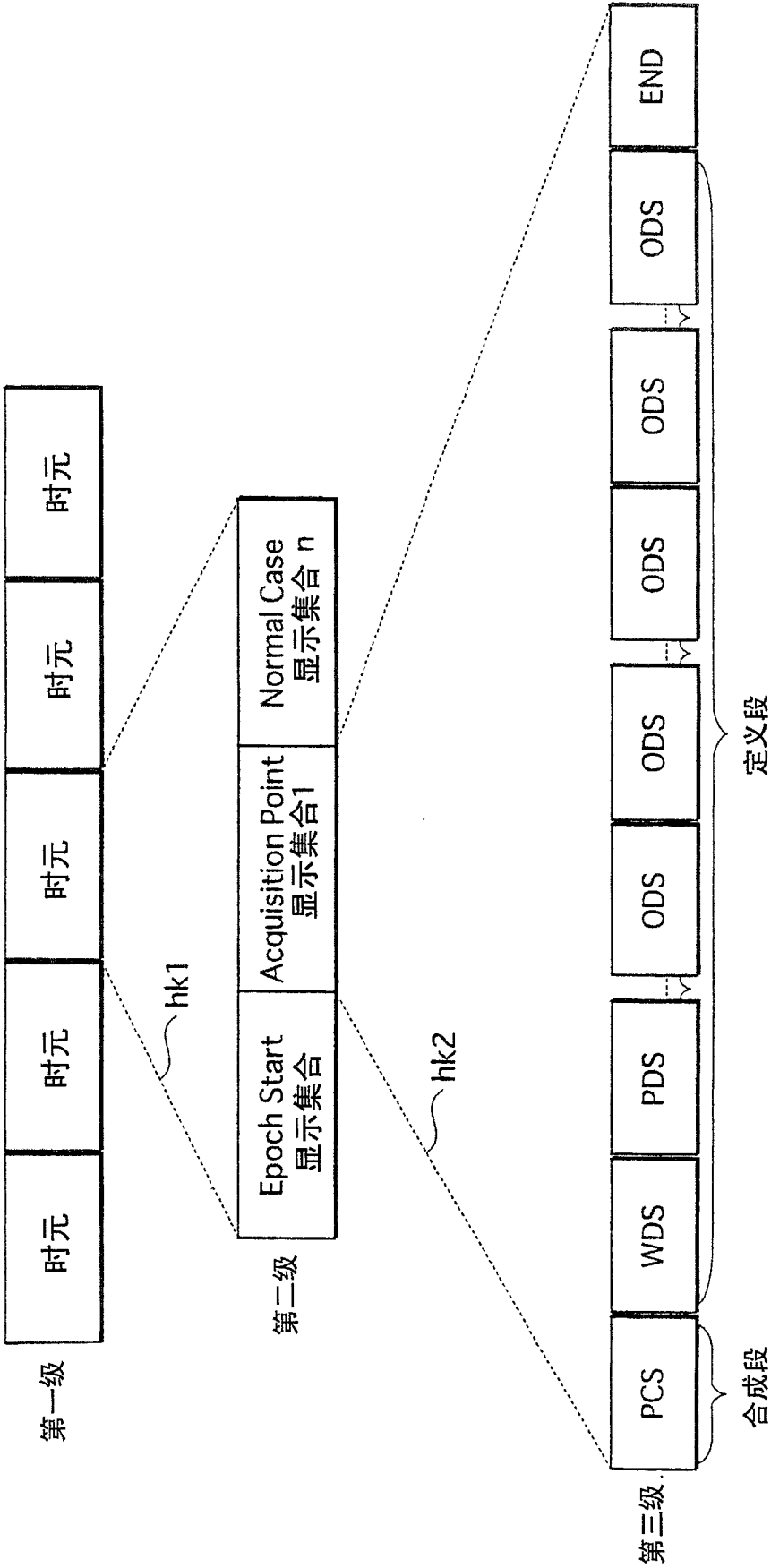


图5

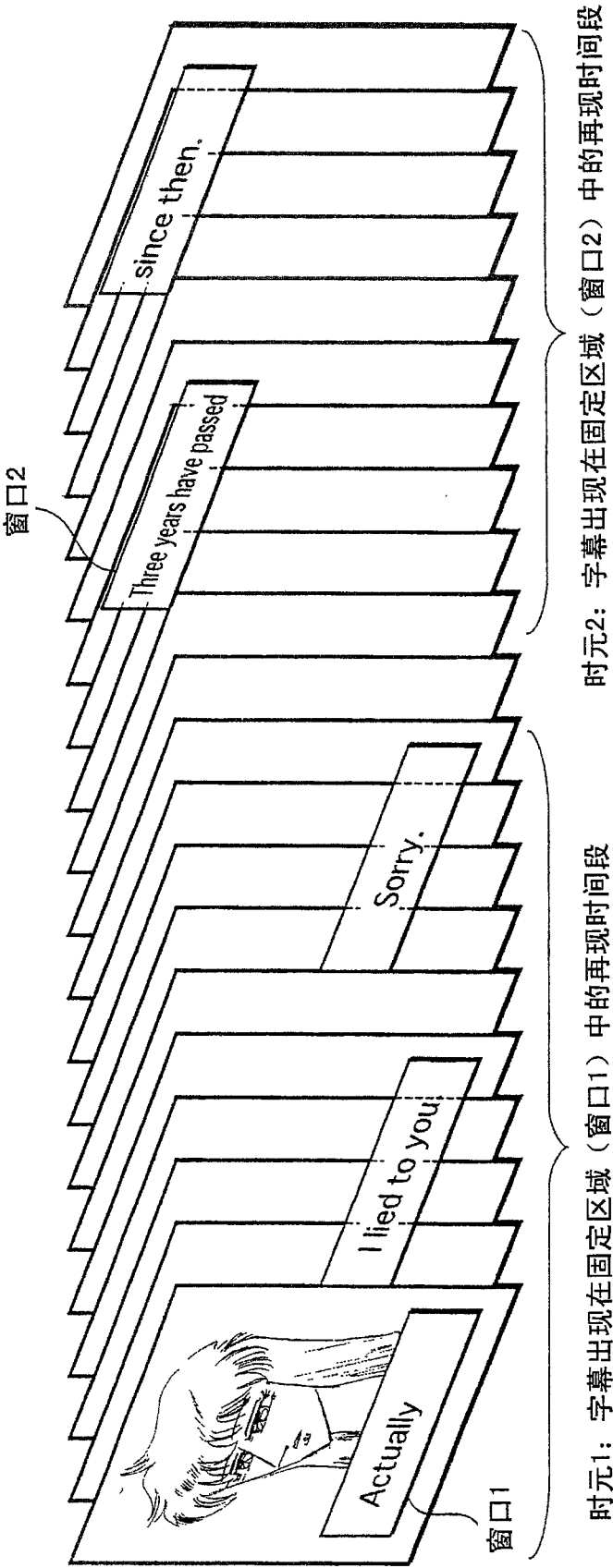


图6

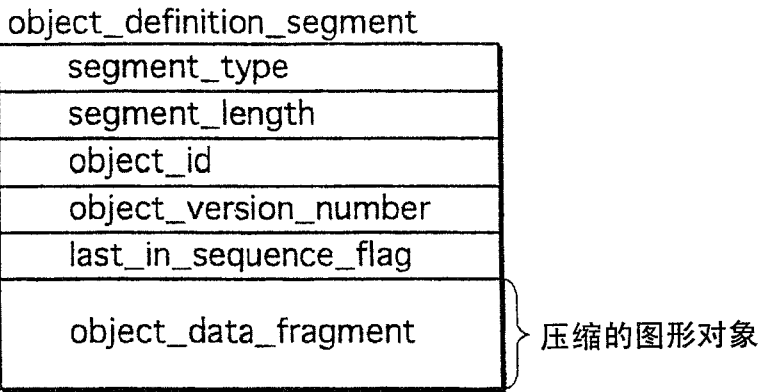


图7A

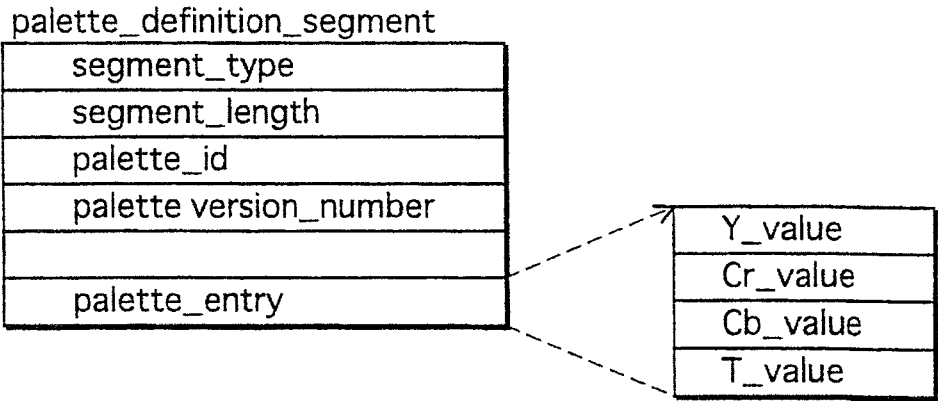
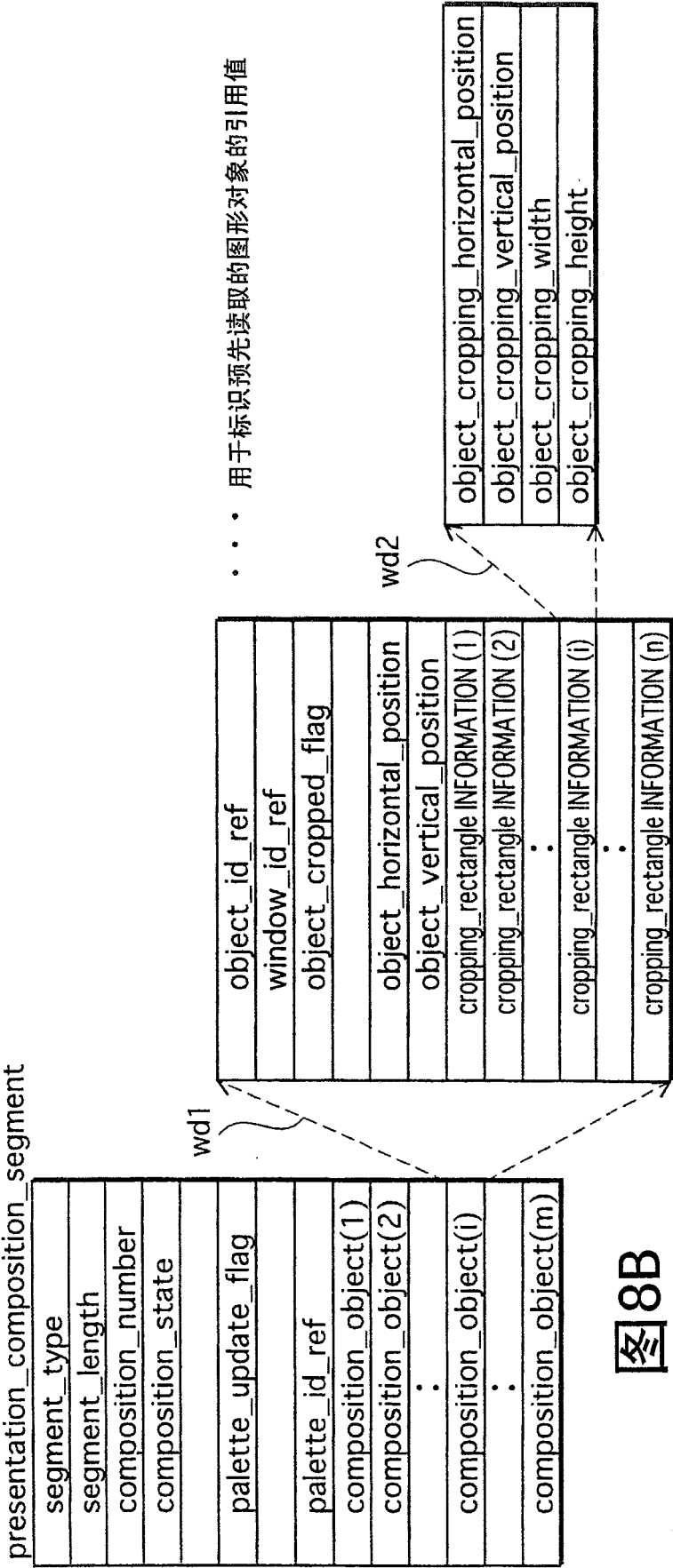
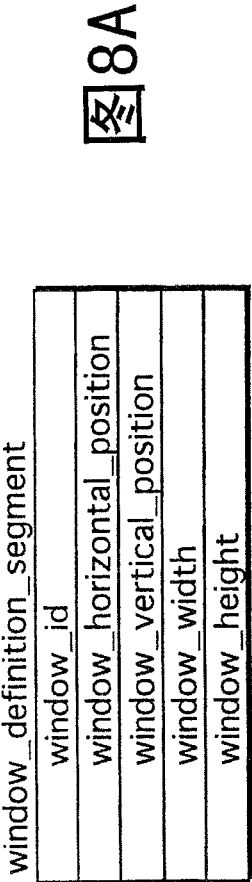


图7B



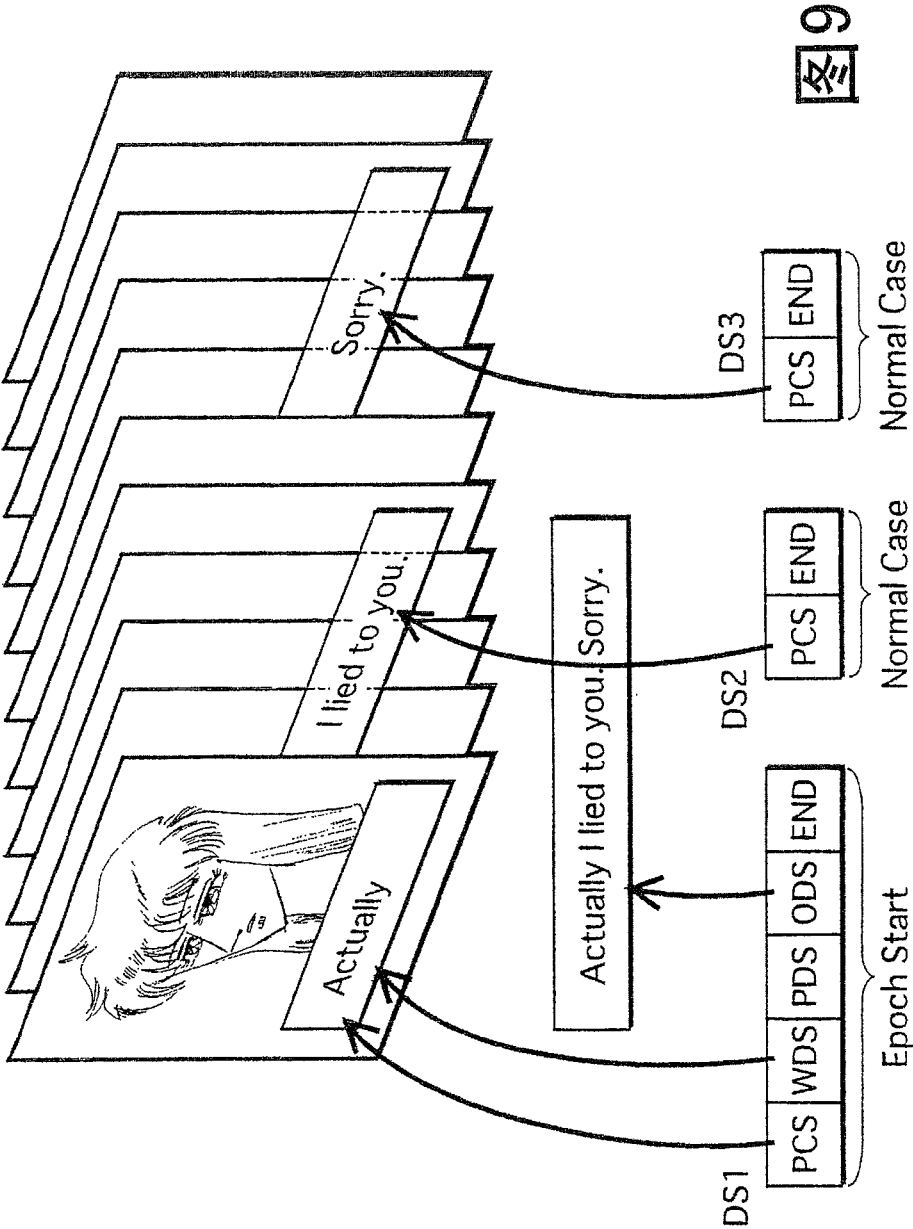


图9

DS1中的PCS和WDS示例性描述

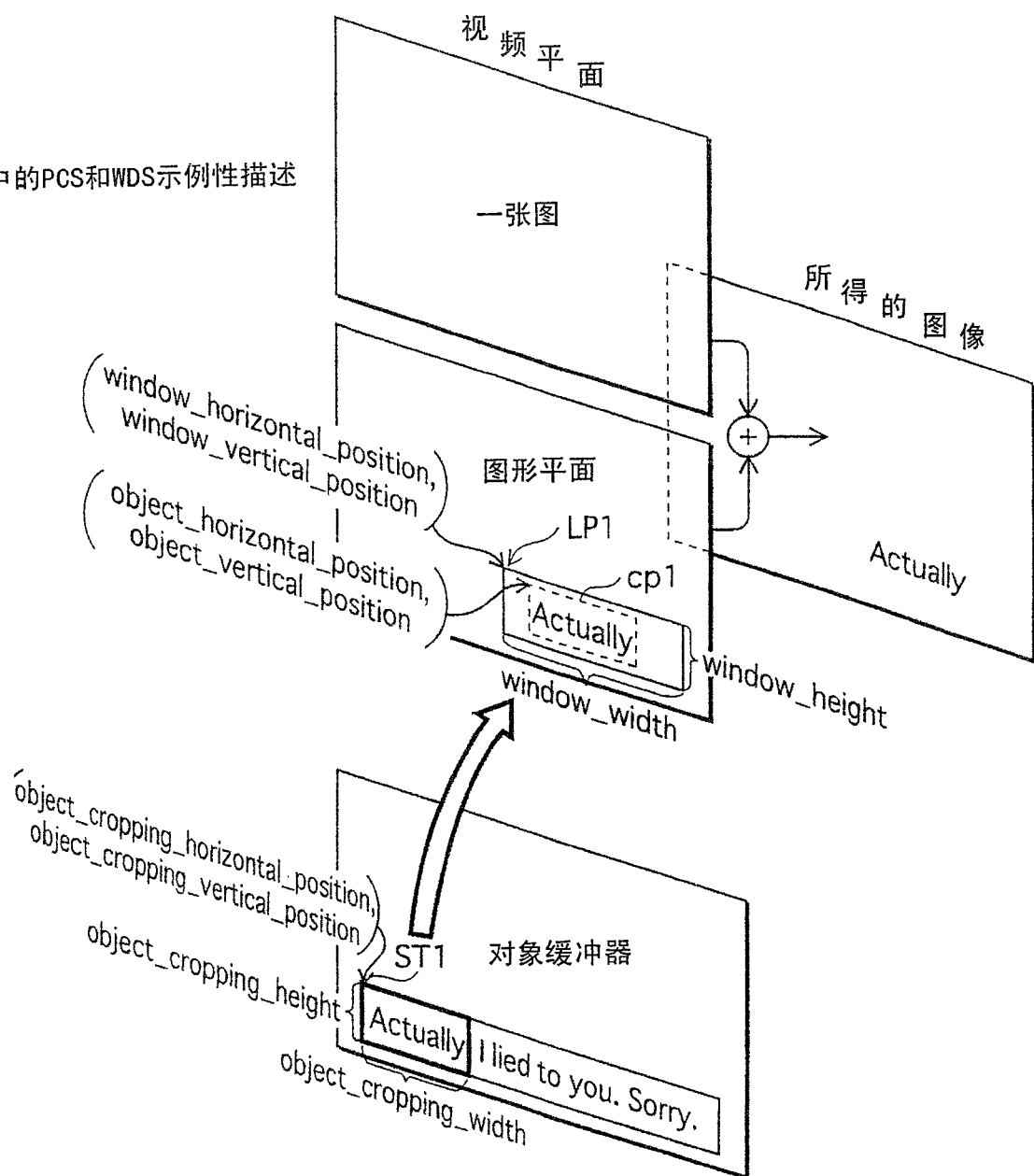


图10

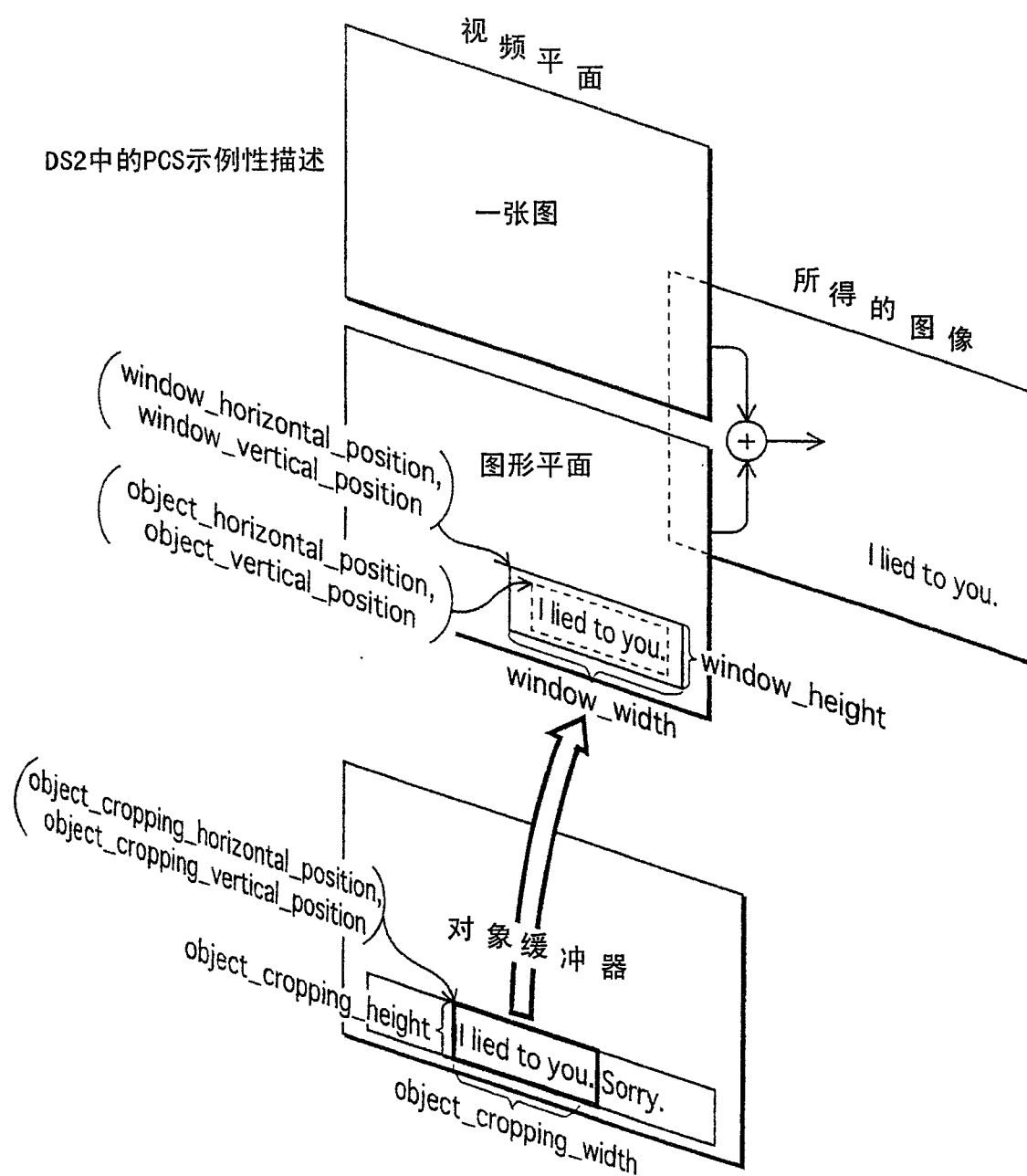


图 11

DS3中的PCS示例性描述

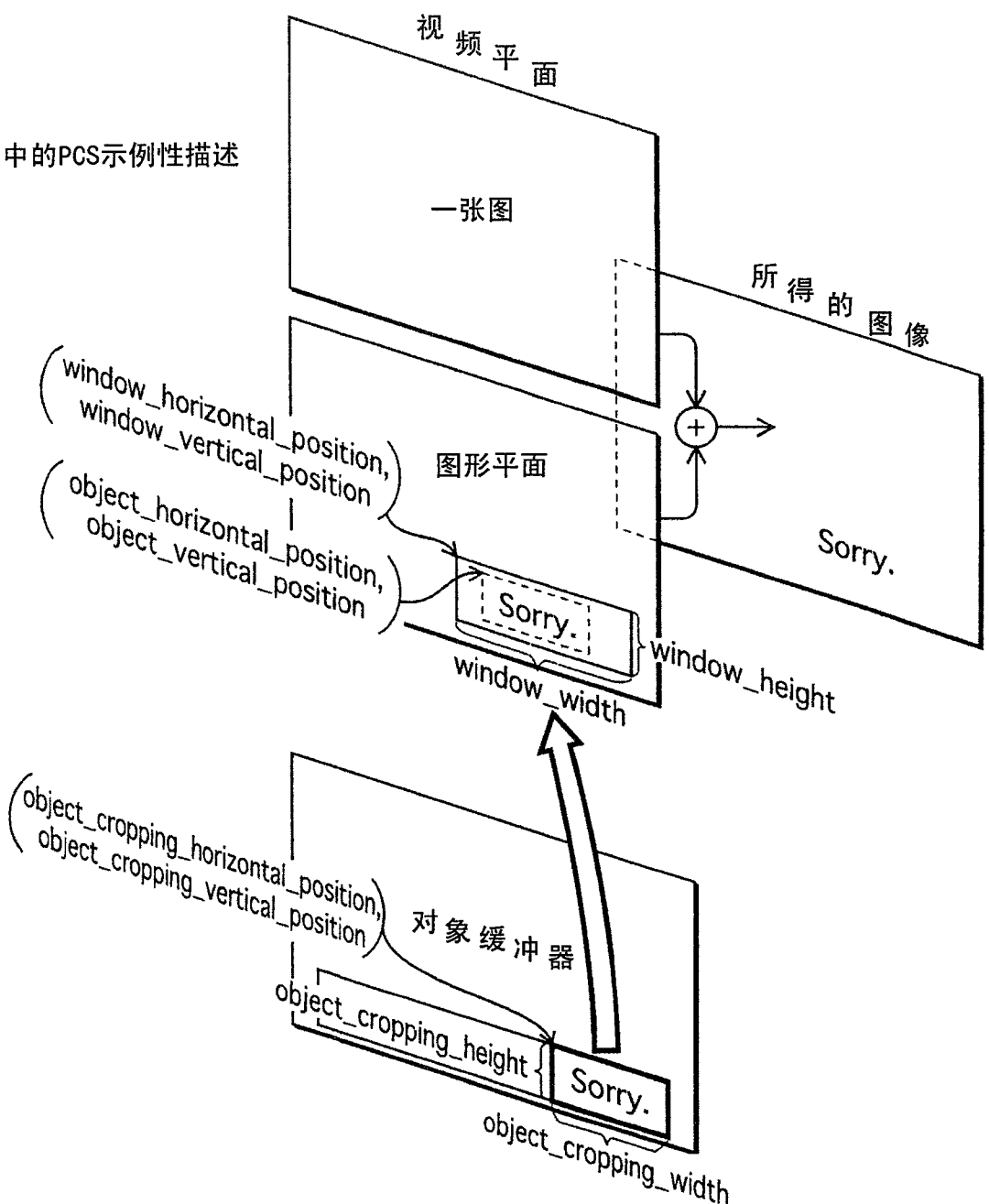


图12

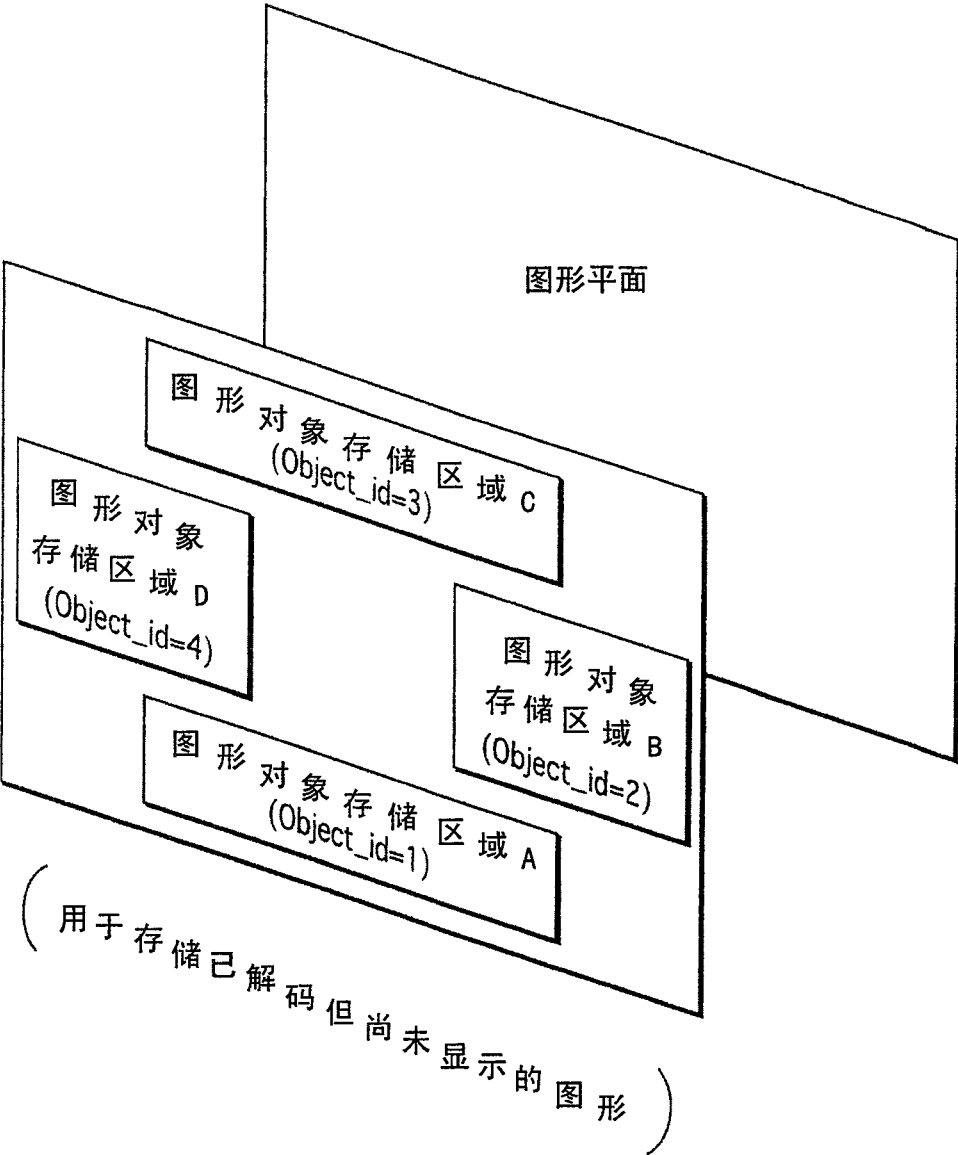


图13

$PTS(DSn[PCS]) \geq DTS(DSn[PCS]) + DECODEDURATION(DSn)$

Where:

- $DECODEDURATION(DSn)$ is calculated as follows:

```

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*106);
    }
    else
    {
        decode_duration += 90000*( SIZE( DSn.PCS.OBJ[0].window_id )//256*106);
        decode_duration += WAIT( DSn, DSn.PCS.OBJ[1], decode_duration ) ;
        decode_duration += 90000*( SIZE( DSn.PCS.OBJ[1].window_id )//256*106);
    }
}
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*106);
}
return decode_duration ;

```
- $PLANEINITIALIZATIONTIME(DSn)$ is calculated as follows:

```

initialize_duration=0 ;
if( DSn.PCS.composition_state== EPOCH_START )
{
    initialize_duration = 90000*( 8*video_width*video_height//256*106);
}
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*106);
    }
}
return initialize_duration ;

```
- $WAIT(DSn, OBJ, current_duration)$ is calculated as follows:

```

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 14

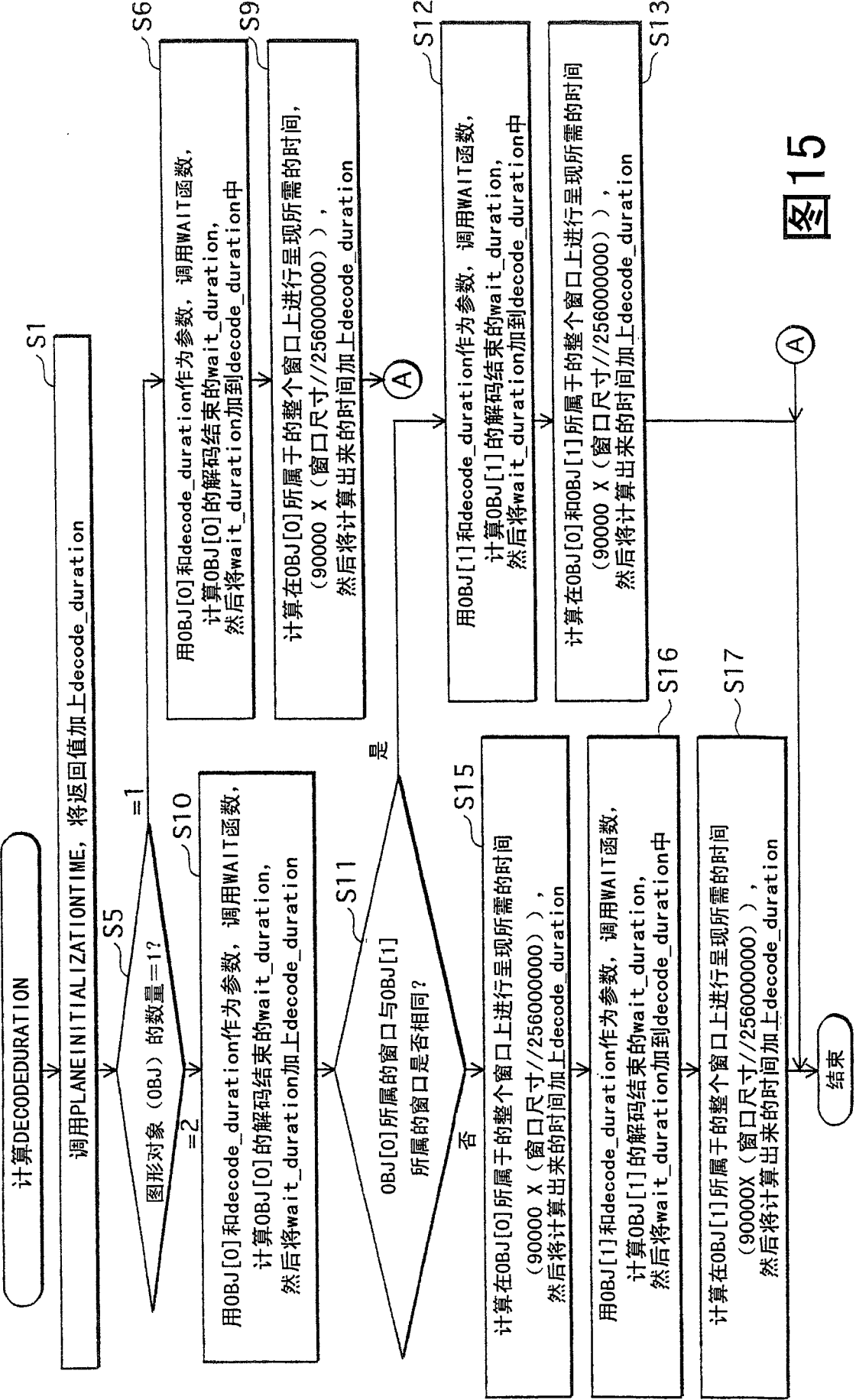
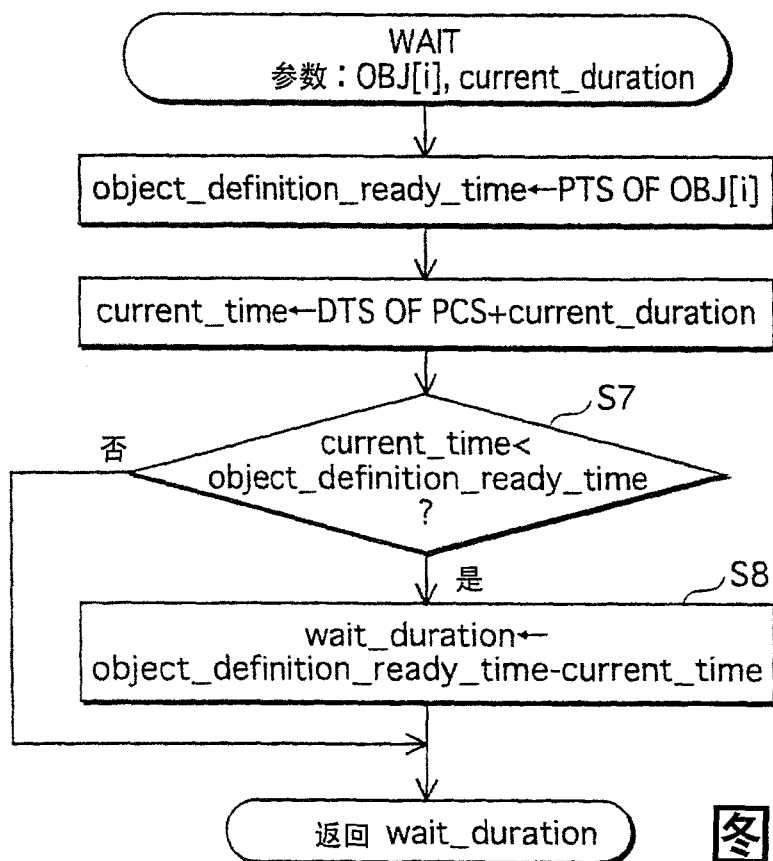
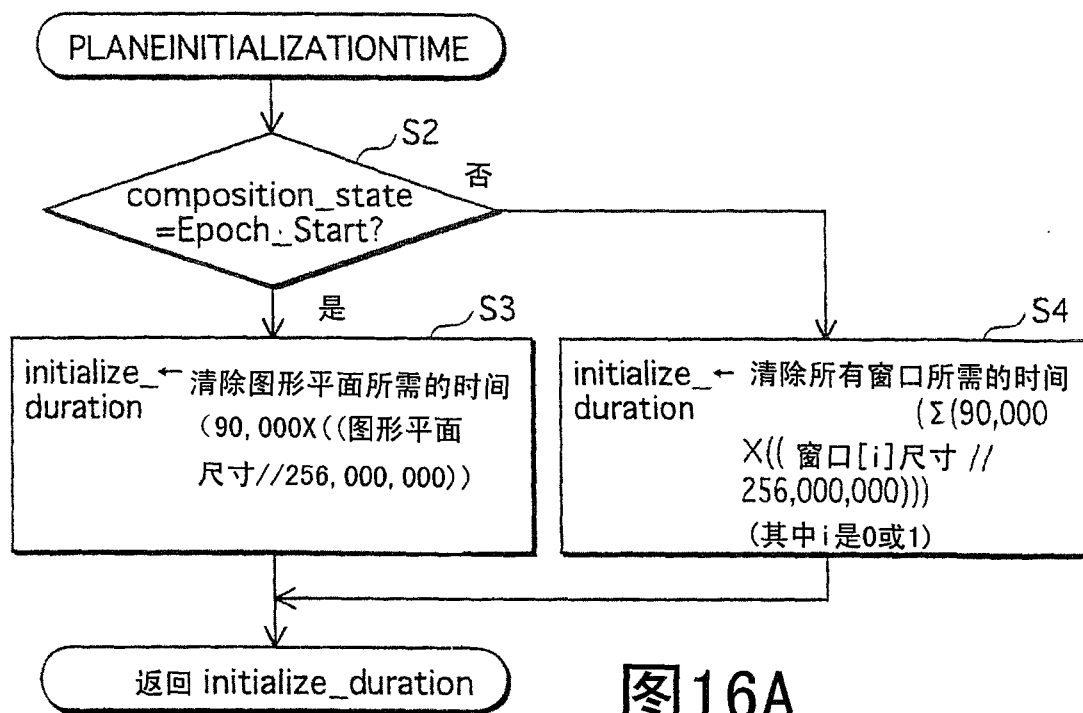
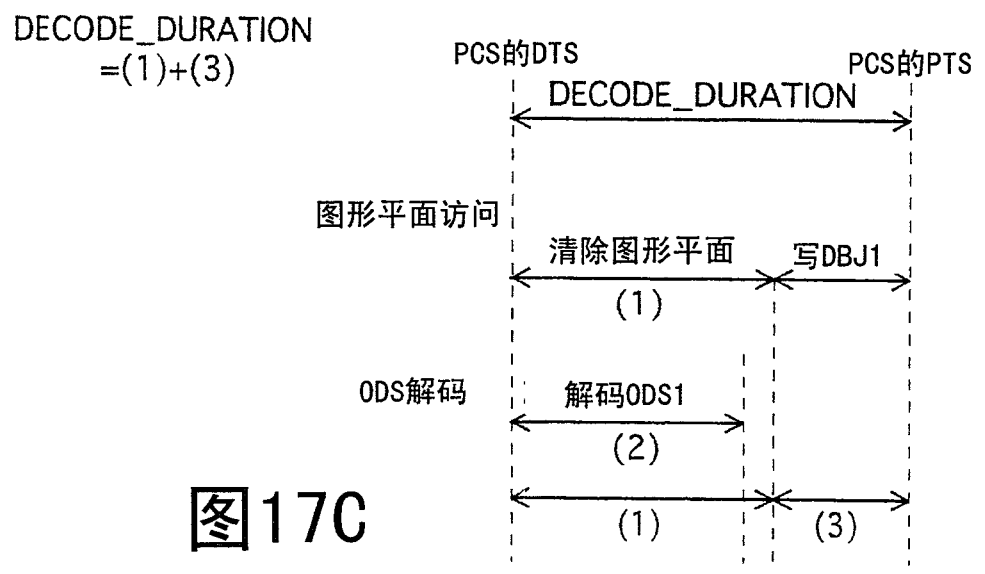
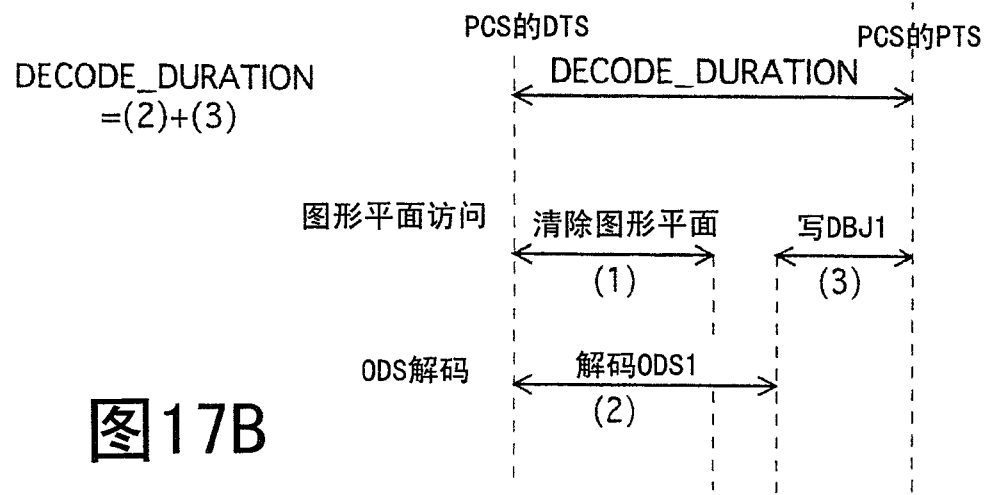
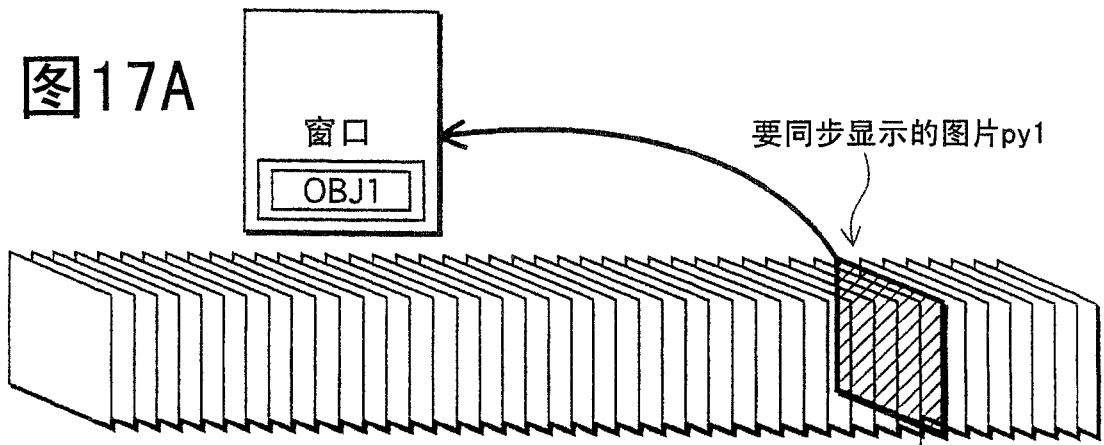


图15





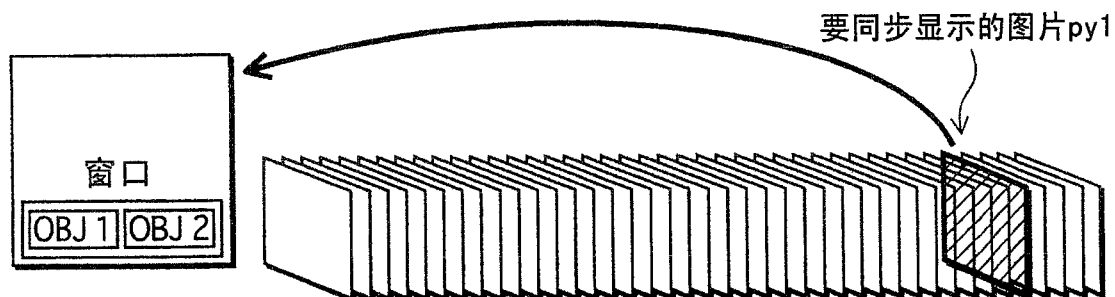


图18A

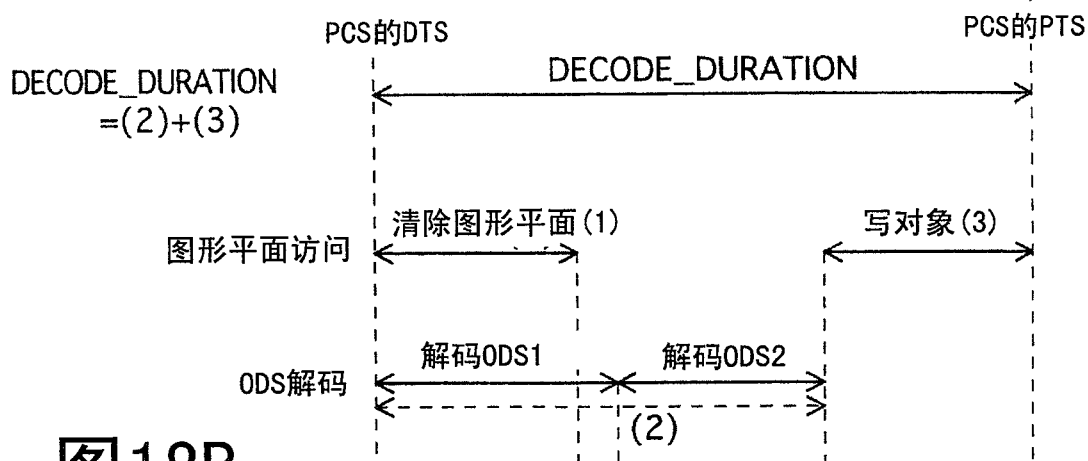


图18B

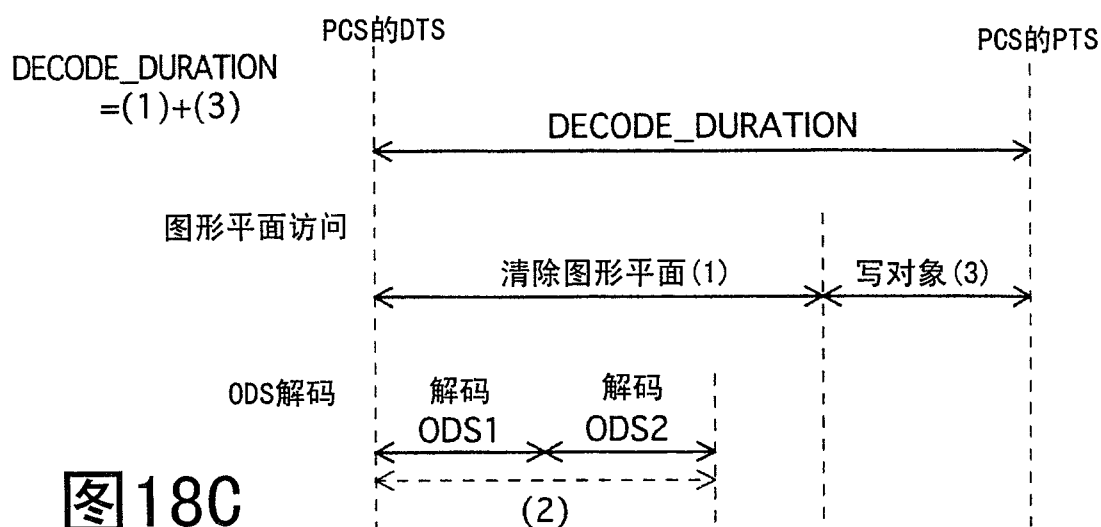
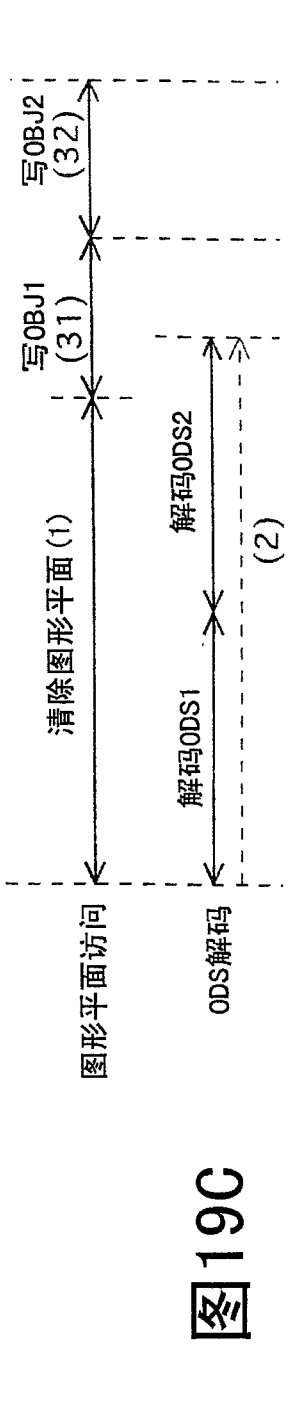
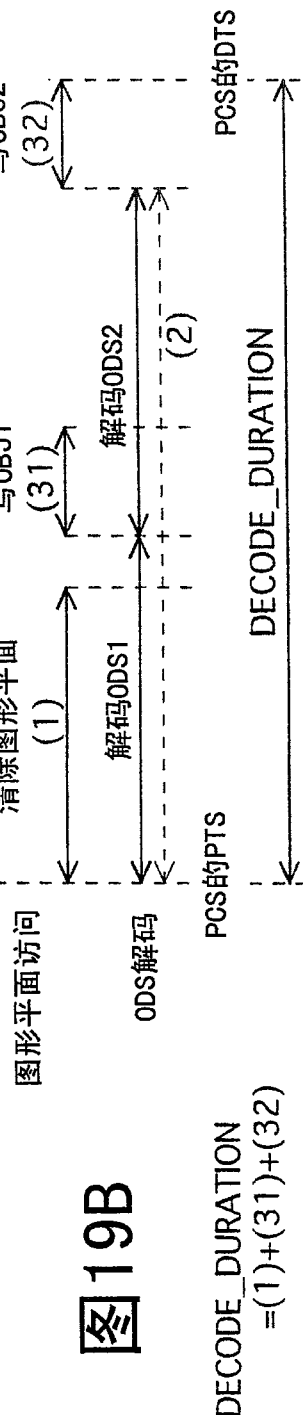
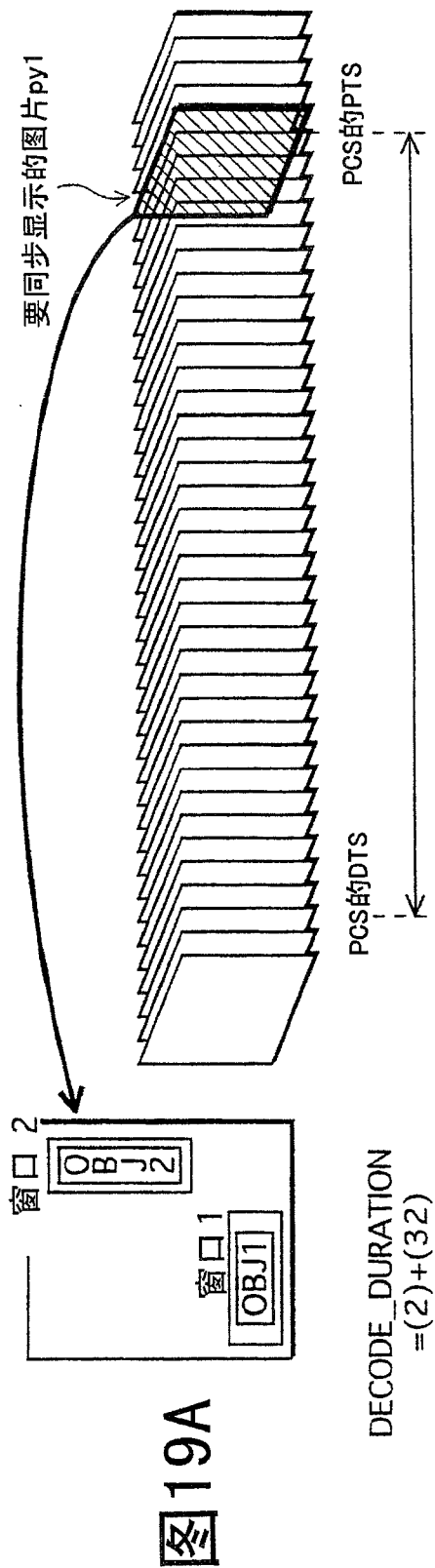


图18C



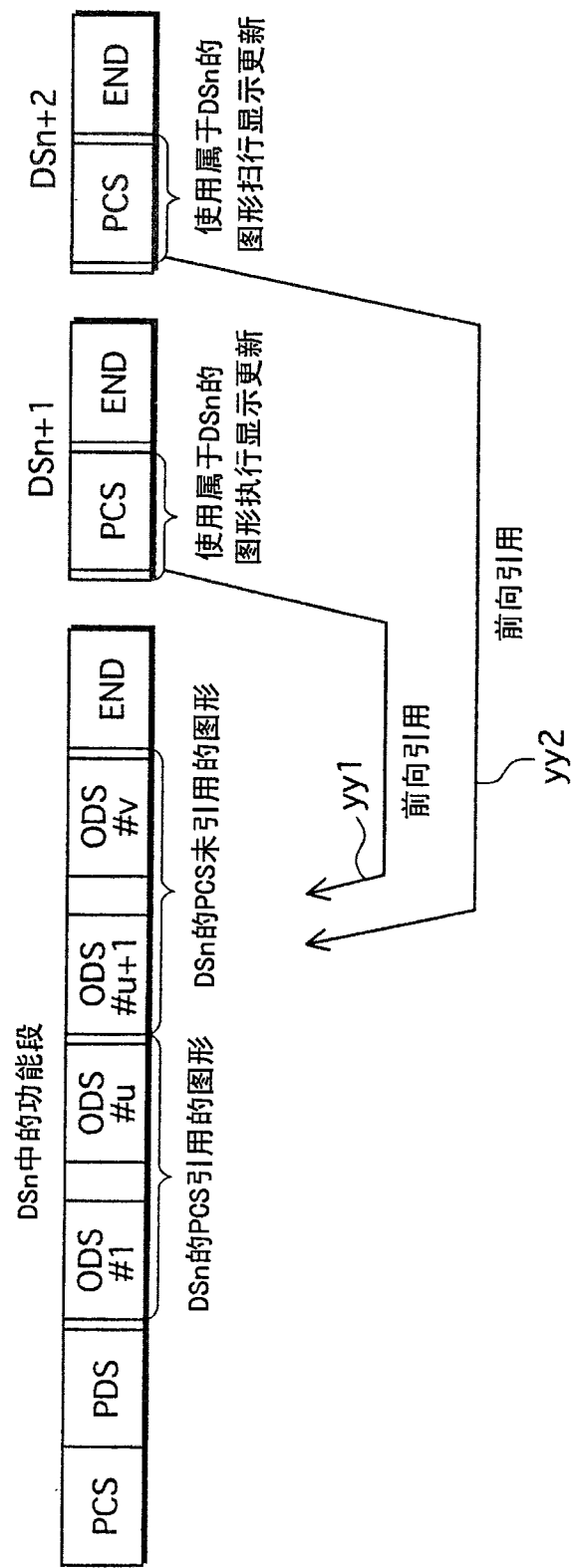
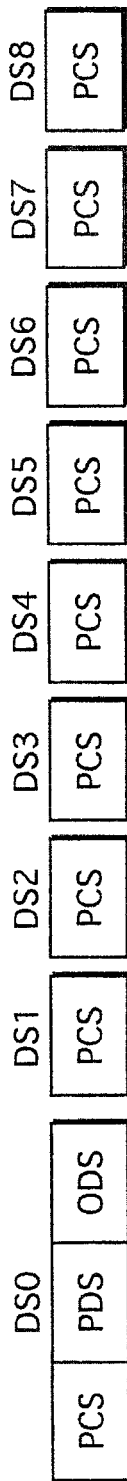


图20



object_id
=1

图21A

composition_object		PTS
DS1中的PCS	object_id_ref=1 object_horizontal_position=x1 object_vertical_position=y1	PTS=t1
DS2中的PCS	object_id_ref=1 object_horizontal_position=x2 object_vertical_position=y2	PTS=t2
DS3中的PCS	object_id_ref=1 object_horizontal_position=x3 object_vertical_position=y3	PTS=t3
DS4中的PCS	object_id_ref=1 object_horizontal_position=x4 object_vertical_position=y4	PTS=t4
DS5中的PCS	object_id_ref=1 object_horizontal_position=x5 object_vertical_position=y5	PTS=t5
DS6中的PCS	object_id_ref=1 object_horizontal_position=x6 object_vertical_position=y6	PTS=t6
DS7中的PCS	object_id_ref=1 object_horizontal_position=x7 object_vertical_position=y7	PTS=t7
DS8中的PCS	object_id_ref=1 object_horizontal_position=x8 object_vertical_position=y8	PTS=t8

图21B

My heart is fluttering

Object_id=1

图 22A

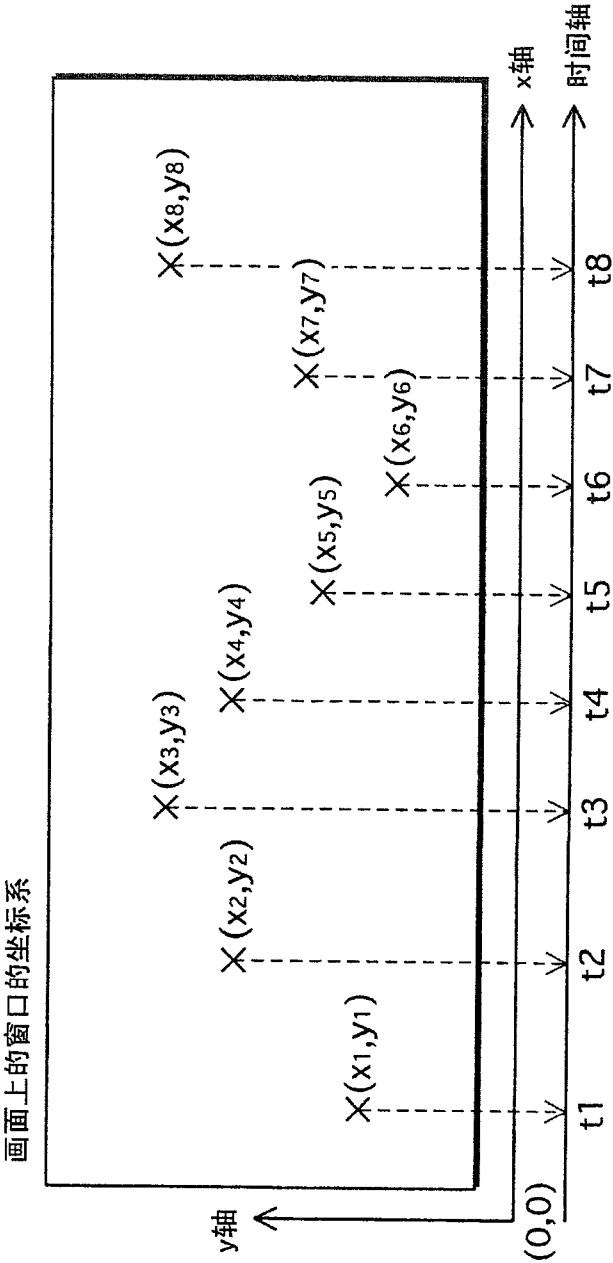


图 22B

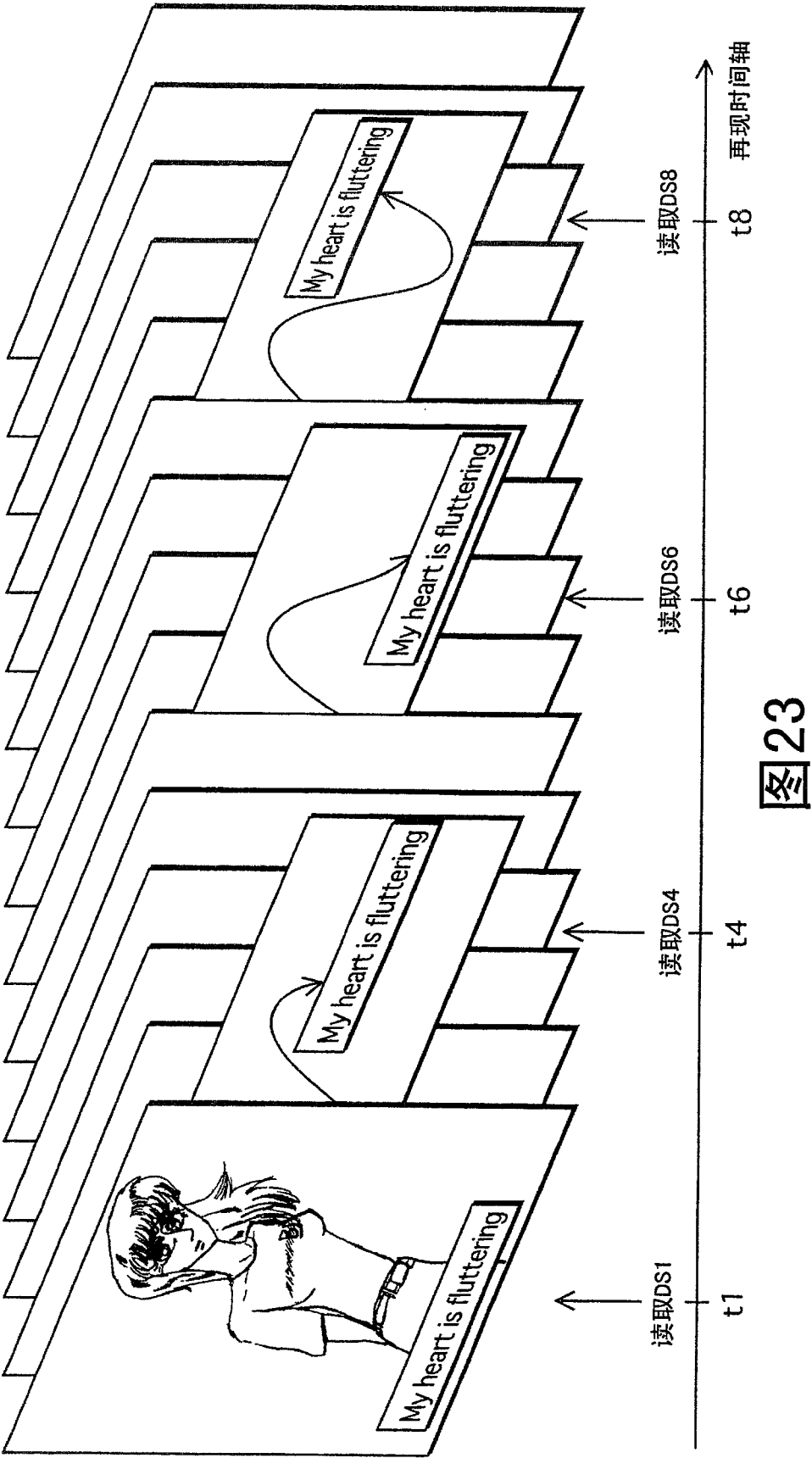


图23

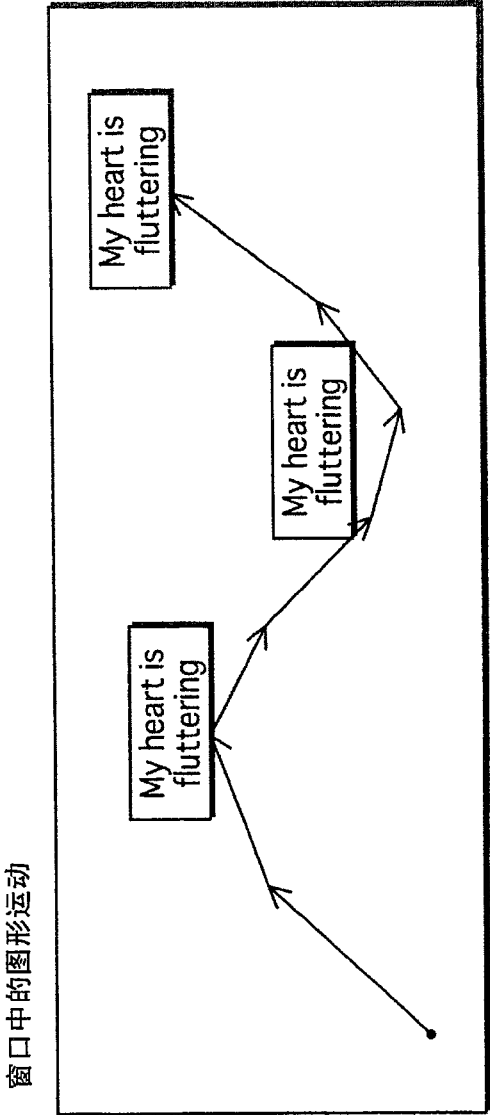


图24

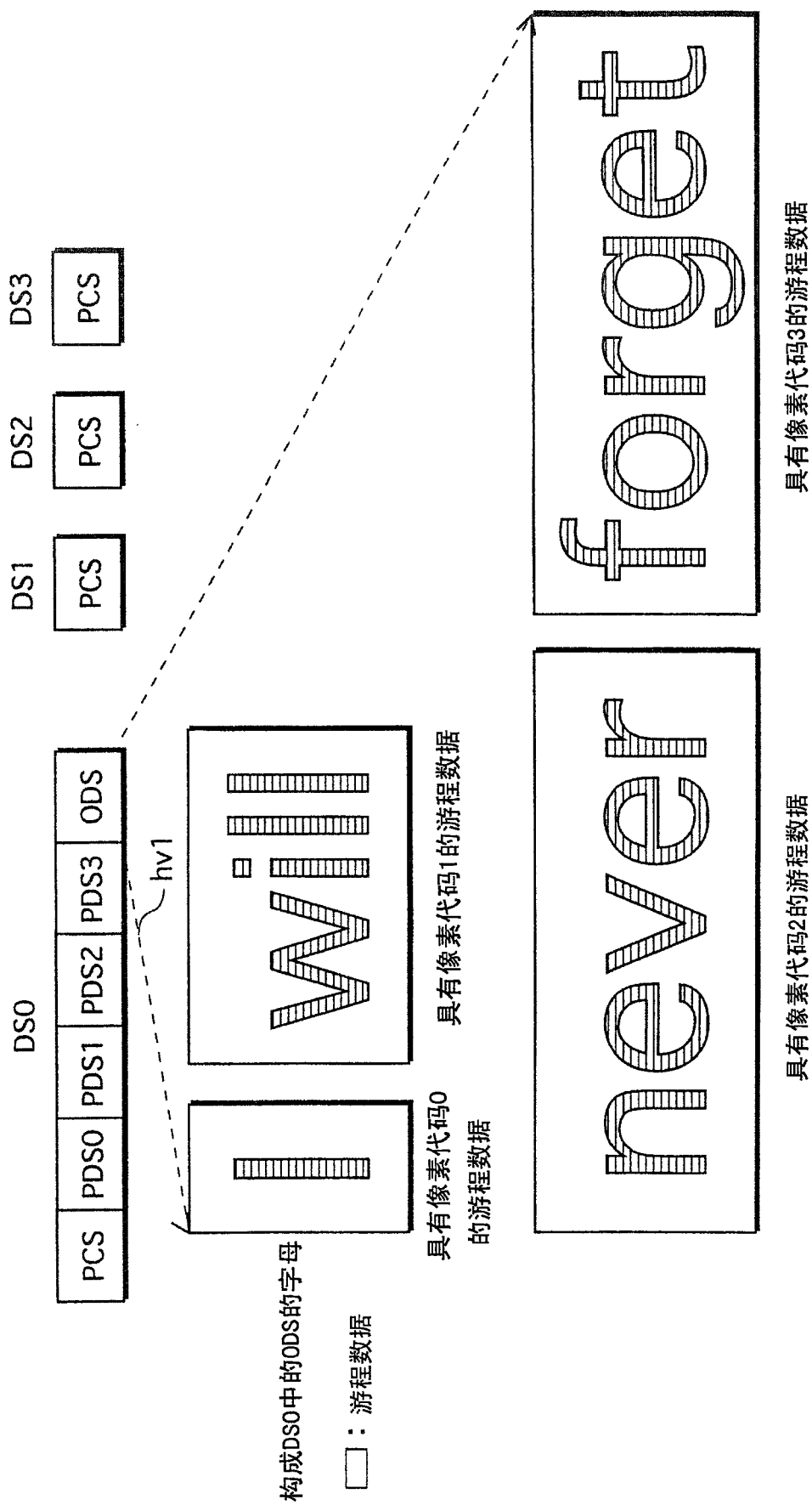


图25

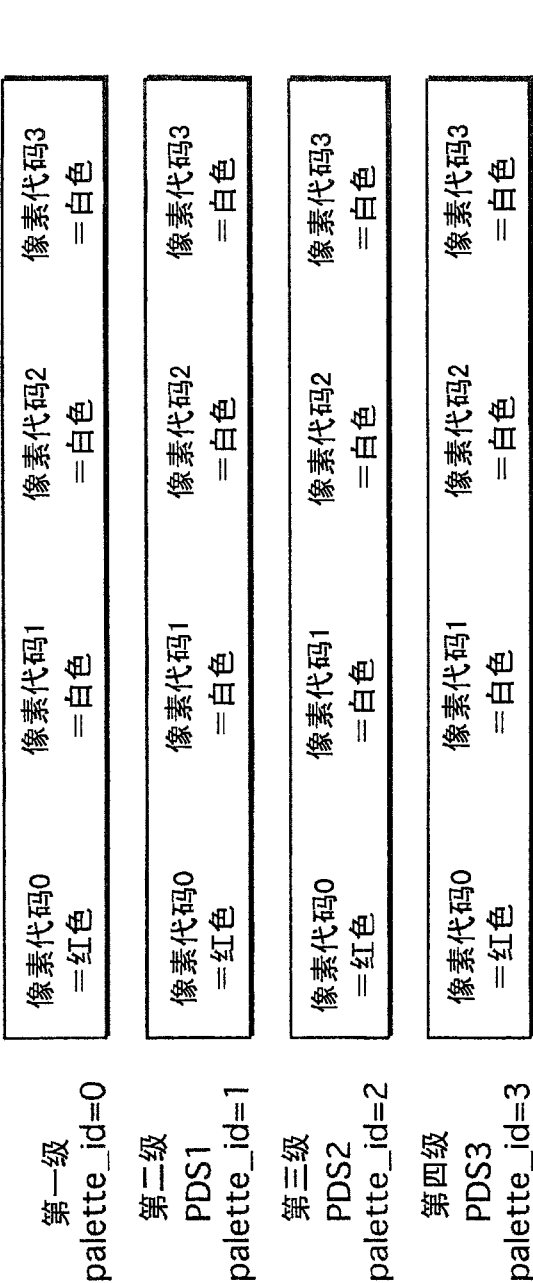


图26A

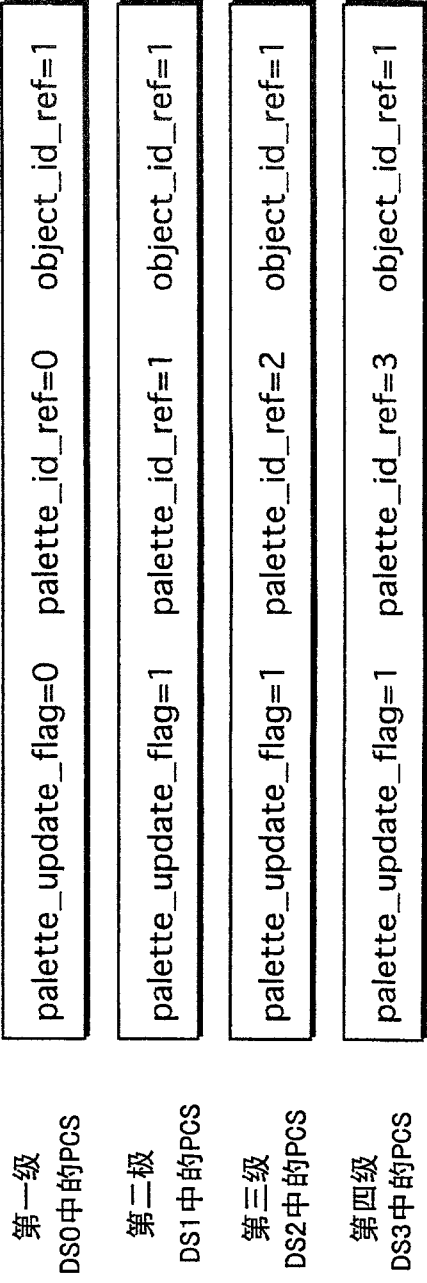


图26B

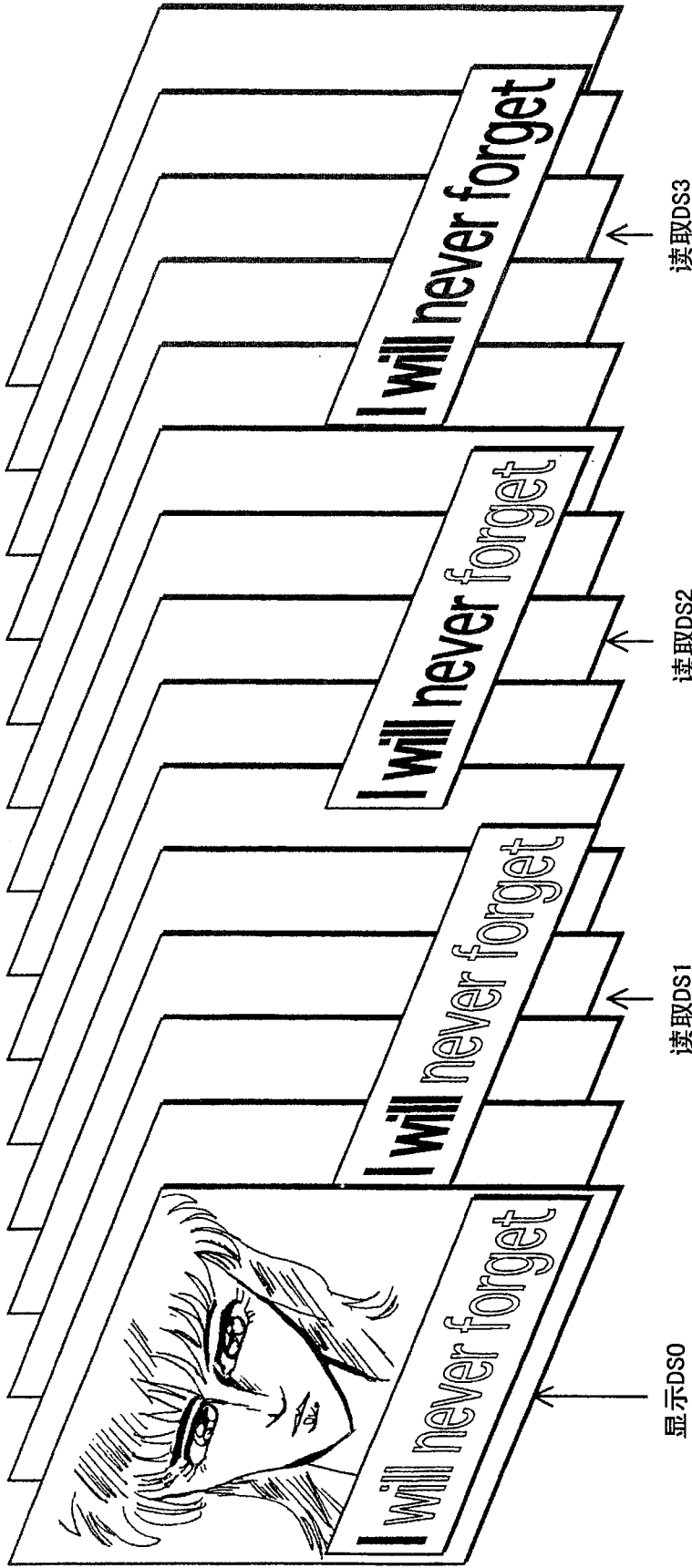


图27

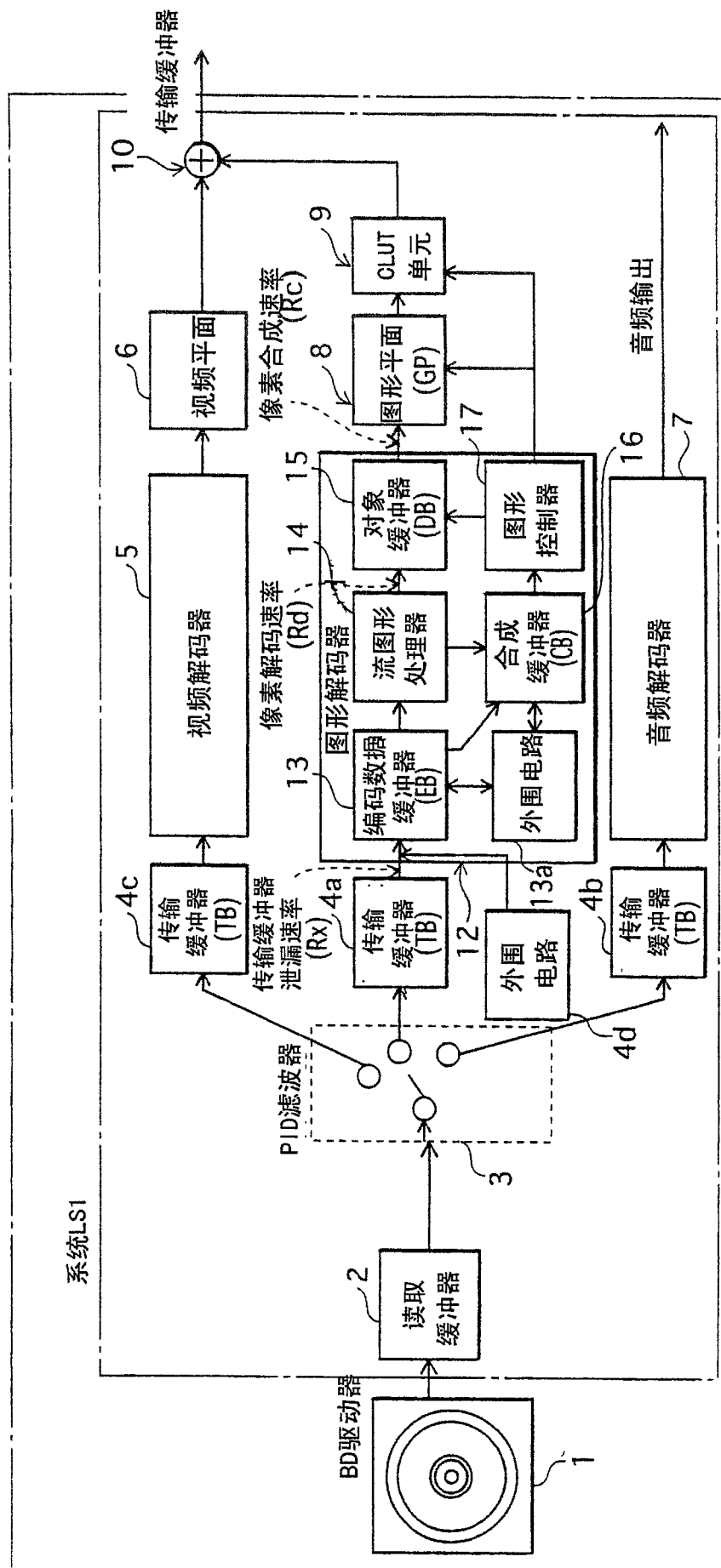


图28

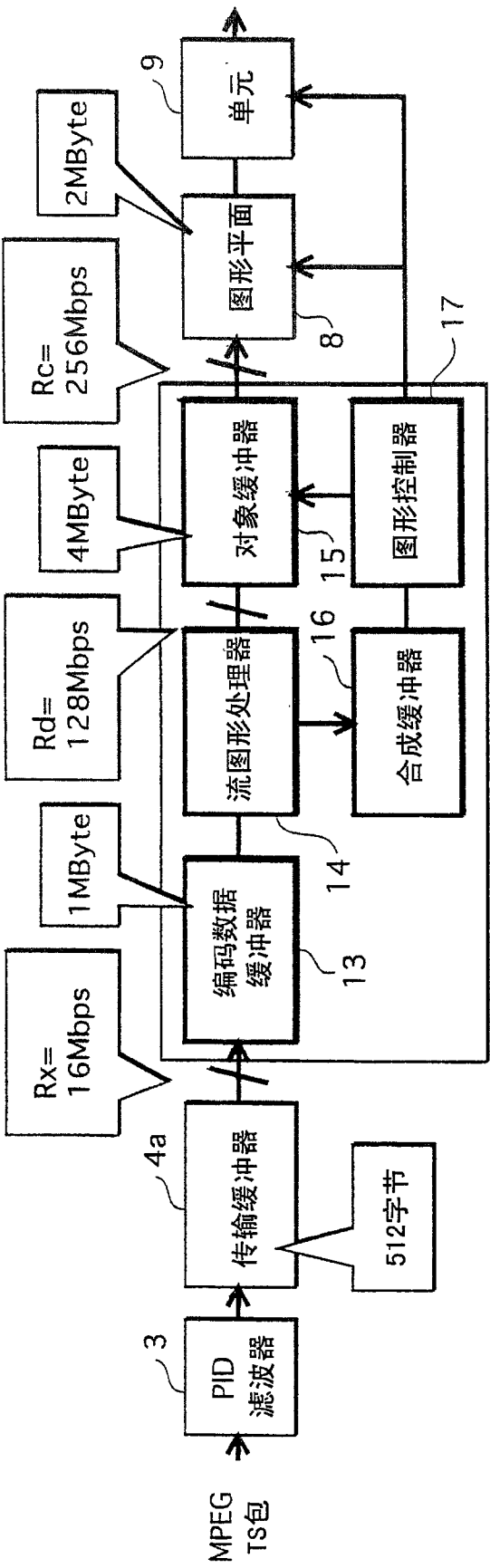


图29

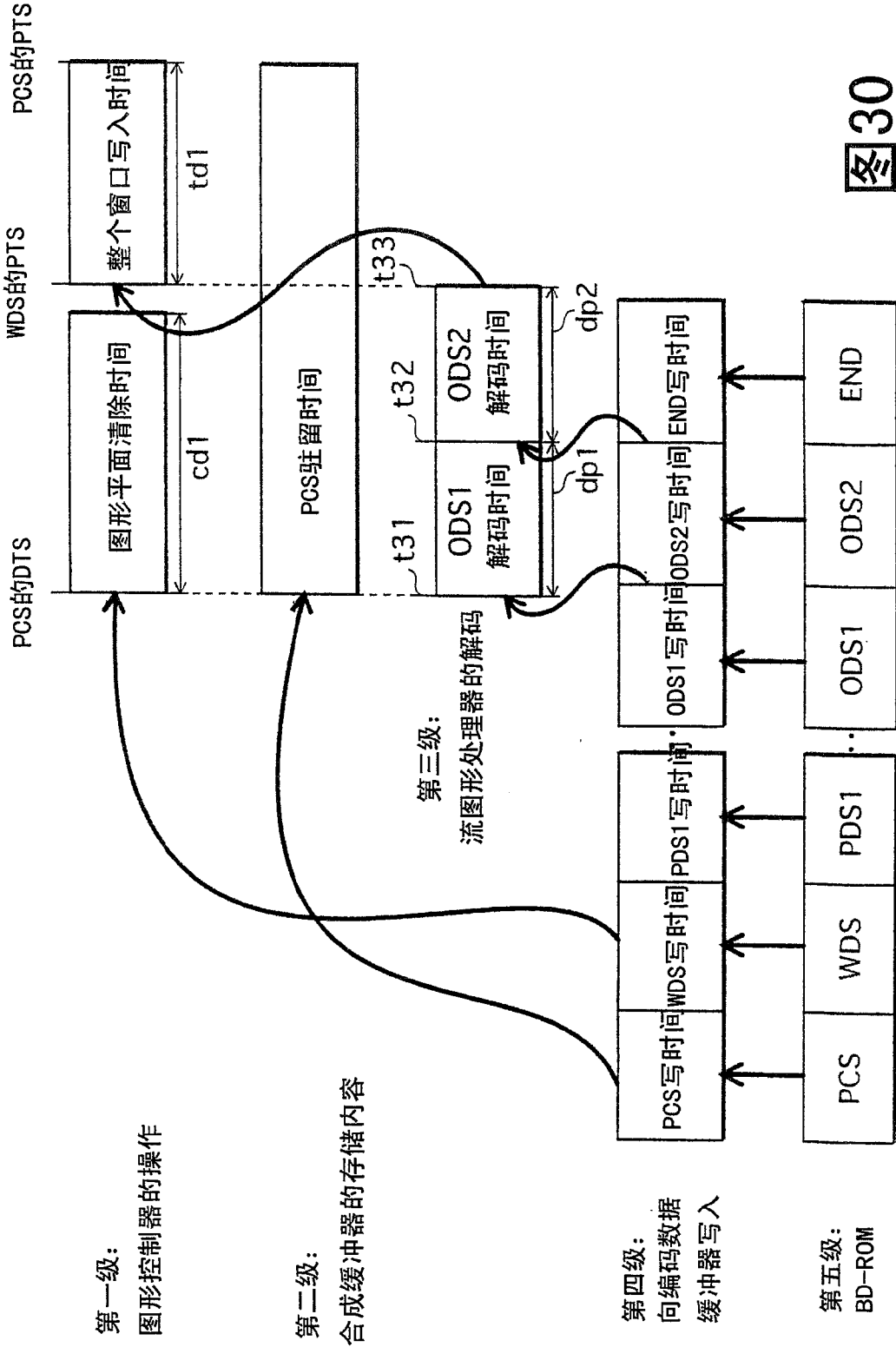


图30

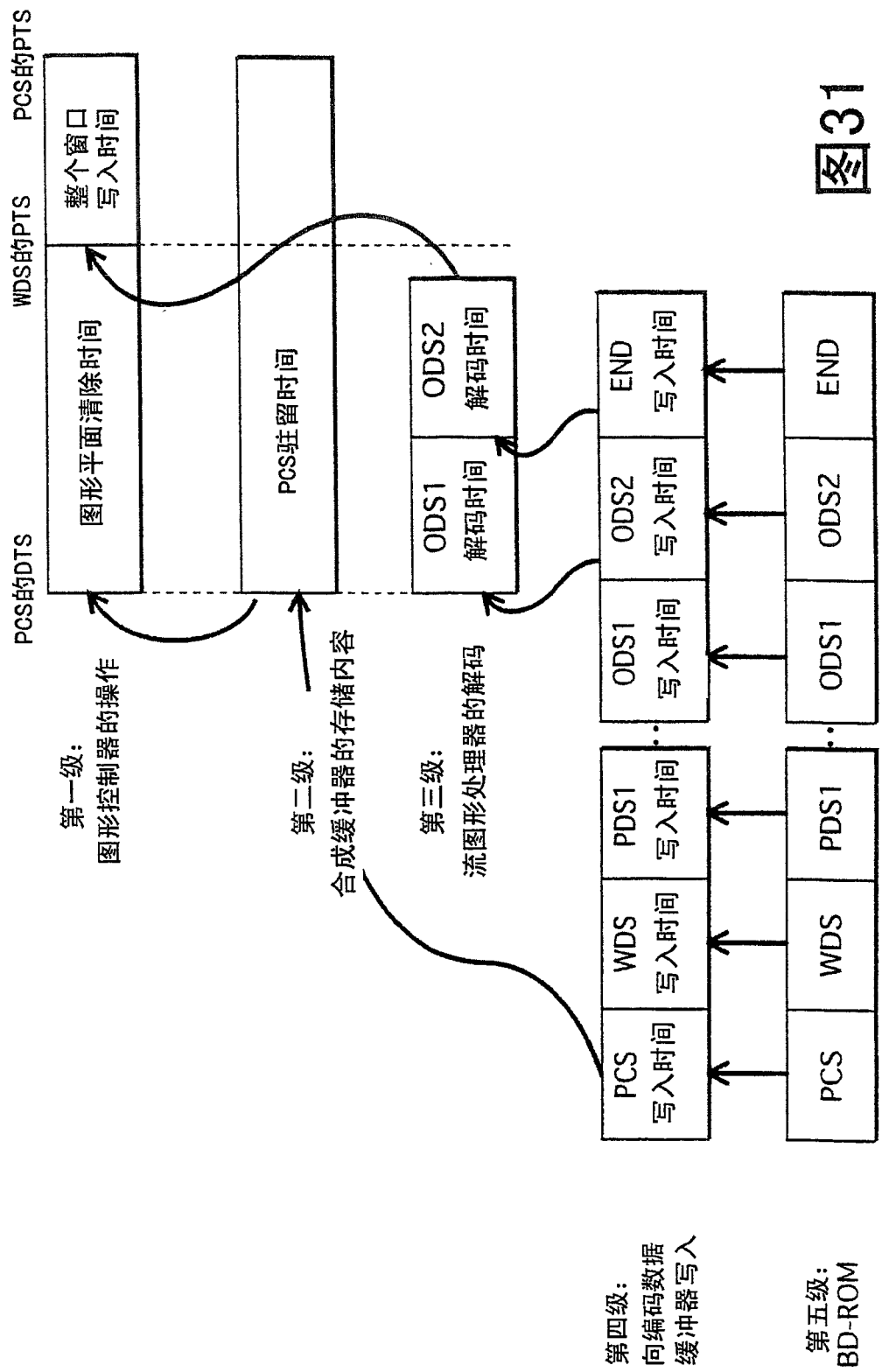


图31

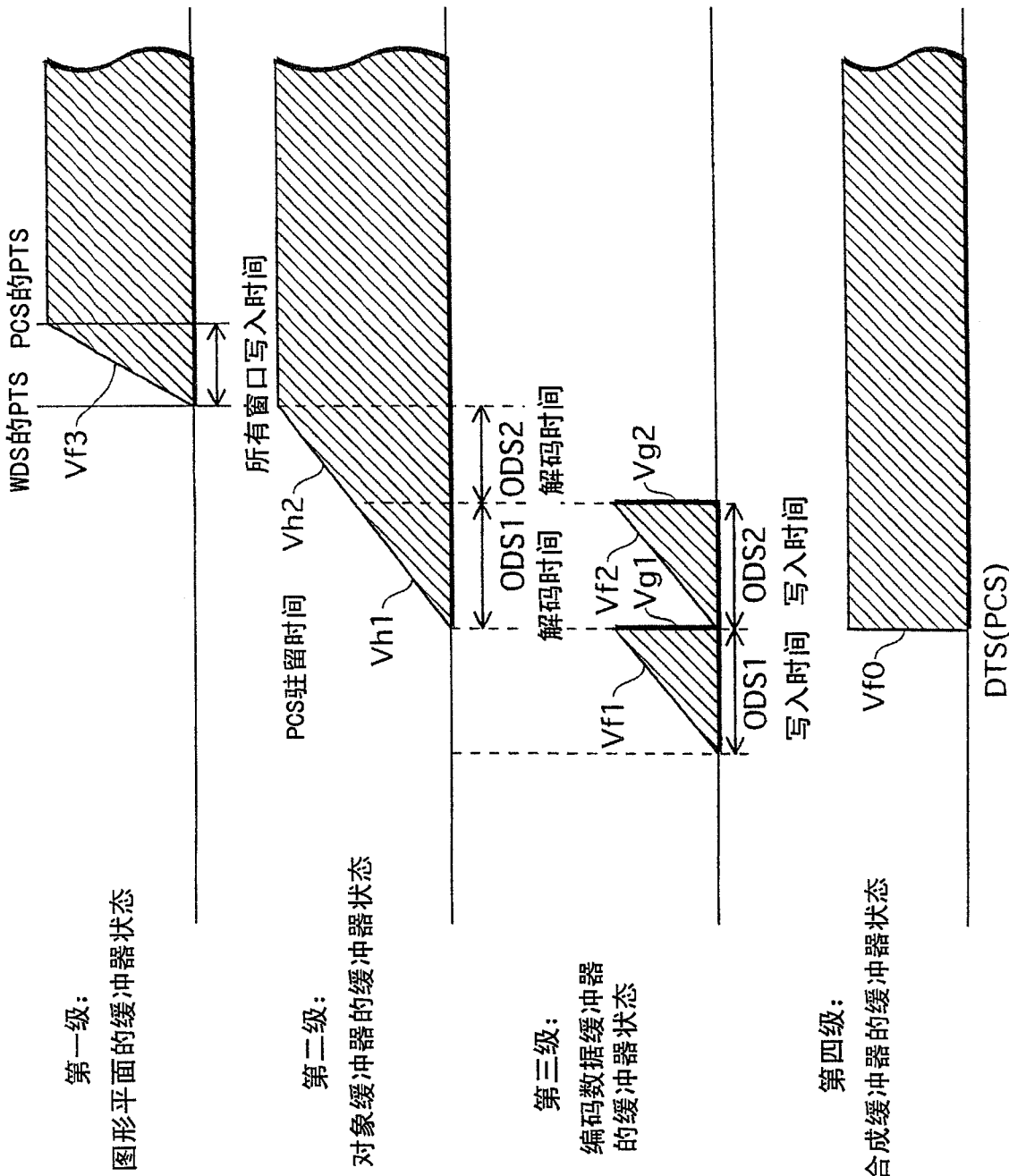


图32

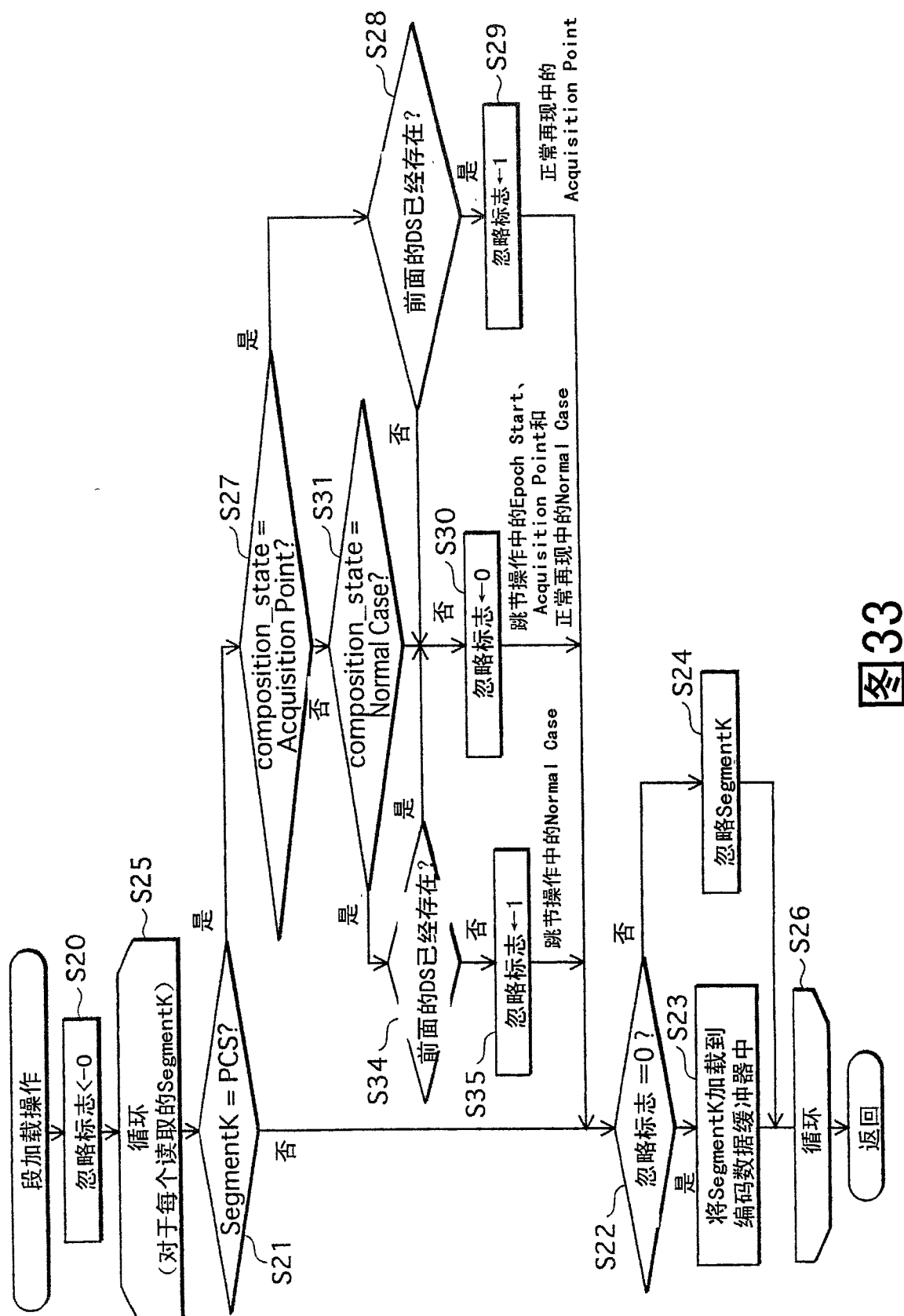


图 33

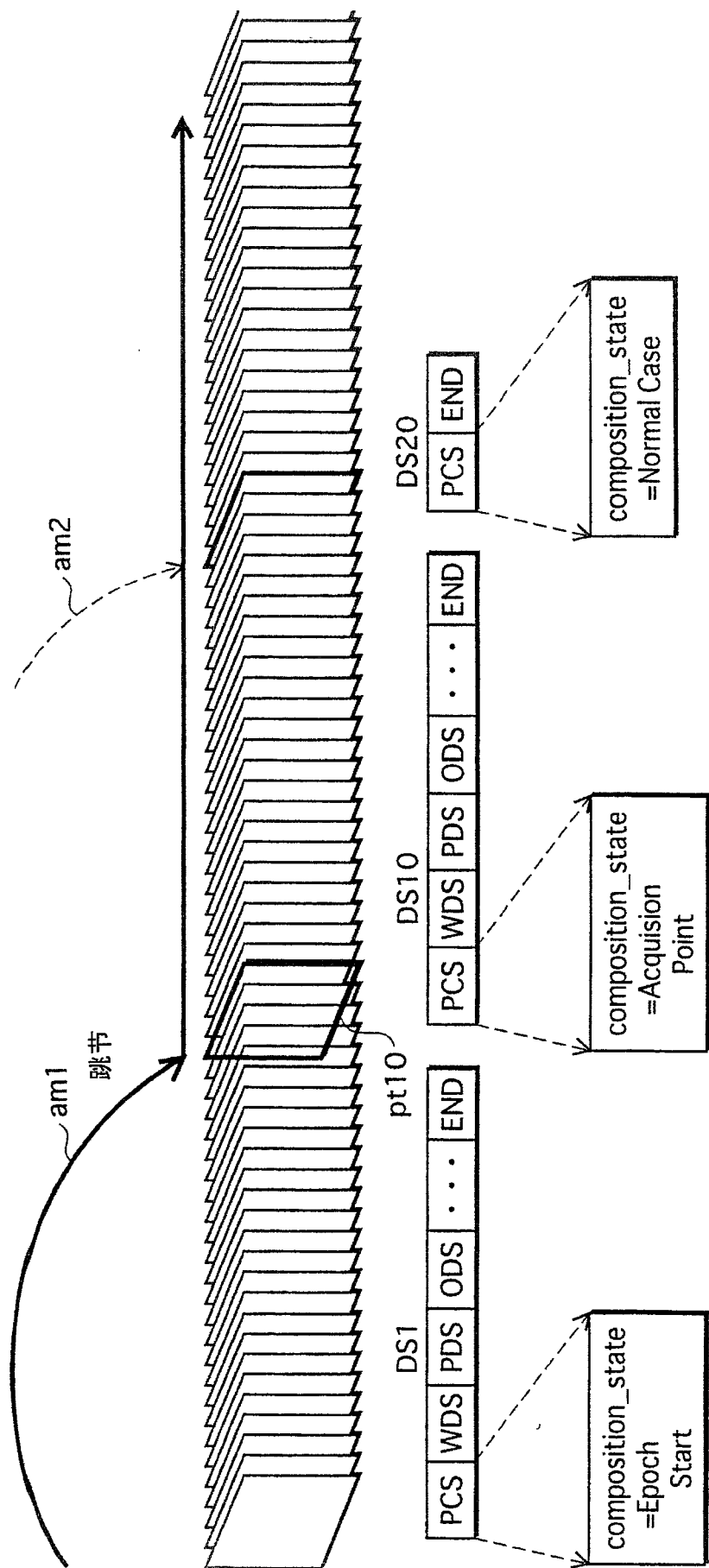


图34

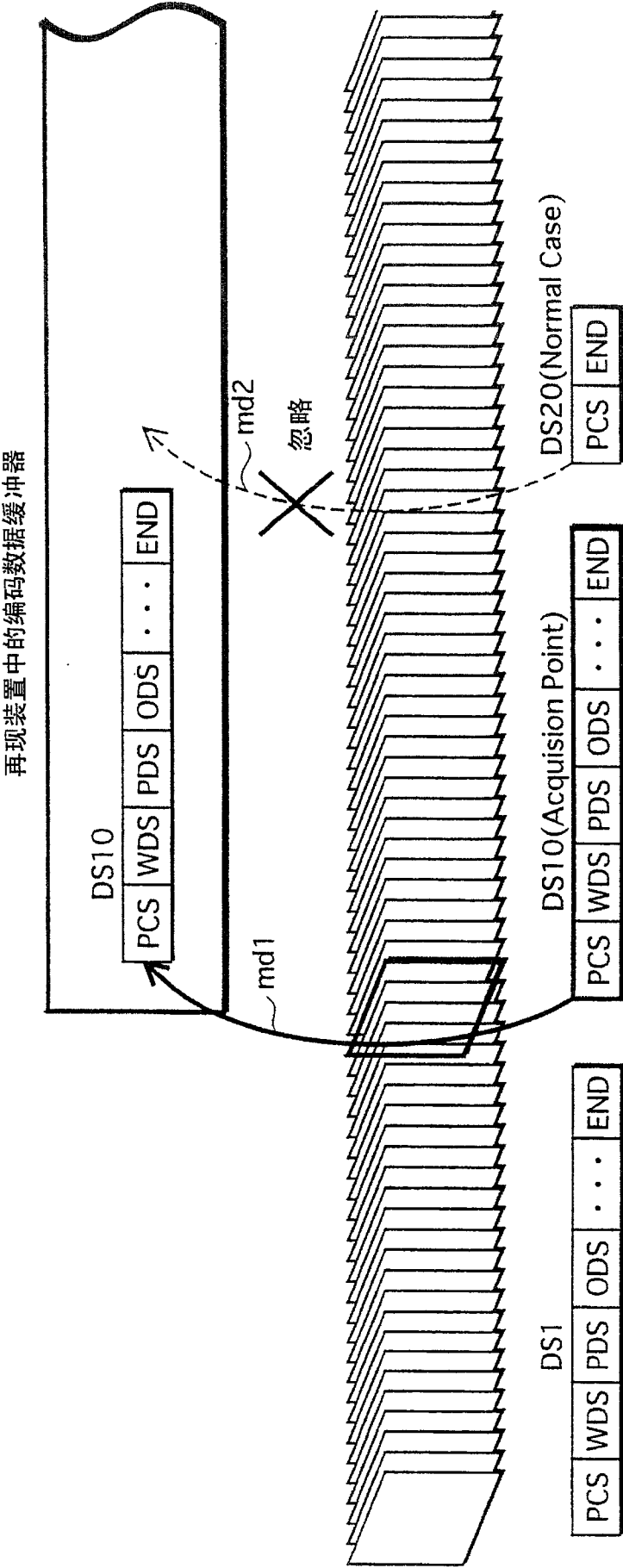


图35

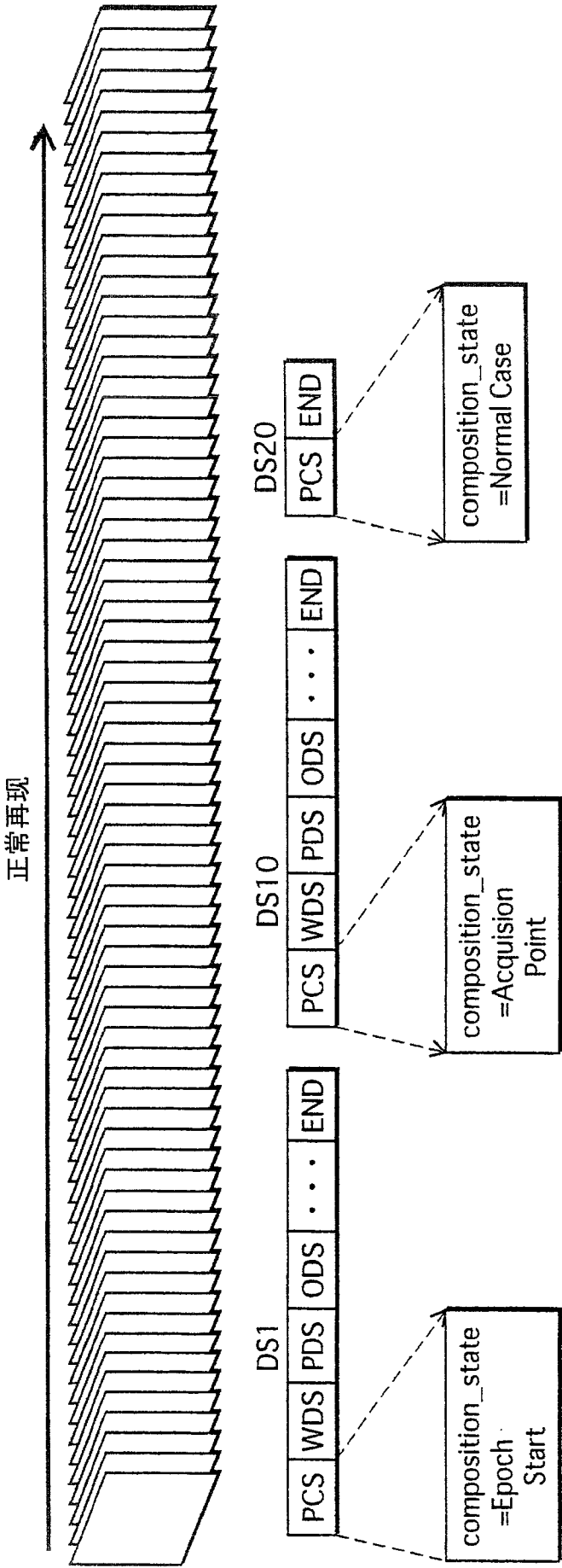


图36

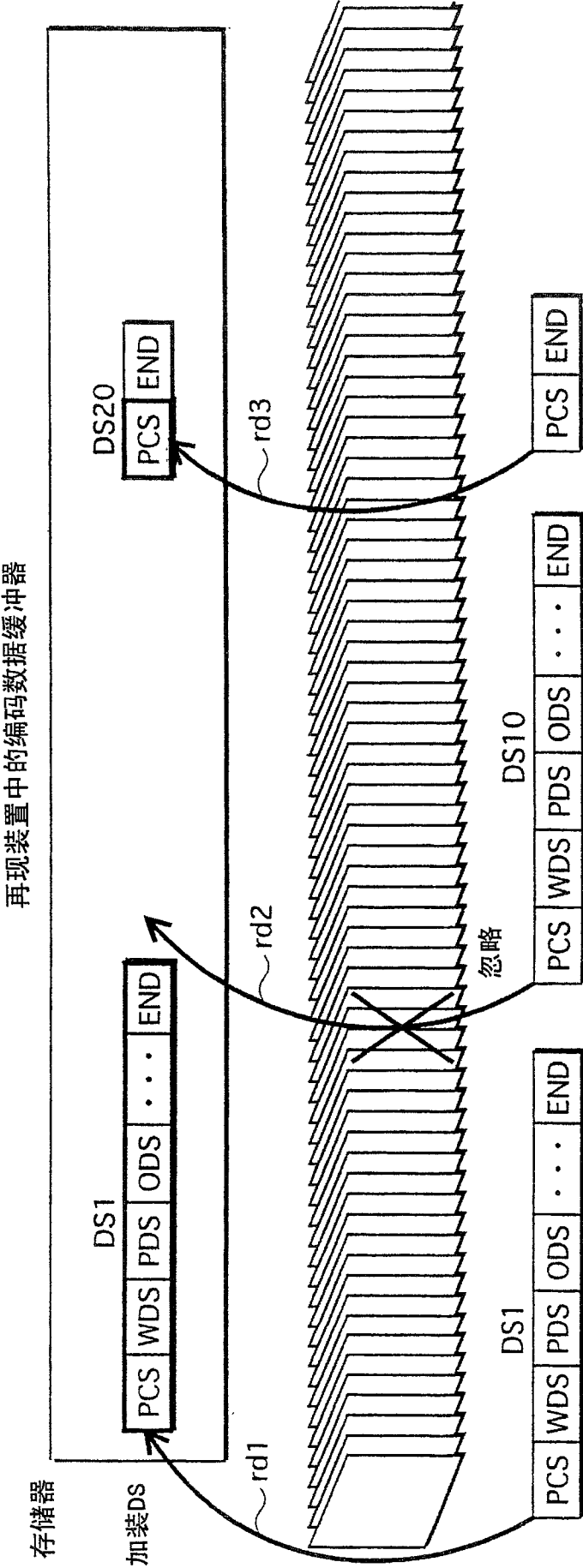


图37

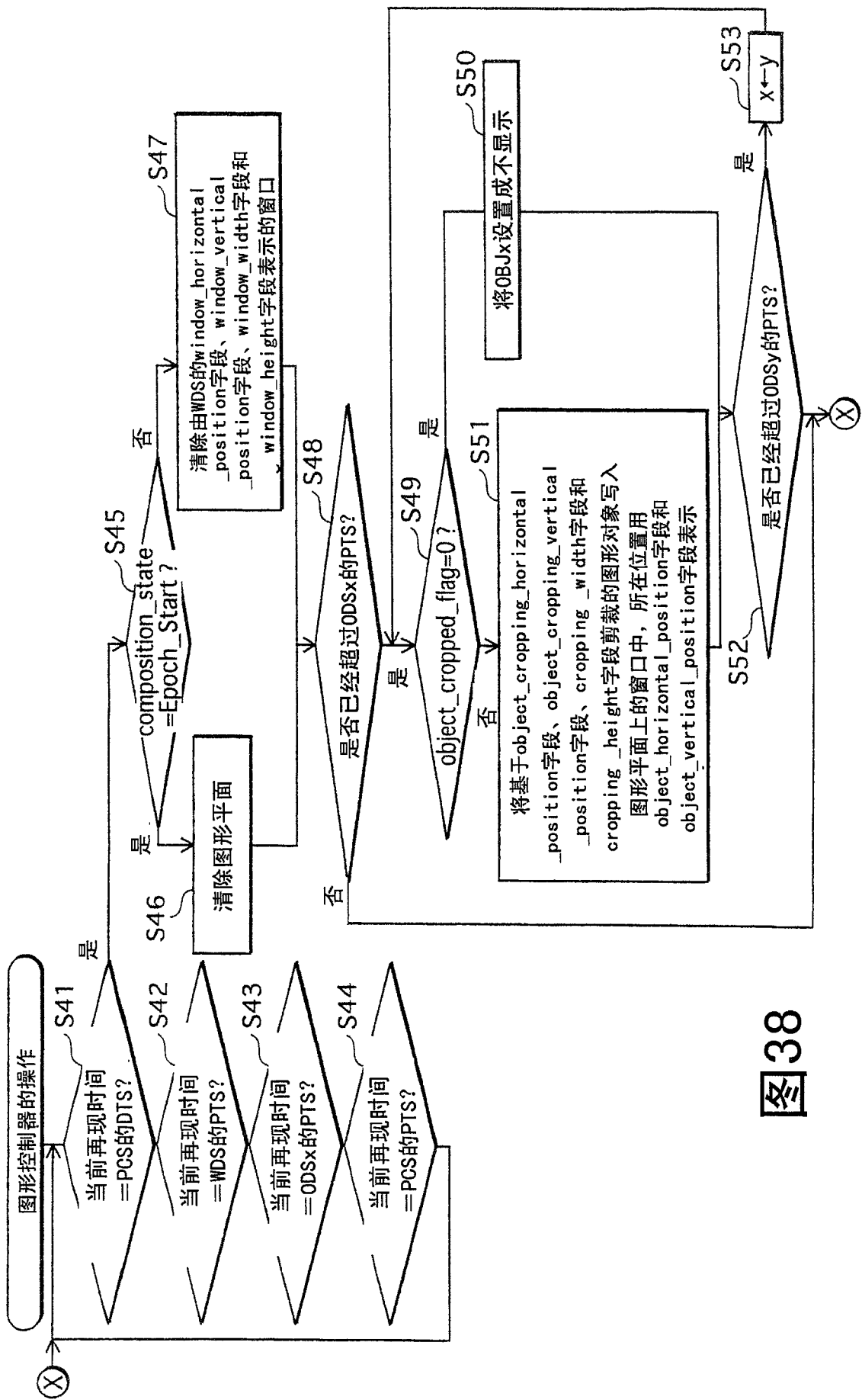


图38

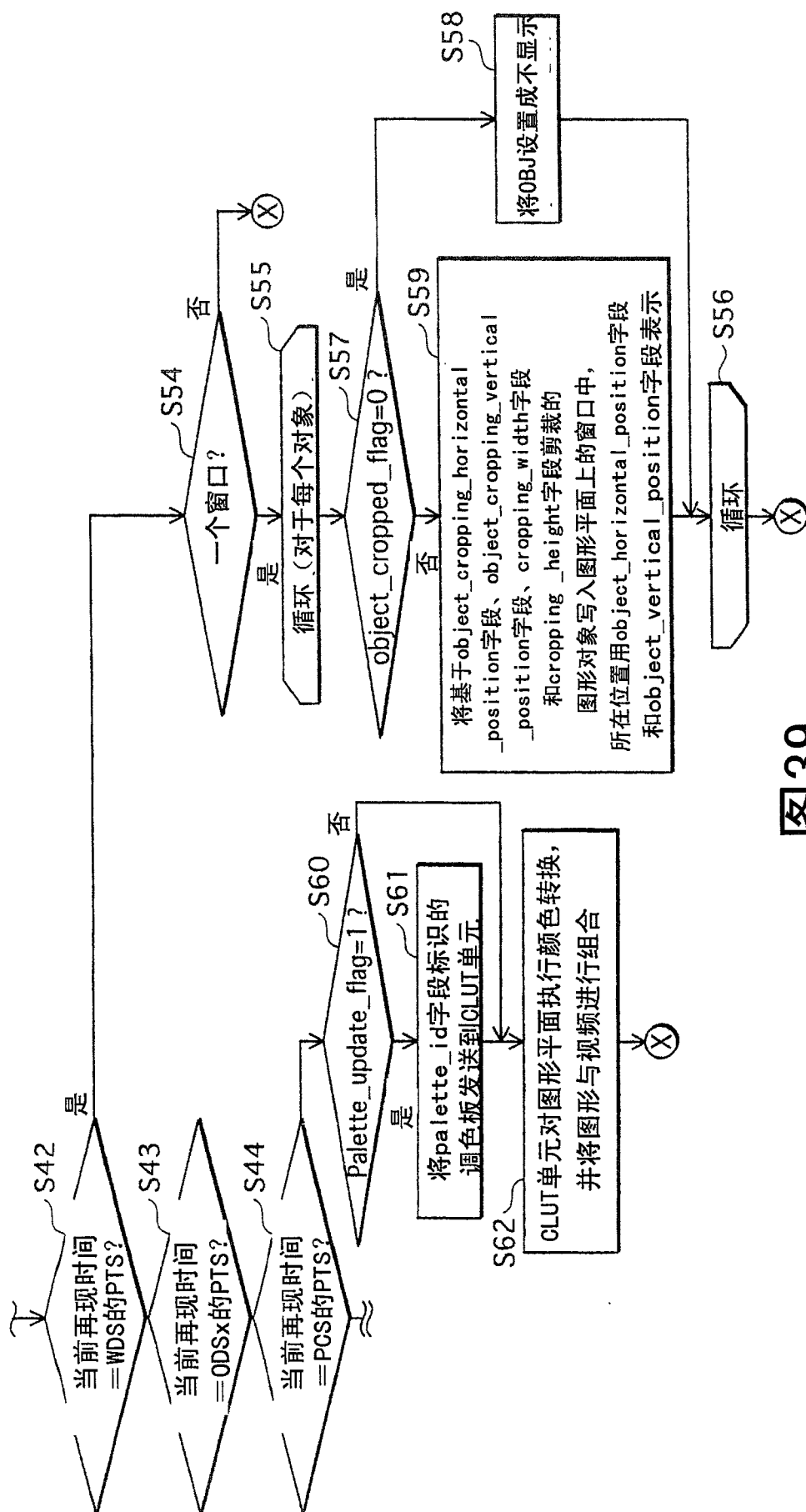


图39

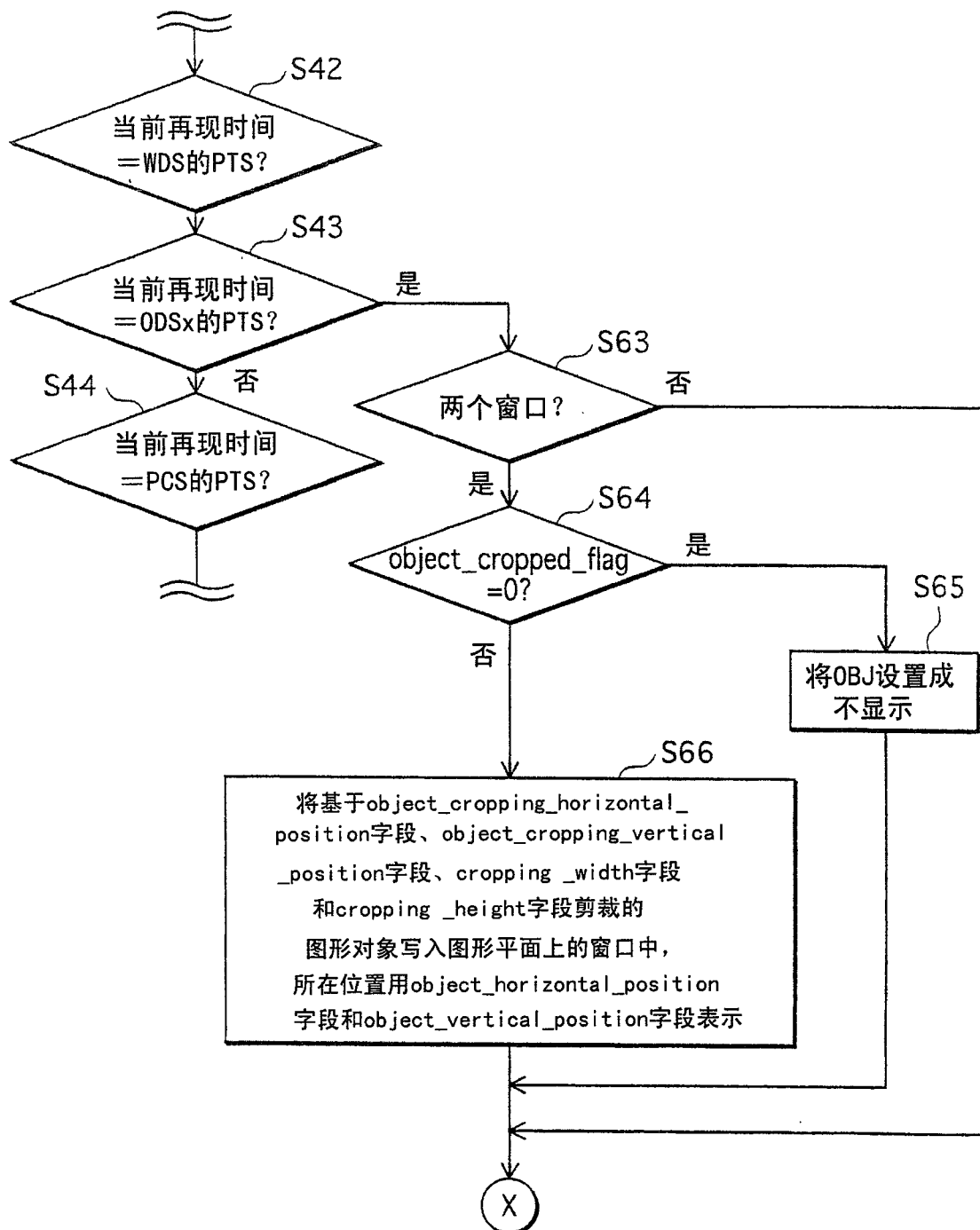


图40

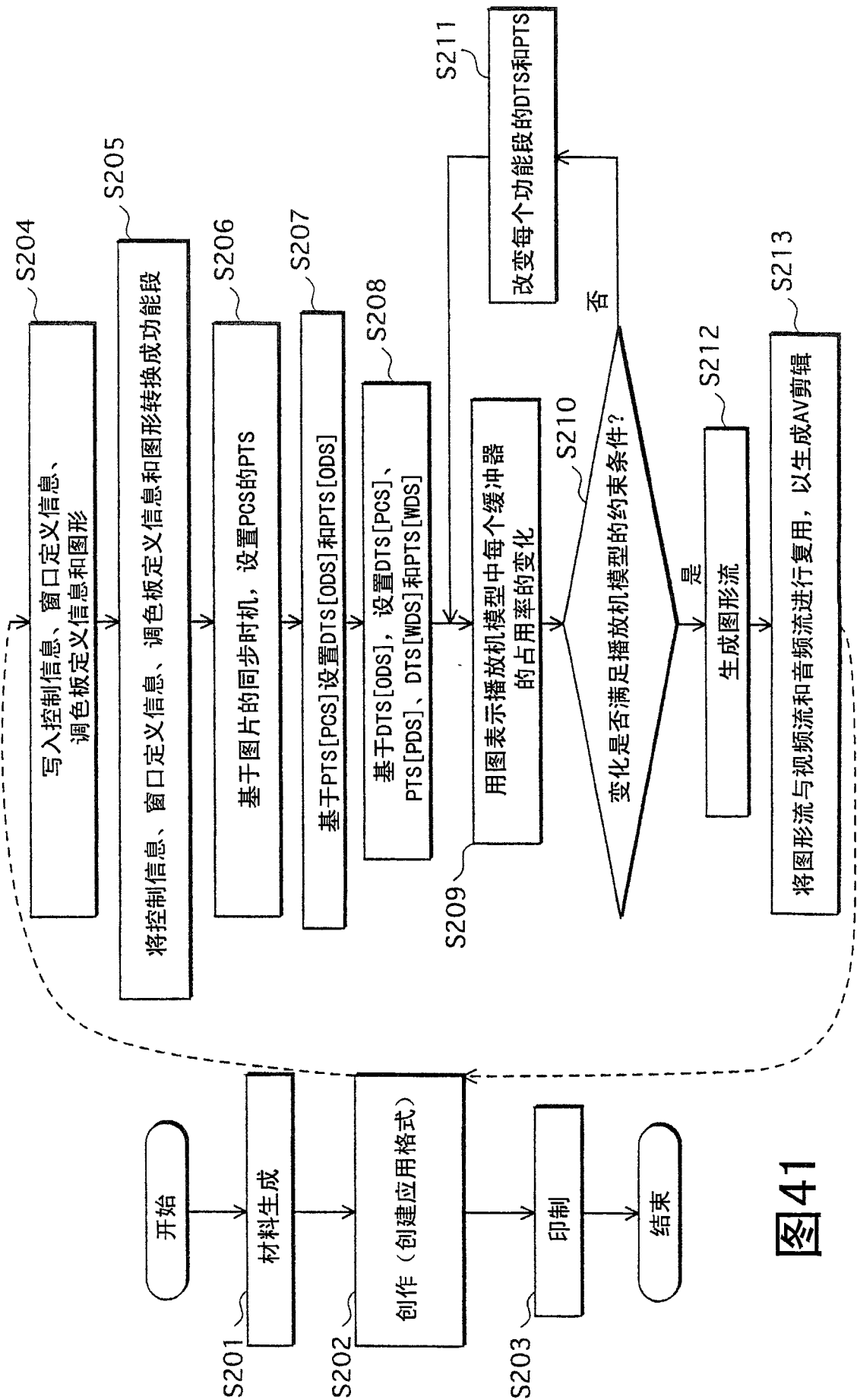


图41