[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G11B 20/10 (2006.01)
G11B 20/12 (2006.01)
H04N 5/92 (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200580027298.4

[43] 公开日 2007 年 7 月 18 日

[11] 公开号 CN 101002264A

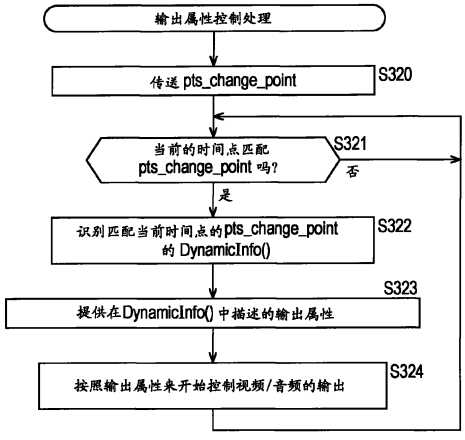
[22] 申请日 2005.5.24
[21] 申请号 200580027298.4
[30] 优先权
[32] 2004. 6.11 [33] JP [31] 174549/2004
[86] 国际申请 PCT/JP2005/009437 2005.5.24
[87] 国际公布 WO2005/122168 日 2005.12.22
[85] 进入国家阶段日期 2007.2.12
[71] 申请人 索尼株式会社
地址 日本东京都
[72] 发明人 藤波靖 浜田俊也
[74] 专利代理机构 北京市柳沈律师事务所
代理人 黄小临 王志森
权利要求书 3 页 说明书 103 页 附图 45 页

[54] 发明名称

数据处理装置和数据处理方法、程序和程序记录介质、数据记录介质和数据结构

[57] 摘要

本发明提供了一种数据处理装置、数据处理方法、程序、程序记录介质、数据记录介质和数据结构，其能够按照数据的播放时间来控制数据的输出。 与其中已经复用至少一个基本流的节目流相关联的元数据 Clip( ) 对于在节目流中复用的每个基本流，包括用于表示基本流的播放时间的 pts_change_point 和包括数据输出属性的 DynamicInfo( ) 的组。步骤 S321 判定是否正在被播放的基本流的播放时间与 pts_change_point 一致，当判定它们一致时，在步骤 S322 中，识别与 pts_change_point 一起构成组的 DynamicInfo( )。 在步骤 S324 中，按照在 DynamicInfo( ) 中包括的输出属性，控制正在被播放的数据的输出。 本发明可以被应用到使用例如 DVD 的游戏装置。



1. 一种数据处理装置，用于处理在数据记录介质上记录的所记录数据，所述所记录数据包括:

复用数据，其中已经复用了一个或多个数据；以及

元数据，其与所述复用数据相关联，

对于在所述复用数据中复用的每个数据，所述元数据包括用于指示所述数据的再现时间点的时间点信息和包括所述数据的输出属性的预定信息的组，

所述数据处理装置包括:

确定部件，用于确定是否被再现的所述数据的再现时间点匹配所述时间点信息;

识别部件，用于在所述确定部件已经确定被再现的所述数据的再现时间点匹配所述时间点信息的情况下，识别与所述时间点信息构成组的所述预定信息；以及，

控制部件，用于按照在由所述识别部件识别的所述预定信息中包括的输出属性来控制被再现的所述数据的输出。

2. 按照权利要求 1 的数据处理装置，其中，所述输出属性是视频长宽比或者音频声道分配。

3. 一种数据处理方法，用于处理在数据记录介质上记录的所记录数据，所述所记录数据包括:

复用数据，其中已经复用了一个或多个数据，以及

元数据，其与所述复用数据相关联，

对于在所述复用数据中复用的每个数据，所述元数据包括用于指示所述数据的再现时间点的时间点信息和包括所述数据的输出属性的预定信息的组，

所述方法包括:

确定步骤，用于确定是否被再现的所述数据的再现时间点匹配所述时间点信息;

识别步骤，用于在所述确定步骤已经确定被再现的所述数据的再现时间点匹配所述时间点信息的情况下，识别与所述时间点信息构成组的所述预定

信息；以及，

控制步骤，用于按照在由所述识别步骤识别的所述预定信息中包括的输出属性来控制被再现的所述数据的输出。

4. 一种程序，用于使得计算机执行用于处理在数据记录介质上记录的所记录数据的数据处理，

所述所记录数据包括：

复用数据，其中已经复用了一个或多个数据；以及

元数据，其与所述复用数据相关联，

对于在所述复用数据中复用的每个数据，所述元数据包括用于指示所述数据的再现时间点的时间点信息和包括所述数据的输出属性的预定信息的组，

所述程序包括：

确定步骤，用于确定是否被再现的所述数据的再现时间点匹配所述时间点信息；

识别步骤，用于在所述确定步骤已经确定被再现的所述数据的再现时间点匹配所述时间点信息的情况下，识别与所述时间点信息构成组的所述预定信息；以及，

控制步骤，用于按照在由所述识别步骤识别的所述预定信息中包括的输出属性来控制被再现的所述数据的输出。

5. 一种程序记录介质，其中记录了程序，用于使得计算机执行用于处理在数据记录介质上记录的所记录数据的数据处理，

所述所记录数据包括：

复用数据，其中已经复用了一个或多个数据；以及

元数据，其与所述复用数据相关联，

对于在所述复用数据中复用的每个数据，所述元数据包括用于指示所述数据的再现时间点的时间点信息和包括所述数据的输出属性的预定信息的组，

所述程序包括：

确定步骤，用于确定是否被再现的所述数据的再现时间点匹配所述时间点信息；

识别步骤，用于在所述确定步骤已经确定被再现的所述数据的再现时间

点匹配所述时间点信息的情况下，识别与所述时间点信息构成组的所述预定信息；以及，

控制步骤，用于按照在由所述识别步骤识别的所述预定信息中包括的输出属性来控制被再现的所述数据的输出。

6. 一种数据记录介质，其中记录了所记录数据，所述所记录数据包括：

复用数据，其中已经复用了一个或多个数据；以及

元数据，其与所述复用数据相关联，

对于在所述复用数据中复用的每个数据，所述元数据包括用于指示所述数据的再现时间点的时间点信息和包括所述数据的输出属性的预定信息的组。

7. 一种数据结构，包括：其中已经复用了一个或多个数据的复用数据，以及与所述复用数据相关联的元数据；

其中，对于在所述复用数据中复用的每个数据，所述元数据包括用于指示所述数据的再现时间点的时间点信息和包括所述数据的输出属性的预定信息的组。

数据处理装置和数据处理方法、程序和程序记录介质、数据记录介质和数据结构

技术领域

本发明涉及数据处理装置和数据处理方法、程序和程序记录介质、数据记录介质和数据结构，特别涉及例如很方便地使得能够进行数据处理的数据处理装置和数据处理方法、程序和程序记录介质、数据记录介质和数据结构。

背景技术

近些年来，例如 DVD(数字通用盘)已经作为具有大容量和可以被随机存取的记录介质而得到广泛使用，而且，使用 DVD 来执行各种类型的处理的 DVD 装置也得到广泛使用。

DVD 装置的类型包括例如：DVD 录像机，它在 DVD 上记录和播放电视广播节目数据等；汽车导航系统，其中，在 DVD 上记录映射信息等并且显示所述映射信息；游戏机，其中，在 DVD 上记录游戏程序等并且执行所述程序。

在例如非专利文件 1 中描述了 DVD 的细节。

非专利文件 1 DVD Specifications for Read-Only Disc Part 3; Version 1.1 December 1997(用于只读盘的 DVD 说明书第 3 部分；版本 1.1，1997 年 12 月)

发明内容

本发明要解决的问题

对于诸如其中可以记录大量数据的 DVD 之类的记录媒体和使用所述数据来执行各种类型处理的 DVD 装置，对于这样的大量数据，要求以高度方便的方式来执行数据处理。

本发明考虑到这样的情况而做出，并且旨在使得数据处理具有高度方便性。

用于解决所述问题的手段

按照本发明的数据处理装置，其中，在与复用数据（其中已经复用了一个或多个数据）相关联的元数据对于在复用数据中被复用的每个数据而包括用于指示所述数据的再现时间点（point-in-time）的时间点信息以及用于包括所述数据的输出属性的预定信息之组（set）的情况下，所述数据处理装置包括：确定部件，用于确定是否被再现的数据的再现时间点匹配所述时间点信息；识别部件，用于在所述确定部件已经确定被再现的数据的再现时间点匹配所述时间点信息的情况下，识别与所述时间点信息构成组的所述预定信息；以及控制部件，用于按照在由所述识别部件识别的预定信息中包括的输出属性来控制被再现的数据的输出。

按照本发明的数据处理方法，其中，在与复用数据（其中已经复用了一个或多个数据）相关联的元数据对于在复用数据中被复用的每个数据而包括用于指示所述数据的再现时间点的时间点信息以及用于包括所述数据的输出属性的预定信息之组的情况下，所述方法包括：确定步骤，用于确定是否被再现的数据的再现时间点匹配所述时间点信息；识别步骤，用于在所述确定步骤中已经确定被再现的数据的再现时间点匹配所述时间点信息的情况下，识别与所述时间点信息构成组的所述预定信息；以及控制步骤，用于按照在由所述识别步骤识别的所述预定信息中包括的输出属性来控制被再现的数据的输出。

按照本发明的程序，其中，在与复用数据（其中已经复用了一个或多个数据）相关联的元数据对于在复用数据中被复用的每个数据而包括用于指示所述数据的再现时间点的时间点信息以及用于包括所述数据的输出属性的预定信息之组的情况下，所述程序包括：确定步骤，用于确定是否被再现的数据的再现时间点匹配所述时间点信息；识别步骤，用于在所述确定步骤已经确定被再现的数据的再现时间点匹配所述时间点信息的情况下，识别与所述时间点信息构成组的所述预定信息；以及控制步骤，用于按照在由所述识别步骤识别的预定信息中包括的输出属性来控制被再现的数据的输出。

按照本发明的在记录介质中记录的程序，其中，在与复用数据（其中已经复用了一个或多个数据）相关联的元数据对于在复用数据中被复用的每个数据而包括用于指示所述数据的再现时间点的时间点信息以及用于包括所述数据的输出属性的预定信息之组的情况下，所述程序包括：确定步骤，用于确定是否被再现的数据的再现时间点匹配所述时间点信息；识别步骤，用于

在所述确定步骤已经确定被再现的数据的再现时间点匹配所述时间点信息的情况下，识别与所述时间点信息构成组的所述预定信息；以及控制步骤，用于按照在由所述识别步骤识别的预定信息中包括的输出属性来控制被再现的数据的输出。

按照本发明的在数据记录介质中记录的被记录数据和数据结构，包括：复用数据，其中已经复用了一个或多个数据；以及元数据，它与所述复用数据相关联，对于在复用数据中被复用的每个数据，所述元数据包括用于指示所述数据的再现时间点的时间点信息以及用于包括所述数据的输出属性的预定信息的组。

使用按照本发明的数据处理装置和数据处理方法、以及所述程序和在程序记录介质中记录的程序，在组包括用于指示所述数据的再现时间点的时间点信息和用于包括所述数据的输出属性的预定信息的情况下，针对与复用数据（其中已经复用了一个或多个数据）相关联的元数据，对于在复用数据中被复用的每个数据，确定是否被再现的数据的再现时间点匹配所述时间点信息。在确定被再现的数据的再现时间点匹配所述时间点信息的情况下，识别与所述时间点信息构成组的预定信息，并且按照已经识别的预定信息中包括的输出属性来控制被再现的数据的输出。

使用按照本发明的在数据记录介质中记录的被记录数据和数据结构，包括其中已经复用了一个或多个数据的复用数据和与所述复用数据相关联的元数据。对于在复用数据中被复用的每个数据，所述元数据包括用于指示所述数据的再现时间点的时间点信息以及用于包括所述数据的输出属性的预定信息的组。

优点

按照本发明，可以实现很方便的数据处理等。特别是，可以对应于所述数据的再现时间点来控制数据的输出。

附图说明

图 1 是用于说明已经应用本发明的盘装置的一个实施例的硬件配置例子的方框图。

图 2 是用于说明由 CPU 112 执行的软件模块组的配置例子的方框图。

图 3 是用于说明缓冲器控制模块 215 的配置例子的方框图。

图 4 是用于说明在盘 101 中的目录配置例子的示意图。

图 5 是用于说明“PLAYLIST.DAT”文件的语法（syntax）的示意图。

图 6 是用于说明 PlayItem()的语法的示意图。

图 7 是用于说明 PlayListMark()的语法的示意图。

图 8 是用于说明在 mark_type 的值和 Mark()的类型之间的关系的示意图。

图 9 是用于说明在 PlayList()、PlayItem()、剪辑（clip）和在剪辑流文件中存储的节目流之间的关系的示意图。

图 10 是用于说明剪辑信息文件 Clip()的语法的示意图。

图 11 是用于说明在用于识别基本流的 stream_id 和专用 stream_id 与基本流之间的关系的示意图。

图 12 是用于说明 StaticInfo()的语法的示意图。

图 13 是用于说明 DynamicInfo()的语法的示意图。

图 14 是用于说明 EP_map()的语法的示意图。

图 15 是用于说明 MPEG-2 系统的节目流、节目流包（pack）和节目流包报头的语法的示意图。

图 16 是用于说明 MPEG-2 系统的 PES 分组的语法的示意图。

图 17 是用于说明 MPEG-2 系统的 PES 分组的语法的示意图。

图 18 是用于说明 MPEG-2 系统的 PES 分组的语法的示意图。

图 19 是用于说明在 MPEG-2 系统的 PES_packet()的 stream_id 中所述的值和基本流的属性(类型)之间的关系的示意图。

图 20 是用于说明由盘装置使用的 stream_id 的示意图。

图 21 是用于说明 private_stream1_PES_payload()的语法的示意图。

图 22 是用于说明在 private_stream_id 的值和 private_payload()中存储的基本流的属性之间的关系的示意图。

图 23 是用于说明 private_stream2_PES_payload()的语法的示意图。

图 24 是用于说明 au_information()的语法的示意图。

图 25 是用于说明“PLAYLIST.DAT”文件的具体例子的示意图。

图 26 是用于说明剪辑信息文件“00001.CLP”、“00002.CLP”和“00003.CLP”的具体例子的示意图。

图 27 是用于说明在剪辑信息文件“00001.CLP”中的 EP_map()的具体例子的示意图。

图 28 是用于说明在 PlayList#0 和 PlayList#1 内的 PlayListMark()的具体例子的示意图。

图 29 是用于描述预先播放处理的流程图。

图 30 是用于描述播放处理的流程图。

图 31 是用于描述 PlayItem 交叉处理的流程图。

图 32 是用于描述时间代码显示处理的流程图。

图 33 是用于描述流转换处理的流程图。

图 34 是用于描述缓冲器控制模块 215 的处理的流程图。

图 35 是用于描述缓冲器控制模块 215 的处理的流程图。

图 36 是用于描述用于读出视频流的处理的流程图。

图 37 是用于描述用于读出音频流的处理的流程图。

图 38 是用于描述用于读出字幕（subtitle）流的处理的流程图。

图 39 是用于描述重新同步处理的流程图。

图 40 是用于描述标记（mark）处理的流程图。

图 41 是用于描述输出属性控制处理的流程图。

图 42 是用于描述在剪辑信息文件“00003.CLP”中所描述的一组 pts_change_point 和 DynamicInfo()的具体例子的示意图。

图 43 是用于描述字幕显示控制处理的流程图。

图 44 是用于描述捕获控制处理和背景/屏幕保护程序（screen-saver）处理的流程图。

图 45 是用于说明 private_stream2_PES_payload()的另一个语法的示意图。

图 46 是用于说明 au_information()的另一个语法的示意图。

附图标号

101 盘

102 盘驱动器

111 总线

112 CPU

113 存储器

114 驱动器接口

115 输入接口

116	视频解码器
117	音频解码器
118	视频输出接口
119	音频输出接口
120	视频输出端
121	音频输出端
201	操作系统
210	视频内容重放程序
211	脚本控制模块
212	播放器控制模块
213	内容数据提供模块
214	解码器控制模块
214A	时钟单元
215	缓冲器控制模块
215A	缓冲器
216	视频解码器控制模块
217	音频解码器控制模块
218	字幕解码器控制模块
219	图形处理模块
220	视频输出模块
220A	FIFO(先入先出)
221	音频输出模块
221A	FIFO
231	数据首部指针存储单元
232	数据写入指针存储单元
233	视频读取功能单元
234	音频读取功能单元
235	字幕读取功能单元
241	视频读取指针存储单元
242	stream_id 寄存器
243	au_information()寄存器

251 音频读取指针存储单元

252 stream_id 寄存器

253 private_stream_id 寄存器

261 字幕读取功能标记存储单元

262 字幕读取指针存储单元

263 stream_id 寄存器

264 private_stream_id 寄存器

具体实施方式

下面参考附图来说明本发明的实施例。

[硬件配置]

图 1 是用于说明已经应用本发明的盘装置的一个实施例的硬件配置例子的方框图。

图 1 所示的盘装置可以被应用到例如盘播放器、游戏装置、汽车导航系统等。

对于在图 1 中所示的盘装置，盘 101 例如是诸如 DVD 的光盘或者磁光盘、磁盘等，并且其中记录了诸如视频数据、音频数据、字幕数据等的内容数据，并且还记录了播放内容数据所需要的数据。

注意，如果需要的话，在盘 101 中记录的数据(记录数据)也包括可以由计算机执行的程序。而且，对于本实施例，使用作为盘形记录媒体的盘 101，但是，可以使用其他的记录媒体，诸如半导体存储器或带形记录媒体。而且，可以向其输入从远程盘 101 读出的并且被发送到图 1 所示盘装置的数据。即，可以进行布置，其中，在连接到盘装置的另一个装置上执行从盘 101 读出的数据，并且在所述盘装置上接收和处理从所述另一个装置读出的数据。而且，所述盘装置可以经由诸如因特网的网络从服务器等（在其存储器中存储了与在盘 101 中记录的数据相同的数据）中接收和处理数据分布。而且，所述盘装置可以从服务器或另一个装置接收数据，将其暂时记录在盘 101 中，然后处理在盘 101 中记录的所述数据。

盘 101 可装卸地被安装到盘驱动器 102。盘驱动器 102 具有内置的未示出接口，并且经由所述接口而连接到驱动器接口 114。所述盘驱动器 102 驱动被安装到其上的盘 101，按照诸如用于从盘接口 114 读出的命令来从盘 101

读出数据，并且执行诸如向驱动器接口 114 等提供之类的处理。

连接到总线 111 的是 CPU(中央处理单元)112、存储器 113、驱动器接口 114、输入接口 115、视频解码器 116、音频解码器 117、视频输出接口 118 和音频输出接口 119。

CPU 112 和存储器 113 构成计算机系统。即，CPU 112 执行作为在存储器 113 中存储的程序的后述的软件模块组，由此控制整个盘装置，并且还执行各种类型的后述的处理。存储器 113 存储要由 CPU 112 执行的软件模块组。而且，存储器 113 暂时存储用于 CPU 112 的操作所需要的数据。注意，存储器 113 可以被仅仅配置非易失性存储器或易失性存储器和非易失性存储器的组合。而且，可以进行布置，其中，在图 1 中所示的盘装置被提供有硬盘，并且在所述硬盘中记录(安装)要由 CPU 112 执行的软件模块组，在这种情况下，存储器 113 可以被单独配置易失性存储器。

现在，可以预先在存储器 113 中记录(存储)要由 CPU 112 执行的程序(软件模块组)，存储器 113 用作为被内置到所述盘装置中的记录介质。

或者，所述程序可以被暂时或者永久地存储(记录)在盘 101 中或者除了盘 101 之外的可装卸记录媒体中，诸如软盘、CD-ROM(致密盘只读存储器)、MO(磁光)盘、磁盘、存储卡等。这样的可装卸记录媒体可以被提供作为所谓的封装软件。

注意，可以预先在存储器 113 中存储程序，或者可以从诸如如上所述的可装卸记录介质在盘装置中安装程序。而且，可以无线地经由数字卫星广播卫星从下载站点向盘装置传送所述程序，或者经由诸如 LAN(局域网)、因特网等的网络来向盘装置传送程序，盘装置使用输入接口 115 来接收如此被传送的程序，以便安装到内置的存储器 113 中。

而且，所述程序可以由单个 CPU 处理，或者可以进行在多个 CPU 之间的分布式处理。

驱动器接口 114 在 CPU 112 的控制下控制盘驱动器 102，并且由此经由总线 111 向 CPU 112、存储器 113、视频解码器 116、音频解码器 117 等提供盘驱动器 102 已经从盘 101 读出的数据。

输入接口 115 接收由用于操作未示出按键(按钮)或遥控器的用户提供的信号，并且经由总线 111 将它们提供到 CPU 112。注意，输入接口 115 也起作为通信接口，诸如调制解调器(包括 ADSL(不对称数字用户线)调制解调器)

和 NIC(网络接口卡)。

视频解码器 116 解码由盘驱动器 102 从盘 101 读出并且经由驱动器接口 114 和总线 111 提供的的视频数据的编码数据(编码的音频数据)，并且经由总线 111 向 CPU 112 和视频输出接口 118 提供作为其结果而获得的视频数据。

音频解码器 117 解码由盘驱动器 102 从盘 101 读出并且经由驱动器接口 114 和总线 111 提供的音频数据的编码数据(编码的音频数据)，并且经由总线 111 向 CPU 112 和音频输出接口 119 提供作为其结果而获得的音频数据。

视频输出接口 118 将经由总线 111 提供的视频数据进行必要的处理，并且从视频输出端 120 输出。音频输出接口 119 将经由总线 111 提供的音频数据进行必要的处理，并且从音频输出端 121 输出。

视频输出端 120 连接到诸如 CRT(阴极射线管)、液晶板等的未示出的视频输出装置，因此，从视频输出端 120 输出的视频数据被提供和显示在所述视频输出装置上。音频输出端 121 连接到诸如扬声器、放大器等的未示出的音频输出装置，因此，从音频输出端 121 输出的音频数据被提供到所述音频输出装置和从所述音频输出装置输出。

注意，可以经由电缆或无线地执行从所述盘装置向视频输出装置和音频输出装置提供视频数据和音频数据。

[软件模块组的配置]

图 2 说明了由图 1 所示的 CPU 112 执行的软件模块组的配置例子。

CPU 112 执行的软件模块组被大致划分为操作系统(OS)201 和用作为应用程序的视频内容播放程序 20。

“操作系统 201”

在接通盘装置的电源时首先激活操作系统 201(CPU 112 执行操作系统 201)，执行诸如初始设置等的必要处理等，并且调用用作为应用程序的视频内容播放程序 210。

操作系统 201 向视频内容播放程序 210 提供诸如文件读出的基础结构之类的服务。即，对于例如文件读出，响应于来自视频内容播放程序 210 的文件读出请求，操作系统 201 经由驱动器接口 114 提供操作盘驱动器 102 的服务，读出盘 101 的数据，并且将其传送到视频内容播放程序 210。而且，操作系统 201 执行文件系统的解释等。

注意，操作系统 201 具有多任务功能，并且能够时间共享(明显的)多个

软件模块的同时操作。即，视频内容播放程序 210 被配置几个软件模块，并且每个软件模块能够并行运行。

“视频内容播放程序 210”

视频内容播放程序 210 被配置脚本控制模块 211、播放器控制模块 212、内容数据提供模块 213、解码控制模块 214、缓冲器控制模块 215、视频解码器控制模块 216、音频解码器控制模块 217、字幕解码器控制模块 218、图形处理模块 219、视频输出模块 220 和音频输出模块 221。

视频内容播放程序 210 是在盘 101 的播放中具有中心作用的软件，并且在盘 101 被安装(插入)到盘驱动器 102 时，进行有关是否盘 101 是其中记录了内容的后述格式的盘的确定。而且，视频内容播放程序 210 从盘 101 读出和执行后述的脚本文件，并且也从盘 101 读出用于播放在盘 101 中记录的内容所需要的元数据(数据库信息)文件，并且基于所述元数据来控制内容的播放。

下面是构成在图 2 中所示的视频内容播放程序 210 的软件模块的说明。注意，在图 2 中，作为规则，实线箭头表示内容的数据，虚线箭头表示控制数据。

“脚本控制模块 211”

脚本控制模块 211 解释和执行在盘 101 中记录的脚本文件中所述的脚本程序(脚本)。可以使用脚本程序来描述动作，诸如“操作图形处理模块 219 以建立和显示诸如菜单的图像”、“按照来自诸如遥控器的 UI(用户界面)的信号来改变菜单显示(例如在菜单等上移动光标)”、“控制播放器控制模块 212”等。

“播放器控制模块 212”

播放器控制模块 212 引用在盘 101 中记录的元数据(数据库信息)等，并且执行与内容的播放相关联的控制。即，播放器控制模块 212 分析在盘 101 中记录的后述 PlayList()或 Clip()，并且按照其分析结果，控制内容数据提供模块 213、解码控制模块 214 和缓冲器控制模块 215。而且，播放器控制模块 212 按照来自脚本控制模块 211 或输入接口 115 的指令执行其中转换要播放的流的后述的流转换等。而且，播放器控制模块 212 从解码控制模块 214 获得时间点，并且执行时间点显示、后述标记(Mark())处理等。

“内容数据提供模块 213”

在播放器控制模块 212 的控制下或者基于在缓冲器控制模块 215 中累积

的数据量，内容数据提供模块 213 请求操作系统 201 以从盘 101 中读出内容数据、元数据等。

注意，操作系统 201 响应于来自内容数据提供模块 213 的请求而从盘 101 中已经读出的元数据等被提供到必要的模块。而且，操作系统 201 基于来自内容数据提供模块 213 的请求而已经从盘 101 中读出的内容数据被提供到缓冲器控制模块 215。

“解码控制模块 214”

解码控制模块 214 在播放器控制模块 212 的控制下控制视频解码器控制模块 216、音频解码器控制模块 217 和字幕解码器控制模块 218 的操作。而且，解码控制模块 214 已经内置了执行所述时间点的计时的时钟单元 214A，并且管理在视频解码器控制模块 216 的控制下输出的视频数据的输出和要与视频数据同步输出的数据(输出数据)的输出——即在这种情况下是在音频解码器控制模块 217 的控制下输出的音频数据的输出——的同步。

“缓冲器控制模块 215”

缓冲器控制模块 215 内置了缓冲器模块 215A，它是在图 1 中所示的存储器 113 的存储区域的一部分，并且在缓冲器 215A 中暂时存储由向操作系统 201 做出请求的内容数据提供模块 213 从盘 101 读出的内容数据。

而且，缓冲器控制模块 215 按照来自视频解码器控制模块 216、音频解码器控制模块 217 或者字幕解码器控制模块 218 的请求，向视频解码器控制模块 216、音频解码器控制模块 217 或者字幕解码器控制模块 218 提供在缓冲器 215A 中存储的数据。

即，缓冲器控制模块 215 内置了视频读取功能单元 233、音频读取功能单元 234 和字幕读取功能单元 235。如下参见图 3 所述。缓冲器控制模块 215 使用视频读取功能单元 233 处理来自视频解码器控制模块 216 的对于数据的请求，由此向视频解码器控制模块 216 提供在缓冲器 215A 中存储的数据。以相同的方式，缓冲器控制模块 215 使用音频读取功能单元 234 处理来自音频解码器控制模块 217 的对于数据的请求，由此向音频解码器控制模块 217 提供在缓冲器 215A 中存储的数据，并且还使用字幕读取功能单元 235 处理来自字幕解码器控制模块 218 的对于数据的请求，由此向字幕解码器控制模块 218 提供在缓冲器 215A 中存储的数据。

“视频解码器控制模块 216”

视频解码器控制模块 216 操作在缓冲器控制模块 215 内的视频读取功能单元 233(图 3)以便以视频访问单元的递增来从缓冲器控制模块 215 的缓冲器 215A 中读出其中视频数据已经被编码的数据(视频编码数据)，并且将其提供到在图 1 中所示的视频解码器 116。而且，视频解码器控制模块 216 控制视频解码器 116 而以视频访问单元的递增来解码数据。而且，视频解码器控制模块 216 向图形处理模块 219 提供作为在视频解码器 116 中的解码结果而获得的视频数据。

现在，视频访问单元例如是视频数据的一个画面(一个帧或一个场)。

“音频解码器控制模块 217”

音频解码器控制模块 217 操作在缓冲器控制模块 215 内的音频读取功能单元 234(图 3)以便以音频访问单元的递增来从缓冲器控制模块 215 的缓冲器 215A 中读出其中音频数据已经被编码的数据(音频编码数据)，并且将其提供到在图 1 中所示的音频解码器 117。而且，音频解码器控制模块 217 控制音频解码器 117 以便以音频访问单元的递增来解码所述数据。而且，音频解码器控制模块 217 向音频输出模块 221 提供作为在音频解码器 117 中的解码结果而获得的音频数据。

现在，音频访问单元是预定数据量的音频数据(例如与一个画面同步输出的)。对于本实施例，让我们说，音频访问单元例如是已知的固定长度。

“字幕解码器控制模块 218”

字幕解码器控制模块 218 操作在缓冲器控制模块 215 内的字幕读取功能单元 235(图 3)以便以字幕访问单元的递增来从缓冲器控制模块 215 的缓冲器 215A 中读出其中字幕数据已经被编码的数据(字幕编码数据)。而且，字幕解码器控制模块 218 内部地具有未示出的字幕解码软件，它解码从缓冲器 215A 读出的数据。而且，字幕解码器控制模块 218 向图形处理模块 219 提供作为在此解码的结果而获得的字幕数据(字幕图像数据)。

现在，字幕访问单元是预定数据量的字幕数据(例如与一个画面同步输出的)。对于本实施例，让我们说，字幕访问单元的大小例如在字幕访问单元的首部中描述。

“图形处理模块 219”

图形处理模块 219 在播放器控制模块 212 的控制(指令)下执行来自字幕解码器控制模块 218 的字幕数据的放大或缩小，并且将其加到(覆盖)来自视频

解码器控制模块216的视频数据。

而且，图形处理模块219将与字幕数据相加后的视频数据的大小(帧)进行放大或缩小等以便匹配连接到在图1中所示的视频输出端120的视频输出装置的显示屏幕，并且向视频输出模块220输出作为其结果而获得的视频数据。

而且,图形处理模块219按照脚本控制模块211或者播放器控制模块212的指令(控制)来产生菜单、消息等，它们被覆盖在输出视频数据上。

而且，图形处理模块219基于连接到在图1中所示的视频输出端120的视频输出装置的长宽比和用于指示在盘101中记录的视频数据的长宽比的信息等来执行要被输出到视频输出模块220的视频数据的长宽比的转换。

即，例如，在视频输出装置的长宽比是例如16:9、并且用于指示视频数据的长宽比的信息表示4:3的长宽比的情况下，图形处理模块219执行用于在横向(水平方向)上压缩(缩小)要输出到视频输出模块220的视频数据的处理，并且在左部和右部用黑色部分（pillarboxing）输出。而且，在视频输出装置的长宽比是例如4:3、并且用于指示视频数据的长宽比的信息表示16:9的长宽比的情况下,图形处理模块219执行用于在垂直方向上压缩(缩小)要输出到视频输出模块220的视频数据的处理，并且在顶部和底部用黑色部分（letterboxing）输出。

注意，在视频输出装置的长宽比和用于指示视频数据的长宽比的信息都同时是在4:3或者16:9的情况下，图形处理模块219原样输出要被输出到视频输出模块220的视频数据，而不执行压缩处理。

另外，图形处理模块219响应于例如来自播放器控制模块212的请求而捕获当前正在被处理的视频数据。而且，图形处理模块219存储所捕获的视频数据，或者提供到播放器控制模块212。

“视频输出模块220”

视频输出模块220专有地占用在图1中所示的存储器113的一部分，以便用作FIFO(先入先出)220A(缓冲器)，以便暂时存储来自图形处理模块219的视频数据，并且也适当地读出在FIFO 220A中存储的视频数据，并且输出到视频输出端120(图1)。

“音频输出模块221”

音频输出模块221专有地占用在图1中所示的存储器113的一部分，以

便使用 FIFO 221A(缓冲器)，以便暂时存储来自音频解码器控制模块 217(音频解码器 117)的音频数据，以及还适当地读出在 FIFO 221A 中存储的音频数据和输出到音频输出端 121(图 1)。

而且，在来自音频解码器控制模块 217 的音频数据是双(Dual)(双语)模式音频数据——其中左声道是“主音频”音频数据并且右声道是“付音频”音频数据——的情况下，音频输出模块 221 按照预先指定的音频输出模式来向音频输出端 121 输出来自音频解码器控制模块 217 的音频数据。

即，例如，在“主音频”被指定为音频输出模式的情况下，音频输出模块 221 在来自音频解码器控制模块 217 的音频数据中将左声道的音频数据复制作为右声道的音频数据，并且向音频输出端 121 输出左声道和右声道的音频数据(“主音频”音频数据)。而且，在“付音频”被指定为音频输出模式的情况下，音频输出模块 221 在来自音频解码器控制模块 217 的音频数据中将右声道的音频数据复制作为左声道的音频数据，并且向音频输出端 121 输出左声道和右声道的音频数据(“付音频”音频数据)。而且，在将“主/付”指定为音频输出模式的情况下，音频输出模块 221 原样地向音频输出端 121 输出来自音频解码器控制模块 217 的音频数据。

现在，在来自音频解码器控制模块 217 的音频数据是立体声（Stereo）模式音频数据的情况下，音频输出模块 221 原样地向音频输出端 121 输出来自音频解码器控制模块 217 的音频数据，而与音频输出模式的规格无关。

在此，可以由操作遥控器等的用户在屏幕等上交互地进行音频输出模式的指定，其中，例如显示由视频内容播放程序 210 产生的菜单。

[缓冲器控制模块 215 的配置]

图 3 说明了在图 2 中所示的缓冲器控制模块 215 的配置例子。

缓冲器控制模块 215 专有地使用在图 1 中所示的存储器 113 的一部分来作为缓冲器 215A，并且暂时在缓冲器 215A 中存储从盘 101 读出的数据。而且，缓冲器控制模块 215 读出在缓冲器 215A 中存储的数据，并且将其提供到视频解码器控制模块 216、音频解码器控制模块 217 或者字幕解码器控制模块 218，如图 2 中所示。

即，除了缓冲器 215A 之外，缓冲器控制模块 215 还具有作为存储器 113 一部分的数据首部指针存储单元 231 和数据写入指针存储单元 232，并且也具有作为内部模块的视频读取功能单元 233、音频读取功能单元 234 和字幕

读取功能单元 235。

缓冲器 215A 例如是环形缓冲器，其中，依序存储从盘 101 读出的数据，并且在存储等同于其存储容量的数据之后，改写最老的数据，由此可以说好像在无限循环中存储最新的数据。

数据首部指针存储单元 231 存储数据首部指针，该数据首部指针指向这个位置（地址），在这里，在缓冲器 215A 所存储的数据中，存储了还没有从缓冲器 215A 读出的最老数据。

数据写入指针存储单元 232 存储写入指针，该写入指针指向写入了从盘 101 读出的最新数据的缓冲器 215A 的位置(地址)。

现在，每次从盘 101 读出的数据被存储到缓冲器 215A 时，数据写入指针所指示的位置被更新到在图中的右部(顺时针)，并且数据首部指针所指示的位置按照从缓冲器 215A 的数据读出被更新到在图中的右部。因此，在缓冲器 215A 所存储的数据中，可以说是有效的数据是从由数据首部指针指示的位置顺时针到数据写入指针指示的位置所存储的数据。

视频读取功能单元 233 响应于来自在图 2 中所示的视频解码器控制模块 216 的请求而从缓冲器 215A 读出视频流(与视频数据相关联的基本流)，并且将其提供到视频解码器控制模块 216。音频读取功能单元 234 也响应于来自在图 2 中所示的音频解码器控制模块 217 的请求而从缓冲器 215A 读出音频流(与音频数据相关联的基本流)，并且将其提供到音频解码器控制模块 217。字幕读取功能单元235也响应于来自在图2中所示的字幕解码器控制模块218的请求而从缓冲器 215A 读出字幕流(与字幕数据相关联的基本流)，并且向字幕解码器控制模块 218 提供它们。

即，例如符合 MPEG(移动图像专家组)2 的节目（program）流(MPEG2-系统节目流)被记录在光盘 101 中，并且从光盘 101 读出的节目流被存储在缓冲器 215A 中。这个节目流具有通过时间共享而复用的一个或多个基本流，诸如视频流、音频流、字幕流等。视频读取功能单元 233 具有用于这种节目流的去复用功能，并且从在缓冲器 215A 存储的节目流中分离出视频流，并且将其读出。

以相同的方式，音频读取功能单元 234 具有用于这种节目流的去复用功能，并且从在缓冲器 215A 存储的节目流中分离出音频流，并且将其读出。字幕读取功能单元 235 也具有用于这种节目流的去复用功能，并且从在缓冲

器 215A 存储的节目流中分离字幕流，并且将其读出。

现在，视频读取功能单元 233 具有视频读取指针存储单元 241、stream_id 寄存器 242 和 au_information()寄存器 243，它们是在图 1 中所示的存储器 113 的一部分。

视频读取指针存储单元 241 存储视频读取指针，该视频读取指针指向其中已经存储视频流的缓冲器 215A 的位置(地址)，并且视频读取功能单元 233 将在缓冲器 215A 中由视频读取指针指向的位置上存储的数据读出作为视频流。stream_id 寄存器 242 分析在缓冲器 215A 中存储的节目流，并且存储用于识别(确定)从节目流读出的视频流的后述的 stream_id。au_information()寄存器 243 存储后述的 au_information()，它是用于从缓冲器 215A 读出视频流所需要的数据(用于读出视频流)。

音频读取功能单元 234 具有音频读取指针存储单元 251、stream_id 寄存器 252 和 private_stream_id 寄存器 253，它们是在图 1 中所示的存储器 113 的一部分。

音频读取指针存储单元 251 存储音频读取指针，该音频读取指针指向其中已经存储音频流的缓冲器 215A 的位置(地址)，并且音频读取功能单元 234 将由音频读取指针指向的位置中存储的数据读出作为音频流。stream_id 寄存器 252 和 private_stream_id 寄存器 253 分析在缓冲器 215A 中存储的节目流，并且存储用于识别从节目流读出的音频流的后述的 stream_id 和 private_stream_id。

字幕读取功能单元 235 具有字幕读取功能标记存储单元 261、字幕读取指针存储单元 262、stream_id 寄存器 263 和 private_stream_id 寄存器 264，它们是在图 1 中所示的存储器 113 的一部分。

字幕读取功能标记存储单元 261 存储字幕读取功能标记。在字幕读取功能标记存储单元 261 中存储的字幕读取功能标记例如是 0 的情况下，字幕读取功能单元 235 不运行，在字幕读取功能标记存储单元 261 中存储的字幕读取功能标记例如是 1 的情况下，字幕读取功能单元 235 起作用。

字幕读取指针存储单元 262 存储字幕读取指针，该字幕读取指针指向其中已经存储字幕流的缓冲器 215A 的位置(地址)，并且，字幕读取功能单元 235 将在缓冲器 215A 中由字幕读取指针指向的位置中存储的数据读出作为字幕流。stream_id 寄存器 263 和 private_stream_id 寄存器 264 分析在缓冲器 215A

中存储的节目流，并且存储用于识别从节目流读出的字幕流的后述的stream_id 和 private_stream_id。

[在盘 101 中记录的数据的数据格式的说明]

下面，将说明盘 101 中记录的数据的数据格式。

图 4 示意地说明了盘 101 的目录结构。

由例如 ISO(国际标准化组织)-9660 或者 UDF(通用盘格式：http://www.osta.org/specs/)等规定的文件系统被用作为盘 101 的文件系统，并且通过目录结构来分层地管理在盘 101 中记录的数据的文件。注意，在此使用的文件系统不限于上述的文件系统。

在图 4 中，在用于指示文件系统的原点的根（root）目录中有“VIDEO(视频)”目录，并且在“VIDEO”目录下有两个目录，它们是“CLIP(剪辑)”目录和“STREAM(流)”目录。

除了所述“CLIP(剪辑)”目录和“STREAM(流)”目录这两个目录之外，在“VIDEO(视频)”目录中，有两个数据文件，它们是“SCRIPT.DAT”文件和“PLAYLIST.DAT”文件。

“SCRIPT.DAT”文件是其中描述脚本程序的脚本文件。即，在“SCRIPT.DAT”文件中描述了用于使得盘 101 的播放形式为交互形式的脚本程序。在“SCRIPT.DAT”文件中描述的脚本程序被在图 2 中所示的脚本控制模块 211 解释和执行。

“PLAYLIST.DAT”文件存储一个或多个播放列表(后述的在图 5 中的 PlayList())，其中，描述了诸如在盘 101 中记录的视频数据等内容的播放规程。

在“CLIP”目录中，有一个或多个剪辑信息文件，并且在“STREAM”目录中是一个或多个剪辑流文件。即，在图 4 中的“CLIP”目录中，有三个剪辑信息文件，它们是“00001.CLP”、“00002.CLP”和“00003.CLP”，并且在“STREAM”目录中是三个剪辑流文件“00001.PS”、“00002.PS”和“00003.PS”。

剪辑流文件存储了将一个或多个基本流时分复用后的节目流，该一个或多个基本流是通过将诸如视频数据、音频数据、字幕数据等的一个或多个数据（流）压缩编码后获得的。

在剪辑信息文件中描述了诸如对应剪辑流文件的特性等的与剪辑流相关联的（文件）元数据。

即，剪辑流文件和剪辑信息文件以一对一方式对应。在图 4 中，按照“5 个数字字符+英文句号+ ‘PS’”的命名规则来命名剪辑流文件，按照与对应的剪辑文件相同的“5 个数字字符+英文句号+ ‘CLP’”的命名规则来命名剪辑信息文件。

因此，可以通过文件名称的后缀(在英文句号右部的部分)来识别是否一个文件是剪辑流文件或者剪辑信息文件，并且，可以通过除了文件名称的后缀之外的部分(在英文句号左侧的部分)是否匹配来识别对应的剪辑流文件或剪辑信息文件。

下面是在盘 101 中记录的文件的详细说明。

“PLAYLIST.DAT”

图 5 说明了在图 4 中所示的“VIDEO”目录下的“PLAYLIST.DAT”文件的内部结构(语法)。

这里，在图 5 中，在用于“语法”的空间中的说明表示所述数据结构，在用于“比特数量”的空间中的说明表示在对应行中用于“语法”的空间中的数据的比特长度。而且，在用于“助记符”的空间中的说明中，“bslbf” (bit string left bit first: 以左部的比特为先的比特串)表示从左部的比特开始提供在对应行中的用于“语法”的空间中的数据，“uimsbf” (unsigned integer most significant bit first: 以最高有效位为先的无符号整数)表示在对应行中的用于“语法”的空间中的数据是无符号的整数值，并且从最高阶比特开始被提供。这对于类似于图 5 的下述的其他附图保持为真。

从“PLAYLIST.DAT”文件的首部依序排列了用于描述诸如其名称(文件名)的信息的 name_length(8 比特)和 name_string(255 字节)。

即，name_length 以字节的数量表示下面的 name_string 的大小。name_string 表示“PLAYLIST.DAT”文件的名称(文件名)。

注意，对于 name_string，从其首部开始到在 name_length 中指示的字节数被用作有效名称。例如，在 name_length 的值是 10 的情况下，从 name_string 的首部开始的 10 个字节被解释为有效名称。

在 name_string 之后配置了 number_of_Playlists(16 比特)。这个 number_of_Playlists 指示随后的 PlayList()的数量。在 number_of_Playlists 之后是由 number_of_Playlists 指示的数量的 PlayList()。

PlayList()是播放列表，其中，描述了在盘 101 中记录的剪辑流文件的播

放规程，并且具有下面的内部结构。

即，PlayList_data_length(32 比特)位于 PlayList() 的首部。PlayList_data_length 表示 PlayList()的大小。

在 PlayList_data_length 之后依序是 reserved_for_word_alignment(15 个比特)和 capture_enable_flag_PlayList(1 个比特)。所述 15 比特的 reserved_for_word_alignment 被布置在随后 1 比特的 capture_enable_flag_PlayList 的位置(用于在 16 比特位置上对齐)，用于所谓的字对齐。所述 capture_enable_flag_PlayList 是 1 比特的标记，用于指示对于要播放盘 101 的图 1 中所示的盘装置，是否允许视频数据的二次使用，该视频数据是对应于使用 PlayList()而播放的视频流的视频数据(属于所述 PlayList()的视频数据)。例如，在 capture_enable_flag_PlayList 是 0 或者 1 中的例如 1 的情况下，这指示允许属于 PlayList() 的视频数据的二次使用，在 capture_enable_flag_PlayList 是 0 或者 1 中的例如 0 的情况下，这指示不允许(禁止)属于 PlayList()的视频数据的二次使用。

注意，虽然在图 5 中 capture_enable_flag_PlayList 是 1 个比特，但是可以另行布置，其中，capture_enable_flag_PlayList 被配置多个比特，并且可以说在阶段上允许属于 PlayList() 的视频数据的二次使用。即，capture_enable_flag_PlayList 可以例如被配置 2 个比特。可以进行布置使得在 capture_enable_flag_PlayList 的值是 00B(B 表示其前面的数字是二进制的)的情况下，禁止视频数据的二次使用，在 capture_enable_flag_PlayList 的值是 01B 的情况下，仅仅当被减少到 64×64 像素的大小时才允许视频数据的二次使用。而且，在 capture_enable_flag_PlayList 的值是 10B 的情况下，可以允许视频数据的二次使用，而没有大小的限制。

而且，在视频数据的二次使用的情况下，可以进行布置，其中，取代如上所述的限制大小，限制其使用。即，可以进行布置，其中，在 capture_enable_flag_PlayList 的值是 01B 的情况下，仅仅对于视频内容再现应用 210(图 2)允许视频内容的二次使用，在 capture_enable_flag_PlayList 的值是 10B 的情况下，对于在图 1 盘装置中的包括视频内容再现应用 210 的选用应用允许视频内容的二次使用。现在，在图 1 中所示的盘装置中的除了视频内容再现应用 210 之外的应用例子包括例如用于执行用来显示壁纸(背景)或者屏幕保护程序的处理的应用等。

注意，在如上所述 capture_enable_flag_PlayList 是 2 比特的情况下，位于之前的 reserved_for_word_alignment 是 14 比特，用于实现字的对齐。

而且，除了通过 capture_enable_flag_PlayList 来允许在盘装置内的视频数据的二次使用之外，也可以允许在盘装置之外的二次使用。现在，在允许盘装置之外的视频数据的二次使用的情况下，视频数据例如被记录在可装卸地安装到盘装置上的记录介质中或者可装卸地安装到可以被连接到所述盘装置的另一个装置上的记录介质中，或者经由诸如因特网的网络被发送(分发)到另一个装置。在这种情况下，在该视频数据上可以已经加上了用于限制在记录介质中记录视频数据的次数或分发的次数的信息。

在 capture_enable_flag_PlayList 之后依序是 PlayList_name_length(8 比特)和 PlayList_name_string(255 字节)。PlayList_name_length 以字节数表示随后的 PlayList_name_string 的大小，PlayList_name_string 表示 PlayList()的名称。

number_of_PlayItems(16 比特)位于 PlayList_name_string 之后。这个 number_of_PlayItems 表示随后的 PlayItem()的数量。

在 number_of_PlayItems 之后描述了与 number_of_PlayItems 的数量一样多的 PlayItem()的结构。

现在，可以以 PlayItem()的递增来在单个 PlayList()中描述内容的播放规程。

而且，number_of_PlayItems 的数量的 PlayItem()的每个被分配有在那个 PlayList()中的唯一 ID(标识)。即，在 PlayList()中的第一 PlayItem()被分配作为 ID 的 0，并且随后的 PlayItem()每个被以第一、第二、...的方式以出现的顺序分配连续的 ID。

单个 PlayListMark()位于在与 number_of_PlayItems 的数量一样多的 PlayItem()之后。PlayListMark()是一组 Mark()，其描述了用于按照 PlayList()而执行的播放的时间轴上的标记，下面参见图 7 来说明其细节。

“PlayItem()的说明”

图 6 说明了在图 5 中所示的 PlayList()中包含的 PlayItem()的内部结构。

在 PlayItem()的首部是长度(16)比特，长度表示包括它的 PlayItem()的大小。

在长度之后依序定位 Clip_Information_file_name_length(16 比特)和 Clip_Informatin_file_name(可变长度)。Clip_Information_file_name_length 以字

节数量来表示随后的 Clip_Information_file_name 的大小。Clip_Informatin_file_name 表示对应于将要由 PlayItem()播放的剪辑流文件(图 4 中后缀是 PS 的文件)的剪辑信息文件(图 4 中的后缀是 CLP 的文件)的文件名称。注意，按照剪辑流文件和剪辑信息文件的文件名的上述命名规则，可以从 Clip_Informatin_file_name 识别要由 PlayItem()播放的剪辑信息文件的文件名，由此确定剪辑流文件。

在 Clip_Informatin_file_name 之后依序是 IN_time(32 比特)和 OUT_time_(32 比特)。

IN_time 和 OUT_time 每个是时间点信息，用于指定从 Clip_Information_file_name 确定的剪辑流文件的播放开始时间和播放结束时间。

对于 IN_time，可以将在剪辑流文件中间的位置(包括首部)指定为播放开始位置，对于 OUT_time，可以将剪辑流文件中间的位置(包括结尾)指定为播放结束位置。

对于 PlayItem()，从自 Clip_Information_file_name 确定的剪辑流文件的 IN_time 到 OUT_time 播放内容，以下在适当时，将由 PlayItem()播放的内容称为剪辑。

“PlayListMark()的说明”

图 7 说明了在图 5 中所示的 PlayList()中包括的 PlayListMark()的内部结构。

如上所述，PlayListMark()是在用于按照包括那个 PlayListMark()的 PlayList()(图 5)而播放的在时间轴上用作为标记的一组 0 个或多个 Mark()。Mark()至少包括用于表示按照 PlayList()而执行的播放的时间轴上的一个时间点(位置)的时间点信息、用于表示 Mark()的类型的类型信息和在所述类型信息表示用于触发事件的类型的情况下用作为事件的变元的变元信息。

即，在 PlayListMark()的首部是长度(16)比特，长度表示包括它的 PlayListMark()的大小。

在长度之后定位了 number_of_PlayList_marks(16 比特)，number_of_PlayList_marks 表示位于其后的 Mark()的数量。在 number_of_PlayList_marks 之后描述了数量上等于 number_of_PlayList_marks 的 Mark()的结构。

在 Mark()的首部定位了 mark_type(8 比特)。这个 mark_type 是上述类型信息的类型信息，用于表示包括它的 Mark()的类型。

对于本实施例，例如提供有三种 Mark()，它们是章节(Chapter)、索引(Index)和事件(Event)。

其类型是章节的 Mark()(以下在适当时被称为章节标记)是章节的首部位置的标记，所述章节是 PlayList()被划分为的用于提示的单位。而且，其类型是索引的 Mark()(以下在适当时称为索引标记)是索引的首部位置的标记，所述索引是章节的子单位。其类型是事件的 Mark()(以下在适当时被称为事件标记)是当按照 PlayList()播放内容时要被触发的事件的位置的标记。由事件标记产生的事件被通知到脚本控制模块 211，如下所述。

图 8 表示了在 mark_type 的值和 Mark()的类型之间的关系。按照图 8，将 1 设置到章节标记的 mark_type。而且，将 2 设置到索引标记的 mark_type，并且将 3 设置到事件标记的 mark_type。注意，在图 8，在可以由 mark_type 表示的 8 比特值中，0 和 4 到 255 被保留用于未来的扩展。

返回图 7，在 mark_type 之后配置了 mark_name_length(8 比特)。而且，在 Mark()的结尾定位了 mark_name_string(24 字节)。所述 mark_name_length 和 mark_name_string 用于描述 Mark()的名称，mark_name_length 表示 mark_name_string 的有效大小，并且 mark_name_string 表示 Mark()的名称。因此，从 mark_name_string 的首部到由 mark_name_length 表示的字节的数量表示 Mark()的有效名称。

在 mark_name_length 之后依序配置了将在 PlayList()上定义的 Mark()与剪辑流文件相关联的四个元素，即，ref_to_PlayItem_id(16 比特)、mark_time_stamp(32 比特)、entry_ES_stream_id(8 比特)和 entry_ES_private_stream_id(8 比特)。

ref_to_PlayItem_id 描述 ID，其中，序号已经被分配到 Mark()所属的 PlayItem()。Mark()所属的 PlayItem()(图 6)由 ref_to_PlayItem_id 确定，因此，如参见图 6 所述，确定剪辑信息文件和剪辑流文件。

mark_time_stamp 表示在由 ref_to_PlayItem_id 确定的剪辑流文件内由 Mark()表示的位置(时间点)。

图 9 说明了在 PlayList()、PlayItem()、剪辑和在剪辑流文件内存储的节目流之间的关系。

在图 9 中，PlayList()被配置 3 个 PlayItem()，所述 3 个 PlayItem()的每个已经被分配了作为序号的 ID#0、#1 和#2。以下，适当地将已经被分配 ID#i 的 PlayItem()描述为 PlayItem#i。

在图 9 中，作为将要由 PlayItem#0、PlayItem#1、和 PlayItem#2 播放的内容的剪辑分别被说明作为剪辑 A、剪辑 B 和剪辑 C。

在从图 6 的 PlayItem()的 Clip_Information_file_name 中确定的(还从剪辑信息文件中确定的)剪辑流文件所存储的节目流中，剪辑实际上是从 IN_time 到 OUT_time 的节目流。在图 9 中，作为剪辑 A、剪辑 B 和剪辑 C 的实际实体的节目流被分别说明为节目流 A、节目流 B 和节目流 C。

例如，在图 9 中，借助作为在按照 PlayList()播放的时间轴上的位置(时间点)t0 之标记的 Mark()，其 ref_to_PlayItem_id 和 mark_time_stamp 被描述如下。

即，时间点 t0 是 PlayItem#1 的播放被执行的时间点，因此在 ref_to_PlayItem_id 中描述了 1，其是 PlayItem#1 的 ID。而且，在时间点 t0，通过 PlayItem#1 来播放作为剪辑 B 的实际实体的节目流 B，因此，在 mark_time_stamp 中描述了在其中存储了节目流 B 的剪辑流文件中等同于 t0 的时间点。

再一次返回图 7，在将 Mark()与特定的基本流相关联的情况下，使用 entry_ES_stream_id 和 entry_ES_private_stream_id 来用于确定基本流。即，在 entry_ES_stream_id 中描述了 Mark()将要相关联的基本流的后述的 stream_id(在后述的图 16 到 18 中所示的 PES_packet()，其中配置了所述基本流)。而且，在需要时，在 entry_ES_private_stream_id 中描述了 Mark()将要相关联的基本流的后述的 private_stream_id(在后述的图 21 中所示的 private_stream1_PES_payload()的中的 private_header()，其中配置了所述基本流)。

例如，在其中复用了视频流#1 和视频流#2 的剪辑中，在播放视频流#1 的情况和播放视频流#2 的情况之间将要改变章节触发时间点的情况下，当播放视频流#1 时在触发章节标记的时间点，在 Mark()的 entry_ES_stream_id 和 entry_ES_private_stream_id 中描述了视频流#1 的 stream_id 和 private_stream_id，当播放视频流#2 时在触发章节标记的时间点，在 Mark()的 entry_ES_stream_id 和 entry_ES_private_stream_id 中描述了视频流#2 的

stream_id 和 private_stream_id。

注意，在不与任何特定的基本流相关联的 Mark()的的 entry_ES_stream_id 和 entry_ES_private_stream_id 两者中例如记述了 0。

在 entry_ES_private_stream_id 之后配置了 mark_data(32 比特)。mark_data 是在 Mark()是事件标记的情况下用作为由事件标记触发的事件的变元的变元信息。注意，在 Mark()是章节标记或者索引标记的情况下，可以将 mark_data 用作所述章节标记或索引标记所表示的章节或者索引的编号。

“Clip()的说明”

接着将说明位于图 4 中的“CLIP”目录中的、后缀是 CLP 的剪辑信息文件的内部结构。

在图 4 中，有三个剪辑信息文件，“00001.CLP”、“00002.CLP”和“00003.CLP”，每个分别存储用于指示在“STREAM”目录中配置的剪辑流文件“00001.PS”、“00002.PS”和“00003.PS”的性质等的元数据。

图 10 说明了这种剪辑信息文件 Clip()的内部结构。

在剪辑信息文件 Clip()的首部依序配置了 presentation_start_time 和 presentation_end_time(两者都是 32 比特)。presentation_start_time 和 presentation_end_time 表示对应于剪辑信息文件 Clip()的剪辑流文件(中存储的节目流的)首部和结尾时间点。注意，所述剪辑流文件的时间点被描述为 90 kHz 的倍数，在 MPEG2 系统中被用作时间点。

在 presentation_end_time 之后依序是 reserved_for_word_alignment(7 比特)和 capture_enable_flag_Clip(1 比特)。所述 7 比特的 reserved_for_word_alignment 用于字对齐，capture_enable_flag_Clip 是用于表示是否允许视频数据的二次使用的标记，与在图 5 中的上述 capture_enable_flag_PlayList 一样。

但是，虽然在图 5 中的 capture_enable_flag_PlayList 表示是否允许对应于通过 PlayList()播放的视频流的视频数据(属于 PlayList()的视频数据)的二次使用，但图 10 中的 capture_enable_flag_Clip 表示是否允许对应于在剪辑流文件中存储的视频流(视频基本流)的视频数据的二次使用，所述剪辑流文件对应于剪辑信息文件 Clip()。因此，允许二次使用的视频数据的递增(范围)在图 5 的 capture_enable_flag_PlayList 和图 10 中的 capture_enable_flag_Clip 之间不同。

注意，在图 10 中的 capture_enable_flag_Clip 也可以是多个比特而不是 1

个比特，这与在图 5 中的 capture_enable_flag_PlayList 相同。

在 capture_enable_flag_Clip 之后配置了 number_of_streams(8 比特)，这个 number_of_streams 描述了跟随它的 StreamInfo()结构的数量。因此，在 number_of_streams 之后，记述了等于 number_of_streams 中数量之数量的 StreamInfo()结构。

在 StreamInfo()的首部配置了长度(16 字节)，这样长度表示包括它的 StreamInfo()的大小。在长度之后配置了 stream_id(8 比特)和 private_stream_id(8 比特)，通过这个 stream_id 和 private_stream_id 来确定(识别)要与 StreamInfo()相关联的基本流。

图 11 说明了在用于识别基本流的 stream_id 和 private_stream_id 与所述基本流之间的关系。

stream_id 与在 MPEG2 系统标准中规定的相同，在 MPEG2 系统标准中预先对于每个基本流(数据)属性(类型)确定了其值。因此，可以通过 stream_id 单独来确定在 MPEG2 系统标准中定义的属性的基本流。

对于本实施例，也可以处理在 MPEG2 系统标准中未规定的属性的基本流，private_stream_id 是用于识别在 MPEG2 系统标准中未规定的属性的基本流的信息。

图 11 说明了在下述四个属性基本流的 stream_id 和 private_stream_id 之间的关系：通过由 MPEG 规定的编码(解码)而编码的视频的基本流；通过 ATRAC(自适应变换声音编码)编码的音频的基本流(以下在适当时称为 ATRAC 音频流)；通过 LPCM(线性脉冲代码调制)而编码的音频的基本流(以下在适当时称为 LPCM 音频流)；以及字幕基本流(以下在适当时称为字幕流)。

MPEG2 系统标准通过使用 0xE0 到 0xEF 范围的值(0x 表示随后的文本串是十六进制的)来作为用于识别基本流的 stream_id，规定了通过由 MPEG 规定的编码系统所编码的视频的基本流的复用。因此，对于通过由 MPEG 规定的编码系统所编码的视频的基本流，可以在节目流上复用能够通过 0xE0 到 0xEF 范围的值的 stream_id 而识别的 16 个视频基本流。

注意，可以通过 0xE0 到 0xEF 范围的值的 stream_id 来执行通过由 MPEG 规定的编码系统所编码的视频的基本流的识别，因此 private_stream_id 是不相关的(可以被忽略)。

另一方面，对于 MPEG2 系统，stream_id 不定义 ATRAC 音频流、LPCM

音频流或者字幕流。

因此，对于本实施例，在 MPEG-2 系统中表示属性 private_stream_1 的值 0xBD 用于在 MPEG2 系统中没有定义 stream_id 的基本流的 stream_id。而且，如图 11 中所示，使用 private_stream_id 来执行识别(确定)。

即，对于 ATRAC 音频流的识别，使用在 0x00 到 0x0F 范围中的值的 private_stream_id。因此，可以在节目流上复用 16 个 ATRAC 音频流。而且，对于 LPCM 音频流的识别，使用在 0x10 到 0x1F 范围中的值的 private_stream_id。因此，可以在节目流上复用 16 个 LPCM 音频流。而且，对于字幕流的识别，使用在 0x80 到 0x9F 范围中的值的 private_stream_id。因此，可以在节目流上复用 32 个字幕流。

现在，稍后进一步说明 stream_id 和 private_stream_id。

返回图 10，在 private_stream_id 之后依序配置了 StaticInfo()和 reserved_for_word_alignment(8 比特)。在 StaticInfo()中描述了在由(在包括那个 StaticInfo()的 StreamInfo()中描述的)stream_id 和 private_stream_id 确定的基本流的播放期间不改变的信息。以下参见图 12 来说明 StaticInfo()的细节。

reserved_for_word_alignment 用于字对齐。

在 reserved_for_word_alignment 之后配置了 number_of_DynamicInfo(8 比特)，这个 number_of_DynamicInfo 表示位于其后的 pts_change_point(32 比特)和 DynamicInfo()的组的数量。

因此，在 number_of_DynamicInfo 之后是按照 number_of_DynamicInfo 数量之数量的 pts_change_point 和 DynamicInfo()结构。

pts_change_point 表示与其为一组的 DynamicInfo()变成有效时的时间点。现在，表示在基本流首部的时间点的 pts_change_point 等于在对应于其中存储了基本流的剪辑流文件的剪辑信息文件 Clip()的开始所描述的 presentation_start_time。

DynamicInfo()存储在由 stream_id 和 private_stream_id 确定的基本流的播放期间变化的信息，它可以说是动态信息。在 DynamicInfo()中所描述的信息在与其形成一组的 pts_change_point 指示的播放时间点变成有效。下面参见图 13 来说明 DynamicInfo()的细节。

EP_map()位于与由 number_of_DynamicInfo 指示一样多的 pts_change_point 和 DynamicInfo()之后。下面参见图 14 来说明 EP_map()。

“StaticInfo()的说明”

下面，将参见图 12 来详细说明在图 10 中所示的 StaticInfo()。

图 12 说明了 StaticInfo()的语法。

StaticInfo()的内容按照对应基本流的属性(类型)而不同。根据包括那个 StaticInfo()的、在图 10 所示的 StreamInfo()中包括的 stream_id 和 private_stream_id，判断对应于 StaticInfo()的基本流的属性。

在对应于 StaticInfo()的基本流是视频流的(stream==VIDEO)情况下，StaticInfo()被配置 picture_size(4 比特)、frame_rate(4 比特)、cc_flag(1 比特)和用于字对齐的 reserved_for_word_alignment。

picture_size 表示对应于视频流的视频数据(由该视频数据显示的图像)的大小。frame_rate 表示对应于视频流的视频数据的帧频率。cc_flag 表示是否视频流包括闭合的标题(Closed Caption)数据。即，例如，在视频流中包括闭合的标题的情况下，将 cc_flag 设置为 1，在视频流中不包括闭合的标题的情况下，将 cc_flag 设置为 0。

在对应于 StaticInfo()的基本流是音频流(stream==AUDIO)的情况下，StaticInfo()被配置 audio_language_code(16 比特)、channel_configuration(8 比特)、lfe_existence(1 比特)、sampling_frequency(4 比特)和用于字对齐的 reserved_for_word_alignment。

在 audio_language_code 中描述了用于表示在音频流中包含的音频数据的语言的代码。channel_configuration 表示在音频流中包含的音频数据的属性，诸如单声道(mono)/立体声(stereo)/多声道等。lfe_existence 表示是否在音频流中存在低范围增强声道。在包括它的情况下 lfe_existence 是 1，如果不包括则 lfe_existence 是 0。sampling_frequency 是用于表示在音频流中包含的音频数据的采样频率的信息。

在对应于 StaticInfo()的基本流是字幕流(stream = = SUBTITLE)的情况下，StaticInfo()被配置 subtitile_language_code(16 比特)、configurable_flag(1 比特)和用于字对齐的 reserved_for_word_alignment。

在 subtitile_language_code 中描述了表示在字幕流中包含的字幕数据的语言的代码。configurable_flag 是用于表示是否允许从默认显示格式改变在字幕流中包括的字幕数据的显示的信息，例如，在允许改变所述显示格式的情况下，描述 1，如果不允许，则描述 0。现在，字幕数据显示格式包括字幕数据

的显示大小、显示位置、显示颜色、显示模式(例如闪烁显示等)、显示方向(例如垂直方向或者水平方向)等。

“DynamicInfo()的说明”

接着，将参见图 13 来详细说明在图 10 中所示的 DynamicInfo()。

图 13 说明了 DynamicInfo()的语法。

在 DynamicInfo()的首部是用于字对齐的 reserved_for_word_alignment(8 比特)，并且随后的分量按照对应于 DynamicInfo()的基本流的属性而不同。通过包括所述 DynamicInfo()的、在图 10 中所示的 StreamInfo()中包括的 stream_id 和 private_stream_id 来判断对应于 DynamicInfo()的基本流的属性，这与对于图 12 所述的 StaticInfo()的情况一样。

如在图 10 中所述，在 DynamicInfo()中描述了当播放基本流时改变的动态信息。这个动态信息不被具体限制，但是对于在图 13 中所示的实施例，在 DynamicInfo()中描述了对应于与 DynamicInfo()对应的基本流的数据，即由被处理的基本流输出的数据的输出属性(从基本流获得的数据的输出属性)。

具体地，在对应于 DynamicInfo()的基本流是视频流(stream==VIDEO)的情况下，DynamicInfo()被配置 display_aspect_ratio(4 个比特)和用于字对齐的 reserved_for_word_alignment。在 display_aspect_ratio 中描述了视频数据的长宽比，例如其用作为对应于视频流的视频数据的输出属性(显示格式)。即，在 display_aspect_ratio 中描述了用于指示是否长宽比是例如 16:9 或者 4:3 的信息。注意，可以使用视频流的 DynamicInfo()来描述除了长宽比之外的内容，诸如由视频数据显示的图像的大小(X 像素×Y 像素)。

在对应于 DynamicInfo()的基本流是音频流(stream==AUDIO)的情况下，DynamicInfo()被配置 channel_assignment(4 个比特)和用于字对齐的 reserved_for_word_alignment。在音频流中包括音频数据的两个声道的情况下，在 channel_assignment 中描述了两个声道的输出属性(输出格式)。即，在 channel_assignment 中描述了用于表示是否音频数据被分配到立体声或双(双语)声道的信息。

在对应于 DynamicInfo()的基本流是字幕流(stream==SUBTITLE)的情况下，DynamicInfo()被配置用于字对齐的 reserved_for_word_alignment。即，对于在图 13 中所示的实施例，没有输出属性被定义为关于字幕流的动态信息。

“EP_map()的说明”

接着，将参见图 14 来详细说明在图 10 中所示的 EP_map()。

图 14 说明了 EP_map()的语法。

对于在对应于包括那个 EP_map()的、在图 10 中所示的剪辑信息文件 Clip()的剪辑流文件中存储的节目流上所复用的基本流的每一个，在 EP_map()中描述了关于解码开始可允许点(进入点)的信息，在这个点，可以开始每个基本流的解码。

现在，对于固定速率流，可以通过计算来获得该解码开始可允许点，但是对于其中其大小对于每个视频访问访问单元不同的流，诸如可变速率流或者按照 MPEG 标准而编码的视频流，不能通过计算来获得该解码开始可允许点，并且需要实际分析所述流以找到这个点。快速识别该解码开始可允许点对于随机访问是重要的，并且对于 EP_map()，可以快速地识别该解码开始可允许点。

注意，对于 MPEG2 视频，包括 Seqeunce_header()(序列报头)等的内画面的首部是该解码开始可允许点。

在 EP_map()的首部是用于字对齐的 reserved_for_word_alignment(8 比特)和随后的 number_of_stream_id_entries(8 比特)。这个 number_of_stream_id_entries 表示基本流的数量，关于所述基本流，该解码开始可允许点信息被描述在 EP_map()中。

在 number_of_stream_id_entries 之后是用于识别基本流的信息和该基本流的解码开始可允许点的信息的重复，重复次数是按照 number_of_stream_id_entries 表示的数量的数量。

即，紧接在 number_of_stream_id_entries 之后配置了用作为用于识别基本流的信息的 stream_id(8 比特)和 private_stream_id(8 比特)，随后配置了 number_of_EP_entries(32 比特)。这个 number_of_EP_entries 表示通过紧前的 stream_id 和 private_stream_id 识别(确定)的基本流的解码开始允许点的数量。

在 number_of_EP_entries 之后是用作为通过紧前的 stream_id 和 private_stream_id 确定的基本流的解码开始可允许点的 PTS_EP_start(32 比特)和 RPN_EP_start(32 比特)的多个组，它们被重复按照由 number_of_EP_entries 表示的数量的次数。

作为一个解码开始可允许点信息的 PTS_EP_start 表示在剪辑流文件内的解码开始可允许点的时间点(播放时间点)，在所述剪辑流文件中存储了节目

流，在所述节目流，已经复用了通过如上所述的 stream_id 和 private_stream_id 确定的基本流。

作为另一个解码开始可允许点信息的 RPN_EP_start 已经在其中以节目流的 pack()的递增而描述了在剪辑流文件内的解码开始可允许点的位置，在所述剪辑流文件中，存储了如上所述由 stream_id 和 private_stream_id 确定的基本流已经被复用的节目流。注意，对于本实施例，让我们说，pack()的大小被固定在 2048 字节。而且，对于本实施例，让我们说，盘 101 的一个扇区是 2048 字节。

现在，在视频流中，在紧接其解码开始可允许点(进入点)之前布置了后述的 private_stream_2 分组(private_stream_2 属性的 PES_packet())。private_stream_2 分组存储了用于解码位于那个 private_stream_2 分组和下一个 private_stream_2 分组之间的视频流的信息。因此，对于视频流，在关于该解码开始可允许点信息的 RPN_RP_start 中描述的不是其实际的解码开始可允许点本身，而是位于紧接该实际解码开始可允许点之前的 private_stream_2 分组的首部位置。

而且，对于 EP_map()，对于由 stream_id 和 private_stream_id 确定的每个基本流，以升序来预先分类用作为解码开始可允许点信息的多组 PTS_EP_start 和 RPN_EP_start。因此，可以对于用作为解码开始可允许点信息的所述多组 PTS_EP_start 和 RPN_EP_start 进行二进制搜索。

注意，在例如日本未审查专利申请公开第 2000-341640(日本专利申请第 11-317738 号)中描述了用于可变速率流或其中每个视频访问访问单元的大小不同的流的随机访问方法。

“剪辑流文件的说明”

接着，将说明在图 4 中的“STREAM”目录中其后缀是 PS 的剪辑流文件(在图 4 中的“00001.PS”、“00002.PS”和“00003.PS”)的内部结构。

基于在 MPEG-2 系统(IOC/IEC 13818-1)中定义的 MPEG2_Program_Stream 来配置剪辑流文件。

即，图 15 说明了在 MPEG－2 系统(IOC/IEC 13818-1：2000)标准中所描述的表 2-31、表 2-32 和表 2-33。

在剪辑流文件中存储的节目流是在 MPEG-2 系统标准的表 2-31 中定义的 MPEG2_Program_Stream()，并且被配置一个或多个 pack()和一个

MPEG_program_end_code。注意，在日本专利第 2785220 等中给出了 MPEG2_Program_Stream(0 的说明。

如在 MPEG-2 系统标准的表 2-32 中所定义的那样，一个 pack()被配置一个 Pack_header()和可选数量的 PES_packet()。在 MPEG-2 系统标准的表 2-33 中定义了 Pack_header()的详细说明。

现在，虽然 MPEG-2 系统将 pack()的大小定义为可变，但是在此，这将理解为被固定在 2048 字节，如在图 14 中所述。而且，在此，单个 pack()的 PES_packet()的数量将是 1、2 或 3。在 Pack()以后述的 private_stream_2 分组开始的情况下，总是有(在同一 Pack()中的)紧接其后的对应视频流的 PES_packet()。另外，可以将 padding_packet()(填充（padding）分组)布置为第三 PES_packet()。注意，private_stream_2 分组总是在 Pack()的首部。

在 Pack()不以 private_stream_2 分组开始的情况下，将用于存储诸如视频、音频、字幕等内容数据的 PES_packet()布置在 Pack()的首部。另外，可以将 padding_packet()(填充分组)布置为第二 PES_packet()。

图 16 至图 18 说明了在 MPEG-2 系统标准的表 2-17 中定义的 PES_packet()。

PES_packet()一般可以被划分为：packet_start_code_prefix、stream_id 和 PES_packet_length(图 16);、具有按照 stream_id 等而改变结构的报头部分(包括 stuffing_byte)(图 16 至图 18)、以及 PES_packet_data_byte(图 18)。注意，在 PES_packet()是 padding_packet (stream_id==padding_stream)的情况下，将 padding_byte(0xFF)(图 18)而不是 PES_packet_data_byte 重复必要的次数。

如在图 16 和图 17 中所示，可以将用于指示被称为 PTS(显示时间标记)的显示定时的信息和用于指示被称为 DTS(解码时间标记)的解码定时的信息布置在 PES_packet()的报头部分中。对于本实施例，让我们说，将 PTS 加到所有的访问单元(在 MPEG2 系统中定义，构成基本流的解码单元)，并且在设置到 MPEG2 系统的情况下加上 DTS。

在 PES_packet()的 PES_packet_data_byte(图 18)中存储了在节目流上要被复用的基本流。在 PES_packet()的 stream_id 中描述了对应于基本流的属性的值，以便识别在 PES_packet_data_byte 中存储的基本流。

在 MPEG-2 系统标准的表 2-18 中定义了在 PES_packet()的 stream_id 中描述的值和基本流的属性(类型)之间的关系。在图 19 中示出了 MPEG-2 系统

标准的表 2-18。

对于本实施例，在图 19 所示的 MPEG-2 系统标准所定义的 stream_id 中，使用例如在图 20 中所示的值。

即，对于本实施例，将 5 个模式 10111101B、10111110B、10111111B、110xxxxxB 和 1110xxxxB 用作 stream_id 的值。注意，“x” 表示 0 或者 1 的任意值。

具有被称为 private_stream_1 的属性的基本流的 PES_packet()的 stream_id 按照图 20 为 10111101B。而且，padding_packet 的 PES_packet()的 stream_id 按照图 20 是 10111110B。而且，具有被称为 private_stream_2 的属性的基本流的 PES_packet()的 stream_id 按照图 20 为 10111111B。

而且，通过 MPEG 定义的音频流(音频基本流)的 PES_packet()的 stream_id 为 110xxxxxB。注意，110xxxxxB 的 5 个低位比特 xxxxx 用作为用于区别音频流的音频流编号，因此可以在节目流(MPEG2_Program_Stream())上复用 32(= 2^5)个音频流(由 MPEG 定义的音频流)，所述数量是可以由这个音频流编号区别的数量。

而且，通过 MPEG 定义的视频流(视频基本流)的 PES_packet()的 stream_id 为 1110xxxxB。注意，1110xxxxB 的 4 个低位比特 xxxx 用作为用于区别视频流的视频流编号，因此可以在节目流上复用 16(= 2^4)个视频流(由 MPEG 定义的视频流)，所述数量是可以由这个视频流编号区别的数量。

现在，使用其 stream_id 是 1110xxxxB 的 PES_packet()来存储由 MPEG 定义的视频流，并且使用其 stream_id 是 110xxxxxB 的 PES_packet()来存储由 MPEG 定义的音频流。另一方面，用于存储按照由 MPEG 未定义的编码格式(例如 ATRAC 格式)的基本流的 PES_packet()的 stream_id 不通过 MPEG 规定，因此，按照由 MPEG 未定义的编码格式的基本流不能够简单地通过指定 stream_id 而存储在 PES_packet()中，这与由 MPEG 定义的视频流和音频流的情况相同。

因此，对于本实施例，扩展了用于 private_stream_1 的 PES_packet()的 PES_packet_data_byte，以存储按照 MPEG 未定义的编码格式的基本流。

现在，将用于 private_stream_1 的 PES_packet()的被扩展的 PES_packet_data_byte 描述为 private_stream1_PES_payload()。

“private_stream1_PES_payload()的说明”

图 21 说明了 private_stream1_PES_payload()的语法。

private_stream1_PES_payload() 被 配 置 private_header() 和 private_payload()。在 private_payload()中存储了按照 MPEG 未定义的编码格式的基本流，诸如 ATRAC 音频流、LPCM 音频流、字幕流等。

private_stream_id(8 比特)位于 private_header()的首部。private_stream_id 是用于识别在 private_payload()中存储的基本流的标识信息，并且按照其属性(类型)而例如具有下面的值。

即，图 22 说明了在 private_stream_id 的值和 private_payload()中存储的基本流的属性之间的关系。

在图 22 中，三个模式 0000xxxxB、0001xxxxB 和 100xxxxxB 被用作 private_stream_id 的值。注意，“x”表示 0 或者 1 的任意值，这与在图 20 中的情况相同。

按照图 22，其 private_payload()存储了 ATRAC 音频流的 private_stream1_PES_payload()的 private_stream_id 是 0000xxxxB。注意，0000xxxxB 的 4 个低位比特 xxxx 用作为用于区别 ATRAC 音频流的音频流编号，因此可以在节目流(MPEG2_Program_Stream())上复用 16(= 2^4)个 ATRAC 音频流，所述数量是可以由这个音频流编号区别的数量。

而且，按照图 22，其 private_payload()存储了 LPCM 音频流的 private_stream1_PES_payload()的 private_stream_id 是 0001xxxxB。注意，0001xxxxB 的 4 个低位比特 xxxx 用作为用于区别 LPCM 音频流的音频流编号，因此可以在节目流上复用 16(= 2^4)个 LPCM 音频流，所述数量是可以由这个音频流编号区别的数量。

而且，按照图 22，其 private_payload()存储了字幕流的 private_stream1_PES_payload()的 private_stream_id 是 100xxxxxB。注意，100xxxxxB 的 5 个低位比特 xxxxx 用作为用于区别字幕流的字幕流编号，因此可以在节目流上复用 32(= 2^5)个字幕流，所述数量是可以由这个字幕流编号区别的数量。

现在，在图 20 和图 22 之间的关系的总结是上述的图 11。

返回图 21，在 private_stream1_PES_payload()中的 private_stream_id 之后的分量按照在 private_payload()中存储的基本流的属性而在内容上不同。通过在 private_header()首部的 private_stream_id 来判断在 private_payload()中存储

的基本流的属性。

在要被存储在 private_payload()中的基本流是 ATRAC 音频流(private_stream_id==ATRAC)的情况下，定位用于未来扩展的 reserved_for_future_use(8 比特)，其后定位 AU_locator(16 比特)。AU_locator 表示在 private_payload()中存储的 ATRAC 音频流的音频访问单元(ATRAC 音频访问单元)(音频帧)的首部位置，以紧接 AU_locator 之后的位置作为基准。在 private_payload()中不存在音频访问单元的情况下，在 AU_locator 中描述了例如 0xFFFF。

在要被存储在 private_payload()中的基本流是 LPCM 音频流(private_stream_id==LPCM)的情况下，依序排列 fs_flag(1 比特)、reserved_for_future_use(3 比特)、ch_flag(4 比特)和 AU_locator(16 比特)。

fs_flag 表示在 private_payload()中存储的 LPCM 音频流的采样频率。即，在采样频率是例如 48 KHz 的情况下，将 fs_flag 设置为 0，并且在采样频率是 44.1 KHz 的情况下，将 fs_flag 设置为 1。

ch_flag 表示在 private_payload()中存储的 LPCM 音频流的声道的数量。例如，在 LPCM 音频流是单声道的情况下，将 ch_flag 设置为 1，并且在 LPCM 音频流是立体声的情况下，将 ch_flag 设置为 2。

AU_locator 表示在 private_payload()中存储的 LPCM 音频流的音频访问单元(LPCM 音频访问单元)(音频帧)的首部位置，以紧接 AU_locator 之后的位置作为基准。在 private_payload()中不存在音频访问单元的情况下，在 AU_locator 中描述例如 0xFFFF。

在要被存储在 private_payload()中的基本流是字幕流(private_stream_id==SUBTITLE)的情况下，配置用于未来扩展的 reserved_for_future_use(8 个比特)和随后配置 AU_locator(16 比特)。AU_locator 表示在 private_payload()中存储的字幕流的字幕访问单元的首部位置，以紧接 AU_locator 之后的位置作为基准。在 private_payload()中不存在字幕访问单元的情况下，在 AU_locator 中描述例如 0xFFFF。

“private_stream2_PES_payload()的说明”

图 23 说明 private_stream2_PES_payload()的语法。

private_stream2_PES_payload()是用于 private_stream_2(图 18)的 PES_packet()的 PES_packet_data_byte 的扩展，即是用于 private_stream_2 的

PES_packet()的扩展的 PES_packet_data_byte，并且描述了用于解码视频流的信息。

对于本实施例，private_stream_2 的 PES_packet()位于紧接视频流的解码开始可允许点之前。因此，对于本实施例，一旦从节目流找到了 private_stream_2 的 PES_packet()，就可以从紧接其后的视频流开始解码。

在此，对于视频流，在图 14 中上述的 EP_map()的 RPN_EP_start 说明了 private_stream_2 的 PES_packet()的首部位置。

在 private_stream2_PES_payload() 的首部是用于未来扩展的 reserved_for_future_use(8 个比特)，随后依序是 video_stream_id(8 个比特)、1stRef_picture(16 比特)、2ndRef_picture(16 比特)、3rdRef_picture(16 比特)、4thRef_picture(16 比特)、au_information()和 VBI()。

在 video_stream_id 中描述了位于紧接 private_stream_2 的 PES_packet()之后的视频流的 PES_packet()的 stream_id(与其相同的值)。这个 video_stream_id 用于确定使用在 private_stream_2 的 PES_packet()(的 private_stream2_PES_payload())中存储的信息所解码的视频流(存储这个视频流的 PES_packet())。

1stRef_picture、2ndRef_picture、3rdRef_picture、4thRef_picture 是在由 video_stream_id 确定的视频流的 private_stream_2 的 PES_packet()和下一个 private_stream_2 的 PES_packet()之间的包括第一、第二、第三和第四基准图像的最后的 pack()的位置的相对值，并且分别表示每一个。注意，1stRef_picture、2ndRef_picture、3rdRef_picture、4thRef_picture 的细节在例如日本未审查专利申请公开第 09-46712(日本专利申请第 07-211420 号)中被公开作为 byte_to_first_P_pic 和 byte_to_second_P_pic。

在 au_information 中描述了与从 private_stream_2 的 PES_packet()到下一个 private_stream_2 的 PES_packet()的视频流中的视频访问单元相关联的信息。下面参见图 24 来说明 au_information()的细节。

VBI()用于描述闭合标题信息。

具有这样的 private_stream2_PES_payload() 的 private_stream_2 的 PES_packet()被定位在每个视频流的每个解码开始可允许点。

图 24 说明了在图 23 中所示的 au_information()的语法。

在 au_information()的首部是长度(16 比特)。这个长度表示包括其本身的

au_information()的大小。在长度之后依序配置 reserved_for_word_alignment(8个比特)和 number_of_access_unit(8 个比特)。reserved_for_word_alignment 用于字对齐。

number_of_access_unit 表示在包括其本身的 private_stream_2 的 PES_packet 到下一个 private_stream_2 的 PES_packet 之间包含的视频访问单元(画面)的数量。

即，number_of_access_unit 表示在其中在图 23 中的 private_stream2_PES_payload()具有相同 video_stream_id 的 private_stream_2 的 PES_packet 内，从这个 au_information()到紧接下一个 au_information()之前(在这个 au_information()是剪辑流文件中的最后的 au_information()的情况下，到所述剪辑流文件的结尾)在由 video_stream_id 指示的视频流中包含的访问单元(画面)的数量。

在 number_of_access_unit 之后，将 for 循环的内容配置与 number_of_access_unit 的数量一样多的次数。即，与在包括 number_of_access_unit 的 private_stream_2 的 PES_packet 到下一个 private_stream_2 的 PES_packet 之间包括的一个或多个视频访问单元的每个相关联地布置信息。

被布置在 for 循环内的信息(与视频访问单元相关联的信息)如下。

即，在 for 循环中布置了 pic_struct_copy(4 比特)、au_ref_flag(1 比特)、AU_length(21 比特)和保留。

在视频流是 MPEG4-AVC(ISO/IEC 14496-10)的情况下，在 pic_struct_copy 中描述了在 ISO/IEC 14496-10, D.2.2 中定义的 pic_struct()的拷贝，它是相对于对应的视频访问单元而被设置的。注意，pic_struct()例如是诸如将画面显示为帧或者显示画面的顶场以及然后显示其底场等等的信息。

au_ref_flag 表示是否对应的访问单元是被引用来解码另一个访问单元(画面)的基准图像，其中，在基准图像的情况下，它被设置到 1，如果不是基准图像，则为 0。

AU_length 以字节的递增来表示对应的访问单元的大小。

[在盘 101 中记录的数据的具体示例]

接着，图 25 到图 28 说明了在图 1 中所示的盘 101 中记录的上述格式的数据的具体示例。

现在，对于图 25 到图 28，使用 MPEG2-Video 来作为视频流，并且使用 ATRAC 音频流来作为音频流。但是，注意，视频流和音频流不限于这些。即，对于视频流，可以使用例如 MPEG4-visual、MPEG4-AVC 等。而且，对于音频流，可以使用例如 MPEG1/2/4 音频、LPCM 音频流等来作为音频流。

注意，字幕流不像视频流或音频流在于，可能不在同一间隔中发生连续的解码/显示(输出)。即，时时地从图 2 中所示的缓冲器控制模块 215 向字幕解码器控制模块 218 提供字幕流，并且解码字幕流。

图 25 到图 28 说明了在下述情况下的“PLAYLIST.DAT”文件、剪辑信息文件“00001.CLP”、“00002.CLP”和“00003.CLP”等的具体例子：其中，三个剪辑信息文件“00001.CLP”、“00002.CLP”和“00003.CLP”被记录在图 4 中所示的盘 101 中的“CLIP”目录中，并且在其“STREAM”目录中记录了对应于所述三个剪辑信息文件“00001.CLP”、“00002.CLP”和“00003.CLP”的三个剪辑流文件“00001.PS”、“00002.PS”和“00003.PS”。但是，注意，已经从图 25 到图 28 省略了数据的一部分，诸如“PLAYLIST.DAT”文件。

即，图 25 说明了对于图 5 所述的“PLAYLIST.DAT”文件的具体例子。

在图 25 中，number_of_PlayLists 是 2，因此，在“PLAYLIST.DAT”文件包括的 PlayList()的数量是 2。在图 25 中，所述两个 PlayList()的第一个被列出为 PlayList#0，并且第二个被列出为 PlayList#1。

对于作为第一 PlayList()的 PlayList#0，capture_enable_flag_PlayList 是 1，因此，允许按照 PlayList#0 而播放的视频数据的二次使用。而且，对于 PlayList#0，number_of_PlayItems 是 2，因此，在 PlayList#0 中包括的 PlayItem()的数量是 2。在图 25 中，在“PlayList#0”空间下列出了作为两个 PlayItem()的 PlayItem #0 和 PlayItem #1 的具体例子。

对于作为在 PlayList#0 中包含的第一 PlayItem()的 PlayItem #0，参见图 6 所述的 Clip_Information_file_name 被设置为“00001.CLP”，IN_time 被设置为 180,090，并且 OUT_time 被设置为 27,180,090。因此，由 PlayList#0 的 PlayItem #0 播放的剪辑是对应于从时间点 180,090 到 27,180,090 的剪辑信息文件“00001.CLP”的剪辑流文件“00001.PS”的部分。

对于作为在 PlayList#0 中包含的第二 PlayItem()的 PlayItem #1，参见图 6 所述的 Clip_Information_file_name 被设置为“00002.CLP”，IN_time 被设置为

90,000，并且 OUT_time 被设置为 27,090,000。因此，由 PlayList#0 的 PlayItem #1 播放的剪辑是对应于从时间点 90,000 到 27,090,000 的剪辑信息文件“00002.CLP”的剪辑流文件“00002.PS”的部分。

另一方面，在图 25 中，对于作为第二 PlayList()的 PlayList#1，capture_enable_flag_PlayList 是 0，因此，不允许(禁止)按照 PlayList#1 而播放的视频数据的二次使用。而且，对于 PlayList#1，number_of_PlayItems 是 1，因此，在 PlayList#1 中包括的 PlayItem()的数量是 1。在图 25 中，在“PlayList#1”空间下列出了作为所述那个 PlayItem()的 PlayItem #0 的具体例子。

对于作为在 PlayList#1 中包含的那个 PlayItem()的 PlayItem #0，参见图 6 所述的 Clip_Information_file_name 被设置为“00003.CLP”，IN_time 被设置为 90,000，并且 OUT_time 被设置为 81,090,000。因此，由 PlayList#1 的 PlayItem #0 播放的剪辑是对应于从时间点 90,000 到 81,090,000 的剪辑信息文件“00003.CLP”的剪辑流文件“00003.PS”的部分。

接着，图 26 说明了参见图 10 所述的剪辑信息文件 Clip()的具体例子。即，图 26 说明了在图 4 中的剪辑信息文件“00001.CLP”、“00002.CLP”和“00003.CLP”的具体例子。

对于剪辑信息文件“00001.CLP”，presentation_start_time 被设置为 90,000，并且 presentation_end_time 被设置为 27,990,000。因此，对于在对应于剪辑信息文件“00001.CLP”的剪辑流文件“00001.PS”中存储的节目流，可以使用 310 秒((27,990,000-90,000)/90 kHz)的内容。

而且，对于剪辑信息文件“00001.CLP”，capture_enable_flag_Clip 是 1，因此，允许对应于剪辑信息文件“00001.CLP”的剪辑流文件“00001.PS”中存储的节目流上复用的视频流(所对应的视频数据)的二次使用。

而且，在图 26 中，剪辑信息文件“00001.CLP”的 number_of_streams 是 4，因此，存在在剪辑流文件“00001.PS”中存储的对应的节目流上复用的 4 个基本流。

现在，参见作为 stream#0、stream#1、stream#2、stream#3（流#0、流#1、流#2、流#3）的四个基本流，图 26 示出了在“00001.CLP”空间下的四个基本流 stream#0、stream#1、stream#2、stream#3 之每一个的每个 StreamInfo()(图 10)的具体例子。

对于剪辑流文件“00001.PS”的第一基本流 stream#0，stream_id 是 0xE0，

因此，这个基本流是视频流，如参见图 20 和图 22(或者图 11)所述的。注意，对于本实施例，视频流与 private_stream_id 不相关联，如上述，并且在图 26 中，这是 0x00。

而且，对于作为剪辑流文件“00001.PS”的第一基本流的视频流 stream#0，在其 StreamInfo()中包括的 StaticInfo()(图 12)中，picture_size 是“720×480”，frame_rate 是“29.97 Hz”，并且 cc_Flag 是“是”。因此，这个视频流 stream#0 是 720×480 像素和 29.97 Hz 帧循环（frame cycle）的视频数据，并且还包括闭合标题数据。

而且，对于作为剪辑流文件“00001.PS”的第一基本流的视频流 stream#0，在 StreamInfo()中的 number_of_DynamicInfo(图 10)是 0，因此，不存在 pts_change_point 和 DynamicInfo()的组。

接着，对于剪辑流文件“00001.PS”的第二基本流 stream#1，stream_id 是 0xBD，private_stream_id 是 0x00。因此，这个基本流 stream#1 是 ATRAC 音频流，如参见图 20 和图 22 所述。

而且，对于作为剪辑流文件“00001.PS”的第二基本流的 ATRAC 音频流 stream#1，在其 StreamInfo()中包括的 StaticInfo()(图 12)中，audio_language_code 是“日语”，channel_configuration 是“STEREO(立体声)”，lfe_existence 是“NO(否)”，并且 sampling_frequency 是“48kHz”。因此，这个 ATRAC 音频流 stream#1 是日语的，并且是立体声音频数据，而且，不存在低范围的增强声道，并且采样频率是 48 kHz。

而且，对于作为剪辑流文件“00001.PS”的第二基本流的 ATRAC 音频流 stream#1，在 StreamInfo()中的 number_of_DynamicInfo(图 10)是 0，因此，不存在 pts_change_point 和 DynamicInfo()的组。

接着，对于剪辑流文件“00001.PS”的第三基本流 stream#2，stream_id 是 0xBD，private_stream_id 是 0x80。因此，这个基本流 stream#2 是字幕流，如参见图 20 和图 22 所述。

而且，对于作为剪辑流文件“00001.PS”的第三基本流的字幕流 stream#2，在其 StreamInfo()中包括的 StaticInfo()(图 12)中，subtitle_langugae_code 是“日语”，并且 configurable_flag 是 0。因此，这个字幕流 stream#2 是日语字幕数据，并且不允许(禁止)改变其显示格式。

而且，对于作为剪辑流文件“00001.PS”的第三基本流的字幕流 stream#2，

在 StreamInfo()中的 number_of_DynamicInfo(图 10)是 0，因此，不存在 pts_change_point 和 DynamicInfo()的组。

接着，对于剪辑流文件“00001.PS”的第四基本流 stream#3，stream_id 是 0xBD，private_stream_id 是 0x81。因此，这个基本流 stream#3 是字幕流，如参见图 20 和图 22 所述。

注意，为了区别作为剪辑流文件“00001.PS”的第三基本流的字幕流 stream#2 和作为剪辑流文件“00001.PS”的第四基本流的字幕流 stream#3，其 private_stream_id 分别是 0x80 和 0x81。

而且，对于作为剪辑流文件“00001.PS”的第四基本流的字幕流 stream#2，在其 StreamInfo()中包括的 StaticInfo()(图 12)中，subtitle_langugae_code 是“日语”，并且 configurable_flag 是 1。因此，这个字幕流 stream#3 是日语字幕数据，并且允许改变其显示格式。

而且，对于作为剪辑流文件“00001.PS”的第四基本流的字幕流 stream#3，在 StreamInfo()中的 number_of_DynamicInfo(图 10)是 0，因此，不存在 pts_change_point 和 DynamicInfo()的组。

接着，在图 26 中，对于剪辑信息文件“00002.CLP”，presentation_start_time 被设置为 90,000，并且 presentation_end_time 被设置为 27,090,000。因此，对于在对应于剪辑信息文件“00002.CLP”的剪辑流文件“00002.PS”中存储的节目流，可以使用 300 秒((27,090,000-90,000)/90 kHz)的内容。

而且，对于剪辑信息文件“00002.CLP”，capture_enable_flag_Clip 是 0，因此，不允许(禁止)在对应于剪辑信息文件“00002.CLP”的剪辑流文件“00002.PS”中存储的节目流上复用的视频流(所对应的视频数据)的二次使用。

而且，在图 26 中，剪辑信息文件“00002.CLP”的 number_of_streams 是 4，因此，以与上述剪辑流文件“00001.PS”的情况相同的方式，存在在剪辑流文件“00002.PS”中存储的对应节目流上复用的 4 个基本流。

现在，参见作为 stream#0、stream#1、stream#2、stream#3 的四个基本流，图 26 示出了在“00002.CLP”空间下的四个基本流 stream#0、stream#1、stream#2、stream#3 之每一个的每个 StreamInfo()(图 10)的具体例子。

现在，在图 26 中，第一到第四基本流 stream#0 到 stream#3 的 StreamInfo()的内容与上述剪辑流文件“00001.PS”的第一到第四基本流 stream#0 到

stream#3 的 StreamInfo()的内容相同，因此，将省略其说明。

注意，如上所述，剪辑流文件“00002.PS”的第一到第四基本流 stream#0 到 stream#3 的 StreamInfo()的内容与上述剪辑流文件“00001.PS”的第一到第四基本流 stream#0 到 stream#3 的 StreamInfo()的内容相同，因此，剪辑流文件“00002.PS”的第一基本流 stream#0 是视频流，第二基本流 stream#1 是 ATRAC 音频流。而且，第三基本流 stream#2 和第四基本流 stream#3 都是字幕流。

接着，在图 26 中，对于剪辑信息文件“00003.CLP”，presentation_start_time 被设置为 90,000，并且 presentation_end_time 被设置为 81,090,000。因此，对于在对应于剪辑信息文件“00003.CLP”的剪辑流文件“00003.PS”中存储的节目流，可以使用 900 秒((81,090,000-90,000)/90 kHz)的内容。

而且，对于剪辑信息文件“00003.CLP”，capture_enable_flag_Clip 是 1，因此，允许在对应于剪辑信息文件“00003.CLP”的剪辑流文件“00003.PS”中存储的节目流上复用的视频流的二次使用。

而且，在图 26 中，剪辑信息文件“00003.CLP”的 number_of_streams 是 3，因此，存在在剪辑流文件“00003.PS”中存储的对应的节目流上复用的 3 个基本流。

现在，参见作为 stream#0、stream#1、stream#2 的 3 个基本流，图 26 示出了在“00003.CLP”空间下的 3 个基本流 stream#0、stream#1、stream#2 之每一个的每个 StreamInfo()(图 10)的具体例子。

对于剪辑流文件“00003.PS”的第一基本流 stream#0，stream_id 是 0xE0，因此，这个基本流 stream#0 是视频流，如参见图 20 和图 22(或者图 11)所述。注意，与剪辑流文件“00001.PS”的第一基本流 stream#0 相同，private_stream_id 是 0x00。

而且，对于作为剪辑流文件“00003.PS”的第一基本流的视频流 stream#0，在其 StreamInfo()中包括的 StaticInfo()(图 12)中，picture_size 是“720×480”，frame_rate 是“29.97 Hz”，并且 cc_Flag 是“否”。因此，这个视频流 stream#0 是 720×480 像素和 29.97 Hz 帧循环的视频数据，并且不包括闭合标题数据。

而且，对于作为剪辑流文件“00003.PS”的第一基本流的视频流 stream#0，在 StreamInfo()中的 number_of_DynamicInfo(图 10)是 2，因此，存在在 StreamInfo()中描述的两组的 pts_change_point 和 DynamicInfo()。

接着，对于剪辑流文件“00003.PS”的第二基本流 stream#1，stream_id 是 0xE1。因此，这个基本流 stream#1 是视频流，如参见图 20 和图 22(或图 11)所述。注意，为了区别作为剪辑流文件“00003.PS”的第一基本流的视频流 stream#0 和作为剪辑流文件“00003.PS”的第二基本流的视频流 stream#1，其 stream_id 分别是 0xE0 和 0xE1。而且，与剪辑流文件“00001.PS”的第一基本流 stream#0 相同，private_stream_id 是 0x00。

而且，对于作为剪辑流文件“00003.PS”的第二基本流的视频流 stream#1，在其 StreamInfo()中包括的 StaticInfo()(图 12)中，picture_size、frame_rate、cc_Flag 与作为剪辑流文件“00003.PS”的第一基本流的视频流 stream#0 相同。因此，作为剪辑流文件“00003.PS”的第二基本流的视频流 stream#1 是 720×480 像素和 29.97 Hz 帧循环的视频数据，并且也不包括闭合标题数据。

而且，对于作为剪辑流文件“00003.PS”的第二基本流的视频流 stream#1，在 StreamInfo()中的 number_of_DynamicInfo(图 10)是 0，因此，不存在 pts_change_point 和 DynamicInfo()的组。

接着，对于剪辑流文件“00003.PS”的第三基本流 stream#2，stream_id 是 0xBD，private_stream_id 是 0x00。因此，这个基本流 stream#2 是 ATRAC 音频流，如参见图 20 和图 22 所述。

而且，对于作为剪辑流文件“00003.PS”的第三基本流的 ATRAC 音频流 stream#2，在其 StreamInfo()中包括的 StaticInfo()(图 12)中，audio_language_code、channel_configuration、lfe_existence、sampling_frequency 与作为剪辑流文件“00001.PS”的第二基本流的 ATRAC 音频流 stream#1 的那些相同。因此，作为剪辑流文件“00003.PS”的第三基本流的这个 ATRAC 音频流 stream#2 是日语的，并且是立体声音频数据，而且，不存在低范围的增强声道，并且采样频率是 48 kHz。

而且，对于作为剪辑流文件“00003.PS”的第三基本流的 ATRAC 音频流 stream#2，在 StreamInfo()中的 number_of_DynamicInfo(图 10)是 3，因此，在 StreamInfo()中描述了三组 pts_change_point 和 DynamicInfo()。

接着，图 27 说明了参见图 10 所述的剪辑信息文件 Clip()中的 EP_map()的具体例子。即，图 27 说明了在图 4 中所示的剪辑信息文件“00001.CLP”内的、在图 14 中的 EP_map()的具体例子。

在图 27 中，EP_map()的 number_of_stream_id_entries 是 1，因此，在这

个 EP_map()中描述了用于一个基本流的解码开始可允许点的信息。

而且，对于在图 27 中的 EP_map()，stream_id 是 0x0E。因此，根据已经参见图 20 和图 22 所述的，在 EP_map()中描述了由 stream_id 为 0xE0 确定的视频流的解码开始可允许点的信息(PTS_EP_start 和 RPN_EP_start(图 14))。即，图 27 是剪辑信息文件“00001.CLP”的 EP_map()，并且在对应于剪辑信息文件“00001.CLP”的剪辑流文件“00001.PS”中具有 stream_id 为 0xE0 的基本流是参见图 26 所述的剪辑信息文件“00001.CLP”的第一视频流 stream#0，因此，在图 27 中的 EP_map()中所述的信息是视频流 stream#0 的解码开始可允许点的 PTS_EP_start 和 RPN_EP_start。

在图 27 中，对于剪辑信息文件“00001.CLP”的第一视频流 stream#0 的解码开始可允许点的前 5 个点，列出了 PTS_EP_start 和 RPN_EP_start，并且省略了对于第六个和随后的点的 PTS_EP_start 和 RPN_EP_start 的列出。

注意，等于在图 27 中的 EP_map()，private_stream_id 是 0x00，但是，在 stream_id 表示视频流的情况下，private_stream_id 是无关的(可以被忽略)，如上所述。

接着，图 28 说明了在参见图 25 所述的 PlayList#0 和 PlayList#1(在图 5 中的 PlayList())内的 PlayList Mark()的具体例子。

图 28 的上部说明了 PlayList#0 的 PlayListMark()(图 7)。

在图 28 的上部，在 PlayList#0 的 PlayListMark() 中的 number_of_PlayList_marks 是 7，因此，在 PlayList#0(的 PlayListMark())中包含的 Mark()的数量是 7。

而且，在图 28 的上部，作为在 PlayList#0 中包含的 7 个 Mark()的第一个 Mark()的 Mark#0 具有“章节（Chapter）”的 mark_type(图 7)，因此是章节标记。而且，Mark#0 的 ref_to_PlayItem_id(图 7)是 0，因此，属于在 PlayList#0 中包含的两个 PlayItem#0 和#1 的 PlayItem#0。而且，Mark#0 的 mark_time_stamp 是 180,090，因此是在由在 PlayList#0 中包含的 PlayItem#0 播放的剪辑流文件的时间点(播放时间点)180,090 上的标记。而且，Mark#0 的 entry_ES_stream_id 和 entry_ES_private_stream_id 是 0，因此，与任何基本流没有相关。而且，Mark#0 的 mark_data 是 1，表示具有第一号的章节。

注意，由在 PlayList#0 中包含的 PlayItem#0 播放的剪辑流文件是已经参见图 25 所描述的、由在 PlayItem#0 的 Clip_Information_file_name 中所述的

“00001.CLP”确定的剪辑流文件“00001.PS”，因此，Mark#0 的 mark_time_stamp 表示的上述时间点 180,090 是剪辑流文件“00001.PS”的时间点。

在图 28 的上部，作为在 PlayList#0 中包含的 7 个 Mark()的第五个 Mark()的 Mark#4 也是章节标记，与第一 Mark#0 相同。

即，作为第五 Mark()的 Mark#4 具有“章节（Chapter）”的 mark_type(标记类型)(图 7)，因此是章节标记。而且，Mark#4 的 ref_to_PlayItem_id(图 7)是 1，因此，属于在 PlayList#0 中的图 25 所示的两个 PlayItem#0 和#1 的 PlayItem#1。而且，Mark#4 的 mark_time_stamp 是 90,000，因此是在由在 PlayList#0 中包含的 PlayItem#1 播放的剪辑流文件的时间点 90,000 上的标记。而且，Mark#4 的 entry_ES_stream_id 和 entry_ES_private_stream_id 是 0，因此，与任何基本流没有相关。而且，Mark#4 的 mark_data 是 1，表示具有第 2 号的章节。

在此注意，由在 PlayList#0 中包含的 PlayItem#1 播放的剪辑流文件是已经参见图 25 所述的、由在 PlayItem#1 的 Clip_Information_file_name 中所述的“00002.CLP”确定的剪辑流文件“00002.PS”，因此，Mark#4 的 mark_time_stamp 表示的上述时间点 90,000 是剪辑流文件“00002.PS”的时间点。

而且，在图 28 的上部，作为在 PlayList#0 中包含的 7 个 Mark()的第 2 个 Mark()的 Mark#1 具有“索引（Index）”的 mark_type(图 7)，因此是索引标记。而且，Mark#1 的 ref_to_PlayItem_id(图 7)是 0，因此，属于在 PlayList#0 中的图 25 所示的两个 PlayItem#0 和#1 的 PlayItem#0。而且，Mark#1 的 mark_time_stamp 是 5,580,090，因此是在由在 PlayList#0 中包含的 PlayItem#0 播放的剪辑流文件的时间点 5,580,090 上的标记。而且，Mark#0 的 entry_ES_stream_id 和 entry_ES_private_stream_id 是 0，因此，与任何基本流没有相关。而且，Mark#1 的 mark_data 是 1，表示具有第 1 号的索引。

在此注意，由在 PlayList#0 中包含的 PlayItem#0 播放的剪辑流文件是如上所述的剪辑流文件“00001.PS”，因此，Mark#1 的 mark_time_stamp 表示的上述时间点 5,580,090 是剪辑流文件“00001.PS”的时间点。

在图 28 的上部，作为在 PlayList#0 中包含的 7 个 Mark()的第三、第六和第七 Mark()的 Mark#2、Mark#5 和 Mark#6 也与第二 Mark#1 相同是索引标记。

而且，在图 28 的上部，作为在 PlayList#0 中包含的 7 个 Mark()的第 4 个 Mark()的 Mark#3 具有“事件（Event）”的 mark_type(图 7)，因此是事件标记。而且，Mark#3 的 ref_to_PlayItem_id(图 7)是 0，因此，属于在 PlayList#0 中的、在图 25 中所示的两个 PlayItem#0 和#1 的 PlayItem#0。而且，Mark#3 的 mark_time_stamp 是 16,380,090，因此是在由在 PlayList#0 中包含的 PlayItem#0 播放的剪辑流文件的时间点 16,380,090 上的标记。而且，Mark#3 的 entry_ES_stream_id 和 entry_ES_private_stream_id 是 0，因此，没有与任何基本流的相关。而且，Mark#3 的 mark_data 是 0，表示事件用作为变元的 0 来触发。

在此注意，由在 PlayList#0 中包含的 PlayItem#0 播放的剪辑流文件是如上所述的剪辑流文件“00001.PS”，因此，Mark#3 的 mark_time_stamp 表示的上述时间点 16,380,090 是剪辑流文件“00001.PS”的时间点。

现在，在图 28 的上部，从 Mark()所属的 PlayItem()的首部开始的时间被表示在 PlayList#0 的 PlayListMark()的列表的外部的右侧，并且，在其右侧进一步给出了从 PlayList#0 的首部开始的时间。

接着，图 28 的下部说明了 PlayList#1 的 PlayListMark()(图 7)。

在图 28 的下部，在 PlayList#1 的 PlayListMark() 中的 number_of_PlayList_marks 是 3，因此，在 PlayList#1（的 PlayListMark()）中包含的 Mark()的数量是 3。

而且，在图 28 的下部，作为在 PlayList#1 中包含的三个 Mark()的第一个 Mark()的 Mark#0 具有“章节（Chapter）”的 mark_type(图 7)，因此是章节标记。而且，Mark#0 的 ref_to_PlayItem_id(图 7)是 0，因此，属于在 PlayList#1 中包含的、图 25 中的那个 PlayItem#0。而且，Mark#0 的 mark_time_stamp 是 90,000，因此是在由在 PlayList#1 中包含的 PlayItem#0 播放的剪辑流文件的时间点 90,000 上的标记。而且，Mark#0 的 entry_ES_stream_id 和 entry_ES_private_stream_id 都是 0，因此，没有与任何基本流的相关，而且，Mark#0 的 mark_data 是 0，表示具有第 0 号的章节。

在此注意，由在 PlayList#1 中包含的 PlayItem#0 播放的剪辑流文件是已经参见图 25 所述的、由在 PlayItem#0 的 Clip_Information_file_name 中所述的“00003.CLP”确定的剪辑流文件“00003.PS”，因此，Mark#0 的 mark_time_stamp 表示的上述时间点 90,000 是剪辑流文件“00003.PS”的时间

点。

而且，在图 28 的下部，作为在 PlayList#1 中包含的 3 个 Mark()的第 2 个 Mark()的 Mark#1 具有“事件（Event）”的 mark_type(图 7)，因此是事件标记。而且，Mark#1 的 ref_to_PlayItem_id(图 7)是 0，因此，属于在 PlayList#1 中的、在图 25 中所示的那个 PlayItem#0。而且，Mark#1 的 mark_time_stamp 是 27,090,000，因此是在由在 PlayList#1 中包含的 PlayItem#0 播放的剪辑流文件的时间点 27,090,000 上的标记。而且，对于 Mark#1，entry_ES_stream_id 是 0xE0，entry_ES_private_stream_id 是 0，因此，与由 0xE0 的 stream_id 确定(识别)的基本流即参见图 20 和图 22 所述的视频流有相关。而且，Mark#1 的 mark_data 是 0，表示事件用作为变元的 1 来触发。

在此注意，由在 PlayList#1 中包含的 PlayItem#0 播放的剪辑流文件是如上所述的剪辑流文件“00003.PS”，因此，Mark#1 的 mark_time_stamp 表示的上述时间点 27,090,000 是剪辑流文件“00003.PS”的时间点。

而且，Mark#1 相关联的、具有 0xE0 的 stream_id 的视频流是 Mark#1 所属的、在图 25 中所示的 PlayList#1 中包含的 PlayItem#0 的 Clip_Information_file_name 中所述的“00003.CLP”中所述的具有 0xE0 的 stream_id 的视频流，即，在从在图 26 中所示的剪辑信息文件“00003.CLP”确定的剪辑流文件“00003.PS”中复用的三个基本流 stream#0 到#2 中，这是第一基本流(视频流)stream#0。

接着，在图 28 的下部，作为在 PlayList#1 中包含的 3 个 Mark()的第 3 个 Mark()的 Mark#2 具有“事件（Event）”的 mark_type(图 7)，因此是事件标记。而且，Mark#2 的 ref_to_PlayItem_id(图 7)是 0，因此，属于在 PlayList#1 中的、在图 25 中所示的那个 PlayItem#0。而且，Mark#1 的 mark_time_stamp 是 27,540,000，因此是在由在 PlayList#1 中包含的 PlayItem#0 播放的剪辑流文件的时间点 27,540,000 上的标记。而且，对于 Mark#2，entry_ES_stream_id 是 0xE1，entry_ES_private_stream_id 是 0，因此，与由 0xE1 的 stream_id 确定(识别)的基本流即参见图 20 和图 22 所述的视频流有相关。而且，Mark#2 的 mark_data 是 2，表示事件用作为变元的 2 来触发。

在此注意，由在 PlayList#1 中包含的 PlayItem#0 播放的剪辑流文件是如上所述的剪辑流文件“00003.PS”，因此，Mark#2 表示的上述时间点 27,540,000 是剪辑流文件“00003.PS”的时间点。

而且，Mark#2 相关联的、具有 0xE1 的 stream_id 的视频流是 Mark#2 所属的、在图 25 中所示的 PlayList#1 中包含的 PlayItem#0 的 Clip_Information_file_name 中所描述的“00003.PS”中所述的具有 0xE1 的 stream_id 的视频流，即，在从在图 26 中所示的剪辑信息文件“00003.CLP”确定的剪辑流文件“00003.PS”中复用的三个基本流 stream#0 到#2 中，这是第二基本流(视频流)stream#1。

现在，在图 28 的下部，从 Mark()所属的 PlayItem()的首部开始的时间被表示在 PlayList#1 的 PlayListMark()的列表之外的右侧。

注意，虽然在图 28 中，章节标记和索引标记指示的章节和索引编号被描述在 mark_data 中，但是可以通过计数例如在 PlayListMark()中的章节标记和索引标记的数量来识别章节标记和索引标记指示的章节和索引编号，即使其没有在 mark_data 中描述也是如此。

[盘装置的操作的说明]

接着，假定诸如图 25 到图 28 所述的数据(文件)被记录在图 1 所示的盘 101 中，说明图 1 中所示的盘装置的操作。

在盘 101 被插入到盘驱动器 102 中时，关于该结果的消息经由驱动器接口 114 和进一步经由图 2 所示的操作系统而被发送到视频内容播放程序 210。在从操作系统 201 接收到关于盘 101 已经被插入到盘驱动器 102 中的消息时，视频内容播放程序 210 开始图 29 中所示的预先播放处理。

“预先播放处理”

即，图 29 是用于说明视频内容播放程序 210 执行的预先播放处理的流程图。

现在，下面参见所述流程图所述的盘装置的操作或处理不必然需要以按照在所述流程图中列出的顺序的时间序列被执行，可以并行或者独立地被执行。但是，注意，对于本说明书，为了便利说明，将按照所述流程图来说明盘装置的操作或者处理。

在所述预先播放处理中，在步骤 S101，视频内容播放程序 210 使用操作系统 201 的文件系统功能来检查盘 101，并且确定是否盘 101 是由视频内容播放程序 210 使用的通常的盘。

现在，如上所述，使用操作系统 201 的文件系统功能来执行对于盘 101 的访问(读出文件)，但是，将在下面适当地省略其说明。

在步骤S101中确定盘101不是正常盘的情况下，即在盘101中使用的文件系统不是与操作系统201兼容的类型的情况下，或者在例如根目录中没有“VIDEO(视频)”目录的情况下，视频内容播放程序210确定它与盘101不兼容，进行到步骤S102，图形处理模块219执行错误处理，并且所述预先播放处理结束。

即，图形处理模块219产生关于盘101是不正常之结果的错误消息(的视频数据)来作为错误处理，并且从视频输出模块220输出它，由此显示错误消息。注意，可以进行其他的布置来用于错误处理，其中，例如，从音频输出模块221发出警告声，从盘驱动器102退出盘101，等等。

另一方面，在步骤S101中确定盘101是通常盘的情况下，流程进行到步骤S103，并且视频内容播放程序210使用内容数据提供模块213来从操作系统201请求并且读出在盘101的“VIDEO”目录中的“SCRIPT.DAT”文件和“PLAYLIST.DAT”文件的两个数据文件(图4)，并且流程进行到步骤S104。在步骤S104中，向脚本控制模块211提供“SCRIPT.DAT”文件，向播放器控制模块212提供“PLAYLIST.DAT”文件。

随后，流程从步骤S104向S105到S107进行，并且播放器控制模块212执行初始化处理。注意，脚本控制模块211等待直到播放器控制模块212的初始化处理结束。

“播放器控制模块212的初始化处理”

关于初始化处理，在步骤S105中，播放器控制模块212分析所述“PLAYLIST.DAT”文件，并且查看在“PLAYLIST.DAT”文件及其文件名中使用的剪辑信息文件的数量。

即，在这种情况下，“PLAYLIST.DAT”文件是在图25中使出的那个，因为在图25中的“PLAYLIST.DAT”文件中的number_of_PlayLists是2，因此播放器控制模块212识别存在两个PlayList()，即第一PlayList#0和第二PlayList#1。而且，关于在图25中的“PLAYLIST.DAT”文件中的第一PlayList#0，number_of_PlayItems是2，因此，播放器控制模块212识别在PlayList#0中存在两个PlayItem()，它们是第一PlayItem#0和第二PlayItem#1。播放器控制模块212然后引用在图25中的“PLAYLIST.DAT”文件中的PlayList#0中的第一PlayItem#0和第二PlayItem#1的每一个的Clip_Information_file_name，由此识别在PlayList#0中包括的第一PlayItem#0

的剪辑信息文件(的文件名)是“00001.CLP”，并且在 PlayList#0 中包括的第二 PlayItem#1 的剪辑信息文件是“00002.CLP”。

以相同的方式，播放器控制模块 212 识别：因为 number_of_PlayItems 对于第二 PlayList#1 是 1，因此在 PlayList#1 中存在一个 PlayItem()(PlayItem#0)，而且，从其 Clip_Information_file_name，PlayItem#0 的剪辑信息文件是“00003.CLP”。

随后，流程从步骤 S105 进行到步骤 S106，在此，播放器控制模块 212 从盘 101 读入在步骤 S105 中识别的剪辑信息文件，即，从在盘 101 的“VIDEO”目录中的“CLIP(剪辑)”目录读入三个剪辑信息文件“00001.CLP”、“00002.CLP”、和“00003.CLP”。

现在，在步骤 S106 中的剪辑信息文件的读入对于被读入的、要首先播放的 PlayList()的 PlayItem 的剪辑信息文件是足够的，但是对于本实施例，预先读入所有 PlayList()的所有 PlayItem 的剪辑信息文件，如上所述。

按照在步骤 S106 中的处理，流程进行到步骤 S107，在此，播放器控制模块 212 确定是否在步骤 S105 中识别的剪辑信息文件的读入已经成功，并且进一步确定(查看)是否在盘 101 中存在对应于已经被读入的剪辑信息文件的剪辑流文件。即，在步骤 S107 中，进行确定是否剪辑信息文件“00001.CLP”、“00002.CLP”、“00003.CLP”的读入已经成功，进一步，确定是否在盘 101 中的“VIDEO”目录下的“STREAM”目录中存在剪辑流文件“00001.PS”、“00002.PS”、“00003.PS”，其中，仅仅其文件名的后缀与剪辑信息文件“00001.CLP”、“00002.CLP”、“00003.CLP”不同。

在步骤 S107 中确定在步骤 S105 中识别的剪辑信息文件的读入已经失败或者在盘 101 中不存在对应于剪辑信息文件的剪辑流文件的情况下，即例如在盘 101 中未记录用于按照“PLAYLIST.DAT”文件的重放所需要的剪辑信息文件或者剪辑流文件的情况下，视频内容播放程序 210 确定它与盘 101 不兼容，并且进行到步骤 S102，在此，执行上述的错误处理，并且预先播放处理结束。

另一方面，在步骤 S107 中确定在步骤 S105 中识别的剪辑信息文件的读入已经成功、并且在盘 101 中存在对应于剪辑信息文件的剪辑流文件的情况下，播放器控制模块 212 结束初始化处理，并且流程进行到步骤 S108。

在步骤 S108，脚本控制模块 211 开始分析和执行“SCRIPT.DAT”文件。

例如，如果我们说在此第一 PlayList()(PlayList#0)的播放已经由执行“SCRIPT.DAT”文件的脚本控制模块 211 指令到播放器控制模块 212 时，则在图 30 中所示的播放处理开始。

“播放处理”

即，图 30 是用于描述视频控制播放程序 210 执行的播放处理的流程图。

“播放准备处理”

在所述播放处理中，首先，在步骤 S121 和 S122 中，播放器控制模块 212 执行已经被脚本控制模块 211 指令播放的 PlayList()、即第一 PlayList()(PlayList#0)的播放准备处理。

即，在步骤 S121 中，播放器控制模块 212 确定在第一 PlayList#0 中包括的第一 PlayItem#0 的 IN_time(图 6)，进行到步骤 S122，并且查看用于在由在第一 PlayList#0 中包括的第一 PlayItem#0 播放的剪辑流文件“00001.PS”上开始播放的播放开始位置，它等同于其 PlayItem#0 的 IN_time。

现在，在 PlayItem()的 IN_time(图 6)指示剪辑流文件的首部时，可以从所述剪辑流文件的首部读出节目流，但是在 IN_time 指示除了剪辑流文件首部之外的位置的情况下，需要找出(查看)对应于 IN_time 的位置，并且从这些读出节目流。

具体上，在图 25 中所示的情况下，第一 PlayList#0 的第一 PlayItem#0 的 IN_time 是 180,090。播放器控制模块 212 从由在第一 PlayList#0 包括的第一 PlayItem#0 播放的剪辑流文件“00001.CLP”的图 27 中所示的 EP_map()中，找出作为 PlayItem#0 的 IN_time 的、适合于 180,090 的播放开始位置。

即，播放器控制模块 212 例如使用二进制搜索等，从表示在 EP_map()中所述的解码开始可允许点的所有 PTS_EP_start 中，搜索满足表达式 PTS_EP_start≤IN_time 的最大 PTS_EP_start。现在，搜索等于或小于 IN_time 的 PTS_EP_start 的原因是由 IN_time 表示的位置不必然是解码开始可允许点。

在这种情况下，如上所述，IN_time 是 180,090。而且，在由在第一 PlayList#0 中包括的第一 PlayItem#0 播放的剪辑流文件“00001.CLP”的在图 27 中所示的 EP_map()中，将 180,090 描述为满足表达式 PTS_EP_start≤IN_time 的最大 PTS_EP_start。因此，播放器控制模块 212 从在图 27 中所示的 EP_map()搜索作为 180,090 的 PTS_EP_start。

而且，对于播放器控制模块 212，读出作为对应于被搜索的 PTS_EP_start

的 RPN_EP_start 的 305(扇区)，并且将由作为 305 的 RPN_EP_start 表示的剪辑流文件“00001.PS”上的位置确定为播放开始位置。

在如上所述确定播放开始位置后，流程从步骤 S122 进行到 S123，并且播放器控制模块 212 控制图形处理模块 219 来显示时间代码。图形处理模块 219 按照播放器控制模块 212 的控制，并且产生时间代码(的视频数据)，并且将其输出到视频输出模块 220。因此，开始时间代码显示。

现在，在步骤 S123 中开始其显示的时间代码是一个值，其中，PlayList()的首部已经被转换为例如 00:00:00(小时：分钟：秒)。注意，可以与时间代码一起或替代时间代码而显示章节和索引编号。

“PlaylistMark()的分析处理”

在已经在步骤 S123 中开始时间代码的显示后，流程进行到步骤 S124，其中，播放器控制模块 212 执行分析处理，用于分析在已经从脚本控制模块 211 指令播放的 PlayList()——即第一 PlayList()(PlayList#0)——中所述的 PlaylistMark()(图 7)。

具体地，播放器控制模块 212 识别在 PlayList#0 中包含的 Mark()的数量是 7，因为在已经被读入的“PLAYLIST.DAT”文件的第一 PlayList#0 中，number_of_PlayList_marks 在图 28 中上部示出的 PlaylistMark()中是 7。

而且，播放器控制模块 212 分析在 PlayList#0 中包括的、在图 28 中上部示出的 7 个 Mark()，并且从其 ref_to_PlayItem_id 识别所述 7 个 Mark()的从第一到第四的四个 Mark()属于 PlayList#0 的第一 PlayItem()(PlayItem#0)。

随后，播放器控制模块 212 提取属于在 PlayList#0 中的第一 PlayItem#0 的四个 Mark()的 mark_time_stamp，并且将它们作为具有元素计数 4 的串传送到解码控制模块 214。即，从播放器控制模块 212 向解码控制模块 214 传送在图 28 上部的 7 个 Mark()的从第一到第四的四个 Mark()中的、每个 mark_time_stamp{180,190}、{5,580,090}、{10,980,090}、{16,380,090}的四个时间点。此时，也从播放器控制模块 212 向解码控制模块 214 传送这些时间点的属性是“标记处理”的事实。在时钟单元 214A 保持的时间点匹配“标记处理”的属性的时间点的时候，解码控制模块 214 向播放器控制模块 212 通知关于其结果的消息、具有“标记处理”的属性的时间点匹配的时间点和“标记处理”的属性。

“用于确定要播放的基本流的处理”

接着，流程从步骤 S124 进行到 S125，并且播放器控制模块 212 确定要播放的基本流。

即，因为 number_of_stream 在其名称被描述在第一 PlayList#0——它是关于已经被脚本控制模块 211 指令播放的 PlayList()——的第一 PlayItem#0(图 25)的 Clip_Information_file_name 中的、在图 26 中的剪辑信息文件“00001.CLP”中是 4，因此播放器控制模块 212 识别在对应的剪辑流文件“00001.PS”上复用四个基本流。而且，播放器控制模块 212 依序查看对应于所述四个基本流的、在图 26 中所示的剪辑信息文件“00001.CLP”的 StaticInfo()的 stream_id 和必要的 private_stream_id，并且识别所述四个基本流是一个视频流、一个 ATRAC 音频流和两个字幕流。即，识别在剪辑流文件“00001.PS”上复用的每个属性的基本流的数量。

注意，在剪辑流文件上复用的每个属性的基本流的数量的信息用于在播放期间转换基本流(音频转换、字幕转换等)。而且，存在下述情况：在剪辑流文件中没有字幕流(在内容中不包括字幕)，并且“字幕流”属性的基本流的数量的信息用于确定是否存在字幕流。

根据这样的 StaticInfo()查看结果，播放器控制模块 212 选择和确定要播放的基本流，但是，在这种情况下，在剪辑流文件“00001.PS”上复用的四个基本流中，对于属性“视频流”和“音频流”的基本流的每一个，仅仅存在一个，因此，没有用于选择属性“视频流”和“音频流”的基本流的空间，并且将单个视频流和音频流(ATRAC 音频流)确定为要播放的基本流。

而且，对于属性“字幕流”的基本流，在剪辑流文件“00001.PS”上复用的四个基本流中存在两个，因此将所述两个字幕流的一个选择和确定为要播放的基本流。在此，例如，让我们说，在所述两个字幕流中，以在剪辑流文件“00001.PS”中的出现顺序来选择第一字幕流。

现在，需要确定所述四个基本流的每一个，以便识别在如上所述的剪辑流文件“00001.PS”上复用的四个基本流的属性和数量，并且播放器控制模块 212 通过 stream_id 和必要的 private_stream_id 来确定在剪辑流文件“00001.PS”上复用的四个基本流。

即，在剪辑流文件“00001.PS”上复用的四个基本流中，播放器控制模块 212 确定对于参见图 26 的剪辑信息文件“00001.CLP”所述的、作为具有 0xE0 的 stream_id 的属性“视频流”的基本流的视频流。

而且，在剪辑流文件“00001.PS”上复用的四个基本流中，播放器控制模块 212 确定对于参见图 26 的剪辑信息文件“00001.CLP”所述的、作为具有 0xBD 的 stream_id 和 0x00 的 private_stream_id 的属性“音频流”的基本流的 ATRAC 音频流。

而且，在剪辑流文件“00001.PS”上复用的四个基本流中，播放器控制模块 212 确定两个字幕流，这两个字幕流是对于参见图 26 的剪辑信息文件“00001.CLP”所述的、作为具有 0xBD 的 stream_id 和 0x80 的 private_stream_id 并且具有 0xBD 的 stream_id 和 0x81 的 private_stream_id 的属性“字幕流”的基本流。

如上所述，可以通过被描述为对应于剪辑流文件的剪辑信息的元数据的 stream_id 和 private_stream_id 的组合来确定在其剪辑流文件上复用的基本流。

现在，stream_id 和 private_stream_id 的组合是用于扩展 MPEG2 系统的复用的机构。可以通过使用用于确定在元数据(数据库)中的基本流的 stream_id 和 private_stream_id 的组合来以可靠的方式确定基本流。这在可扩展性上也是优越的，因为可以原样地使用当前的机构，即使在扩展 private_stream_id 的含义和增加对应的基本流的数量和类型(属性)的未来情况下也是如此。

即，例如，对于 BD(蓝光盘)标准，MPEG2 标准传送流(Transport Stream)PID(分组 ID)用于确定数据，因此这限于 MPEG2 标准。而且，例如，对于 DVD-VIDEO 标准，定义了类似 private_stream_id 的 sub_stream_id，但是 sub_stream_id 未被布置成在用于确定流的数据库中描述，并且仅仅可以描述在固定区域中的 8 或 32 个流的流信息(例如参见 VI4-49，表 4.2.1-2(VTS-AST_ATRT)和 VI4-52，表 4.2.1-3(VTS-SPST/ATRT)等)，因此可扩展性差。

相反，对于 stream_id 和 private_stream_id 的组合，将说明与例如其中描述了元数据的在图 10 中所示的剪辑信息文件 Clip()中的 number_of_streams 表示的数量一样多的数量，因此，可以从用作为在剪辑信息文件 Clip()中所述的元数据的 stream_id 和 private_stream_id 的组合中来确定在剪辑流文件上复用的基本流，而不论其数量如何(虽然在能够被 number_of_streams 表示的范围内)。

注意，除了用于确定在图 10 中所示的剪辑信息文件中的对应剪辑流文件

上复用的基本流之外，作为例如在图 7 中所示的 PlayListMark()中的 entry_ES_stream_id 和 entry_ES_private_stream_id 的组合，在本实施例中的 stream_id 和 private_stream_id 的组合也用于确定具有 Mark()的相关的基本流。而且，stream_id 和 private_stream_id 的组合也用于在例如图 14 的 EP_map()中确定具有解码可允许开始点的说明的基本流。

“输出属性控制处理”

随后，流程从步骤 S125 进行到 S126，在此，播放器控制模块 212 执行要播放的基本流、即已经在步骤 S125 中被确定播放的基本流的输出属性的控制处理。

具体地，播放器控制模块 212 首先查看 number_of_DynamicInfo(图 10)，它表示 DynamicInfo()的数量(图 13)，在 DynamicInfo()中描述了关于要播放的基本流、即已经在步骤 S125 中被确定播放的基本流的视频流、ATRAC 音频流和字幕流的输出属性。

在此，在这种情况下，要播放的视频流、ATRAC 音频流和字幕流是在剪辑流文件“00001.PS”上被复用的每个基本流，并且如在图 26 中的“00001.CLP”的说明中所述的，这些 number_of_DynamicInfo 对于每个是 0。在 number_of_DynamicInfo 对于要播放的所有基本流是 0 的情况下，播放器控制模块 212 不执行处理，特别是对于要播放的基本流的输出属性的控制处理。

注意，在 number_of_DynamicInfo 对于所有要播放的基本流不是 0 的情况下，下面说明作为基本流的输出属性的控制而执行的处理。

“用于开始播放的准备处理”

在步骤 S126 的处理之后，流程进行到步骤 S127，其中，播放器控制模块 212 执行用于开始要播放的基本流的播放的准备处理。

即，播放器控制模块 212 向内容数据提供模块 213 提供在其上要播放的基本流被复用的剪辑流文件“00001.PS”的文件名和在作为在步骤 S122 中确定的播放开始位置的 EP_map()中所述的 RPN_EP_start(= 305)。

而且，播放器控制模块 212 在开始向缓冲器控制模块 215 提供在其上要播放的基本流被复用的剪辑流文件“00001.PS”中存储的节目流之前初始化缓冲器控制模块 215。

具体地，在缓冲器控制模块 215(图 3)，相同的值被代入在数据首部指针存储单元 231 中存储的数据首部指针、在数据写入指针存储单元 232 中存储

的数据写入指针、在视频读取指针存储单元 241 中存储的视频读取指针、在音频读取指针存储单元 251 中存储的音频读取指针和在字幕读取指针存储单元 262 中存储的字幕读取指针中。

因此，在数据首部指针存储单元 231 中存储的数据首部指针和在数据写入指针存储单元 232 中存储的数据写入指针指向在缓冲器控制模块 215 的缓冲器 215A 中的同一位置。这表示在缓冲器 215A 中不存储有效数据的状态。

而且，播放器控制模块 212 向缓冲器控制模块 215 提供用作为用于识别(确定)要播放的基本流的标识信息的 stream_id，并且如果必要的话也提供 private_stream_id。

即，如上所述，在要播放的基本流中，通过 stream_id 0xE0 来确定属性“视频流”的视频流，通过 stream_id 0xBD 和 0x00 的 private_stream_id 来确定属性“音频流”的 ATRAC 音频流，并且通过 stream_id 0xBD 和 0x80 的 private_stream_id 来确定属性“字幕流”的字幕流。播放器控制模块 212 向缓冲器控制模块 215 提供这些 stream_id 和 private_stream_id。

在缓冲器控制模块 215(图 3)，视频读取功能单元 233 在 stream_id 寄存器 242 中存储来自播放器控制模块 212 的用于视频流的 0xE0 的 stream_id 。而且，音频读取功能单元 234 分别在 stream_id 寄存器 252 和 private_stream_id 寄存器 253 中存储来自播放器控制模块 212 的 0xBD 的 stream_id 和 0x00 的 private_stream_id。而且，字幕读取功能单元 235 分别在 stream_id 寄存器 263 和 private_stream_id 寄存器 264 中存储来自播放器控制模块 212 的 0xBD 的 stream_id 和 0x80 的 private_stream_id。

注意，播放器控制模块 212 存储已经被提供到缓冲器控制模块 215 的、要播放的基本流的 stream_id 和 private_stream_id，用于以后的处理。播放器控制模块 212 在后述的流转换请求消息发生的情况下或者在后述的标记处理中使用这些 stream_id 和 private_stream_id 来确定当前被播放的流。

作为缓冲器控制模块 215(图 3)的进一步的初始化，播放器控制模块 212 在字幕读取功能标记存储单元 261 中设置对应于在其上要播放的基本流被复用的剪辑流文件的值的字幕读取功能标记。

即，在这种情况下，在其上要播放的基本流被复用的剪辑流文件“00001.PS”中包含字幕流，因此，在字幕读取功能标记存储单元 261 中设置字幕读取功能标记 1，以使得字幕读取功能单元 235 起作用。注意，在其

上要播放的基本流被复用的剪辑流文件中不包含字幕流的情况下，在字幕读取功能标记存储单元 261 中设置字幕读取功能标记 0。在这种情况下，字幕读取功能单元 235 不起作用(具体地不执行任何处理)。

而且，播放器控制模块 212 向解码控制模块 214 传送在在已经从脚本控制模块 211 指定播放的第一 PlayList#0 中包含的第一 PlayItem#0 的 IN_time 180,090 和 OUT_time 27,180,090。在解码控制模块 214，IN_time 被用于控制要通过 PlayItem()播放的剪辑的解码的开始，OUT_time 被用于结束所述剪辑的解码，并且还用于后述的 PlayItem 交叉控制。

而且，播放器控制模块 212 初始化向图形处理模块 219 发出的字幕流显示格式指令。即，播放器控制模块 212 控制图形处理模块 219 以便将字幕流显示格式设置为例如默认的显示格式。

“开始接收读取”

随后，流程从步骤 S127 进行到 S128，并且播放器控制模块 212 控制内容数据提供模块 213，由此内容数据提供模块 213 使用操作系统 201 的功能来读出其中存储了节目流的剪辑流文件，在该节目流上，要播放的基本流被复用。即，内容数据提供模块 213 指定在盘 101(图 4)中的“VIDEO”目录下的“STREAM”目录中的剪辑流文件“00001.PS”，并且进一步指定作为在步骤 S122 中确定的播放开始位置的扇区 305，以及请求操作系统 201 读出所述文件。而且，内容数据提供模块 213 指定从盘 101 读出的数据应该被提供给缓冲器控制模块 215。

因此，开始从盘 101 读出在剪辑流文件“00001.PS”中存储的节目流，并且向缓冲器控制模块 215 提供所述节目流。

缓冲器控制模块 215(图 3)向数据写入指针存储单元 232 的数据写入指针指向的、在缓冲器 215A 中的位置写入从盘 101 读出并且被提供的节目流，并且将数据写入指针递增被写入的数据的大小。

现在，以下除非特别地被拒绝，我们将说内容数据提供模块 213 从盘 101 读出数据，并且提供到和存储在缓冲器控制模块 215 的缓冲器 215A 中，只要在缓冲器控制模块 215 的缓冲器 215A 中存在空间。因此，我们在此说，在缓冲器 215A 中总是存储了足够的数据。

“开始解码控制”

在如上所述开始从盘 101 读出数据并且在缓冲器控制模块 215 的缓冲器

215A 中存储了所述数据后，流程从步骤 S128 进行到步骤 S129，在此，解码控制模块 214 控制视频解码器控制模块 216、音频解码器控制模块 217 和字幕解码器控制模块 218 以开始从缓冲器 215A 读出数据，这作为解码操作的预备阶段。

即，因此，视频解码器控制模块 216 向缓冲器控制模块 215(图 3)的视频读取功能单元 233 请求数据，并且获得在缓冲器 215A 中存储的单个视频访问单元、被附加到视频访问单元的 PTS 和 DTS(以下在适当时称为时间标记)和作为在位于紧接解码开始可允许点之前的 private_stream_2 的 PES_packet()中所述的信息(以下称为附加信息)的 pict_struct_copy、au_ref_flag、AU_length 等，它们响应于所述请求而被从缓冲器控制模块 215 传送。注意，每次视频解码器控制模块 216 获得视频访问单元时，从视频解码器控制模块 216 向解码控制模块 214 传送时间标记。

另一方面，音频解码器控制模块 217 向缓冲器控制模块 215(图 3)的音频读取功能单元 234 请求数据，并且获得响应于所述请求从缓冲器控制模块 215 传送的在缓冲器 215A 中存储的单个(ATRAC)音频访问单元和被附加到音频访问单元的时间标记(PTS 和 DTS)。注意，每次音频解码器控制模块 217 获得视频访问单元时，从音频解码器控制模块 217 向解码控制模块 214 传送时间标记。

而且，字幕解码器控制模块 218 向缓冲器控制模块 215(图 3)的字幕读取功能单元 235 请求数据，并且获得响应于所述请求而从缓冲器控制模块 215 传送的、在缓冲器 215A 中存储的单个字幕访问单元和被附接到字幕访问单元的时间标记。注意，每次字幕解码器控制模块 218 获得字幕访问单元时，从字幕解码器控制模块 218 向解码控制模块 214 传送时间标记。而且，在要播放的基本流中没有字幕流的情况下，或者在没有在缓冲器 215A 中存储的字幕访问单元的情况下，不从缓冲器控制模块 215 向字幕解码器控制模块 218 传送数据。

现在，每次视频解码器控制模块 216、音频解码器控制模块 217 和字幕解码器控制模块 218 从缓冲器控制模块 215 请求数据时，数据请求的结果被传送到解码控制模块 214。

而且，下面将说明从缓冲器控制模块 215 向视频解码器控制模块 216、音频解码器控制模块 217 和字幕解码器控制模块 218 传送的数据被从缓冲器

215A 读出的细节。

“开始解码”

如上所述，在视频解码器控制模块 216、音频解码器控制模块 217 和字幕解码器控制模块 218 开始从缓冲器控制模块 215 的缓冲器 215A 读出数据时，流程从步骤 S129 进行到 S130，并且数据的解码开始。

即，根据在已经在步骤 S127 中从播放器控制模块 212 提供的第一 PlayList#0 中包含的第一 PlayItem#0 的 IN_time 180,090 和如在步骤 S129 中所述的从视频解码器控制模块 216、音频解码器控制模块 217 和字幕解码器控制模块 218 传送的时间标记，解码控制模块 214 指令视频解码器控制模块 216、音频解码器控制模块 217 和字幕解码器控制模块 218 开始解码，如果必要的话移位定时，以保证同步。

现在，存在在例如日本专利第 3,496,725 中所述的一种用于指令解码开始并且移位时间以保证同步的方法，简而言之，将从视频解码器控制模块 216、音频解码器控制模块 217 和字幕解码器控制模块 218 传送的时间标记的最小值设置为由时钟单元 214A 定时的时间点初始值，开始所述时间点的计时，并且在由时钟单元 214A 保持的时间点与所述时间标记匹配时指令解码。

视频解码器控制模块 216 从解码控制模块 214 接收用于开始解码的指令，并且响应于这个指令，向视频解码器 116(图 1)传送从缓冲器控制模块 215（图 3）的视频读取功能单元 233 获得的单个视频访问单元以解码。而且，视频解码器控制模块 216 向图形处理模块 219 提供作为由视频解码器 116 解码的结果而获得的视频数据。

随后，视频解码器控制模块 216 在视频解码器 116 依序解码从缓冲器控制模块 215 的视频读取功能单元 233 一次一个地获得的视频访问单元，并且向图形处理模块 219 提供作为解码的结果而获得的视频数据。

另一方面，音频解码器控制模块 217 从解码控制模块 214 接收开始解码的指令，并且响应于这个指令，向音频解码器 117（图 1）传送从缓冲器控制模块 215(图 3)的音频读取功能单元 234 获得的单个音频访问单元以解码。而且，音频解码器控制模块 217 向音频输出模块 221 提供作为由音频解码器 117 解码的结果而获得的音频数据。

随后，音频解码器控制模块 217 在音频解码器 117 依序解码从缓冲器控制模块 215 的音频读取功能单元 234 一次一个地获得的音频访问单元，并且

向音频输出模块221提供作为解码结果而获得的音频数据。

而且，字幕解码器控制模块218从解码控制模块214接收用于开始解码的指令，并且响应于这个指令，使用内置的字幕解码软件来解码从缓冲器控制模块215(图3)的字幕读取功能单元235获得的单个字幕访问单元，并且向图形处理模块219提供作为其解码结果而获得的字幕数据(字幕图像数据)。

随后，字幕解码器控制模块218使用内置的字幕解码软件来依序解码从缓冲器控制模块215的字幕读取功能单元235一次一个地获得的字幕访问单元，并且向图形处理模块219提供作为解码结果而获得的字幕数据。

“图形处理”

随后，流程从步骤S130进行到步骤S131，并且图形处理模块219对于如上所述从视频解码器控制模块216提供的视频数据和在必要时对于从字幕解码器控制模块218提供的字幕数据执行图形处理。

即，图形处理模块219首先执行字幕处理，其中，按照来自播放器控制模块212的关于显示格式的指令，将来自字幕解码器控制模块218的字幕数据放大、缩小等。在没有来自播放器控制模块212的关于显示格式的指令的情况下，或者在有用于默认显示格式的指令的情况下，图形处理模块219原样保存来自字幕解码器控制模块218的字幕数据。

而且，图形处理模块219相加来自视频解码器控制模块216的视频数据和来自字幕解码器控制模块218的字幕数据或者字幕处理后的字幕数据，以便获得输出视频数据，其中，将字幕数据覆盖在被提供到视频输出模块220的、来自视频解码器控制模块216的视频数据上。

注意，在从脚本控制模块211或者播放器控制模块212接收到用于诸如菜单、消息、时间代码、章节、索引编号等显示信息的指令的情况下，图形处理模块219产生那个信息，将其覆盖在输出视频数据上，并且将其提供到视频输出模块220。

“输出处理”

在步骤S131的处理后，流程进行到步骤S132，其中，视频输出模块220依序在FIFO 220A中存储如在步骤S131中所述的从图形处理模块219提供的输出视频数据，并且以预定的输出速率依序输出在FIFO 220A中存储的所述输出视频数据。

视频输出模块220从图形处理模块219接收输出视频数据，只要在FIFO

220A 的存储容量(剩余量)上存在空间，但是，在没有更多空间的情况下，视频输出模块 220 请求图形处理模块 219 停止接收输出视频数据。因此，图形处理模块 219 停止处理，并且也请求视频解码器控制模块 216 和字幕解码器控制模块 218 停止处理。因此，视频解码器控制模块 216 和字幕解码器控制模块 218 停止处理。

在请求所述图形处理模块 219 停止输出视频数据的接收处理之后，其中来自 FIFO 220A 的输出视频数据的输出已经处理到在 FIFO 220A 中有空间，视频输出模块 220 请求从图形处理模块 219 接收输出视频数据。这个请求被从图形处理模块 219 发送到视频解码器控制模块 216 和字幕解码器控制模块 218，这与请求停止接收输出视频数据的情况一样。因此，图形处理模块 219 以及视频解码器控制模块 216 和字幕解码器控制模块 218 恢复已经被停止的处理。

另一方面，音频输出模块 221 也依序在 FIFO 221A 中存储如在步骤 S130 中所述的从音频解码器控制模块 227 提供的输出音频数据，并且以预定的输出速率(采样频率)依序输出在 FIFO 221A 中存储的音频数据。

音频输出模块 221 从音频解码器控制模块 217 接收音频数据，只要在 FIFO 221A 的存储容量(剩余量)上存在空间，但是在没有更多空间的情况下，音频输出模块 221 请求音频解码器控制模块 217 停止接收音频数据。因此，音频解码器控制模块 217 停止处理。

在请求所述音频解码器控制模块 217 停止音频数据的接收处理之后，其中来自 FIFO 221A 的数据的输出已经处理到在 FIFO 220A 中有空间，音频输出模块 221 请求从音频解码器控制模块 217 接收音频数据。因此，音频解码器控制模块 217 恢复已经被停止的处理。

因此，当从视频输出模块 220 和音频输出模块 221 输出数据时，执行基本流的解码。

当使用在图 1 中所示的盘装置来播放盘 101 时的整体处理和操作的流程是如已经参见图 29 和图 30 所述的，现在，将说明当使用盘装置来播放盘 101 时执行的其他处理和操作。

[PlayItem 交叉]

如参见图 29 和图 30 所述，开始在图 25 中的第一 PlayList#0 的第一 PlayItem#0 的播放，但是按照 PlayList#0，在第一 PlayItem#0 的播放结束时，

第二 PlayItem#1 的播放开始。即，执行 PlayItem 交叉，其中，从 PlayItem#0 向 PlayItem#1 执行交叉。

现在，参见图 31 中的流程图来描述 PlayItem 交叉的处理。

在如在图 29 和图 30 中所述开始在图 25 中的 PlayList#0 中的第一 PlayItem#0(的剪辑)的播放时，解码控制模块 214(图 2)在播放第一 PlayItem#0 的同时继续确认由内置的时钟单元 214A 计时的时间点。

“结束 PlayItem#0 的播放”

在由时钟单元 214A 计时的时间点变得等于 27,180,090(图 25)——它是在图 30 中的步骤 S127 中从播放器控制模块 212 提供的第一 PlayList#0 的 OUT_time——的时候，在步骤 S151 中，解码控制模块 214 执行解码中断处理，并且结束 PlayItem#0 的播放。

即，解码控制模块 214 操作视频解码器控制模块 216、音频解码器控制模块 217 和字幕解码器控制模块 218，以便停止解码操作。而且，解码控制模块 214 控制视频输出模块 220 继续输出当前正被输出的输出视频数据。

而且，解码控制模块 214 向播放器控制模块 212 发送关于已经结束第一 PlayItem#0 的播放的结果的消息。

“开始 PlayItem#1 的播放”

如上所述，播放器控制模块 212 在图 29 的步骤 S105 中识别在第一 PlayList#0 中存在第一 PlayItem#0 和第二 PlayItem#1，因此在从解码控制模块 214 向其发送关于已经结束第一 PlayItem#0 的播放的结果的消息时，流程从步骤 S151 进行到步骤 S152，并且以与上述的第一 PlayItem#0 相同的方式来开始第二 PlayItem#1 的播放。

即，为了给出第二 PlayItem#1 的播放规程的一般说明，首先，播放器控制模块 212 以与在图 30 中的步骤 S122 中的情况相同的方式，将在 EP_map() 中所述的 RPN_EP_start 之一确定为第二 PlayItem#1 的播放开始位置。

而且，播放器控制模块 212 进行如在图 30 中的步骤 S124 中所述的属于第二 PlayItem#1 的 Mark()的识别，并且识别在将要由 PlayItem#1 播放的剪辑流文件“00002.PS”上复用的每个属性的基本流的数量，以及进一步确定要播放（要被播放）的基本流，如在图 30 的步骤 S125 中所述。

播放器控制模块 212 然后执行与在图 30 中的步骤 S127 中的情况相同的处理。

即，播放器控制模块 212 向内容数据提供模块 213 提供被确定为播放开始位置的 EP_map()的 RPN_EP_start 和其上复用了要播放的基本流的剪辑流文件的文件名，即，在这种情况下，为对应于在第二 PlayItem#1(图 25)的 Clip_Information_file_name 中所述的“00002.CLP”的剪辑流文件“00002.PS”的文件名。

而且，播放器控制模块 212 在开始向缓冲器控制模块 215 提供在其上复用了要被播放的基本流的剪辑流文件“00002.PS”中存储的节目流之前，初始化缓冲器控制模块 215。

即，在缓冲器控制模块 215(图 3)，将相同的值代入在数据首部指针存储单元 231 中存储的数据首部指针、在数据写入指针存储单元 232 中存储的数据写入指针、在视频读取指针存储单元 241 中存储的视频读取指针、在音频读取指针存储单元 251 中存储的音频读取指针和在字幕读取指针存储单元 262 中存储的字幕读取指针中。

而且，播放器控制模块 212 向缓冲器控制模块 215 提供用作为用于识别要播放的基本流的标识信息的 stream_id，并且在必要时也提供 private_stream_id。

在缓冲器控制模块 215(图 3)，视频读取功能单元 233 在 stream_id 寄存器 242 中存储来自播放器控制模块 212 的要播放的基本流的视频流的 stream_id。而且，音频读取功能单元 234 存储来自播放器控制模块 212 的要播放的基本流的音频流的 stream_id 和 private_stream_id，并且存储在 stream_id 寄存器 252 和 private_stream_id 寄存器 253 中。

而且，因为字幕流被包含在其上复用了要播放的基本流的剪辑流文件“00002.PS”中，因此从播放器控制模块 212 向字幕读取功能单元 235 提供要播放的基本流的字幕流的 stream_id 和 private_stream_id，并且字幕读取功能单元 235 分别在 stream_id 寄存器 263 和 private_stream_id 寄存器 264 中存储 stream_id 和 private_stream_id。

作为缓冲器控制模块 215(图 3)的初始化，播放器控制模块 212 还向字幕读取功能标记存储单元 261 设置对应于在其上复用了要播放的基本流的剪辑流文件的字幕读取功能标记的值。

即，在这种情况下，因为字幕流被包含在其上复用了要播放的基本流的剪辑流文件“00002.PS”中，因此向字幕读取功能标记存储单元 261 设置具

有值 1 的字幕读取功能标记，以便使得字幕读取功能单元 235 起作用。

而且，播放器控制模块 212 向解码控制模块 214 提供要播放的第二 PlayItem#1(图 25)的 IN_time 90,000 和 OUT_time 27,090,000。

而且，播放器控制模块 212 初始化对于图形处理模块 219 的字幕流的显示格式的指令。即，播放器控制模块 212 控制图形处理模块 219 以便使用用于字幕流显示格式的默认显示格式。

注意，进行布置，其中，在要播放的字幕流的 configurable_flag(图 12)是用于允许改变显示格式的 1 的情况下，从播放器控制模块 212 到图形处理模块 219 的字幕流的显示格式的指令可以保持当前的显示格式。

随后，以与第一 PlayItem#0 的播放相同的方式来执行第二 PlayItem#1 的播放。在第二 PlayItem#1 的播放期间，解码控制模块 214 继续确认由内置的时钟单元 214A 计时的时间点，并且在由时钟单元 214A 计时的时间点变成等于 27,090,000(图 25)——它是在步骤 S152(图 31)中从播放器控制模块 212 提供的第二 PlayList#1 的 OUT_time——的时候，以与在步骤 S151 中相同的方式，解码控制模块 214 执行解码中断处理，并且结束 PlayItem#1 的播放。

[时间代码的显示]

接着，在图 30 的步骤 S123 中，开始时间代码的显示，但是，依序更新时间代码的显示。

现在，将参见图 32 中的流程图来描述用于时间代码显示的处理。

在内置的时钟单元 214A 计时 1 秒后，在步骤 S171 中，解码控制模块 214(图 2)向播放器控制模块 212 提供关于已经过去一秒的结果的消息以及由时钟单元 214A 计时的当前时间点，并且流程进行到步骤 S172。在步骤 S172，播放器控制模块 212 从解码控制模块 214 接收所述消息和所述当前时间点，将所述当前时间点转换为时间代码，并且流程进行到步骤 S173。

在步骤 S173，播放器控制模块 212 控制图形处理模块 219 以便显示在步骤 S172 中获得的时间代码，并且流程返回到步骤 S171。

因此，每秒更新所述时间代码。注意，在时间代码更新之间的时间间隔不限于一秒。

[流转换]

接着，如在图 26 中所述，两个字幕流在由第一 PlayItem#0 播放的剪辑流文件“00001.PS”和由第二 PlayItem#1 播放的剪辑流文件“00002.PS”上

被复用，它们构成在参见图 25 所述的第一 PlayList#0。

因此，在剪辑流文件上复用相同属性的多个基本流的情况下，可以从所述多个基本流的一个向相同属性的另一个基本流转换要播放的基本流。

现在，将参见图 33 中所示的流程图来说明流转换处理。

在将流转换指令描述为在“SCRIPT/DAT”文件(图 4)中的脚本程序的情况下由执行脚本程序的脚本控制模块 211 或者由操作例如遥控器的用户向播放器控制模块 212 给出流转换请求。

即，在执行其中描述了流转换指令的脚本程序时，脚本控制模块 211 向播放器控制模块 212 提供请求流转换的消息。而且，在通过操作遥控器的用户从遥控器接收到单个指令流转换时，输入接口 115 向播放器控制模块 212 提供请求流转换的消息。

例如，如果我们现在说已经向播放器控制模块 212 提供了作为用于请求字幕流的转换的消息的字幕流转换消息，则在步骤 S191 中，播放器控制模块 212 查看在图 30 中的步骤 S125 中执行的、在确定要播放的基本流时识别的字幕流的数量。

在作为查看结果而发现字幕流的数量是一个或更少的情况下，播放器控制模块 212 忽略所述字幕流转换消息，因此，不执行步骤 S192 到 S194 的随后的处理。

另一方面，在字幕流的数量是 2 个或更多的情况下，流程依序进行通过步骤 S192 到 S194，并且要播放的字幕流从当前正在播放的字幕流向另一个字幕流转换。

即，在步骤 S192 中，播放器控制模块 212 在剪辑信息文件上确定当前正在播放的字幕流。具体地，如果我们说已经在剪辑流文件“00002.PS”上被复用的、具有 0xBD 的 stream_id 和 0x80 的 private_stream_id 的字幕流通过构成在图 25 中所述的第一 PlayList#0 的第二 PlayItem#1 当前正在被播放，则在步骤 S192 中，进行确定当前正在被播放的字幕流，在剪辑流文件“00002.PS”上复用的两个字幕流中，是在图 26 中的剪辑信息文件“00002.CLP”上的第三字幕流 stream#2。

流程然后进行到步骤 S193，在此，播放器控制模块 212 将在步骤 S192 中确定的字幕流的下一个字幕流识别(确定)为在剪辑信息文件上要下一个播放的字幕流。在图 26 中，剪辑信息文件“00002.CLP”上的来自第三字幕流

stream#2 的下一个字幕流是第四字幕流 stream#3，因此在步骤 S193 中，将第四字幕流 stream#3 识别为下一个要播放的字幕流。

注意，在确定当前当前正在播放的字幕流在剪辑流文件“00002.PS”上复用的两个字幕流中是在图 26 中的剪辑信息文件“00002.CLP”上的第四字幕流 stream#3 的情况下，例如，将第三字幕流 stream#2 识别为下一个要播放的字幕流。

随后，流程进行到步骤 S194，在此，播放器控制模块 212 向缓冲器控制模块 215(图 3)的字幕读取功能单元 235 提供已经在步骤 S193 中识别的下一个要播放的字幕流的 stream_id 和 private_stream_id，并且给出指令以将这些 stream_id 和 private_stream_id 用于从缓冲器 215A 对字幕访问单元的下一个和随后的读出。

在缓冲器控制模块 215(图 3)的字幕读取功能单元 235，在步骤 S194 中，分别在 stream_id 寄存器 263 和 private_stream_id 寄存器 264 中新设置从播放器控制模块 212 提供的 stream_id 和 private_stream_id，并且，对于由在 stream_id 寄存器 263 和 private_stream_id 寄存器 264 中新设置的 stream_id 和 private_stream_id 确定的字幕访问单元，执行从其上的缓冲器 215A 的读出。

因此，从当前被播放的字幕流向另一个字幕流转换要播放的字幕流。

[在缓冲器控制模块 215 的处理]

接着，将参见图 34 到图 38 来说明在缓冲器控制模块 215(图 3)的处理，即向缓冲器 215A 写入数据和从缓冲器 215A 读出数据。

缓冲器控制模块 215 具有 5 个指针，用于向和从缓冲器 215A 中读/写数据，如参见图 3 所述。

即，如图 34 和图 35 中所示，缓冲器控制模块 215 具有在数据首部指针存储单元 231 中存储的数据首部指针、在数据写入指针存储单元 232 中存储的数据写入指针、在视频读取指针存储单元 241 中存储的视频读取指针、在音频读取指针存储单元 251 中存储的音频读取指针和在字幕读取指针存储单元 262 中存储的字幕读取指针。

注意，图 34 和图 35 省略了在图 3 中的视频读取功能单元 233 的 stream_id 寄存器 242 和 au_information()寄存器 243、音频读取功能单元 234 的 stream_id 寄存器 252 和 private_stream_id 寄存器 253 以及在字幕读取功能单元 235 中的字幕读取功能标记存储单元 261、stream_id 寄存器 263 和 private_stream_id

寄存器 264。

在数据首部指针存储单元 231 中存储的数据首部指针表示在缓冲器 215A 中剩余的最老数据(需要被读出并且还没有被读出的数据的最老数据)的位置。在数据写入指针存储单元 232 中存储的数据写入指针指示向缓冲器 215A 的数据的写入位置，并且这个位置是在缓冲器 215A 中的最新数据被写入的位置。

在视频读取指针存储单元 241 中存储的视频读取指针表示要从缓冲器 215A 读出的视频流的位置。而且，在音频读取指针存储单元 251 中存储的音频读取指针表示要从缓冲器 215A 读出的音频流的位置，在字幕读取指针存储单元 262 中存储的字幕读取指针表示要从缓冲器 215A 读出的字幕流的位置。

如在图 3 中所示，所述数据首部指针、数据写入指针、视频读取指针、音频读取指针和字幕读取指针都在顺时针方向上围绕缓冲器 215A 移动。

而且，对于本实施例，总是更新数据首部指针以便指向与视频读取指针、音频读取指针和字幕读取指针中指向所述最老数据的那个相同的位置，如图 35 中所示。现在，在图 35 中，在视频读取指针、音频读取指针和字幕读取指针中，音频读取指针指向最老数据的位置，因此数据首部指针匹配音频读取指针。

使用具有这样的数据首部指针、数据写入指针、视频读取指针、音频读取指针和字幕读取指针的缓冲器控制模块 215，在新数据被从盘 101 读出并且被写入到缓冲器 215A 时，在顺时针方向更新所述数据写入指针，以便指向在紧接已经被写入的新数据之后的位置。

而且，在从缓冲器 215A 读出视频流、音频流或者字幕流后，视频读取指针、音频读取指针或字幕读取指针在顺时针方向上被更新对应于所读出的数量的数量。在此，对应于所读出的数量的数量是对应于实际被读出的视频、音频或者字幕数据的部分和在所读出数据之间包括的并且已经在读取中被跳过的其他流的数据部分的总和。

而且，在更新了视频读取指针、音频读取指针或者字幕读取指针时，更新数据首部指针以便指向与视频读取指针、音频读取指针和字幕读取指针的、指向最老数据的位置的那个相同的位置。

现在，关于向缓冲器 215A 的数据的写入，缓冲器控制模块 215 控制向

缓冲器 215A 的数据的写入，以便数据写入指针不超过数据首部指针。

即，只要数据写入指针不超过数据首部指针，则使用缓冲器控制模块 215，从盘 101 读出的数据被写入到由数据写入指针指示的缓冲器 215A 的位置，并且更新数据写入指针。另一方面，在数据首部指针就要被数据写入指针超过的情况下，在缓冲器控制模块 215A，向内容数据提供模块 213 请求停止(中断)从盘 101 读出数据，并且进一步停止向缓冲器 215A 的数据的写入。因此，可以防止缓冲器 215A 的溢出。

如上所述，仅仅通过数据首部指针和数据写入指针这两个指针的位置关系来控制向缓冲器 215A 写入从盘 101 读出的数据。

另一方面，对于从缓冲器 215A 的数据的读出，缓冲器控制模块 215 控制从缓冲器 215A 的数据的读出，以便视频读取指针、音频读取指针或者字幕读取指针并且因此数据首部指针不超过数据写入指针。

即，只要视频读取指针、音频读取指针或者字幕读取指针不超过数据写入指针，则在缓冲器控制模块 215，响应于来自视频解码器控制模块 216、音频解码器控制模块 217 或者字幕解码器控制模块 218 的请求，从由视频读取指针、音频读取指针或者字幕读取指针指向的缓冲器 215A 位置读出数据，更新视频读取指针、音频读取指针或者字幕读取指针，并且也在必要时更新数据首部指针。另一方面，在视频读取指针、音频读取指针或者字幕读取指针就要超过数据写入指针的情况下，在缓冲器控制模块 215，来自视频解码器控制模块 216、音频解码器控制模块 217 和字幕解码器控制模块 218 的请求例如被冻结，并且等待直到准备了足够的数据。因此，可以防止缓冲器 215A 的下溢。

因此，要被提供到视频解码器控制模块 216、音频解码器控制模块 217 和字幕解码器控制模块 218 的数据被存储在缓冲器 215A 中的从数据首部指针指向的位置开始、在顺时针方向上到数据写入指针指向位置的范围上(在图 34 和图 35 中的阴影部分)，并且视频读取指针、音频读取指针和字幕读取指针存在于那个范围中。

注意在上述的情况下，数据首部指针被更新，以便从视频读取指针、音频读取指针和字幕读取指针指向的位置指向最老数据的位置，但是，可以进行另一种布置，其中，执行数据首部指针的更新以便指向从最老数据的位置间隔预定时间量(例如 1 秒)的过去的数据位置。

即，可以假设，一般，在视频读取指针、音频读取指针和字幕读取指针中，视频读取指针或者音频读取指针最常指向最老数据的位置。

因此，在更新数据首部指针以便指向在过去的数据位置的情况下，例如从由视频读取指针或者音频读取指针指向的最老数据的位置开始 1 秒的情况下，从由视频读取指针或者音频读取指针指向的最老数据位置开始过去的一秒的数据可以被剩余在缓冲器 215A 中，如图 34 中所示。在此，在图 34 中，音频读取指针正在指向最老数据的位置，并且数据首部指针正在指向从那个位置开始过去一秒的数据的位置。

如上所述，可以通过更新数据首部指针以便指向过去一秒的数据的位置来改善盘装置的响应。

即，在更新数据首部指针以便指向音频读取指针指向的最老数据的位置的情况下，如图 35 中所示，如果例如在逆向指令特殊的重放，则需要从盘 101 再一次读出其从缓冲器 215 的读出已经结束的数据，因此，从指令特殊重放到使能特殊重放，需要某个时间量。

相反，在更新数据首部指针以便指向从音频读取指针指向的最老数据的位置开始过去一秒的数据的位置的情况下，如图 34 中所示，如果在逆向方向指令特殊重放并且用于开始所述特殊重放所需要的数据是被存储在缓冲器 215A 中的过去一秒的数据，则如上所述需要再一次从盘 101 读出数据，并且可以立即开始特殊重放。

注意，即使在更新数据首部指针以便指向从音频读取指针指向的最老数据的位置开始过去一秒的数据的位置的情况下，可能有这样的情况：其中，用于开始特殊重放所需要的数据未被存储在缓冲器 215A 中。在这种情况下，再一次从盘 101 读出用于开始所述特殊重放所需要的数据。

接着，将说明从缓冲器 215A 读出视频流、音频流和字幕流的每一个的细节。

如在图 30 中的步骤 S127 中所述，在开始播放剪辑流文件时，在缓冲器控制模块 215，数据首部指针、数据写入指针、视频读取指针、音频读取指针和字幕读取指针都被初始化以便指向在缓冲器 215A 上的同一位置。

然后从盘 101 读出在剪辑流文件中存储的节目流(MPEG2 系统节目流)，并且在将其提供到缓冲器控制模块 215 后，在缓冲器控制模块 215，将节目流存储在缓冲器 215A 的数据写入指针指向的位置，并且在顺时针方向上更

新数据写入指针。

而且，在缓冲器控制模块 215(图 3)，视频读取功能单元 233 对于在缓冲器 215A 中存储的节目流执行语法分析，响应于来自视频解码器控制模块 216 的请求而从在缓冲器 215A 中存储的节目流提取(分离)和读出视频流(视频访问单元)，并且将其提供到视频解码器控制模块 216。

以相同的方式，音频读取功能单元 234 也对于在缓冲器 215A 中存储的节目流执行语法分析，响应于来自音频解码器控制模块 217 的请求而从在缓冲器 215A 中存储的节目流提取(分离)和读出音频流(音频访问单元)，并且将其提供到音频解码器控制模块 217。字幕读取功能单元 235 也对于在缓冲器 215A 中存储的节目流执行语法分析，响应于来自字幕解码器控制模块 218 的请求而从在缓冲器 215A 中存储的节目流提取(分离)和读出字幕流(字幕访问单元)，并且将其提供到字幕解码器控制模块 218。

“读出视频流”

接着，将参见在图 36 中的流程图来说明由视频读取功能单元 233(图 3)从缓冲器 215A 的视频流读出处理的细节。

在步骤 S211，视频读取功能单元 233 首先在缓冲器 215A 中存储的节目流中搜索和找到 private_stream_2 的 PES_packet()。即，private_stream_2 的 PES_packet()的 stream_id 是参见图 20 所述的 10111111B(= 0xBF)，因此视频读取功能单元 233 搜索和找到其 stream_id 是 10111111B 的 PES_packet()。

现在，如果我们说在剪辑流文件“00001.PS”中存储的节目流上复用的基本流是要播放的基本流，如上所述，例如在从盘 101 读出节目流并且将其存储在缓冲器 215A 时，在图 3 的步骤 S122，将扇区 305 确定为来自在剪辑流文件“00001.PS”的 EP_map()(图 27)中所述的解码开始允许点的信息的播放开始位置，并且进一步在图 30 的步骤 S128 中，指定作为播放开始位置的扇区 305，并且向操作系统 201 进行对于在剪辑流文件“00001.PS”中存储的节目流的读取请求。

而且，对于视频流，在 EP_map()中所述的解码开始允许点信息表示位于紧接在实际解码开始允许点之前的 private_stream_2 的 PES_packet()的位置。

因此，紧接在剪辑流文件“00001.PS”中存储的节目流被从盘 101 读出并且被存储在缓冲器 215A 之后，private_stream_2 的 PES_packet()被存储在数据首部指针和视频读取指针所指向的缓冲器 215A 的位置中。

在步骤 S211 中找到 private_stream_2 的 PES_packet()时，流程进行到步骤 S212，在此，在图 30 的步骤 S127 中，视频读取功能单元 233 提取作为 private_stream_2 的 PES_packet() 的 PES_packet_data_byte 的 private_stream2_PES_payload()(图 23)中所述的 video_stream_id，并且确定是否 video_stream_id 匹配在 stream_id 寄存器 242(图 3)中存储的要播放的视频流的 stream_id。

在步骤 S212 中确定在 private_stream2_PES_payload() 中所述的 video_stream_id 不匹配在 stream_id 寄存器 242 中存储的 stream_id 的情况下，即在紧前的步骤 S211 中找到的 private_stream_2 的 PES_packet()不位于要播放的视频流的解码开始点的情况下，流程返回到步骤 S211，在缓冲器 215A 中存储的节目流中搜索 private_stream_2 的另一个 PES_packet()，随后重复相同的处理。

另一方面，在步骤 S212 中确定在 private_stream2_PES_payload()中所述的 video_stream_id 匹配在 stream_id 寄存器 242 中存储的 stream_id 的情况下，即在紧前的步骤 S211 中找到的 private_stream_2 的 PES_packet()位于要播放的视频流的解码开始点的情况下，流程进行到步骤 S213，其中，视频读取功能单元 233 从缓冲器 215A 读出在 private_stream_2 的 PES_packet()的 private_stream2_PES_payload() 中所述的 au_information()，将其存储在 au_information()寄存器 243(图 3)中，并且流程进行到步骤 S214。

在步骤 S214 中，视频读取功能单元 233 将在视频读取指针存储单元 231 中存储的视频读取指针更新等同于在紧前的步骤 211 中找到的 private_stream_2 的 PES_packet()的大小的数量(其 video_stream_id(图 23)匹配在 stream_id 寄存器 242(图 3)中存储的 stream_id 的 private_stream_2 的 PES_packet())。

即，对于剪辑流文件，其 stream_id 匹配 private_stream_2 的 PES_packet() 的 video_stream_id 的视频流(PES_packet())位于紧后，因此，在步骤 S214 中更新视频读取指针，以便指向视频流的实际解码开始允许点。

随后，流程从步骤 S214 进行到 S215，在此，视频读取功能单元 233 确定是否已经有来自视频解码器控制模块 216 的对于数据的请求，并且在确定还没有请求的情况下，流程返回到步骤 S215，并且重复相同的处理。

而且，在步骤 S215 中确定有来自视频解码器控制模块 216 的对于数据的

请求的情况下，流程进行到步骤 S216，在此，视频读取功能单元 233 从缓冲器 215A 读出在 au_information()寄存器 243 中存储的 au_information()的 AU_length 中所描述的字节数量的数据，即一个视频访问单元，同时从视频读取指针指向的在缓冲器 215A 中的位置开始执行节目流的语法分析，并且向视频解码器控制模块 216 提供所述视频访问单元，同时将视频读取指针更新等同于从缓冲器 215A 读出的一个视频访问单元大小的数量。

即，如参见图 24 所述，在 au_information() 中描述了 number_of_access_unit，它表示从其所属的 private_stream_2 的 PES_packet()到下一个 private_stream_2 的 PES_packet()包括的视频访问单元(画面)的数量。

在 au_information()中描述了 pict_struct_copy、au_ref_flag 和 AU_length，它们是与按照参见图 24 所述的 number_of_access_unit 的数量的视频访问单元的每一个相关联的信息。

在按照 number_of_access_unit 的数量的 au_information()中描述的每个 AU_length 表示从其所属的 private_stream_2 的 PES_packet()到下一个 private_stream_2 的 PES_packet()包括的 number_of_access_unit 的数量的视频访问单元的每一个的大小，如参见图 24 所述，因此，视频读取功能单元 233 可以通过使用 AU_length 来提取访问单元，而不必执行视频流的语法分析。

即，在提取 MPEG2-Video 或 MPEG4-AVC 的访问单元的情况下，传统上，需要知道视频流的语法，并且根据其来执行视频流的语法分析，但是，在盘 101 中记录的剪辑流文件中存储的节目流包含 private_stream_2 的 PES_packet()，其中描述了用于表示视频访问单元的大小的 AU_length，它以视频访问单元的递增紧接在视频流中的一个或多个实际解码开始允许点的每一个之前，因此，视频读取功能单元 233 可以基于在 private_stream_2 的 PES_packet()中描述的 AU_length 从缓冲器 215A 读出视频访问单元(的递增的视频流)，并且提供到视频解码器控制模块 216，而不执行视频流的语法分析。

注意，当在步骤 S216 中向视频解码器控制模块 216 提供视频访问单元时，视频读取功能单元 233 也向视频解码器控制模块 216 提供在 au_information()中被描述为与视频访问单元相关联的信息的 pict_struct_copy、au_ref_flag 和 AU_length 以及以视频访问单元的递增而增加的时间标记(PTS，DTS)。

当在步骤 S216 中单个视频访问单元被从缓冲器 215A 读出并且被提供到视频解码器控制模块 216 之后，流程进行到步骤 S217，在此，视频读取功能

单元 233 确定是否已经处理了由在 au_information()寄存器 243 中存储的 au_information()(图 24)中的 number_of_access_unit 表示的数量的访问单元。

当在步骤S217中确定还没有处理由number_of_access_unit表示的数量的访问单元的情况下，即在由 number_of_access_unit 表示的数量的访问单元还没有被从缓冲器 215A 读出并且提供到视频解码器控制模块 216 的情况下，流程返回到步骤 S215，并且随后重复相同的处理。

而且，在步骤 S217 中确定已经处理了由 number_of_access_unit 表示的数量的访问单元的情况下，即在由 number_of_access_unit 表示的数量的访问单元已经被从缓冲器 215A 读出并且提供到视频解码器控制模块 216 的情况下，流程返回到步骤 S211，并且搜索下一个 private_stream_2 的 PES_packet()，随后重复相同的处理。

“读出音频流”

接着，将参见在图 37 中的流程图来说明由音频读取功能单元 234(图 3)从缓冲器 215A 的音频流读出处理的细节。

首先，在步骤 S230 中，音频读取功能单元 234 确定是否在图 30 中的步骤 S127 中在 stream_id 寄存器 252(图 3)中存储的、要播放的音频流的 stream_id 表示 private_stream_1 的 PES_packet()。

在步骤 S230 中确定在 stream_id 寄存器 252 中存储的 stream_id 不表示 private_stream_1 的 PES_packet()的情况下，即，在 stream_id 寄存器 252 中存储的 stream_id 是被分配到按照如参见图 20 所述的 MPEG 标准而编码的音频流的 110xxxxxB 的情况下，流程进行到步骤 S231，并且音频读取功能单元 234 从在缓冲器 215A 中存储的节目流搜索同步代码，用于表示由 MPEG 音频规定的音频帧的首部。同步代码的位置是音频帧的首部，因此，音频读取功能单元 234 更新音频读取指针，以便指示音频帧的首部的位置，并且流程从步骤 S231 进行到 S232。在步骤 S232 中，音频读取功能单元 234 从在缓冲器 215A 中存储的节目流搜索和找到匹配在音频读取指针指示的位置上的在 stream_id 寄存器 252 中存储的 stream_id 的 PES_packet()，并且流程进行到步骤 S233。

在步骤 S233，音频读取功能单元 234 更新在音频读取指针存储单元 251 中存储的音频读取指针，以便指向在紧前的步骤 S232 中找到的 PES_packet()的 PES_packet_data_byte(图 16 到 18)的首部，并且流程进行到步骤 S237。

在步骤 S237 中，音频读取功能单元 234 确定是否已经有来自音频解码器

控制模块 217 的对于数据的请求，并且在没有的情况下，流程返回到步骤 S237，并且重复相同的处理。

而且，在步骤 S237 中确定有来自音频解码器控制模块 217 的对于数据的请求时，流程进行到步骤 S238，在此，音频读取功能单元 234 从缓冲器 215A 读出已知固定长度的单个音频访问单元，同时从所述音频读取指针指向的在缓冲器 215A 中的位置执行节目流的语法分析，并且向音频解码器控制模块 217 提供所述音频访问单元以及附加其上的时间标记(PTS，DTS)。

音频读取功能单元 234 然后将音频读取指针更新等同于从缓冲器 215A 读出的一个音频访问单元的大小的数量，流程返回到步骤 S237，随后重复相同的处理。

另一方面，在步骤 S230 中确定在 stream_id 寄存器 252 中存储的 stream_id 表示 private_stream_1 的 PES_packet()的情况下，即，在 stream_id 寄存器 252 中存储的 stream_id 是 10111101B(= 0xBD)、并且表示在图 20 中所述的 private_stream_1 的 PES_packet()的情况下，流程进行到步骤 S234，在此，音频读取功能单元 234 搜索和找到在缓冲器 215A 中存储的节目流内的 private_stream_1 的 PES_packet()。即，音频读取功能单元 234 搜索和找到 stream_id 是 10111101B 的 PES_packet()。

在已经在步骤 S234 中找到 private_stream_1 的 PES_packet()的情况下，流程进行到步骤 S235，并且音频读取功能单元 234 提取作为 private_stream_1 的 PES_packet()的 PES_packet_data_byte 的 private_stream1_PES_payload()(图 21)中所述的 private_stream_id，并且确定是否 private_stream_id 匹配在图 30 中的步骤 S127 中的 private_stream_id 寄存器 253(图 3)中存储的要播放的音频流的 private_stream_id。

在步骤 S235 中确定在 private_stream1_PES_payload() 中所述的 private_stream_id 不匹配在 private_stream_id 寄存器 253 中存储的 private_stream_id 的情况下，即在紧前的步骤 S234 中找到的 private_stream_1 的 PES_packet()不是要播放的音频流的情况下，流程返回到步骤 S234，在缓冲器 215A 中存储的节目流中搜索 private_stream_1 的另一个 PES_packet()，随后重复相同的处理。

另一方面，在步骤 S235 中确定在 private_stream1_PES_payload()中所述的 private_stream_id 匹配在 private_stream_id 寄存器 253 中存储的

private_stream_id 的情况下，即在紧前的步骤 S234 中找到的 private_stream_1 的 PES_packet()是要播放的音频流的情况下，流程进行到步骤 S236，在此，音频读取功能单元 234 从缓冲器 215A 读出在 private_stream_1 的 PES_packet()的 private_stream1_PES_payload()中所述的 AU_locator()，并且向 AU_locator()表示的值加上在紧接 AU_locator()之后的位置，由此获得音频访问单元的首部位置。

即，可以通过向 AU_locator()表示的值加上紧随 AU_locator()的位置而获得音频访问单元的(绝对)首部位置，因为如图 21 中所述，以紧随 AU_locator()的位置作为基准，AU_locator 表示在 private_stream1_PES_payload()的 private_payload()中存储的音频访问单元(或者字幕访问单元)的首部位置。

而且，在步骤 S236 中，音频读取功能单元 234 更新在音频读取指针存储单元 251 中存储的音频读取指针，以便指向如上所述获得的音频访问单元的首部位置，流程进行到步骤 S237。

在步骤 S237 中，音频读取功能单元 234 确定是否有来自音频解码器控制模块 217 的对于数据的请求，并且在确定没有的情况下，流程返回到步骤 S237，并且重复相同的处理。

而且，在步骤 S237 中确定已经有来自音频解码器控制模块 217 的对于数据的请求的情况下，流程进行到步骤 S238，在此，音频读取功能单元 234 从缓冲器 215A 读出已知固定长度的单个音频访问单元，同时执行来自音频读取指针所指向的缓冲器 215A 中的位置的节目流的语法分析，并且向音频解码器控制模块 217 提供所述音频访问单元以及附加到其上的时间标记。

音频读取功能单元 234 然后将音频读取指针更新等同于从缓冲器 215A 读出的一个音频访问单元的大小的数量，流程返回到步骤 S237，随后重复相同的处理。

“读出字幕流”

接着，将参见在图 36 中的流程图来说明由字幕读取功能单元 235(图 3)从缓冲器 215A 的字幕流的读出处理的细节。

首先，在步骤 S251 中，字幕读取功能单元 235 确定在图 30 的步骤 S127 中在字幕读取功能标记存储单元 261 中存储的字幕读取功能标记。在步骤 S251 中确定字幕读取功能标记是 0 的情况下，即在其上要播放的基本流被复用的剪辑流文件中不包括字幕流、并且已经在图 30 的步骤 S127 中的字幕读

取功能标记存储单元 261 中例如设置了 0 的情况下，字幕读取功能单元 235 不具体执行任何处理。

另一方面，在步骤 S251 中确定字幕读取功能标记是 1 的情况下，即，在具有其上要播放的基本流被复用的剪辑流文件中包括的字幕流、并且已经在图 30 的步骤 S127 中的字幕读取功能标记存储单元 261 中例如设置了 1 的情况下，流程进行到步骤 S252，在此，字幕读取功能单元 235 从在缓冲器 215A 中存储的节目流搜索匹配在 stream_id 寄存器 263(图 3)中存储的要播放的字幕流的 stream_id 的 PES_packet()。

现在，如在图 30 中的步骤 S127 中所述，要播放的字幕流的 stream_id 被存储在 stream_id 寄存器 263(图 3)中，并且字幕流的 stream_id 如参见图 20 所述是 10111101B(= 0xBD)，它表示 private_stream_1 的 PES_packet()。

因此，在缓冲器 215A 中存储的节目流中的 private_stream_1 的 PES_packet()是在步骤 S252 中搜索的内容。

在已经在步骤 S252 中搜索了 private_stream_1 的 PES_packet()并且已经找到 private_stream_1 的 PES_packet()后，流程进行到步骤 S253，并且字幕读取功能单元 235 提取作为 private_stream_1 的 PES_packet() 的 PES_packet_data_byte 的 private_stream1_PES_payload()(图 21) 中所述的 private_stream_id，并且确定是否 private_stream_id 匹配在图 30 中的步骤 S127 中在 private_stream_id 寄存器 264(图 3) 中存储的要播放的字幕流的 private_stream_id。

在步骤 S253 中确定在 private_stream1_PES_payload() 中所述的 private_stream_id 不匹配在 private_stream_id 寄存器 264 中存储的 private_stream_id 的情况下，即在紧前的步骤 S252 中找到的 private_stream_1 的 PES_packet()不是要播放的字幕流的情况下，流程返回到步骤 S252，在缓冲器 215A 中存储的节目流内搜索另一个 private_stream_1 的 PES_packet()，随后重复相同的处理。

另一方面，在步骤 S253 中确定在 private_stream1_PES_payload()中所述的 private_stream_id 匹配在 private_stream_id 寄存器 264 中存储的 private_stream_id 的情况下，即在紧前的步骤 S252 中找到的 private_stream_1 的 PES_packet()是要播放的字幕流的情况下，流程进行到步骤 S254，在此，字幕读取功能单元235从缓冲器215A读出在private_stream_1的PES_packet()

的 private_stream1_PES_payload()中所述的 AU_locator(图 21)，并且向 AU_locator 表示的值加上紧随 AU_locator 的位置，由此获得字幕访问单元的首部位置。

即，可以通过向 AU_locator 表示的值加上紧随 AU_locator 的位置来获得字幕访问单元的(绝对)首部位置，因为如图 21 中所述，以紧随 AU_locator()的位置作为基准，AU_locator 表示在 private_stream1_PES_payload()的 private_payload()中存储的字幕访问单元(或者音频访问单元)的首部位置。

而且，在步骤 S54 中，字幕读取功能单元 235 更新在字幕读取指针存储单元 262 中存储的字幕读取指针，以便指向如上所述获得的字幕访问单元的首部位置，并且流程进行到步骤 S255。

在步骤 S255 中，字幕读取功能单元 235 确定是否有来自字幕解码器控制模块 218 的对于数据的请求，并且在确定没有的情况下，流程返回到步骤 S255，并且重复相同的处理。

而且，在步骤 S255 中确定已经有来自字幕解码器控制模块 218 的对于数据的请求的情况下，流程进行到步骤 S256，在此，字幕读取功能单元 235 从缓冲器 215A 读取用于描述在字幕访问单元的首部描述的大小的单个字幕访问单元，同时执行来自字幕读取指针所指向的缓冲器 215A 中的位置的节目流的语法分析，并且向字幕解码器控制模块 218 提供所述字幕访问单元以及附加到其上的时间标记。即，如图 2 中所述，在其首部描述了字幕访问单元的大小，因此，字幕读取功能单元 235 从字幕读取指针指向的缓冲器 215A 的位置读出那个大小的数据，并且向字幕解码器控制模块 218 提供作为已经被读出的数据的字幕访问单元以及在其上附加的时间标记。

字幕读取功能单元 235 然后将字幕读取指针更新等同于从缓冲器 215A 读出的所述那个字幕访问单元的大小的数量，流程返回到步骤 S255，随后重复同一处理。

“重新同步处理”

接着，将说明通过在图 2 中所示的解码控制模块 214 而进行的视频数据和音频解码器的同步处理。

如在图 30 中的 S130 中所述，解码控制模块 214 向视频解码器控制模块 216、音频解码器控制模块 217 和字幕解码器控制模块 218 指令解码开始，如果必要的话则用移位的解码开始以保证同步，但是存在这样的情况：其中，

依赖于例如在视频解码器 116 和音频解码器 117 的解码处理的进程，视频数据的输出和作为应当与该视频数据同步输出的输出数据的音频数据的输出之间有偏移。

因此，在解码控制模块 214 上执行重新同步处理，以校正在视频数据的输出和应当与视频数据同步地被输出的音频数据的输出之间产生的偏移，以便视频数据和音频数据同步地被输出。

将参见在图 39 中的流程图来描述重新同步处理。

对于重新同步处理，首先，在步骤 S271 中，解码控制模块 214 确定是否在来自视频解码器控制模块 216 的视频访问单元的时间标记和来自音频解码器控制模块 217 的音频访问单元的时间标记之间的偏移大。

即，如对于在图 30 中的步骤 S129 所述，每次从缓冲器控制模块 215 获得视频访问单元，视频解码器控制模块 216 就向解码控制模块 214 提供视频访问单元的时间标记。以相同的方式，每次从缓冲器控制模块 215 获得音频访问单元，音频解码器控制模块 217 就向解码控制模块 214 提供音频访问单元的时间标记。

在步骤 S271 中，解码控制模块 214 将从视频解码器控制模块 216 和音频解码器控制模块 217 同时(在可以被看作同一定时的某个时间内)提供的时间标记彼此相比较，并且确定是否在时间标记之间的偏移大。

在步骤 S271 中确定在来自视频解码器控制模块 216 的视频访问单元的时间标记和来自音频解码器控制模块 217 的音频访问单元的时间标记之间的偏移不大的情况下，即，在视频访问单元的时间标记和音频访问单元的时间标记之间的偏移处于在被认为已经实现预定同步的范围——例如两个视频帧(大约 66 毫秒)——内的情况下，流程返回到步骤 S271，并且继续确定(监控)在时间标记之间的偏移。

另一方面，在步骤 S271 中确定在来自视频解码器控制模块 216 的视频访问单元的时间标记和来自音频解码器控制模块 217 的音频访问单元的时间标记之间的偏移大的情况下，即，在视频访问单元的时间标记和音频访问单元的时间标记之间的偏移超过其中被认为已经实现预定同步的范围的情况下，流程进行到步骤 S272，其中，解码控制模块 214 将来自视频解码器控制模块 216 的视频访问单元的时间标记与来自音频解码器控制模块 217 的音频访问单元的时间标记相比较，由此确定视频数据的输出(解码)和音频数据的输出的

那个滞后。

在步骤 S272 中确定视频数据的输出滞后于音频数据的输出的情况下，流程进行到步骤 S273，在此，解码控制模块 214 向视频解码器控制模块 216 发送关于不对于单个视频访问单元执行视频访问单元的解码和输出(显示)之结果的指令，以便加速视频访问单元的处理，即输出用于跳过视频访问单元的处理的指令，并且流程进行到步骤 S274。

在步骤 S274 中，视频解码器控制模块 216 从解码控制模块 214 接收所述跳过指令，并且响应于所述跳过指令，查看从缓冲器控制模块 215 与视频访问单元一起提供的 au_ref_flag。

即，au_ref_flag 作为与访问单元相关联的信息被包括在位于 private_stream_2 的 PES_packet()的 private_stream2_PES_payload()图 23)上的 au_information()(图 24)，并且缓冲器控制模块 215 向视频解码器控制模块 216 与视频访问单元一起提供视频访问单元的 au_ref_flag，如对于在图 30 的步骤 S129 和在图 36 中的步骤 S216 所述的。

在步骤 S274 中，查看如此与访问单元一起被提供的访问单元的 au_ref_flag。

流程然后从步骤 S274 进行到 S275，在此，视频解码器控制模块 216 根据从缓冲器控制模块 215 提供的视频访问单元的 au_ref_flag 的查看结果，确定是否视频访问单元是在解码另一个画面时不参考的非基准图像。

现在，如参见图 24 所述，视频访问单元的 au_ref_flag 表示是否访问单元是基准图像，并且在这是基准图像的情况下被设置为 1，在这不是基准图像即是非基准图像的情况下，被设置为 0。

在步骤 S275 中确定从缓冲器控制模块 215 提供的视频访问单元不是非基准图像(的视频访问单元)的情况下，即在从缓冲器控制模块 215 提供的视频访问单元是基准图像的情况下，流程进行到步骤 S276，视频解码器控制模块 216 以通常的方式使用视频解码器 116 来处理视频访问单元，等待从缓冲器控制模块 215 提供的下一个视频访问单元，并且流程返回到步骤 S274。

而且，在步骤 S275 中确定从缓冲器控制模块 215 提供的视频访问单元是非基准图像的情况下，流程进行到步骤 S277，在此，视频解码器控制模块 216 跳过使用视频解码器 116 的视频访问单元的处理，等待从缓冲器控制模块 215 提供的下一个视频访问单元，并且流程返回到步骤 S271。

因此，通过已经跳过的视频访问单元的处理，视频访问单元的处理被加速大约一个视频访问单元(降低了处理时间)。因此，加快了已经滞后于音频数据的输出的视频数据的输出。

另一方面，在步骤 S272 中确定视频数据的输出不滞后于音频数据的输出的情况下，即在音频数据的输出滞后于视频数据的输出的情况下，流程进行到步骤 S278，在此，解码控制模块 214 向视频解码器控制模块 216 输出重复输出的指令，以重复输出对应于当前被解码的视频访问单元的视频数据，以便使得视频访问单元的处理等待，并且流程进行到步骤 S279。

在步骤 S279 中，视频解码器控制模块 216 从解码控制模块 214 接收所述重复输出指令，响应于所述重复输出指令而重复向图形处理模块 219 输出对应于在视频解码器控制模块 216 中当前被解码的视频访问单元的视频数据，等待从缓冲器控制模块 215 提供的下一个视频访问单元，并且流程返回到步骤 S271。

因此，解码控制模块 214 确定是否视频数据的输出滞后于音频数据的输出，并且在视频数据的输出滞后于音频数据的输出的情况下，指令视频解码器控制模块 216 跳过一个访问单元的处理。根据已经被指令跳过的访问单元的 au_ref_flag，视频解码器控制模块 216 确定是否访问单元是基准图像或者是非基准图像，并且在非基准图像的情况下，使得视频解码器 116 跳过已经被指令跳过的访问单元的处理。因此，可以容易地获得视频数据的输出和音频数据的输出的同步。

即，在要跳过其处理的访问单元是基准图像的情况下，对应于那个访问单元的视频数据需要被解码，以在解码以后要解码的另一个访问单元时参考。因此，在用于同步视频数据的输出和音频数据的输出的同步控制中跳过基准图像的访问单元的处理的情况下，不能解码参考所述基准图像的其他访问单元，因此，当显示视频数据时，将同步控制显示为噪声。

因此，优选的是，跳过不是基准图像的访问单元即非基准图像的访问单元的处理。

另一方面，为了搜索具有传统基本流的非基准图像访问单元，需要执行基本流的语法分析，但是，通过按照 MPEG4-AVC 编码而获得的基本流的语法极为复杂，因此语法分析代价很大。

相反，private_stream_2 的 PES_packet()与在盘 101 中记录的剪辑流文件

中存储的节目流复用，所述节目流与位于 PES_packet_data_byte 中的 PES_packet()(图 16 到 18)相分离，在 private_stream_2 的 PES_packet()中，配置了 private_stream2_PES_payload()(图 23)，它是 PES_packet_data_byte 的扩展，并且，对于在那个 private_stream2_PES_payload()的 au_information()(图 24)中的每个视频访问单元，描述了用于表示是否视频访问单元是基准图像或者非基准图像的 au_reference_flag。从缓冲器控制模块 215 向视频解码器控制模块 216 与对应的视频访问单元一起提供 au_ref_flag。因此，视频解码器控制模块 216 可以通过查看与视频访问单元一起被提供的视频访问单元的 au_ref_flag 来几乎没有代价地识别视频访问单元是基准图像或者是非基准图像。

[标记处理]

接着，将参见图 40 的流程图来说明基于在 PlayListMark()(图 7)中所述的 Mark()而执行的标记处理。

解码控制模块 214 总是确认由内置的时钟单元 214A 保持的当前时间点，并且在步骤 S301 确定是否当前的时间点匹配在 PlayListMark()(图 7)中所述的任何 Mark()的 mark_time_stamp。

即，如在图 30 的步骤 S124 中所述，播放器控制模块 212 在播放在图 25 中所示的第一 PlayList#0 的第一 PlayItem#0 时，在被表示到图 28 的上部的 PlayListMark()中包括的 7 个 Mark()中，识别第一到第四 Mark()属于 PlayList#0 的第一 PlayItem#0，并且向解码控制模块 214 传送所述四个 Mark()的 mark_time_stamp {180,090}、{5，580，090}、{10，980，090}、{16，380，090}以及关于 mark_time_stamp 表示的时间点的属性是“标记处理”的指示。

在步骤 S301 中，解码控制模块 214 确定是否当前的时间点匹配如上所述从播放器控制模块 212 提供的属性“标记处理”的时间点(mark_time_stamp)之一。

在步骤 S301 中，在确定当前的时间点不匹配属性“标记处理”的时间点的任何一个时，流程返回到步骤 S301，并且重复相同的处理。

而且，在步骤 S301 中确定当前的时间点匹配属性“标记处理”的时间点的任何一个时，解码控制模块 214 向播放器控制模块 212 提供关于当前的时间点是“标记处理”属性的时间点之结果的消息和当前的时间点已经匹配的“标记处理”属性的时间点，流程进行到步骤 S302。

在步骤 S302，播放器控制模块 212 从解码控制模块 214 接收关于当前的时间点是“标记处理”属性的时间点之结果的消息和当前的时间点已经匹配的“标记处理”属性的时间点（mark_time_stamp），并且将其 mark_time_stamp 已经匹配当前时间点的 Mark()识别为作为标记处理的对象的 Mark()(以下称为要处理的标记)。

即，播放器控制模块 212 识别当前被播放的 PlayList()的 PlayItem()，并且通过参考来自 PlayList()和 PlayItem()的“PLAYLIST.DAT”文件(图 5)的 PlayListMark()(图 7)和当前的时间点已经匹配的“标记处理”属性的时间点(mark_time_stamp)而识别要被处理的标记。

具体地，如果我们说例如当前正在播放在图 25 中所示的第一 PlayList#0 的第一 PlayItem#0，则这使得播放器控制模块 212 识别标记时间点是在图 28 的上侧示出的 PlayListMark()中包括的 7 个 Mark()的第一到第四这四个 Mark()之一的 mark_time_stamp。

现在，如果我们说从解码控制模块 214 向播放器控制模块 212 提供的标记时间点是例如 16,380,090，则播放器控制模块 212 将在图 28 的上部示出的、PlayListMark()中包括的第一到第四个 Mark()中的第四 Mark()——其 mark_time_stamp 匹配标记时间点 16,380,090——识别为要处理的标记。

在播放器控制模块 212 如上所述识别要处理的标记后，流程从步骤 S302 进行到 S303，并且确定是否在要处理的标记中描述了用于确定基本流的 entry_ES_stream_id 和 entry_ES_private_stream_id(图 7).

在步骤 S303 中确定在要处理的标记中未描述用于确定基本流的 entry_ES_stream_id 和 entry_ES_private_stream_id(图 7)的情况下，即在 entry_ES_stream_id 和 entry_ES_private_stream_id 都是 0x00 的情况下，跳过步骤 S304，并且流程进行到步骤 S305，随后执行对应于要处理的标记的处理。

而且，在步骤 S303 中确定在要处理的标记中描述了用于确定基本流的 entry_ES_stream_id 和 entry_ES_private_stream_id(图 7)的情况下，流程进行到步骤 S304，在此，播放器控制模块 212 确定是否被播放的基本流包括由 entry_ES_stream_id——如果必要的话由 entry_ES_private_stream_id——确定的基本流。

在步骤 S304 中确定正在被播放的基本流不包括由 entry_ES_stream_id 和 entry_ES_private_stream_id 确定的基本流的情况下，流程返回到步骤 S301。

即，在没有正在播放由要被处理的标记的 entry_ES_stream_id 和 entry_ES_private_stream_id 确定的基本流的情况下，忽略要被处理的标记。

另一方面，在步骤 S304 中确定正在被播放的基本流包括由 entry_ES_stream_id 和 entry_ES_private_stream_id 确定的基本流的情况下，即在正在播放由要被处理的标记的 entry_ES_stream_id 和 entry_ES_private_stream_id 确定的基本流的情况下，将要处理的标记被认为是有效的，流程进行到步骤 S305，并且随后执行对应于要处理的标记的处理。

即，在步骤 S305，播放器控制模块 212 通过参考要被处理的标记的 mark_type(图 7)来确定要处理的标记。

在步骤 S305，在将要处理的标记被确定为章节标记或者索引标记的情况下，即在要处理的标记的 mark_type 是“章节”或者“索引”的情况下，流程进行到步骤 S306，在此，播放器控制模块 212 向图形处理模块 219 发出命令以将章节或者索引编号的显示更新为其章节标记或者索引标记是要处理标记的的章节或者索引编号，并且流程返回到步骤 S301。

而且，在步骤 S305 中将要处理的标记已经被确定为事件标记的情况下，即在要处理的标记的 mark_type 是“事件”的情况下，流程进行到步骤 S307，播放器控制模块 212 向脚本控制模块 211 通知(提供)用于表示事件发生的事件消息和要处理的标记的 mark_data，并且流程进行到步骤 S308。

在步骤 S308，脚本控制模块 211 从播放器控制模块 212 接收事件消息和 mark_data，并且使用所述事件消息作为中断消息，以 mark_data 作为变元来预先执行在“SCRIPT.DAT”文件中所描述的系列处理，并且流程返回到步骤 S301。

即，脚本控制模块 211 执行对应于 mark_data 的处理。

具体地，例如，对于被示出到图 28 中的下部的 PlayList#1 中的 PlayListMark()，第二 Mark()(Mark#1)和第三 Mark()(Mark#2)具有作为 mark_type 的“事件”，但是 mark_data 在 1(Mark#1)和 2(Mark#2)之间不同。

在接收对应于第二 Mark()的事件消息并且接收对应于第三 Mark()的事件消息的任何一种情况下，脚本控制模块 211 使用同一事件处理器(中断处理例程)来执行对应于事件消息的处理。但是在所述事件处理器内，通过查看被提供有事件消息的 mark_data 来对于事件消息执行对于每个 mark_data 不同的处理。

具体地，例如，在 mark_data 是 1 的情况下，脚本控制模块 211 控制图形处理模块 219 以显示第一类型的图标，而且，例如，在 mark_data 是 2 的情况下，脚本控制模块 211 控制图形处理模块 219 以显示第二类型的图标。

注意，mark_data 不限于 1 或者 2，而且注意，对应于 mark_data 而执行的处理不限于诸如如上所述的简单图标显示。

即，例如，在 mark_data 具有在 3 到 18 的范围内的值的情况下，脚本控制模块 211 控制图形处理模块 219 来显示第一类型的图标，它具有对应于通过从 mark_data 减去 2 而获得的值(1 到 16 的数值)的亮度。而且，例如，在 mark_data 具有在 19 到 34 范围内的值的情况下，脚本控制模块 211 控制图形处理模块 219 来显示第二类型的图标，它具有对应于通过从 mark_data 减去 18 而获得的值(1 到 16 的数值)的亮度。

可以进行另一种布置，其中，例如，用户操作的控制器连接到输入接口 115(图 1)，并且控制器内置了振动电机，其中，偏心的重量被附加到 DC(直流)电机的轴，它在操作所述 DC 电机时产生振动，并且在 mark_data 具有在 35 到 42 范围中的值的情况下，通过对应于通过从 mark_data 减去 34 而获得的值(1 到 8 的数值)的持续时间来操作所述振动电机。

所述 mark_data 是数值，并且可以通过由脚本控制模块 211 执行的脚本程序来描述使用和算法。因此，可以在预定规则下使用 mark_data，或者可以在由盘 101 的制造商、提供要被记录在盘 101 中的数据的内容提供商等独立地设置的多个规则下使用 mark_data。

如上所述，对于标记处理，在当前时间点匹配属性“标记处理”的时间点时，从作为属性“标记处理”的时间点的标记时间点识别要处理的标记。而且，在要处理的标记中没有描述用于确定基本流的的 entry_ES_stream_id 或者 entry_ES_private_stream_id 的情况下，执行对应于要处理的标记的 mark_type 的处理。而且，即使在要处理的标记中描述了用于确定基本流的的 entry_ES_stream_id 或者 entry_ES_private_stream_id 的情况下，如果当前正在播放由 entry_ES_stream_id 和 entry_ES_private_stream_id 确定的基本流，则执行对应于要处理的标记的 mark_type 的处理。

因此，如果我们说正在执行例如在图 25 中所示的第二 PlayList#1 的播放，则执行下面的标记处理。

即，在第二 PlayList#1 的 PlayListMark()中描述了第一 Mark()(Mark#0)、

第二 Mark()(Mark#1)和第三 Mark()(Mark#2)，其 mark_time_stamp 分别是 90,000、27,090,000 和 27,540,000，如图 28 的下部所示。

而且，在图 28 的下部所示的 PlayListMark()中，在第二 Mark()和第三 Mark()的各自的 entry_ES_stream_id 中描述了 0xE0 和 0xE1，因此，第二 Mark()和第三 Mark()与其各自的 stream_id 被 0xE0 和 0xE1 确定的基本流相关联。

现在，如图 25 所述，在第二 PlayList#1 中描述了仅仅一个 PlayItem()(PlayItem#0)，并且根据该 PlayItem#0，播放剪辑流文件“00003.PS”。如对于对应于剪辑流文件“00003.PS”的在图 26 中的剪辑信息文件“00003.CLP”所述，在剪辑流文件“00003.PS”上复用了三个基本流：由作为 0xE0 的 stream_id 确定的视频 stream#0，由作为 0xE1 的 stream_id 确定的视频 stream#1，由 stream_id 0xBD 和作为 0x00 的 private_stream_id 确定的音频 stream#2。

因此，在剪辑流文件“00003.PS”上复用的、其 stream_id 为 0xE0 的视频 stream#0 与图 28 的下部的 PlayListMark()的第二 Mark()相关联，并且，在剪辑流文件“00003.PS”上复用的、其 stream_id 为 0xE1 的视频 stream#1 与第三 Mark()相关联。

在要开始在图 25 中的第二 PlayList#1 的 PlayItem#0 的播放的情况下，如在图 30 中的步骤 S124 中所述，播放器控制模块 212 识别在图 28 的下部所示的 PlayListMark()中包括的三个 Mark()属于 PlayList#1 的 PlayItem#0，并且向解码控制模块 214 传送作为三个 Mark()的 mark_time_stamp 的{90,000}、{27,090,000}和{27,540,000}以及关于 mark_time_stamp 表示的时间点的属性是“标记处理”之结果的指示。

在标记处理中，解码控制模块 214 总是确认是否在 PlayList#1 的 PlayItem#0 的播放期间由时钟单元 214A 保持的当前时间点匹配时间点{90,000}、{27,090,000}和{27,540,000}之一(步骤 S301)，并且在当前的时间点匹配具有属性“标记处理”的时间点的情况下，向播放器控制模块 212 提供作为匹配当前时间点的属性“标记处理”的时间点的标记时间点以及关于当前时间是“标记处理”属性的时间点之结果的消息。

即，如果我们说当前时间点已经匹配 27,090,000，则在“标记处理”属性时间点{90,000}、{27,090,000}和{27,540,000}中，解码控制模块 214 向播放器控制模块 212 提供作为匹配当前时间点的属性“标记处理”的时间点的标

记时间点 27,090,000 以及关于当前时间是“标记处理”属性的时间点之结果的消息。

播放器控制模块 212 识别当前正在播放 PlayList#1 的 PlayItem#0，并且通过在 PlayList#1 中的图 28 的下部所示的 PlayListMark()所述的 Mark()中将作为属于 PlayItem#0 的三个 Mark()的 mark_time_stamp 的 90,000、27,090,000 和 27,540,000 的每一个与作为来自解码控制模块 214 的标记时间点的 27,090,000 进行比较，识别其 mark_time_stamp 匹配标记时间点 27,090,000 的 Mark()即在图 28 的下部所示的 PlayListMark()中所述的第二 Mark()(Mark#1)是要处理的标记(步骤 S302)。

在图 28 的下部所示的 PlayListMark()中所述的第二 Mark()中，将 0xE0 指定为 entry_ES_stream_id。根据上面的说明，这个 0xE0 的 entry_ES_stream_id 表示在剪辑流文件“00003.PS”上被复用的其 stream_id 为 0xE0 的视频 stream#0(图 26)，并且播放器控制模块 212 确定是否视频流 stream#0 被包括在被播放的基本流中(步骤 S303，S304)。

在视频流 stream#0 未被包括在被播放的基本流中的情况下，忽略要处理的标记(步骤 S304)。

另一方面，在视频流 stream#0 被包括在被播放的基本流中的情况下，将要处理的标记被认为是有效的，并且执行对应于要处理的那个标记的处理(步骤 S305 到 308)。

即，在这种情况下，在图 28 的下部所示的 PlayListMark()中所述的第二 Mark()具有“事件”的 mark_type，因此是事件标记，因此播放器控制模块 212 向脚本控制模块 211 提供用于表示事件发生的事件消息和要被处理的标记的 mark_data(步骤 S305，S307)。使用来自播放器控制模块 212 的事件消息来作为中断消息，以与事件消息一起提供的 mark_data 作为变元，脚本控制模块 211 预先执行在“SCRIPT.DAT”文件中描述的一系列处理（步骤 S308）。

如上所述，按照标记处理，确定是否当前时间点匹配 mark_time_stamp，所述当前时间点是正在按照 PlayList()(图 5)播放的剪辑流文件的播放时间点，所述 PlayList()包括 PlayList Mark()(图 7)，它具有 0 个或多个 Mark()，所述 Mark()包括用于表示在 PlayList()的时间轴上的一个播放时间点的 mark_time_stamp、用于表示 Mark()的类型的 mark_type 和用作事件标记的变元的 mark_data，并且在当前时间点匹配 mark_time_stamp 的情况下，具有等

于作为匹配的当前时间点的标记时间点的 mark_time_stamp 的 Mark()被识别为要处理的标记。而且，在要处理的标记具有的 mark_type 表示用于产生事件的类型的情况下，即在要处理的标记是事件标记的情况下，通知要处理的标记具有的 mark_data 和事件消息，并且执行对应于 mark_data 的处理。因此，可以对应于剪辑流文件的播放时间点来执行对应于 mark_data 的处理。

[输出属性的控制处理]

接着，将参见在图 41 中所示的流程图来说明在图 30 的步骤 S126 中执行的输出属性控制处理等的细节。

如在图 30 中的步骤 S126 中所述，播放器控制模块 212 对于要播放的一个或多个基本流，即对于已经在图 30 的步骤 S125 中确定播放的一个或多个基本流，首先查看用于表示其输出属性被描述的 DynamicInfo()(图 13)的 number_of_DynamicInfo(图 10)。

在 number_of_DynamicInfo 对于所有的所述一个或多个要播放的基本流是 0 的情况下，播放器控制模块 212 不具体执行处理。

另一方面，在 number_of_DynamicInfo 对于要播放的基本流不是 0 的情况下，播放器控制模块 212 按照图 41 中的流程图来执行输出属性控制处理。

因此，在盘 101 中记录的三个剪辑信息文件“00001.CLP”、“00002.CLP”和“00003.CLP”是例如如图 26 中所示的情况下，在正在播放对应于剪辑信息文件“00001.CLP”的剪辑流文件“00001.PS”的时候(在正在播放用于播放对应于剪辑信息文件“00001.CLP”的剪辑流文件“00001.PS”的第一 PlayList#1 的第一 PlayItem#0 的时候)，number_of_DynamicInfo 对于在剪辑流文件“00001.PS”上复用的所有四个基本流 stream#0 到 stream#3 是 0，因此，不对于剪辑信息文件“00001.CLP”执行输出属性控制处理(图 26)。

以相同的方式，在正在播放对应于剪辑信息文件“00002.CLP”的剪辑流文件“00002.PS”的时候(在正在播放用于播放对应于剪辑信息文件“00002.CLP”的剪辑流文件“00002.PS”的第一 PlayList#1 的第二 PlayItem#1 的时候)，number_of_DynamicInfo 对于在剪辑流文件“00002.PS”上复用的所有四个基本流 stream#0 到 stream#3 是 0，因此，不对于剪辑信息文件“00002.CLP”执行输出属性控制处理(图 26)。

另一方面，在正在播放对应于剪辑信息文件“00003.CLP”(图 26)的剪辑流文件“00003.PS”的时候(在正在播放用于播放对应于剪辑信息文件

“00003.CLP”的剪辑流文件“00003.PS”的第二PlayList#1的PlayItem#0的时候)，在剪辑流文件“00003.PS”上复用的三个基本流stream#0到stream#2中，作为第一基本流的视频流stream#0和作为第三基本流的音频流stream#2分别具有被设置到非0的2和3的number_of_DynamicInfo，因此，执行输出属性控制处理。

即，对于输出属性控制处理，首先，在步骤S320，播放器控制模块212向解码控制模块214传送在对应于要播放的剪辑流文件的剪辑信息文件Clip()(图10)中所述的pts_change_point以及关于这是属性“DynamicInfo()处理”的时间点的指示，并且解码控制模块214从播放器控制模块212接收作为属性“DynamicInfo()处理”的时间点的pts_change_point，并且流程进行到步骤S321。

在步骤S321，解码控制模块214确定是否由时钟单元214A保持的当前时间点匹配属性“DynamicInfo()处理”的pts_change_point(之一)，并且在确定不匹配的情况下，流程返回到步骤S321。

而且，在步骤S321，在确定当前时间点匹配属性“DynamicInfo()处理”的pts_change_point(之一)的情况下，解码控制模块214向播放器控制模块212提供关于当前时间点是属性“DynamicInfo()处理”的时间点之结果的消息和已经匹配当前时间点的属性“DynamicInfo()处理”的时间点(以下在适当时被称为DynamicInfo时间点)，并且流程进行到步骤S322。

在步骤S322中，播放器控制模块212接收关于当前时间点是属性“DynamicInfo()处理”的时间点之结果的消息和DynamicInfo时间点，识别与匹配DynamicInfo时间点的pts_change_point(图10)成为一组的DynamicInfo()是作为要被处理的DynamicInfo()的要处理的DynamicInfo()，并且流程进行到步骤S323。

在步骤S323，播放器控制模块212向图形处理模块219或者音频输出模块221提供在作为要处理的DynamicInfo()的DynamicInfo()(图13)中描述的输出属性，并且流程进行到步骤S324。

在步骤S324中，图形处理模块219或者音频输出模块221开始按照从在紧前的步骤S323中的播放器控制模块212提供的输出属性来控制视频数据或者音频数据的输出，并且流程返回到步骤S321。

因此，按照例如被描述为输出属性(显示格式)的长宽比来输出视频数据，

或者按照例如被描述为输出属性(输出格式)的立体声或者双声道(双语)来输出音频数据。

接着，将参见图 42 来进一步说明输出属性控制处理的细节。

即，图 42 说明了在图 26 中所示的剪辑信息文件“00003.CLP”中所述的 pts_change_point 和 DynamicInfo()的组(图 10)。

现在，如上所述，在剪辑流文件“00003.PS”上复用的三个基本流 stream#0 到 stream#2 中，作为第一基本流的视频流 stream#0 和作为第三基本流的音频流 stream#2 分别具有被设置为 2 和 3 的剪辑信息文件“00003.CLP”中的 number_of_DynamicInfo。因此，在剪辑信息文件“00003.CLP”中，在剪辑流文件“00003.PS”中的第一视频流 stream#0 中描述了两组 pts_change_point 和 DynamicInfo()，在剪辑流文件“00003.PS”中的第三音频流 stream#2 中描述了三组 pts_change_point 和 DynamicInfo()。

图 42 的上部说明了在剪辑流文件“00003.PS”中的第一视频流 stream#0 的两组 pts_change_point 和 DynamicInfo()，图 42 的下部说明了在剪辑流文件“00003.PS”中的第三个音频流 stream#2 的三组 pts_change_point 和 DynamicInfo()。

注意，在图 42 的上部，除了关于第一视频流 stream#0 所描述的两组 pts_change_point 和 DynamicInfo()之外，还说明了在关于那个视频流 stream#0 的剪辑信息文件“00003.CLP”中所述的 stream_id(= 0xE0)、private_stream_id(0x00)和 number_of_DynamicInfo(= 2)。以相同的方式，在图 42 的下部，除了关于第三音频流 stream#2 所述的三组 pts_change_point 和 DynamicInfo()之外，还说明了在关于音频流 stream#2 的剪辑信息文件“00003.CLP”中描述的 stream_id(= 0xBD)、private_stream_id(= 0x00)和 number_of_DynamicInfo(= 3)。

在图 42 的上部，在关于视频流 stream#0 所述的两组 pts_change_point 和 DynamicInfo()中，对于第一组，pts_change_point 是 90,000，并且 DynamicInfo()的 display_aspect_ratio(图 13)是“4:3”。而且，对于第二组，pts_change_point 是 54,090,000，并且 DynamicInfo()的 display_aspect_ratio(图 13)是“16:9”。

另一方面，在图 42 的下部，在关于音频流 stream#2 所述的三组 pts_change_point 和 DynamicInfo()中，对于第一组，pts_change_point 是 90,000，并且 DynamicInfo()的 channel_assignment(图 13)是“双声道”。而且，对于第

二组，pts_change_point是27,090,000，并且DynamicInfo()的channel_assignment是“立体声”。而且，对于第三组，pts_change_point 是 32,490,000，并且DynamicInfo()的 channel_assignment 是“双声道”。

让我们现在说，例如，在图 30 的步骤 S125 中，剪辑流文件“00003.PS”的由 0xE0 的 stream_id 确定的第一视频流 stream#0 和由 0xBD 的 stream_id 和 0x00 的 private_stream_id 确定的第三音频流 stream#2 已经被确定为要播放的流。

在这种情况下，播放器控制模块 212 查看由 0xE0 的 stream_id 确定的图 42 的上部所示的关于视频流 stream#0 所述的两组 pts_change_point 和 DynamicInfo()和由 0xBD 的 stream_id 和 0x00 的 private_stream_id 确定的在图 42 的下部所示的关于音频流 stream#2ss 的三组 pts_change_point 和 DynamicInfo()，并且识别初始值。

即，在由 0xE0 的 stream_id 确定的图 42 的上部所示的关于视频流 stream#0 所述的两组 pts_change_point 和 DynamicInfo()中，对于第一组，pts_change_point 是 90,000。这个时间点 90,000 在对应于其上视频流 stream#0 被复用的剪辑流文件“00003.PS”的图 26 中的剪辑信息文件“00003.CLP”中，匹配用于表示剪辑流文件“00003.PS”的首部时间点的 presentation_start_time 上描述的时间点 90,000。

以相同的方式，在由 0xBD 的 stream_id 和 0x00 的 private_stream_id 确定的在图 42 的下部所示的关于音频流 stream#2 所述的三组 pts_change_point 和 DynamicInfo()中，对于第一组，pts_change_point 是 90,000。这个时间点 90,000 在对应于其上音频流 stream#2 被复用的剪辑流文件“00003.PS”的图 26 中的剪辑信息文件“00003.CLP”中，匹配用于表示剪辑流文件“00003.PS”的首部时间点的 presentation_start_time 上描述的时间点 90,000。

播放器控制模块 212 将匹配用于表示剪辑流文件“00003.PS”的首部时间点的 presentation_start_time 上描述的时间点 90,000 的 pts_change_point 识别为初始值。因此，在图 42 的上部的两组 pts_change_point 和 DynamicInfo()的第一组的 pts_change_point 和在图 42 的下部的三组 pts_change_point 和 DynamicInfo()的第一组的 pts_change_point 被识别为初始值。

播放器控制模块 212 然后在剪辑流文件“00003.PS”的播放开始之前(在图 30 的步骤 S126)，按照与被识别为初始值的 pts_change_point 成为一组的

DynamicInfo()来指令对应的基本流的输出属性。

即，对于由 0xE0 的 stream_id 确定的视频流 stream#0，在图 42 的上部，在与在初始值 90,000 的 pts_change_point 成为一组的 DynamicInfo()中的 display_aspect_ratio 是“4:3”。在这种情况下，播放器控制模块 212 使用作为 display_aspect_ratio 是“4:3”即视频流 stream#0 是具有长宽比 4:3 的视频数据之结果的输出属性信息来控制图形处理模块 219。

而且，对于由 0xBD 的 stream_id 和 0x00 的 private_stream_id 确定的音频流 stream#2，在图 42 的下部，与初始值 90,000 的 pts_change_point 成为一组的 DynamicInfo()中的 channel_assignment 是“双声道”。在这种情况下，播放器控制模块 212 向音频输出模块 221 提供关于 channel_assignment 是“双声道”即音频流 stream#2 是双声道音频数据的输出属性信息。

现在，在图 30 的步骤 S126 中，执行关于作为初始值的 pts_change_point 的输出属性控制处理。

随后，在关于视频流 stream#0 的图 42 的上部的两个 pts_change_point 90,000 和 54,090,000 和关于音频流 stream#2 的图 42 的下部的三个 pts_change_point 90,000、27,090,000 和 32,490,000 中，播放器控制模块 212 采用除了初始值 90,000 之外的时间点，它们是{27,090,000}、{32,490,000}和{54,090,000}，并且将这些与关于这些是“DynamicInfo()处理”属性时间点的指示一起传送到解码控制模块 214(步骤 S320)。

解码控制模块 214 从播放器控制模块 212 接收“DynamicInfo()处理”属性时间点{27,090,000}、{32,490,000}和{54,090,000}，并且进一步在视频流 stream#0 和音频流 stream#2 的播放(用于播放剪辑流文件“00003.PS”的第二 PlayList#1 的 PlayItem#0 的播放)的开始之后，开始监控由时钟单元 214A 保持的当前时间点。

在当前时间点匹配“DynamicInfo()处理”属性时间点{27,090,000}、{32,490,000}和{54,090,000}之一的情况下，解码控制模块 214 向播放器控制模块 212 提供作为匹配当前时间点的“DynamicInfo()处理”属性时间点的 DynamicInfo 时间点(步骤 S321)。

即，例如，在当前时间点是 27,090,000 的情况下，解码控制模块 214 向播放器控制模块 212 提供匹配当前时间点的“DynamicInfo()处理”属性时间点 27,090,000。

播放器控制模块 212 从解码控制模块 214 接收 DynamicInfo 时间点 27,090,000，从图 42 上部的关于视频流 stream#0 的两个 pts_change_point 和图 42 下部的关于音频流 stream#2 的三个 pts_change_point，查看匹配 DynamicInfo 时间点 27,090,000 的 pts_change_point，并且将与匹配 27,090,000 的 pts_change_point 成为一组的 DynamicInfo()即在图 42 下部的关于音频流 stream#2 的第二 DynamicInfo()识别为要处理的 DynamicInfo()(步骤 S322)。

在要处理的 DynamicInfo()是关于视频流的 DynamicInfo()的情况下，播放器控制模块 212 向图形处理模块 219 提供在要处理的 DynamicInfo()中所述的输出属性(步骤 S323)。而且，在要处理的 DynamicInfo()是关于音频流的 DynamicInfo()的情况下，播放器控制模块 212 向音频输出模块 221 提供在要处理的 DynamicInfo()中所述的输出属性(步骤 S323)。

在输出属性被从播放器控制模块 212 提供之后，图形处理模块 219 开始按照这些输出属性来控制视频数据的输出(步骤 S324)。

即，图形处理模块 219 根据来自播放器控制模块 212 的输出属性所表示的视频数据的长宽比(display_aspect_ratio(图 13))的指令和连接到在图 1 中所示的视频输出端 120 的视频输出装置的长宽比来执行要被输出到视频输出模块 220 的视频数据的长宽比的转换。

具体地，例如，在其中视频输出装置的长宽比是 16:9 的情况下，在作为输出属性的视频数据的长宽比的指令表示 4: 3 的长宽比时，图形处理模块 219 将要输出到视频输出模块 220 的视频数据进行水平压挤，并且在左部和右部以黑色部分（pillarboxing）来输出。而且，在视频输出装置的长宽比是例如 4:3 的情况下，作为输出属性的视频数据的长宽比的指令表示 16:9 的长宽比，图形处理模块 219 执行用于在垂直方向上压挤要被输出到视频输出模块 220 的视频数据的处理，并且在顶部和底部以黑色部分（letterboxing）输出。而且，例如，在视频输出装置的长宽比和以视频数据的长宽比指令表示的长宽比都同为 4:3 或者 16:9 的情况下，图形处理模块 219 原样输出要输出到视频输出模块 220 的视频数据，而不执行压挤处理。

现在，对于由 0xE0 的 stream_id 确定的图 42 的上部所示的关于视频流 stream#0 所述的两组 pts_change_point 和 DynamicInfo()，从作为视频流 stream#0 的播放开始时间的时间点 90,000 到紧接时间点 54,090,000 之前，从视频流 stream#0 获得具有长宽比 4:3 的视频数据。在时间点 54,090,000 之后，

从视频流 stream#0 获得具有长宽比 16:9 的视频数据。

因此，例如，如果我们说连接到在图 1 中所示的视频输出端 120 的视频输出装置的长宽比是 4:3，则在图形处理模块 219，从时间点 90,000 到紧接时间点 54,090,000 之前，具有从视频流 stream#0 获得的长宽比 4:3 的视频数据被原样提供到具有长宽比 4:3 的视频输出装置，并且被显示。

然后，在时间点 54,090,000 之后，从视频流 stream#0 获得的具有长宽比 16:9 的视频数据经过垂直压挤处理，并且进一步被转换为具有长宽比 4:3 的视频信号，在顶部和底部具有黑色部分（letterboxing），并且被提供到具有长宽比 4:3 的视频输出装置和被显示。

另一方面，在输出属性被从播放器控制模块 212 提供后，音频输出模块 221 开始按照这些输出属性来控制音频数据的输出(步骤 S324)。

即，音频输出模块 221 根据来自播放器控制模块 212 的输出属性所表示的音频数据的信道分配指令(channel_assignment(图 13))和由操作遥控器的用户经由输入接口 115(图 1)从播放器控制模块 212 提供的音频输出模式，来处理来自音频解码器控制模块 217 的音频数据，并且输出到音频输出端 121(图 1)。

具体地，例如，在输出属性表示的音频数据的信道分配的指令是双声道(双语)模式——“主音频”音频数据用于左声道并且“付音频”音频数据用于右声道——的情况下，音频输出模块 221 按照从播放器控制模块 212 提供的音频输出模式来处理来自音频解码器控制模块 217 的音频数据。

即，例如，在“主音频”被指定为音频输出模式的情况下，音频输出模块 221 在来自音频解码器控制模块 217 的音频数据中，将左声道的音频数据复制为右声道的音频数据，并且向音频输出端 121 输出左声道和右声道的音频数据(“主音频”音频数据)。而且，在“付音频”被指定为音频输出模式的情况下，音频输出模块 221 在来自音频解码器控制模块 217 的音频数据中，将右声道的音频数据复制为左声道的音频数据，并且向音频输出端 121 输出左声道和右声道的音频数据(“付音频”音频数据)。而且，在“主/付”被指定为音频输出模式的情况下，音频输出模块 221 原样向音频输出端 121 输出来自音频解码器控制模块 217 的音频数据。

而且，例如，在输出属性表示的音频数据的信道分配指令表示立体声(Stereo)模式音频数据的情况下，音频输出模块 221 原样向音频输出端 121 输

出来自音频解码器控制模块 217 的音频数据，而与从播放器控制 212 提供的音频输出模式的指定无关。

现在，对于由 0xBD 的 stream_id 和 0x00 的 private_stream_id 确定的图 42 的下部所示的关于音频流 stream#2 所述的三组 pts_change_point 和 DynamicInfo()，从作为音频流 stream#2 的播放开始时间的时间点 90,000 到紧接时间点 27,090,000 之前，从音频流 stream#2 获得双声道音频数据。在时间点 27,090,000 之后到紧接时间点 32,490,000 之前，从音频流 stream#2 获得立体声音频数据，并且在时间点 32,490,000 之后，从音频流 stream#2 获得双声道音频数据。

因此，例如，在“主音频”被指定为音频输出模式的情况下，音频输出模块 221 从时间点 90,000 到紧接时间点 27,090,000 之前，在从音频流 stream#2 获得的双声道音频数据中，将左声道的音频数据复制为右声道的音频数据，并且向音频输出端 121 输出左声道和右声道的音频数据。

而且，从时间点 27,090,000 到紧接时间点 32,490,000 之前，从音频流 stream#2 获得的立体声音频数据被原样输出到音频输出端 121。

然后，在时间点 32,490,000 后，在从音频流 stream#2 获得的双声道音频数据中，左声道的音频数据被复制为右声道的音频数据，并且左声道和右声道的音频数据被输出到音频输出端 121。

如上所述，对于输出属性控制处理，对于在剪辑流文件上复用的每个基本流，基于关于包括 0 个或多个用于表示基本流的播放时间点的 pts_change_point 和包括基本流的输出属性的 DynamicInfo()的组的剪辑信息文件 Clip()的说明，确定是否被播放的基本流的播放时间点匹配 pts_change_point。在被播放的基本流的播放时间点匹配 pts_change_point 的情况下，识别与 pts_change_point 成为一组的 DynamicInfo()，并且按照在所识别的 DynamicInfo()中包括的输出属性来控制被播放的基本流的输出。因此，可以按照基本流的播放时间点和输出属性来控制基本流的输出。

[字幕显示控制处理]

接着，将参见在图 43 中的流程图来说明用于控制对应于字幕流的字幕数据的显示的字幕显示控制处理。

在开始 PlayList()(图 5)(的 PlayItem())的播放后，在步骤 S341，播放器控制模块 212 对于图形处理模块 219 初始化关于字幕数据的显示格式的指令。

即，播放器控制模块 212 控制图形处理模块 219 以便使用字幕数据的显示格式来作为默认显示格式。注意，在步骤 S341 中执行的显示格式的指令的初始化对应于在图 30 中的 S127 中描述的显示格式的指令的初始化。

在步骤 S341 的处理之后，流程进行到步骤 S342。在此，播放器控制模块 212 确定是否已经有由操作遥控器的用户从输入接口 115 的、关于相对于字幕数据的显示的新的显示格式的指令。

在步骤 S342 中确定已经有新的显示格式的指令的情况下，流程进行到步骤 S343，在此，播放器控制模块 212 确定是否字幕流(对应于字幕流的字幕数据)当前正在被播放。

在步骤 S343 中确定没有字幕流正在播放的情况下，流程返回到步骤 S342。

而且，在步骤 S343 中确定字幕流正在播放的情况下，流程进行到步骤 S345，在此，播放器控制模块 212 确定是否新的显示格式指令是用于默认显示格式的指令，在步骤 S343 中确定新的显示格式指令是用于默认显示格式的指令的情况下，流程返回到步骤 S341，并且如上所述，播放器控制模块 212 控制图形处理模块 219 以便字幕数据显示格式是默认显示格式。

另一方面，在步骤 S345 中确定新的显示格式的指令不是用于默认显示格式的指令的情况下，即，在新的显示格式的指令是非默认的默认显示指令——诸如字幕数据的放大或者缩小显示或者改变亮度以便利观看等——的情况下，流程进行到步骤 S346，并且播放器控制模块 212 从对应于其上已经复用了当前正在播放的字幕流的剪辑流文件的剪辑信息文件 Clip()（图 10）中的 StaticInfo()(图 12)获得关于当前正在播放的字幕流的 StaticInfo()，并且流程进行到步骤 S347。

在步骤 S347 中，播放器控制模块 212 确定在步骤 S346 中获得的 StaticInfo()的 configurable_flag。

在步骤 S347 中确定 configurable_flag 被设置为 0——表示不允许字幕数据的显示格式的改变——的情况下，流程进行到步骤 S348，在此，播放器控制模块 212 控制图形处理模块 219 来覆盖关于不能改变字幕数据的显示格式之结果的错误消息，并且流程返回到步骤 S342。因此，显示错误消息。

另一方面，在步骤 S347 中确定 configurable_flag 被设置为 1——表示允许字幕数据的显示格式的改变——的情况下，流程进行到步骤 S349，在此，

播放器控制模块 212 向图形处理模块 219 提供由操作遥控器的用户从输入接口 115 提供的、新的显示格式指令，并且流程进行到步骤 S350。

在步骤 S350，图形处理模块 219 按照在紧前的步骤 S349 中从播放器控制模块 212 提供的显示格式指令来开始从字幕解码器控制模块 218 提供的字幕数据的放大或缩小等或者改变亮度等的处理，并且流程返回到步骤 S342。因此，以按照由操作遥控器的用户指令的显示格式的显示大小、显示位置、显示颜色等来显示字幕数据。

另一方面，在步骤 S342 中确定没有新的显示格式指令的情况下，流程进行到步骤 S351，在此，播放器控制模块 212 确定是否已经执行了参见图 31 所述的 PlayItem()的交叉，并且在确定还没有执行此的情况下，流程返回到步骤 S342。

另一方面，在步骤 S351 中确定已经执行了 PlayItem()的交叉的情况下，流程返回到步骤 S341，并且如上所述，播放器控制模块 212 控制图形处理模块 219 以便字幕数据显示格式是默认显示格式。即，在这种情况下，在已经执行了 PlayItem()的交叉的情况下，字幕数据显示格式被返回到默认显示格式。

如上所述，对于字幕显示控制处理，仅仅在 configurable_flag 被设置为 1——表示允许字幕数据的显示格式的改变——的情况下，才按照例如由操作遥控器的用户输入的显示格式的指令来改变对应于字幕流的字幕数据的显示格式。

因此，例如，使用在图 26 中所示的剪辑信息文件“00001.CLP”，作为在对应的剪辑流文件“00001.PS”上复用的四个基本流的第三个基本流的字幕流 stream#2 的 configurable_flag 被设置为 0，用于表示不允许显示格式的改变，因此，即使当正在显示字幕流 stream#2 的同时用户操作遥控器来改变字幕的显示，也不改变显示。

另一方面，例如，作为在对应的剪辑流文件“00001.PS”上复用的四个基本流的第四个基本流的字幕流 stream#3 的 configurable_flag 被设置为 1，用于表示允许显示格式的改变，因此，如果当正在显示字幕流 stream#3 的同时用户操作遥控器来改变字幕的显示，则字幕的显示大小等按照所进行的操作被改变。

即，例如，让我们说，剪辑流文件“00001.PS”现在正在按照在图 25 中

的第一 PlayList#1 的第一 PlayItem#0 而被播放。而且，如参见在图 26 中的剪辑流文件“00001.PS”所述，在剪辑流文件“00001.PS”上复用的四个基本流中，第三和第四是字幕流，并且我们说，在第三字幕流 stream#2 和第四字幕流 stream#3 中，当前正在播放例如第三字幕流 stream#2。

在用户通过操作遥控器来输入用于字幕显示格式的指令后(步骤 S242)，从输入接口 115(图 1)向播放器控制模块 212 提供显示格式指令。在被提供有显示格式指令后，播放器控制模块 212 从剪辑信息文件搜索对应于正在播放的字幕流的 StaticInfo()(图 10)(步骤 S346)。

即，在这种情况下，被播放的字幕流是在剪辑流文件“00001.PS”上复用的第三字幕流 stream#2，并且播放器控制模块 212 从对应的剪辑信息文件“00001.CLP”搜索与第三字幕流 stream#2 相关联的 StaticInfo()。

而且，播放器控制模块 212 判断被设置到 0 的在关于在图 26 中的第三字幕流 stream#2 的 StaticInfo()中所述的 configurable_flag(步骤 S347)，因此确认对于第三字幕流 stream#2 不允许显示格式的改变。

在这种情况下，播放器控制模块 212 确定对应于正在被播放的字幕流的字幕数据不能处理放大/缩小等，并且控制图形处理模块 219 产生关于此的错误消息(步骤 S348)，所述错误消息被覆盖在视频数据上，并且被输出。

另一方面，在剪辑流文件“00001.PS”上复用的四个基本流中，在第三字幕流 stream#2 和第四字幕流 stream#3 的情况下，例如第四字幕流 stream#3 而不是第三字幕流 stream#2 当前正在被播放的情况下，播放器控制模块 212 从对应的剪辑信息文件“00001.CLP”搜索与第四字幕流 stream#3 相关联的 StaticInfo()。

而且，播放器控制模块 212 判断被设置到 1 的在关于在图 26 中的第四字幕流 stream#3 的 StaticInfo()中所述的 configurable_flag(步骤 S347)，因此确认对于第四字幕流 stream#3 允许显示格式的改变。

在这种情况下，播放器控制模块 212 确定正在被播放的字幕流(对应于正在被播放的字幕流的字幕数据)可以处理放大/缩小等，并且向图形处理模块 219 提供由操作遥控器的用户提供的显示格式指令。

因此，图形处理模块 219 按照来自播放器控制模块 212 的显示格式指令来将来自字幕解码器控制模块 218 的字幕数据放大、缩小等，并且覆盖在来自视频解码器控制模块 216 的视频数据上和输出。

注意，播放器控制模块 212 在开始播放 PlayList()的第一 PlayItem()时初始化到图形处理模块 219 的字幕数据显示格式指令(步骤 S341)。即，播放器控制模块 212 控制图形处理模块 219 以便字幕数据显示格式是默认显示格式。

而且，也在 PlayItem()交叉时，播放器控制模块 212 初始化到图形处理模块 219 的字幕数据显示格式指令(步骤 S341，S351)。

但是，注意，可以进行一种布置，其中，在 PlayItem()交叉时，搜索关于按照此后要新播放的 PlayItem()而播放的新字幕流的 configurable_flag，并且在 configurable_flag 是 0 的情况下，初始化到图形处理模块 219 的字幕数据显示格式指令，并且在 configurable_flag 是 1 的情况下，将到图形处理模块 219 的字幕数据显示格式指令保持为与在 PlayItem()交叉之前相同。

而且，对于在图 43 中的字幕显示控制处理，在由操作遥控器的用户输入新的显示格式指令的情况下，向图形处理模块 219 提供新的显示格式指令(步骤 S349)，但是，显示格式指令可以被存储在构成存储器 113(图 1)的非易失性存储器中，并且在非易失性存储器中存储的所述显示格式指令被提供到图形处理模块 219。

即，可以进行一种布置，其中，例如，用户设置的显示格式指令被存储在非易失性存储器中来作为在图 1 中所示的盘装置的初始设置，并且在新的显示格式指令被操作遥控器的用户输入的情况下，在非易失性存储器中存储的显示格式指令被更新为新的显示格式指令，并且在非易失性存储器中存储的显示格式指令被提供到图形处理模块 219。在这种情况下，在前一次结束播放时间的显示格式的指令被保存在非易失性存储器中，因此，在下一次播放 PlayList()时，字幕数据的显示以那个显示格式开始，甚至不用操作遥控器的用户在前一次结束播放时输入所述显示格式。

注意，在这种情况下，要被存储在非易失性存储器中的显示格式的指令包括例如用于放大或者缩小字幕数据的放大比或者缩小比等。

如上所述，按照字幕显示控制处理，在基本流的播放期间不改变的 StaticInfo()中，对于在剪辑信息文件 Clip()(图 10)中包括的每个基本流获得字幕数据的 StaticInfo()，并且根据用于表示是否允许从默认显示格式改变正在播放的字幕数据显示的 configurable_flag 来确定是否允许从默认显示格式改变正在播放的字幕数据显示。在允许从默认显示格式改变正在播放的字幕数据显示的情况下，通过例如放大或者缩小字幕数据或者改变显示颜色等来执

行字幕数据的显示处理、即显示处理。因此，可以控制字幕数据显示格式的改变。

[捕获控制处理]

接着，将参见在图 44 中的流程图来说明用于控制对应于视频流的视频数据的捕获的捕获控制处理。注意，图 44 说明了与用于描述捕获控制处理的流程图一起的、用于描述背景/屏幕保护程序处理的流程图，它是由捕获控制处理捕获的视频数据的二次使用的例子。

例如，通过由操作遥控器的用户经由输入接口 115 提供到播放器控制模块 212 的、用于指令捕获视频数据的捕获指令来开始捕获控制处理。

即，对于捕获控制处理，首先，在步骤 S371 中，播放器控制模块 212 确定是否正在播放视频流，并且在确定未播放时，捕获控制处理结束。

另一方面，在步骤 S371 中确定正在播放视频流的情况下，流程进行到步骤 S372，在此，播放器控制模块 212 从对应于正在被播放的视频流的 PlayList()(图 5)获得 capture_enable_flag_PlayList，并且也从对应于正在被播放的视频流的剪辑信息文件 Clip()(图 10)获得 capture_enable_flag_Clip。

现在，在 PlayList()中的 capture_enable_flag_PlayList 表示是否允许对应于由 PlayList()播放的视频流的视频数据(属于 PlayList()的视频数据)的二次使用，如参见图 5 所述。而且，在剪辑信息文件 Clip()中的 capture_enable_flag_Clip 表示是否允许对应于在与剪辑信息文件 Clip()对应的剪辑流文件中存储的视频流的视频数据的二次使用，如参见图 10 所述。

在步骤 S372 中的处理后，流程进行到步骤 S373，在此，播放器控制模块 212 根据在紧前的步骤 S373 中获得的 capture_enable_flag_PlayList 和 capture_enable_flag_Clip 确定是否允许捕获在从输入接口 115(图 1)输入捕获指令时播放的视频数据画面。

在步骤S373中确定不允许捕获在从输入接口115输入捕获指令时播放的视频数据画面的情况下，即在紧前的步骤 S373 中获得的 capture_enable_flag_PlayList 和 capture_enable_flag_Clip 的至少一个是 0——表示不允许视频数据的二次使用——的情况下，流程进行到步骤 S374，在此，播放器控制模块 212 控制图形处理模块 219 来覆盖关于不允许捕获视频数据之结果的错误消息，并且捕获控制处理结束。因此，显示错误消息。

另一方面，在步骤 S373 中确定允许捕获在从输入接口 115 输入捕获指令

时播放的视频数据画面的情况下，即在紧前的步骤 S373 中获得的 capture_enable_flag_PlayList 和 capture_enable_flag_Clip 两者是 1——表示允许视频数据的二次使用——的情况下，流程进行到步骤 S375，在此，播放器控制模块 212 向图形处理模块 219 提供在从输入接口 115 输入捕获指令时播放的视频数据画面的捕获指令，并且流程进行到步骤 S376。

在步骤 S376 中，图形处理模块 219 按照来自播放器控制模块 212 的捕获指令来捕获来自视频解码器控制模块 216 的视频数据的画面，将其存储在存储器 113(图 1)中，并且结束捕获控制处理。注意，capture_enable_flag 是多个比特的配置，并且在存在多个使用条件的限制的情况下，在此处处理它。即，在存在对于捕获图像的大小的多个限制的情况下，在此捕获缩小的图像。而且，在所使用的应用中存在多个限制的情况下，同时记录用于进行关于那个结果的通知的标记。

如上所述，使用捕获控制处理，对于对应于当存在来自用户的捕获指令时正在播放的视频流的 PlayList()(图 5)和剪辑信息文件 Clip()(图 10)的每一个，获得 capture_enable_flag_PlayList 和 capture_enable_flag_Clip 的逻辑与，并且在所述逻辑与是 1 的情况下，即仅仅在 capture_enable_flag_PlayList 和 capture_enable_flag_Clip 两者都被设置为允许二次使用的 1 的情况下，确定允许视频数据的二次使用，并且执行捕获。

因此，在正在按照例如在图 25 中的第一 PlayList#0 的第一 PlayItem#0 来播放视频流时有来自用户的捕获指令的情况下，即，在正在执行在剪辑流文件“00001.PS”上复用的视频流的情况下，在第一 PlayList#0 中的 capture_enable_flag_PlayList 是 1，并且在对应于由第一 PlayItem#0 播放的剪辑流文件“00001.PS”的、在图 26 中的剪辑信息文件“00001.CLP”中的 capture_enable_flag_Clip 是 1，因此确定允许正在被播放的视频数据(对应于在剪辑流文件“00001.PS”上复用的视频流)的二次使用，因此执行捕获。

而且，在正在按照例如在图 25 中的第一 PlayList#0 的第二 PlayItem#1 来播放视频流时有来自用户的捕获指令的情况下，即，在正在执行在剪辑流文件“00002.PS”上复用的视频流的情况下，在第一 PlayList#0 中的 capture_enable_flag_PlayList 是 1，并且在对应于由第二 PlayItem#1 播放的剪辑流文件“00002.PS”的、在图 26 中的剪辑信息文件“00002.CLP”中的 capture_enable_flag_Clip 是 0，因此确定不允许正在被播放的视频数据(对应于

在剪辑流文件“00002.PS”上复用的视频流的视频数据)的二次使用，因此不执行捕获。

而且，在正在按照例如在图 25 中的第二 PlayList#0 的 PlayItem#0 来播放视频流时有来自用户的捕获指令的情况下，即，在正在执行在剪辑流文件“00003.PS”上复用的视频流的情况下，在第二 PlayList#1 中的 capture_enable_flag_PlayList 是 0，并且在对应于由第二 PlayList#1 中的 PlayItem#0 播放的剪辑流文件“00003.PS”的、在图 26 中的剪辑信息文件“00003.CLP”中的 capture_enable_flag_Clip 是 1，因此确定不允许正在被播放的视频数据(对应于在剪辑流文件“00003.PS”上复用的视频流的视频数据)的二次使用，因此不执行捕获。

注意，在这种情况下，可以确定在发现在第二 PlayList#1 中的 capture_enable_flag_PlayList 是 0 时不允许视频数据的二次使用，因此，可以省略对于对应于在第二 PlayList#1 中的 PlayItem#0 播放的剪辑流文件“00003.PS”的、在图 26 中的剪辑信息文件“00003.CLP”中的 capture_enable_flag_Clip 的确认。

已经由捕获控制处理捕获并且被存储在存储器 113 中的画面可以在背景/屏幕保护程序处理中进行二次使用。

在下述情况下执行背景/屏幕保护程序处理：其中，例如，播放器控制模块 212 正在运行，但是不执行基本流的播放，即，盘 101 没有被插入到盘驱动器 102(图 1)中，或者已经结束了基本流的播放等等。

即，对于背景/屏幕保护程序处理，在步骤 S381 中，播放器控制模块 212 控制图形处理模块 219 以便显示通过捕获控制处理而存储在存储器 113 中的画面，图形处理模块 219 按照播放器控制模块 212 的控制来显示通过捕获控制处理而存储在存储器 113 中的画面。

现在，使用图形处理模块 219 来将在存储器 113 中存储的画面显示为静止图像实现了所谓的壁纸(背景)，并且按照某个周期来在放大、缩小、移动等的同时显示实现了屏幕保护程序。而且，可以由取代播放器控制模块 212 的另一个独立应用来执行用于显示通过捕获控制处理而存储在存储器 113 中的画面的背景/屏幕保护程序处理。

而且，在表示使用限制的标记被加到在存储器 113 中存储的画面的情况下，遵循所述限制。

如上所述，获得了正在被播放的视频数据的 capture_enable_flag_PlayList 和 capture_enable_flag_Clip，用于表示是否允许对应于大于视频访问单元的递增的例如对应于 PlayList()或者 PlayItem()的视频数据的二次使用，并且根据 capture_enable_flag_PlayList 和 capture_enable_flag_Clip 来确定是否允许正在播放的视频数据的二次使用。在确定允许被播放的视频数据的二次使用的情况下，捕获正在播放的视频数据，并且执行使用所捕获的视频数据的背景/屏幕保护程序处理。因此，使能视频数据的二次使用的控制。

注意，虽然已经对于在图 44 中的捕获控制处理进行了布置，其中对于 PlayList()(图 5)提供了 capture_enable_flag_PlayList，并且对于对应于由 PlayItem()播放的剪辑流文件的剪辑信息文件 Clip()(图 10)提供了 capture_enable_flag_Clip，但是，对于用于确定二次使用的许可(允许/不允许)的 capture_enable_flag_PlayList 和 capture_enable_flag_Clip，可以进行一种布置，其中，仅仅对于 PlayList()(图 5)提供了 capture_enable_flag_PlayList，或者仅仅对于对应于由 PlayItem()播放的剪辑流文件的剪辑信息文件 Clip()(图 10)提供了 capture_enable_flag_Clip，以便仅仅使用 capture_enable_flag_PlayList 和 capture_enable_flag_Clip 的一个来确定二次使用的许可。

注意，虽然已经对于在图 44 中的捕获控制处理进行了布置，其中，在步骤 S376 中，图形处理模块 219 按照来自播放器控制模块 212 的捕获指令来捕获来自视频解码器控制模块 216 的视频数据的画面，即仅仅捕获一个画面，但是，可以进行一种布置，其中，捕获多个画面。即，可以捕获以时间顺序(作为移动画面的多个画面的顺序)从视频解码器控制模块 216 输出的多个画面。在这种情况下，可以例如预先确定一次捕获的画面的数量，或者，capture_enable_flag_PlayList 和 capture_enable_flag_Clip 可以被扩展来包括可以在 capture_enable_flag_PlayList 或者 capture_enable_flag_Clip 中一次捕获的画面的数量的信息。

而且，在上述的情况下，在 PlayList()或者剪辑信息文件 Clip()中描述了用于允许视频数据的二次使用的使用许可信息(capture_enable_flag_PlayList，capture_enable_flag_Clip)，并且根据该使用许可信息来对于由 PlayList()播放的视频数据的整体或者对应于在与剪辑信息文件 Clip()对应的剪辑流文件上复用的视频流的视频数据的整体确定二次使用的许可，但是，可以进行一种

布置，其中，对于其他选用递增的视频数据，进行使用许可信息的说明，以便确定由其使用许可信息的视频数据的选用递增的二次使用的许可。

即，图 45 说明了使用许可信息所位于的 private_stream2_PES_payload()的语法，并且图 46 说明了使用许可信息所位于的 au_information()的语法。

注意，对于在图 45 中的 private_stream2_PES_payload()，作为使用许可信息的 capture_enable_flag_ps2 位于紧接 video_stream_id 之前，并且以与在图 23 中相同的方式被配置。而且，在图 46 中的 au_information()被以与在图 24 中相同的方式配置，除了作为使用许可信息的 capture_enable_flag_AU 位于紧接 pic_struct_copy 之前。

位于在图 45 中的 private_stream2_PES_payload() 中的 capture_enable_flag_ps2 表示是否对于对应于来自包括所述 private_stream2_PES_payload()到紧接下一个 private_stream2_PES_payload()之前的 private_stream_2 的 PES_packet()的视频流的视频数据允许二次使用。因此，可以从位于在图 45 中的 private_stream2_PES_payload() 的 capture_enable_flag_ps2 确定是否对于从某个解码开始可允许点到下一个解码开始可允许点的视频数据允许二次使用。

而且，位于在图 46 中的 au_information()中的 capture_enable_flag_AU 表示是否允许对应于 capture_enable_flag_AU 的视频访问单元的视频数据的二次使用。因此，可以从位于在图 46 中的 au_information() 的 capture_enable_flag_AU 确定是否对于以视频访问单元递增、即以画面递增的视频数据允许二次使用。

现在，可以复制地使用作为在 PlayList()(图 5)中的使用许可信息的 capture_enable_flag_PlayList 的两个或者多个、作为在剪辑信息文件 Clip()(图 10) 中的使用许可信息的 capture_enable_flag_Clip 、作为在 private_stream2_PES_payload()(图 45) 中的使用许可信息的 capture_enable_flag_ps2 和作为在 au_information()(图 46)中的使用许可信息的 capture_enable_flag_AU，并且在这种情况下，可以根据复制使用的所述两个或者多个使用许可信息的逻辑与等来确定某个视频数据的画面的二次使用的许可。

如参见图 36 中的步骤 S211 所述，缓冲器控制模块 215(图 3)的视频读取功能单元 233 从在缓冲器 215A 中存储的节目流搜索包括图 23 或者图 45 中

所示的 private_stream2_PES_payload()的 private_stream_2 的 PES_packet()，在此，配置了在图 46 中所示的 au_information()。因此，在使用其中配置了 capture_enable_flag_ps2 的图 45 中所示的 private_stream2_PES_payload()或者其中配置了 capture_enable_flag_AU 的图 46 中所示的 au_information()的情况下，播放器控制模块 212 必须向视频读取功能单元 233 查询用于确定视频数据的二次使用许可的 capture_enable_flag_ps2 或 capture_enable_flag_AU。

注意，虽然通过在本实施例中的软件来执行上述的系列处理，但是也可以通过专用硬件来执行上述的系列处理。

而且，虽然对于本实施例使用硬件解码器来用于视频解码器 116(图 1)，但是可以使用软件解码器来用于视频解码器 116。这对于音频解码器 117(图 1)也为真。

而且，虽然对于本实施例将软件解码器用作字幕解码器，但是可以使用硬件解码器来作为字幕解码器。

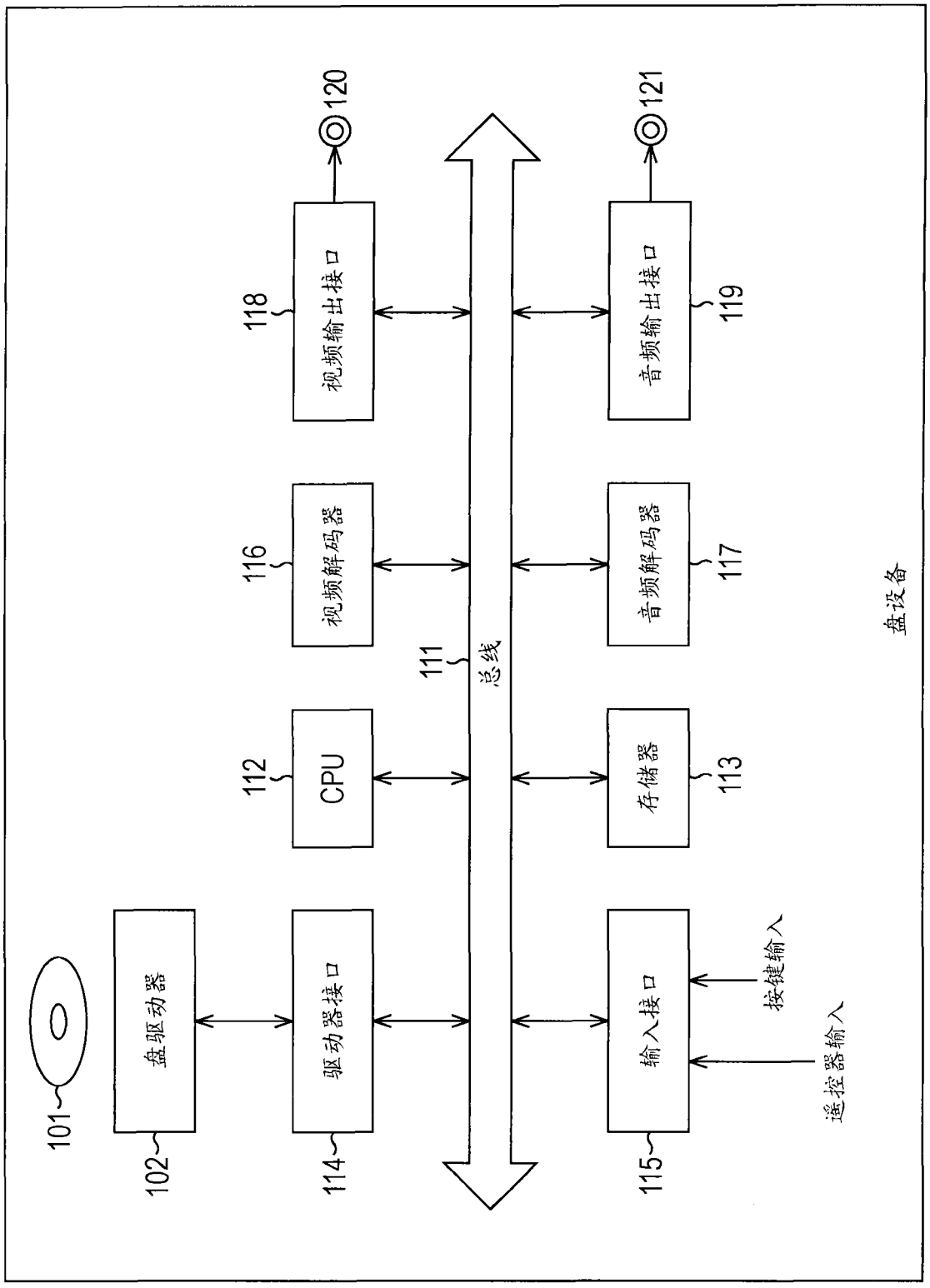

101
盘驱动器
102
驱动器接口
114
CPU
112
视频解码器
116
视频输出接口
118
120
总线
111
存储器
113
音频解码器
117
音频输出接口
119
121
输入接口
115
按键输入
遥控器输入
盘设备

图 1

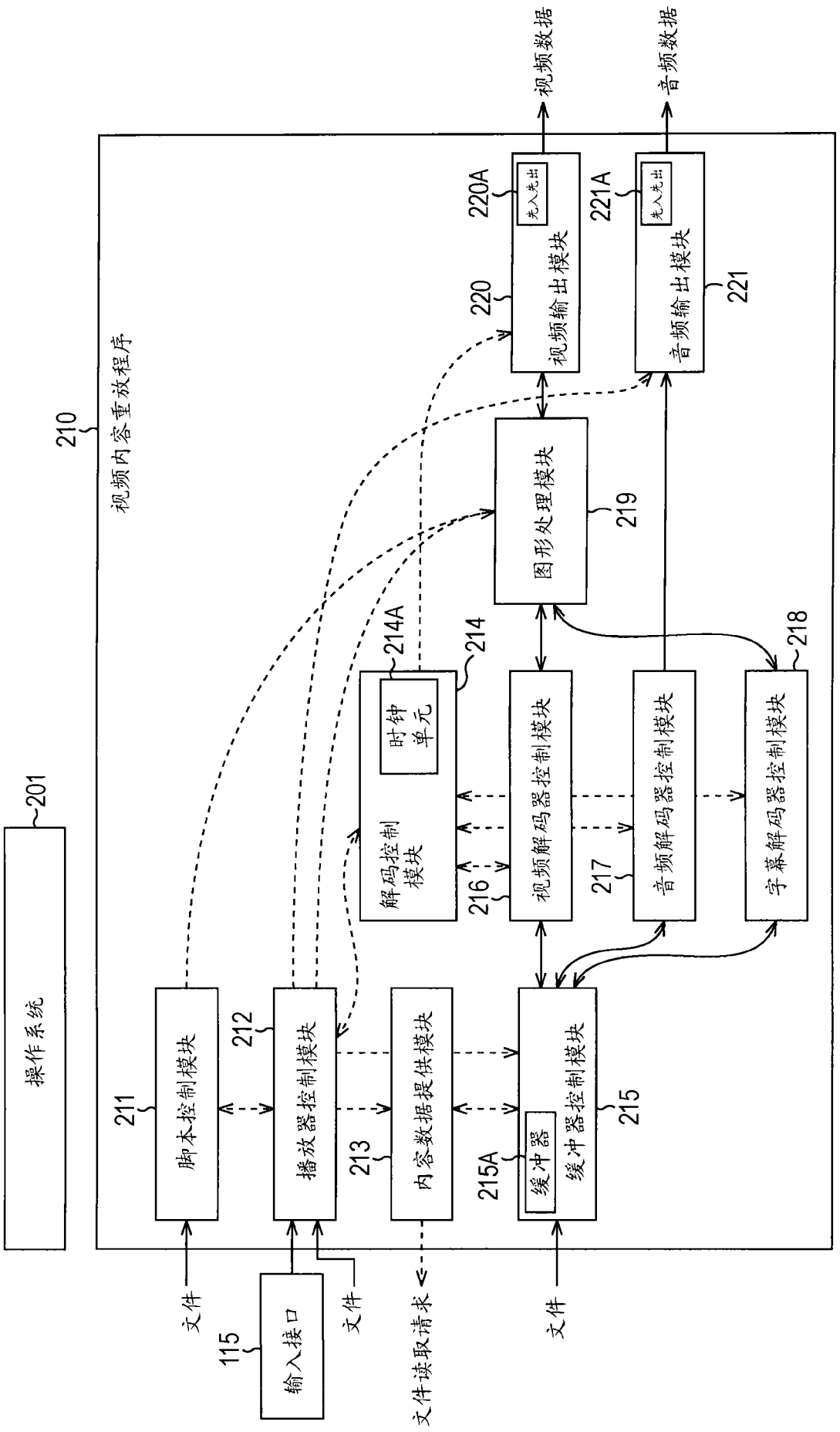

操作系统
201
视频内容重放程序
210
脚本控制模块
211
播放器控制模块
212
内容数据提供模块
213
缓冲器
缓冲器控制模块
215A
215
解码控制模块
时钟单元
214A
214
视频解码器控制模块
216
音频解码器控制模块
217
字幕解码器控制模块
218
图形处理模块
219
先入先出
视频输出模块
220A
220
先入先出
音频输出模块
221A
221
视频数据
音频数据
输入接口
115
文件
文件
文件读取请求
文件

图 2

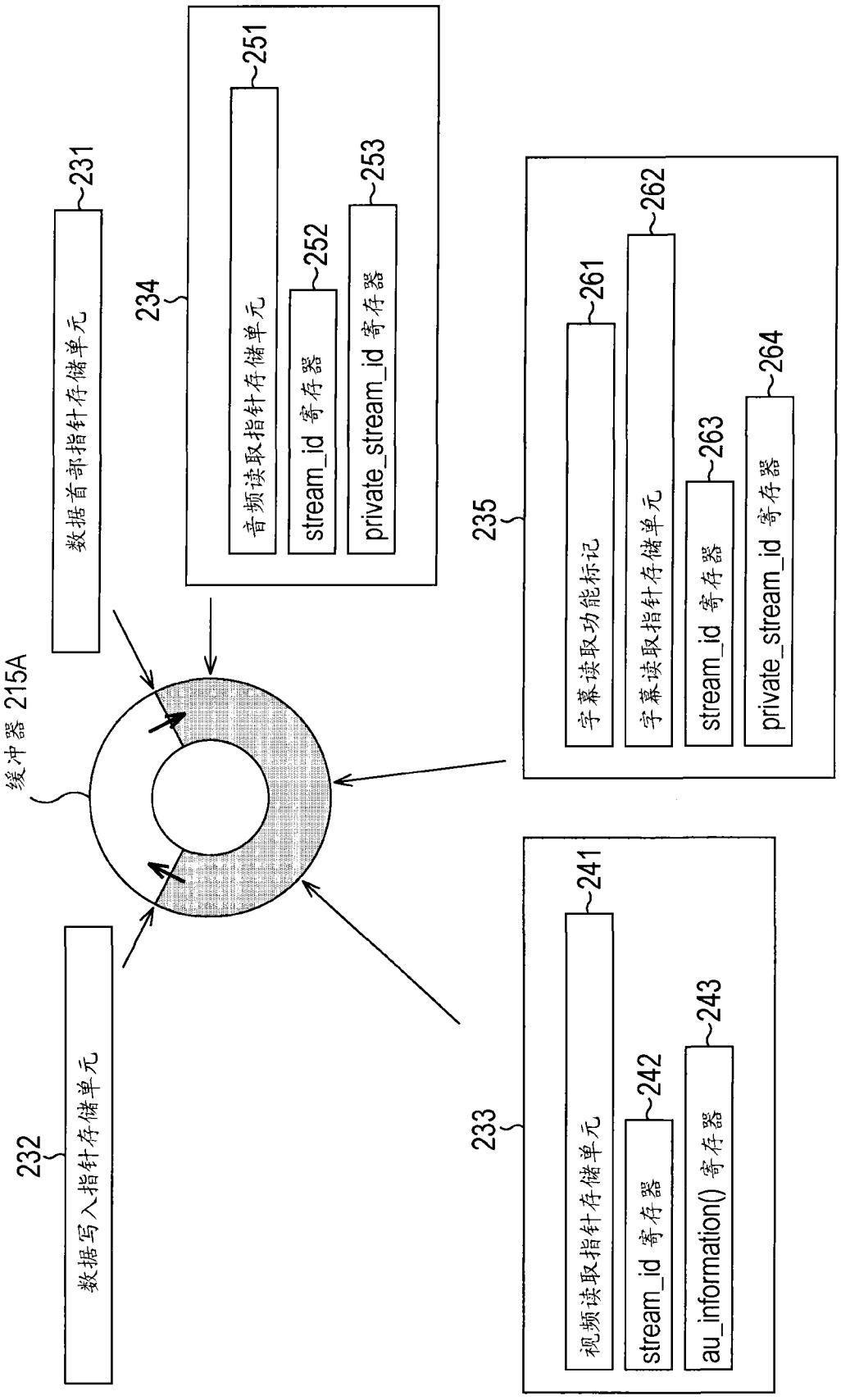

232
数据写入指针存储单元
缓冲器 215A
231
数据首部指针存储单元
234
251
音频读取指针存储单元
252
stream_id 寄存器
253
private_stream_id 寄存器
235
261
字幕读取功能标记
262
字幕读取指针存储单元
263
stream_id 寄存器
264
private_stream_id 寄存器
233
241
视频读取指针存储单元
242
stream_id 寄存器
243
au_information() 寄存器

图 3

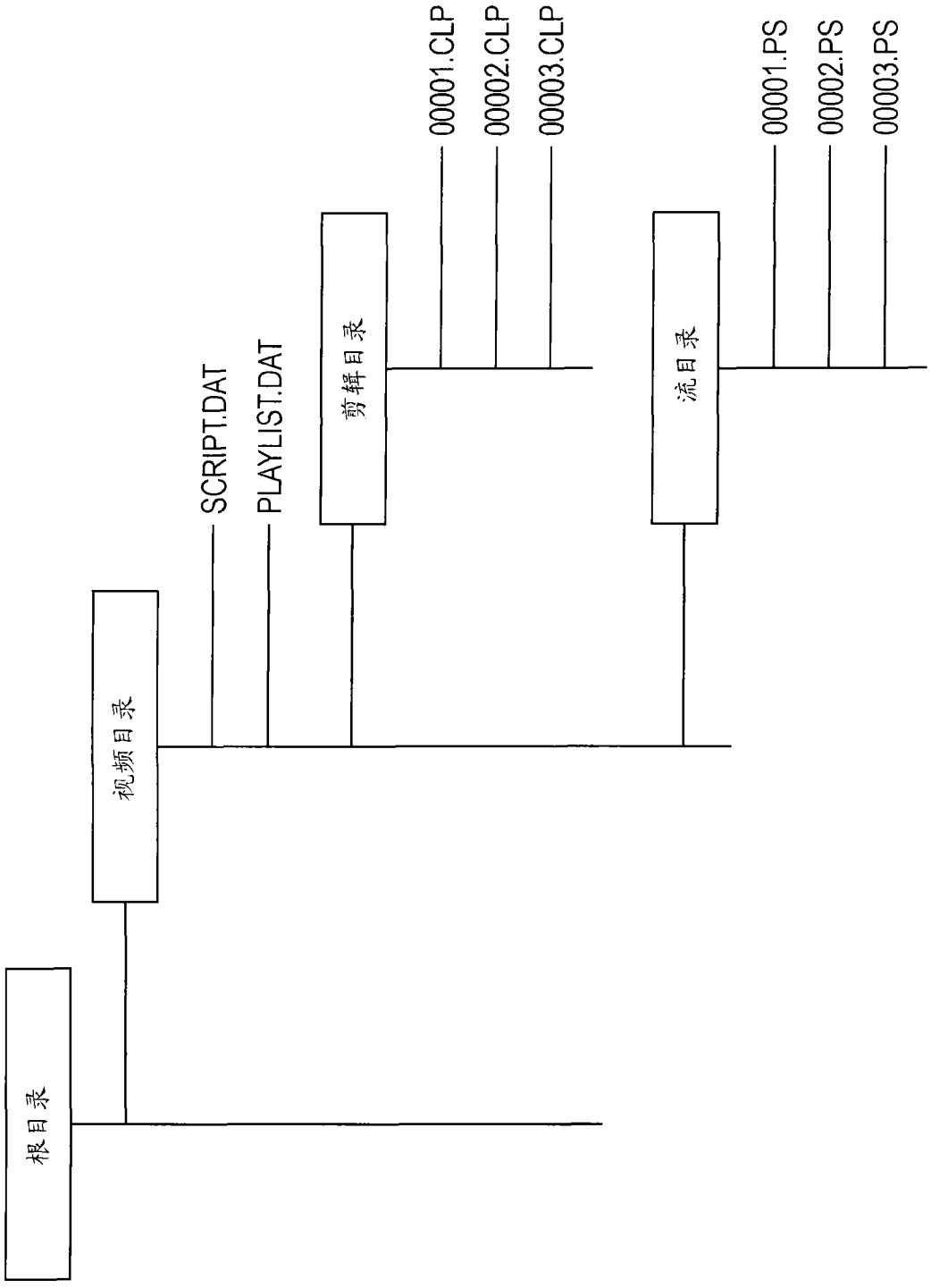

根目录
视频目录
SCRIPT.DAT
PLAYLIST.DAT
剪辑目录
00001.CLP
00002.CLP
00003.CLP
流目录
00001.PS
00002.PS
00003.PS

图 4

PlayList.PAT 文件

语法	比特数量	助记符
"PLAYLIST.DAT" {		
name_length	8	uimsbf
name_string	8x255	bslbf
number_of_PlayLists	16	uimsbf
for (i=0 ; i < number_of_PlayLists ; i++) {		
PlayList(){ // A PlayList()		
PlayList_data_length	32	uimsbf
//ATTRIBUTE INFORMATION		
reserved_for_word_alignment	15	bslbf
capture_enable_flag_PlayList	1	bslbf
PlayList_name_length	8	uimsbf
PlayList_name_string	8*255	bslbf
//		
number_of_PlayItems	16	uimsbf
for (i=0 ; i < number_of_PlayItems ; i++) {		
PlayItem()		
}		
PlayListMark()		
}		
}		
}		

图 5

PlayItem ( )

语法	比特数量	助记符
PlayItem() {		
length	16	uimsbf
Clip_Information_file_name_length	16	uimsbf
Clip_Information_file_name	8*Clip_Information_file_name_length	bslbf uimsbf uimsbf
IN_time	32	
OUT_time	32	
}		

图 6

PlayListMark ()

语法	比特数量	助记符
PlayListMark() {		
length	32	uimsbf
number_of_PlayList_marks	16	uimsbf
for (i=0 ; i<number_of_PlayList_marks ; i++) {		
Mark() {		
mark_type	8	uimsbf
mark_name_length	8	uimsbf
ref_to_PlayItem_id	16	uimsbf
mark_time_stamp	32	uimsbf
entry_ES_stream_id	8	uimsbf
entry_ES_private_stream_id	8	uimsbf
mark_data	32	bslbf
mark_name_string	8*24	bslbf
}		
}		
}		

图 7

mark_type	流编码
0	保留
1	"章节(Chapter)"标记
2	"索引(index)"标记
3	"事件(Event)"标记
4 - 255	保留

图 8

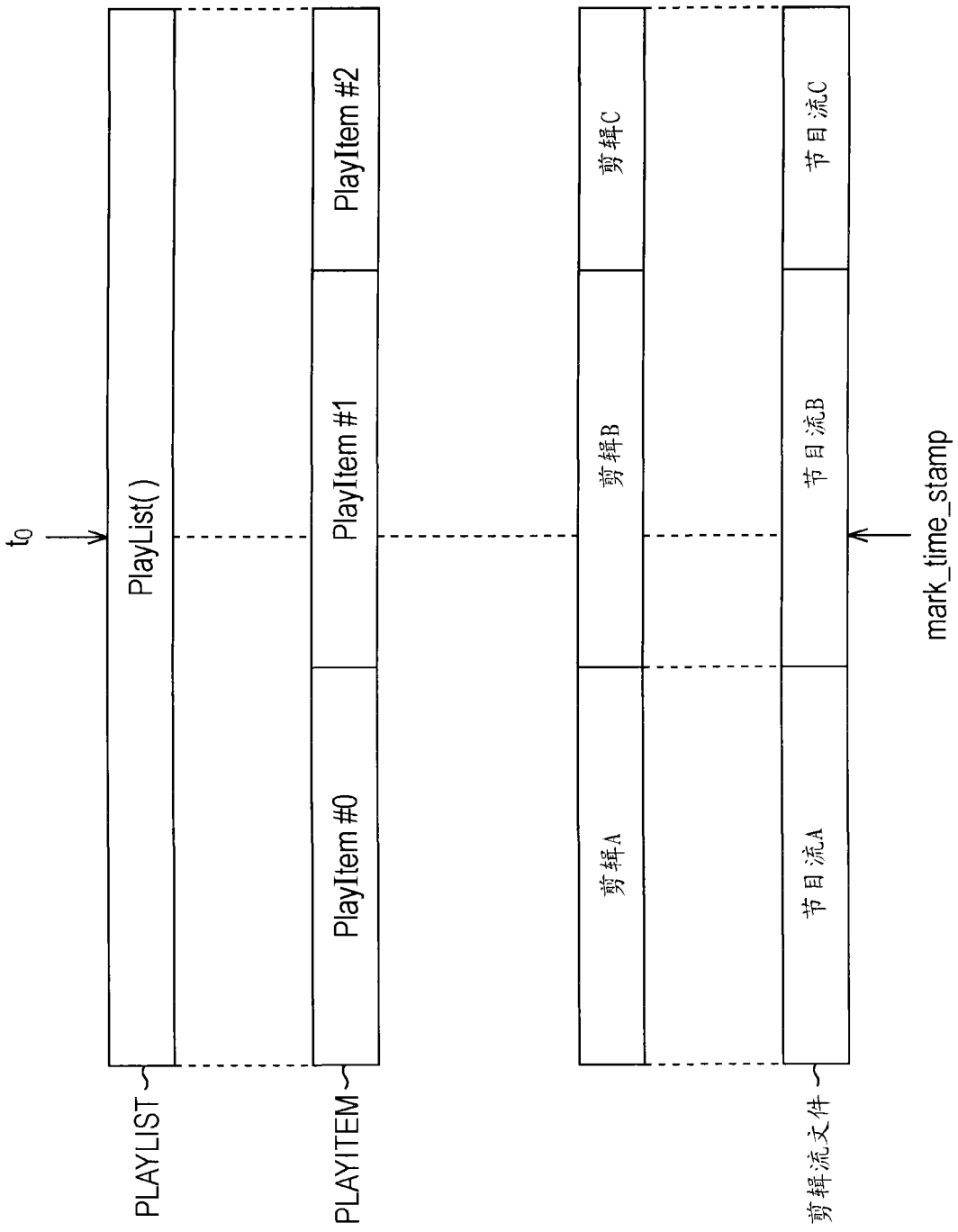

t0
PlayList( )
PlayItem #0
PlayItem #1
PlayItem #2
剪辑A
剪辑B
剪辑C
节目流A
节目流B
节目流C
mark_time_stamp
PLAYLIST
PLAYITEM
剪辑流文件

图 9

clip() "剪辑信息文件" (.CLP)

语法	比特数量	助记符
XXXXX.CLP {		
presentation_start_time	32	uimsbf
presentation_end_time	32	uimsbf
reserved_for_word_alignment	7	bslbf
capture_enable_flag_Clip	1	bslbf
number_of_streams	8	uimsbf
for (i=0 ; i < number_of_streams ; i++) {		
StreamInfo() {		
length	16	uimsbf
stream_id	8	uimsbf
private_stream_id	8	uimsbf
StaticInfo()		
reserved_for_word_alignment	8	bslbf
number_of_DynamicInfo	8	uimsbf
for (j=0 ; j < number_of_DynamicInfo ; j++) {		
pts_change_point	32	uimsbf
DynamicInfo()		
}		
}		
}		
EP_map()		
}		

图 10

基本流类型	stream_id	private_stream_id
视频 ATRAC 音频 LPCM 音频 字幕	0xE0-0xEF 0xBD 0xBD 0xBD	(none) 0x00-0x0F 0x10-0x1F 0x80-0x9F

图 11

StaticInfo ()

语法	比特数量	助记符
StaticInfo() {		
if (stream == VIDEO) {		
reserved_for_word_alignment	16	bslbf
picture_size	4	uimsbf
frame_rate	4	uimsbf
reserved_for_word_alignment	7	bslbf
cc_flag	1	bslbf
} else if (stream == AUDIO) {		
audio_language_code	16	bslbf
channel_configuration	8	uimsbf
reserved_for_word_alignment	3	bslbf
lfe_existence	1	bslbf
sampling_frequency	4	uimsbf
} else if (stream == SUBTITLE) {		
subtitle_language_code	16	bslbf
reserved_for_word_alignment	15	bslbf
configurable_flag	1	uimsbf
}		
}		

图 12

DynamicInfo

语法	比特数量	助记符
DynamicInfo (i , j) {		
reserved_for_word_alignment	8	bslbf
if (stream == VIDEO) {		
reserved_for_word_alignment	4	bslbf
display_aspect_ratio	4	uimsbf
} else if (stream == AUDIO) {		
reserved_for_word_alignment	4	bslbf
channel_assignment	4	uimsbf
} else if (stream == SUBTITLE) {		
reserved_for_word_alignment	8	bslbf
}		
}		

图 13

EP_map ()

语法	比特数量	助记符
EP_map() {		
reserved_for_word_alignment	8	bslbf
number_of_stream_id_entries	8	uimsbf
for (k=0 ; k < number_of_stream_id_entries ; k++) {		
stream_id	8	bslbf
private_stream_id	8	bslbf
number_of_EP_entries	32	uimsbf
for (i=0 ; i < number_of_EP_entries ; i++) {		
PTS_EP_start	32	uimsbf
RPN_EP_start	32	uimsbf
}		
}		
}		

图 14

表2-31 - 节目流

语法	比特数量	助记符

MPEG2_program_stream() {
do {
pack()
}while (nextbits() == pack_start_code)
MPEG_program_end_code
}
	32	bslbf

表2-32 - 节目流包

语法	比特数量	助记符

pack() {
pack_header()
while (nextbits() == packet_start_code_prefix) {
PES_packet()
}
}		

表2-33 - 节目流包报头

语法	比特数量	助记符

pack_header() {
pack_start_code
'01'
system_clock_reference_base[32..30]
marker_bit
system_clock_reference_base[29..15]
marker_bit
system_clock_reference_base[14..0]
marker_bit
system_clock_reference_extension
marker_bit
program_mux_rate
marker_bit
marker_bit
reserved
pack_stuffing_length
for (i=0 ; i < pack_stuffing_length ; i++) {
stuffing_byte
}
if (nextbits() == system_header_start_code) {
system_header()
}
}
	32
	2
	3
	1
	15
	1
	15
	1
	9
	1
	22
	1
	1
	5
	3
	8
		bslbf
		bslbf
		bslbf
		bslbf
		bslbf
		bslbf
		bslbf
		bslbf
		uimsbf
		bslbf
		uimsbf
		bslbf
		bslbf
		bslbf
		uimsbf
		bslbf

图 15

表2-17 - PES分组

语法	比特数量	助记符
PES_packet() {		
packet_start_code_prefix	24	bslbf
stream_id	8	uimsbf
PES_packet_length	16	uimsbf
if (stream_id !=program_stream_map		
&& stream_id !=padding_stream		
&& stream_id !=private_stream_2		
&& stream_id !=ECM		
&& stream_id !=EMM		
&& stream_id !=program_stream_directory		
&& stream_id !=DSMCC_stream		
&& stream_id !=ITU-T Rec. H.222.1 type E stream) {		
'10'	2	bslbf
PES_scrambling_control	2	bslbf
PES_priority	1	bslbf
data_alignment_indicator	1	bslbf
copyright	1	bslbf
original_or_copy	1	bslbf
PTS_DTS_flags	2	bslbf
ESCR_flag	1	bslbf
ES_rate_flag	1	bslbf
DSM_trick_mode_flag	1	bslbf
additional_copy_info_flag	1	bslbf
PES_CRC_flag	1	bslbf
PES_extension_flag	1	bslbf
PES_header_data_length	8	uimsbf
if (PTS_DTS_flags == '10') {		
'0010' 4 bslbf	4	bslbf
PTS [32..30]	3	bslbf
marker_bit	1	bslbf
PTS [29..15]	15	bslbf
marker_bit	1	bslbf
PTS [14..0]	15	bslbf
marker_bit	1	bslbf
}		

图 16

表2-17 - PES分组(续)

语法	比特数量	助记符
if (PTS_DTS_flags == '11') {		
'0011'	4	bslbf
PTS[32..30]	3	bslbf
marker_bit	1	bslbf
PTS [29..15]	15	bslbf
marker_bit	1	bslbf
PTS [14..0]	15	bslbf
marker_bit	1	bslbf
'0001'	4	bslbf
DTS [32..30]	3	bslbf
marker_bit	1	bslbf
DTS [29..15]	15	bslbf
marker_bit	1	bslbf
DTS [14..0]	15	bslbf
marker_bit	1	bslbf
}		
if (ESCR_flag == '1') {		
reserved	2	bslbf
ESCR_base[32..30]	3	bslbf
marker_bit	1	bslbf
ESCR_base[29..15]	15	bslbf
marker_bit	1	bslbf
ESCR_base[14..0]	15	bslbf
marker_bit	1	bslbf
ESCR_extension	9	uimsbf
marker_bit	1	bslbf
}		
if (ES_rate_flag == '1') {		
marker_bit	1	bslbf
ES_rate	22	uimsbf
marker_bit	1	bslbf
}		
if (DSM_trick_mode_flag == '1') {		
trick_mode_control	3	uimsbf
if (trick_mode_control == fast_forward) {		
field_id	2	bslbf
intra_slice_refresh	1	bslbf
frequency_truncation	2	bslbf
}		
else if (trick_mode_control == slow_motion) {		
rep_cntrl	5	uimsbf
}		
else if (trick_mode_control == freeze_frame) {		
field_id	2	uimsbf
reserved	3	bslbf
}		
else if (trick_mode_control == fast_reverse) {		
field_id	2	bslbf
intra_slice_refresh	1	bslbf
frequency_truncation	2	bslbf
else if (trick_mode_control == slow_reverse) {		
rep_cntrl	5	uimsbf
}		
else		
reserved	5	bslbf
}		
if (additional_copy_info_flag == '1') {		
marker_bit	1	bslbf
additional_copy_info	7	bslbf
}		
if (PES_CRC_flag == '1') {		
previous_PES_packet_CRC	16	bslbf
}		

图 17

表2-17 - PES分组(续)

语法	比特数量	助记符
if (PES_extension_flag == '1') {		
PES_private_data_flag	1	bslbf
pack_header_field_flag	1	bslbf
program_packet_sequence_counter_flag	1	bslbf
P-STD_buffer_flag	1	bslbf
reserved	3	bslbf
PES_extension_flag_2	1	bslbf
if (PES_private_data_flag == '1') {		
PES_private_data	128	bslbf
}		
if (pack_header_field_flag == '1') {		
pack_field_length	8	uimsbf
pack_header()		
}		
if (program_packet_sequence_counter_flag == '1') {		
marker_bit	1	bslbf
program_packet_sequence_counter	7	uimsbf
marker_bit	1	bslbf
MPEG1_MPEG2_identifier	1	bslbf
original_stuff_length	6	uimsbf
}		
if (P-STD_buffer_flag == '1') {		
'01'	2	bslbf
P-STD_buffer_scale	1	bslbf
P-STD_buffer_size	13	uimsbf
}		
if (PES_extension_flag_2 == '1') {		
marker_bit	1	bslbf
PES_extension_field_length	7	uimsbf
for (i=0 ; i<PES_extension_field_length ; i++) {		
reserved	8	bslbf
}		
}		
}		
for (i=0 ; i<N1 ; i++) {		
stuffing_byte	8	bslbf
}		
for (i=0 ; i<N2 ; i++) {		
PES_packet_data_byte	8	bslbf
}		
}		
else if (stream_id == program_stream_map		
||stream_id == private_stream_2		
||stream_id == ECM		
||stream_id == EMM		
||stream_id == program_stream_directory		
||stream_id == DSMCC_stream		
||stream_id == ITU-T Rec.H.222.1 type E stream) {		
for (i=0 ; i<PES_packet_length ; i++) {		
PES_packet_data_byte	8	bslbf
}		
}		
else if (stream_id == padding_stream) {		
for (i=0 ; i<PES_packet_length ; i++) {		
padding_byte	8	bslbf
}		
}		
}		

图 18

表2-18-Stream_id 分配

stream_id	注意	流编码
1011 1100	1	program_stream_map
1011 1101	2	private_stream_1
1011 1110		padding_stream
1011 1111	3	private_stream_2
110x xxxx		ISO/IEC 13818-3 或ISO/IEC 11172-3 或ISO/IEC 13818-7 或ISO/IEC 14496-3 音频流编号 x xxxx
1110 xxxx		ITU-T Rec. H.262 | ISO/IEC 13818-2 或ISO/IEC 11172-2 或ISO/IEC 14496-2 视频流编号 xxxx
1111 0000	3	ECM_stream
1111 0001	3	EMM_stream
1111 0010	5	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Annex A 或ISO/IEC 13818-6_DSMCC_stream
1111 0011	2	ISO/IEC_13522_stream
1111 0100	6	ITU-T Rec. H.222.1 类型A
1111 0101	6	ITU-T Rec. H.222.1 类型B
1111 0110	6	ITU-T Rec. H.222.1 类型C
1111 0111	6	ITU-T Rec. H.222.1 类型D
1111 1000	6	ITU-T Rec. H.222.1 类型E
1111 1001	7	ancillary_stream
1111 1010		ISO/IEC14496-1_SL-packetized_stream
1111 1011		ISO/IEC14496-1_FlexMux_stream
1111 1100…1111 1110		保留的数据流
1111 1111	4	program_stream_directory

符号 x 表示允许值“0”或者“1”，并且值“0”或者“1”产生同一流类型，通过由 x 采用的值来给出流编号。

注意 1——类型 program_stream_map 的 PES 分组具有在 2.5.4.1 中指定的独特语法。

注意 2——类型 private_stream_1 的 PES 分组和 ISO/IEC_13552_stream 按照与用于 ITU-T Rec. H.262|ISO/IEC 13818-2 视频和 ISO/IEC 13818-3 音频流的那些相同的 PES 分组语法。

注意 3——类型 private_stream_2、ECM_stream 和 EMM_stream 的 PES 分组类似于 private_stream_1，除了在 PES_packet_length 字段后未指定语法。

注意 4——类型 program_stream_directory 的 PES 分组具有在 2.5.5 中指定的独特语法。

注意 5——类型 DSM-CC_stream 的 PES 分组具有在 ISO/IEC 13818-6 中指定的独特语法。

注意 6——这个 stream_id 与在表 2-29 中的 stream_type 0x09 相关联。

注意 7——这个 stream_id 仅仅用于 PES 分组中，它在传送流中承载来自节目流或者 ISO/IEC 11172-1 系统流的数据(参见 2.4.3.7)。

图 19

stream_id	流编码(流类型)
1011 1101	private_stream_1
1011 1110	padding_stream
1011 1111	private_stream_2
110x xxxx	音频流编号 x xxxx
1110 xxxx	视频流编号 xxxx

图 20

语法	比特数量	助记符
private_stream1_PES_payload() {		
private_header() {		
private_stream_id	8	uimsbf
if (stream == ATRAC) {		
reserved_for_future_use	8	bslbf
AU_locator	16	uimsbf
if (stream == LPCM) {		
fs_flag	1	uimsbf
reserved_for_future_use	3	bslbf
ch_flag	4	uimsbf
AU_locator	16	uimsbf
} else if (stream == SUBTITLE) {		
reserved_for_future_use	8	bslbf
AU_locator	16	uimsbf
}		
}		
private_payload()		
}		

图 21

private_stream_id	流编码(流类型)
0000 xxxx	ATRAC 音频流编号 XXXX
0001 xxxx	LPCM 流编号 XXXX
100x xxxx	字幕流编号 X XXXX

图 22

语法	比特数量	助记符
private_stream2_PES_payload() {		
reserved_for_future_use	8	bslbf
video_stream_id	8	uimsbf
1stRef_picture	16	uimsbf
2ndRef_picture	16	uimsbf
3rdRef_picture	16	uimsbf
4thRef_picture	16	uimsbf
au_information()		
VBI()		
for (i=0 ; i<N1 ; i++) {		
padding_word	16	bslbf
}		
}		

图 23

语法	比特数量	助记符
au_information() {		
length	16	uimsbf
reserved_for_word_alignment	8	bslbf
number_of_access_unit	8	uimsbf
for (i=0 ; i<number_of_access_unit ; i++) {		
reserved	4	bslbf
pic_struct_copy	4	uimsbf
au_ref_flag	1	uimsbf
reserved	2	bslbf
AU_length	21	uimsbf
}		uimsbf
}		

图 24

// "PLAYLIST.DAT"		
number_of_PlayLists	2	
// PlayList()	PlayList #0	PlayList #1
capture_enable_flag_PlayList	1	0
number_of_PlayItems	2	1
// PlayItem #0		
Clip_Information_file_name	"00001.CLP"	"00003.CLP"
IN_time	180,090	90,000
OUT_time	27,180,090	81,090,000
// PlayItem #1		
Clip_Information_file_name	"00002.CLP"	
IN_time	90,000	
OUT_time	27,090,000	

图 25

// Clip()	"00001.CLP"	"00002.CLP"	"00003.CLP"
presentation_start_time	90,000	90,000	90,000
presentation_end_time	27,990,000	27,090,000	81,090,000
capture_enable_flag_Clip	1	0	1
number_of_streams	4	4	3
// stream #0	StreamInfo()	StreamInfo()	StreamInfo()
stream_id	0xE0 (video stream)	0xE0 (video stream)	0xE0 (video stream)
private_stream_id	0x00	0x00	0x00
	picture_size = '720x480'	picture_size = '720x480'	picture_size = '720x480'
	frame_rate = '29.97Hz'	frame_rate = '29.97Hz'	frame_rate = '29.97Hz'
	cc_flag = 'Yes'	cc_flag = 'Yes'	cc_flag = 'No'
number_of_DynamicInfo	0	0	2
// stream #1	StreamInfo()	StreamInfo()	StreamInfo()
stream_id	0xBD (private_stream_1)	0xBD (private_stream_1)	0xE1 (video stream)
private_stream_id	0x00 (ATRAC)	0x00 (ATRAC)	0x00
	audio_language_code = 'JAPANESE'	audio_language_code = 'JAPANESE'	picture_size = '720x480'
	channel_configuration = 'STEREO'	channel_configuration = 'STEREO'	frame_rate = '29.97Hz'
	life_existence = 'NO'	life_existence = 'NO'	cc_flag = 'No'
	sampling_frequency = '48KHz'	sampling_frequency = '48KHz'	
number_of_DynamicInfo	0	0	0
// stream #2	StreamInfo()	StreamInfo()	StreamInfo()
stream_id	0xBD (private_stream_1)	0xBD (private_stream_1)	0xBD (private_stream_1)
private_stream_id	0x80 (Subtitle)	0x80 (Subtitle)	0x00 (ATRAC)
	subtitle_language_code = 'JAPANESE'	subtitle_language_code = 'JAPANESE'	audio_language_code = 'JAPANESE'
	configurable_flag=0	configurable_flag=0	channel_configuration = 'STEREO'
			life_existence = 'NO'
			sampling_frequency = '48KHz'
number_of_DynamicInfo	0	0	3
// stream #3	StreamInfo()	StreamInfo()	
stream_id	0xBD (private_stream_1)	0xBD (private_stream_1)	
private_stream_id	0x81 (Subtitle)	0x81 (Subtitle)	
	subtitle_language_code = 'JAPANESE'	subtitle_language_code = 'JAPANESE'	
	configurable_flag = 1	configurable_flag = 1	
number_of_DynamicInfo	0	0	

图 26

"00001.CLP" 的 EP_map()		
number_of_stream_id_entries	1	
stream_id	0xE0 (视频流)	
private_stream_id	0x00	
	PTS_EP_start	RPN_EP_start
	90000	0
	135045	244
	180090	305
	225135	427
	270180	701

图 27

PlayList #0 PlayListMark()

number_of_PlayList_marks == 7

Mark()	mark_type	ref_to_ PlayItem_id	mark_time_ stamp	entry_ES_ stream_id	entry_ES_ private_ stream_id	mark_data		
#0	章节	0	180,090	0	0	1	顶部	顶部
#1	索引	0	5,580,090	0	0	1	1分钟	1分钟
#2	索引	0	10,980,090	0	0	2	2分钟	2分钟
#3	事件	0	16,380,090	0	0	0	3分钟	3分钟
#4	章节	1	90,000	0	0	2	顶部	5分钟
#5	索引	1	5,490,000	0	0	1	1分钟	6分钟
#6	索引	1	10,890,000	0	0	2	2分钟	7分钟

PlayList #1 PlayListMark()

number_of_PlayList_marks == 3

Mark()	mark_type	ref_to_ PlayItem_id	mark_time_ stamp	entry_ES_ stream_id	entry_ES_ private_ stream_id	mark_data		
#0	章节	0	90,000	0	0	0	顶部	
#1	事件	0	27,090,000	0xE0	0	1	5分钟	
#2	事件	0	27,540,000	0xE1	0	2	5分钟	5秒 (305 秒)

图 28

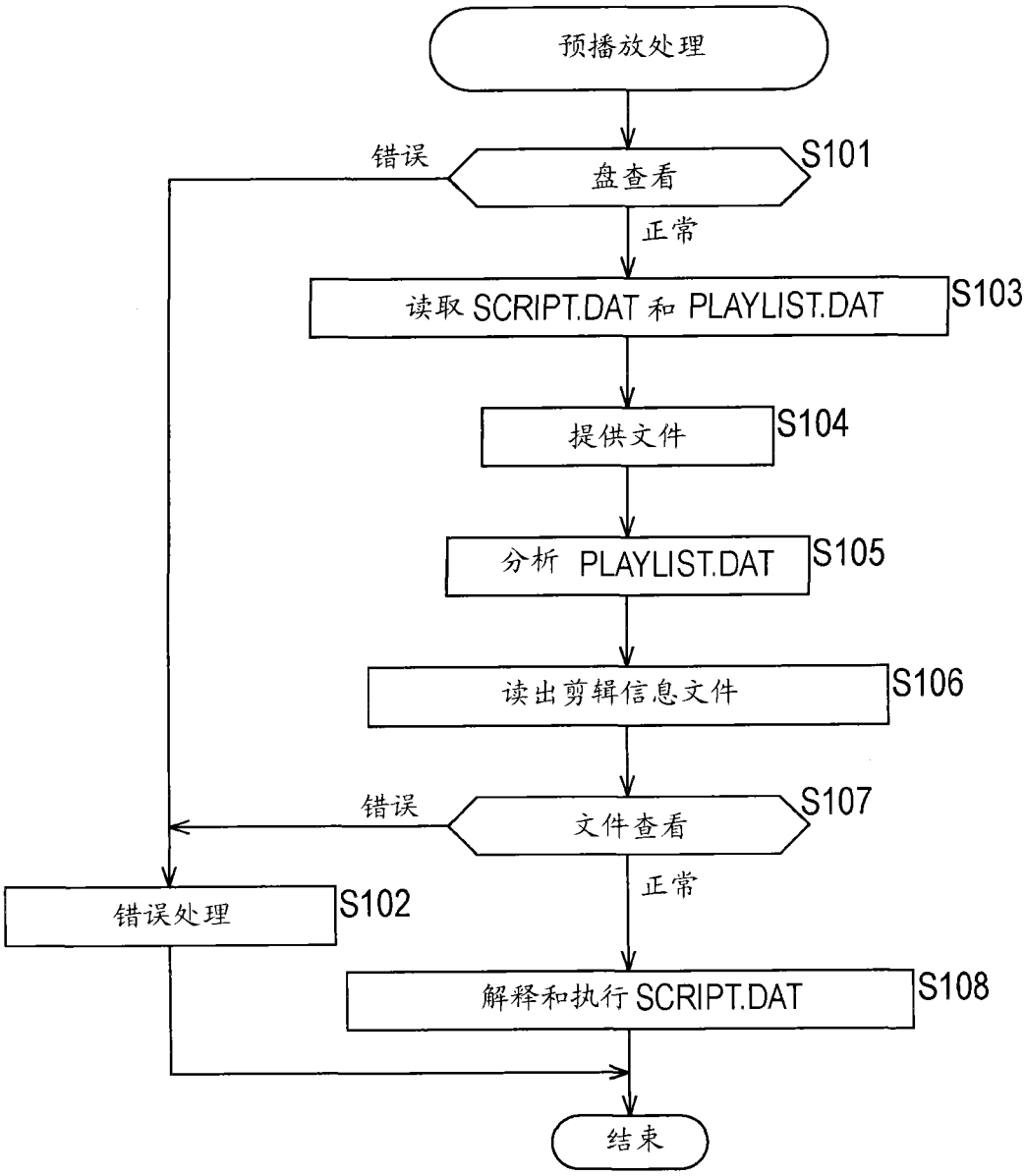

预播放处理
盘查看
S101
错误
正常
读取 SCRIPT.DAT 和 PLAYLIST.DAT
S103
提供文件
S104
分析 PLAYLIST.DAT
S105
读出剪辑信息文件
S106
文件查看
S107
错误
正常
错误处理
S102
解释和执行 SCRIPT.DAT
S108
结束

图 29

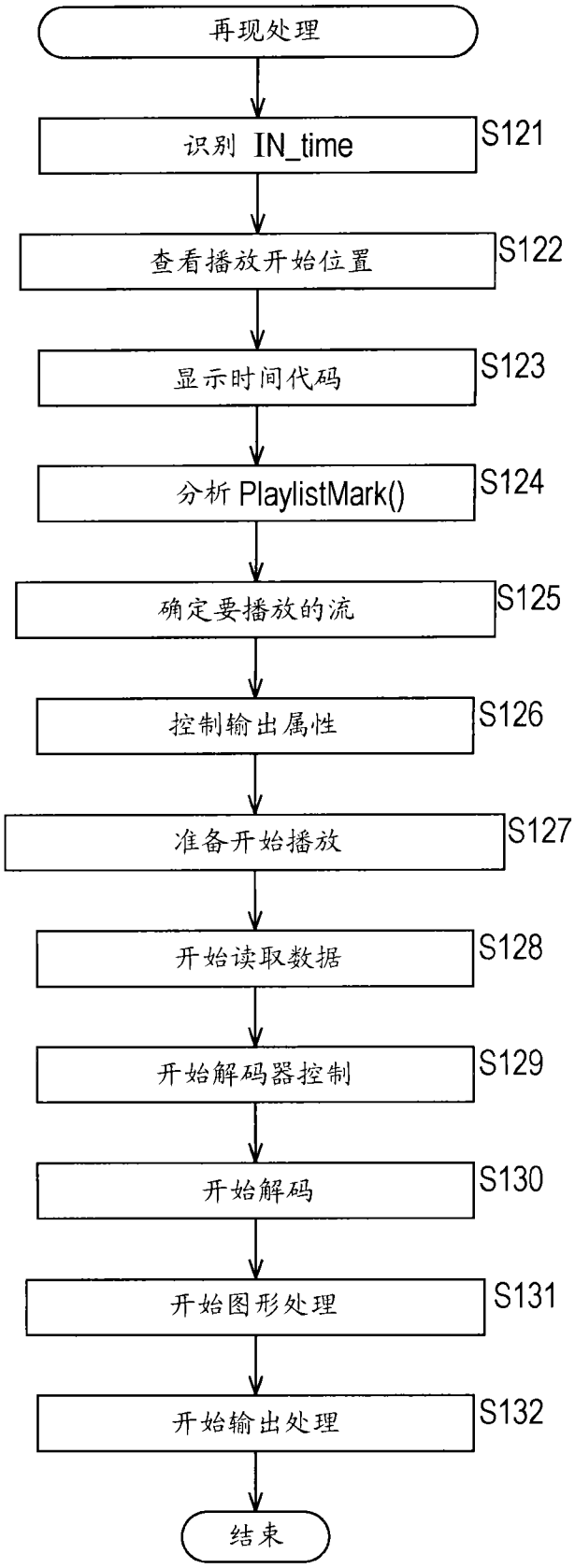

再现处理
识别 IN_time
S121
查看播放开始位置
S122
显示时间代码
S123
分析 PlaylistMark()
S124
确定要播放的流
S125
控制输出属性
S126
准备开始播放
S127
开始读取数据
S128
开始解码器控制
S129
开始解码
S130
开始图形处理
S131
开始输出处理
S132
结束

图 30

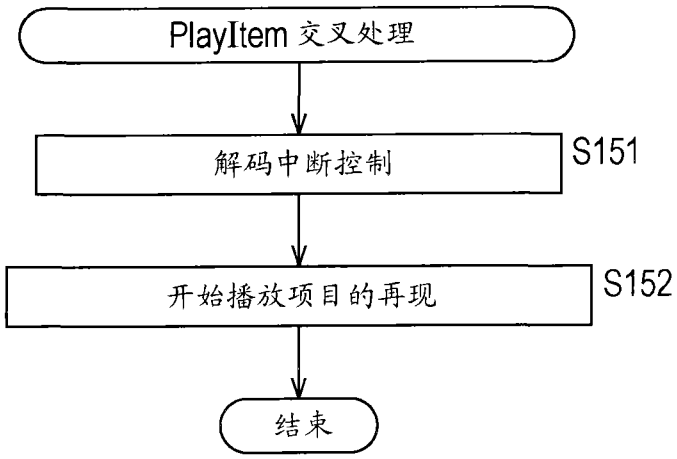

PlayItem 交叉处理
解码中断控制
S151
开始播放项目的再现
S152
结束

图 31

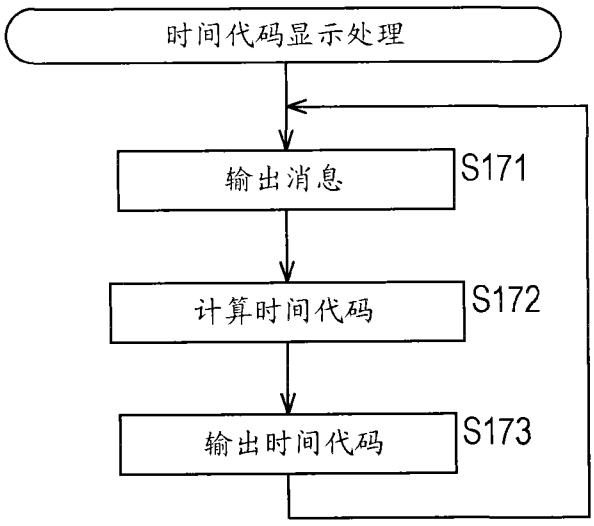

时间代码显示处理
输出消息
S171
计算时间代码
S172
输出时间代码
S173

图 32

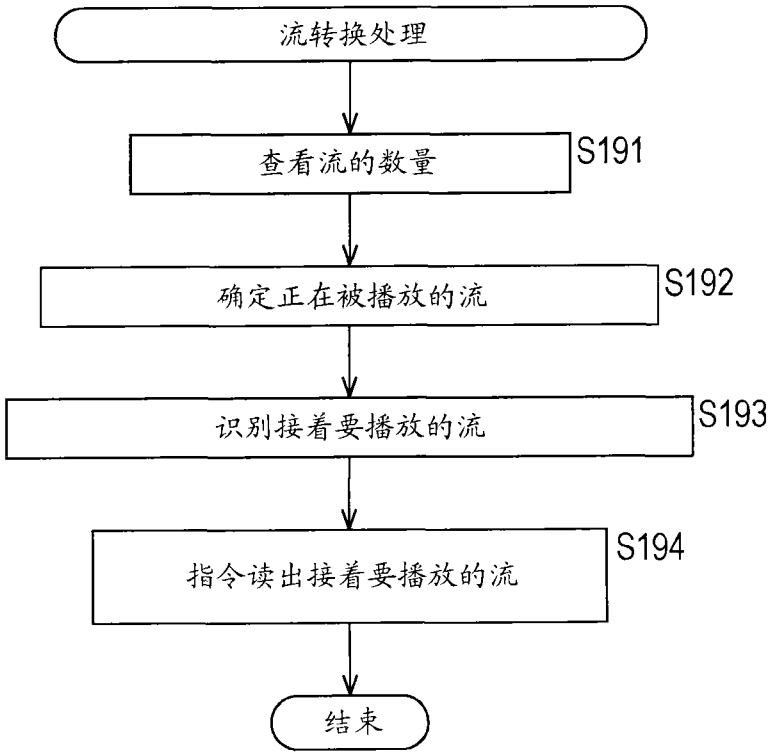

流转换处理
查看流的数量
S191
确定正在被播放的流
S192
识别接着要播放的流
S193
指令读出接着要播放的流
S194
结束

图 33

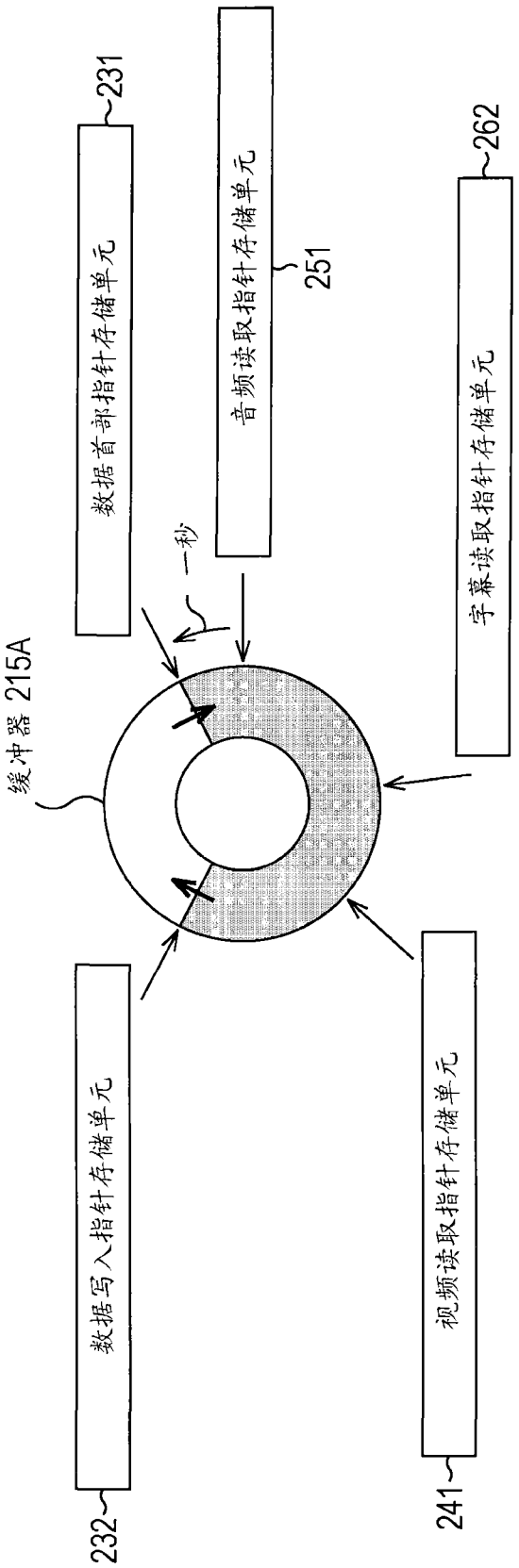

数据首部指针存储单元
231
音频读取指针存储单元
251
262
字幕读取指针存储单元
一秒
缓冲器 215A
数据写入指针存储单元
232
视频读取指针存储单元
241

图 34

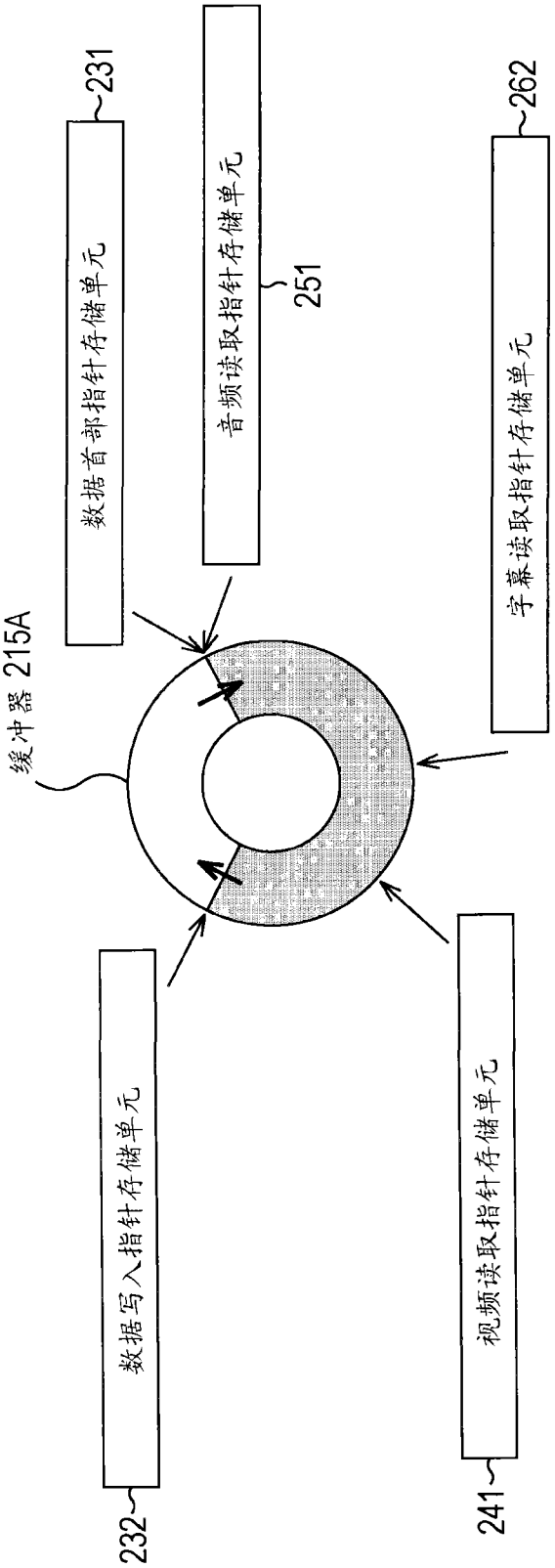

缓冲器 215A
数据首部指针存储单元
231
音频读取指针存储单元
251
字幕读取指针存储单元
262
数据写入指针存储单元
232
视频读取指针存储单元
241

图 35

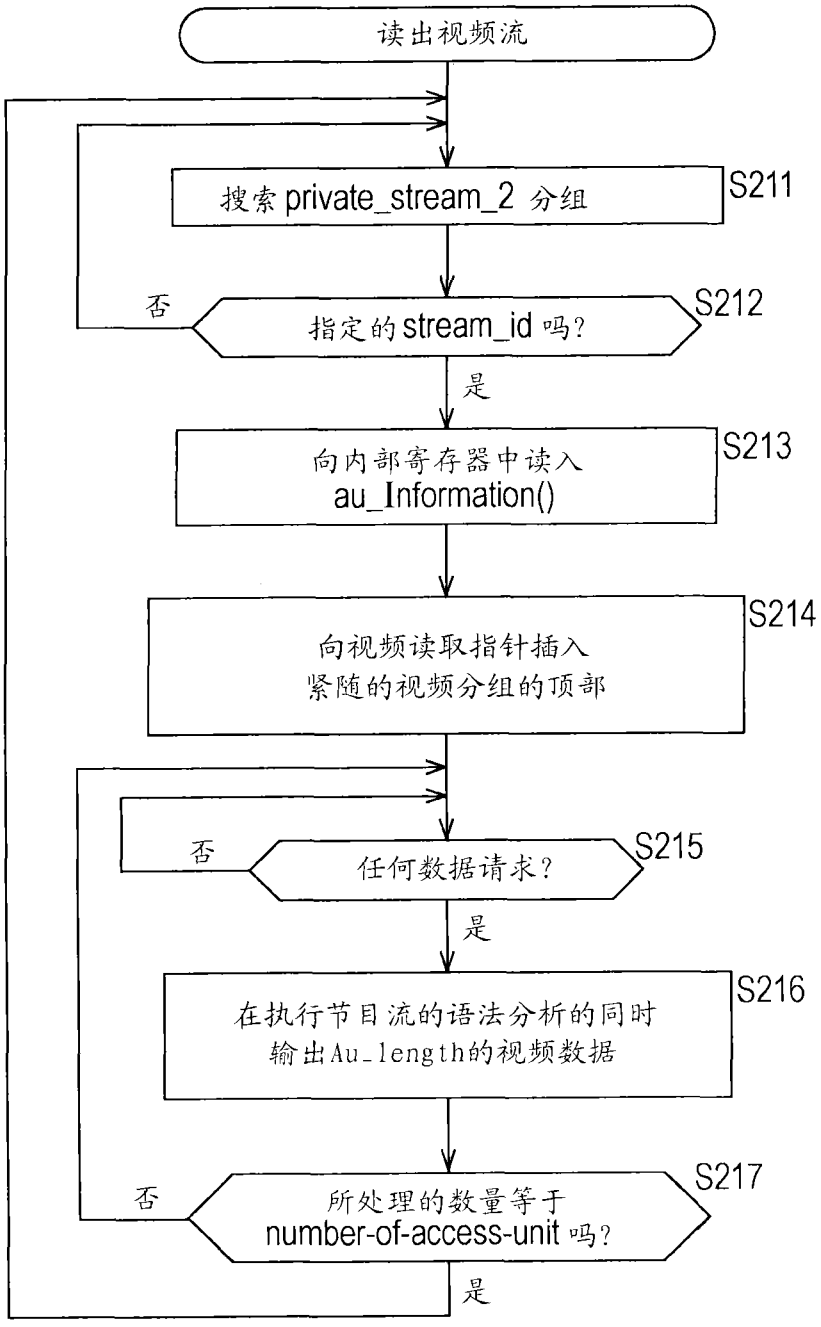

读出视频流
搜索 private_stream_2 分组
S211
否
指定的 stream_id 吗?
S212
是
向内部寄存器中读入
au_Information()
S213
向视频读取指针插入
紧随的视频分组的顶部
S214
否
任何数据请求?
S215
是
在执行节目流的语法分析的同时
输出Au_length的视频数据
S216
否
所处理的数量等于
number-of-access-unit 吗?
S217
是

图 36

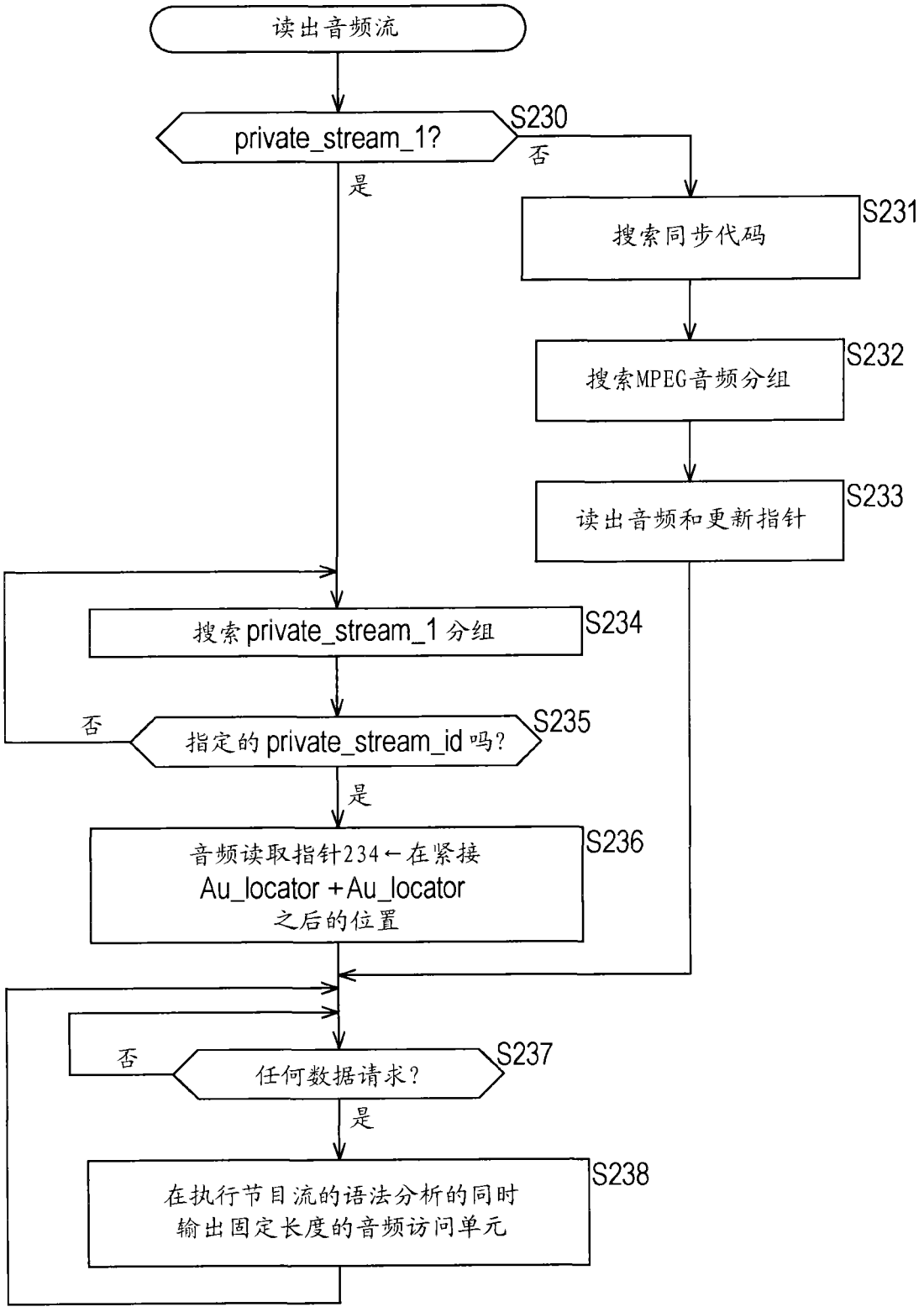

读出音频流
private_stream_1?
S230
否
是
搜索同步代码
S231
搜索MPEG音频分组
S232
读出音频和更新指针
S233
搜索 private_stream_1 分组
S234
否
指定的 private_stream_id 吗?
S235
是
音频读取指针234←在紧接
Au_locator +Au_locator
之后的位置
S236
否
任何数据请求?
S237
是
在执行节目流的语法分析的同时
输出固定长度的音频访问单元
S238

图 37

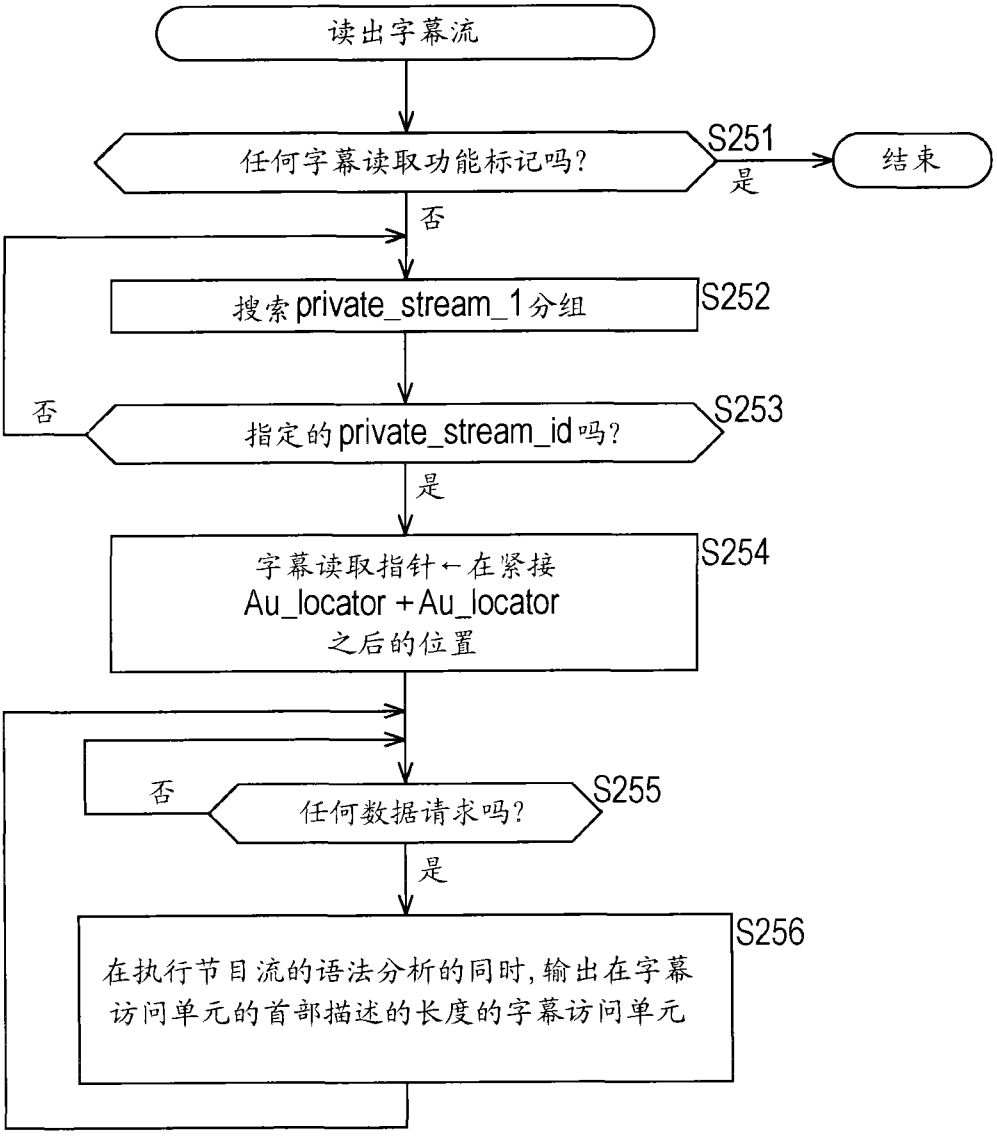

读出字幕流
任何字幕读取功能标记吗?
S251
是
结束
否
搜索private_stream_1分组
S252
否
指定的private_stream_id吗?
S253
是
字幕读取指针←在紧接
Au_locator +Au_locator
之后的位置
S254
否
任何数据请求吗?
S255
是
在执行节目流的语法分析的同时,输出在字幕访问单元的首部描述的长度的字幕访问单元
S256

图 38

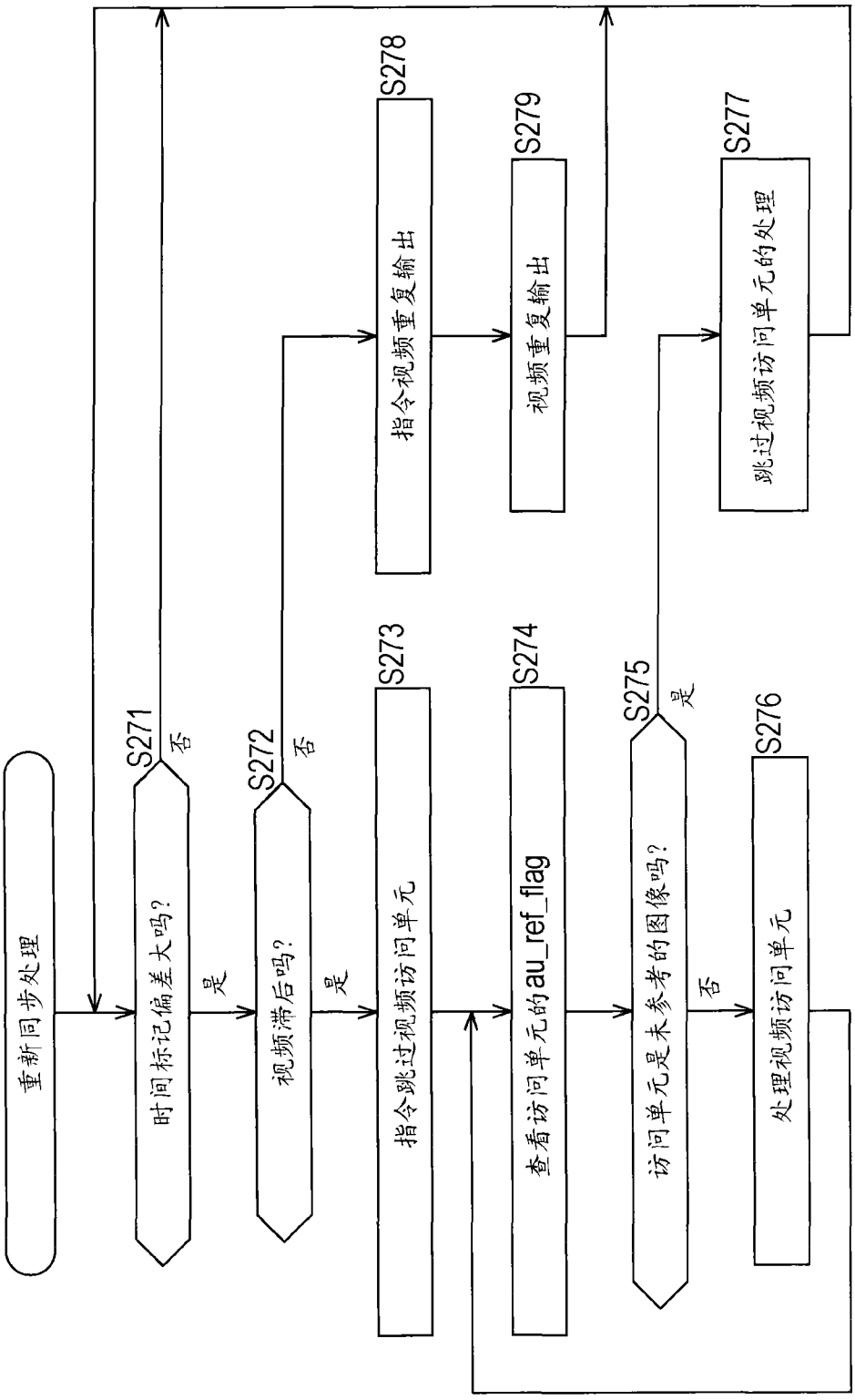

重新同步处理
S271
时间标记偏差大吗?
否
是
S272
视频滞后吗?
否
是
S273
指令跳过视频访问单元
S274
查看访问单元的au_ref_flag
S275
访问单元是未参考的图像吗?
是
否
S276
处理视频访问单元
S277
跳过视频访问单元的处理
S278
指令视频重复输出
S279
视频重复输出

图 39

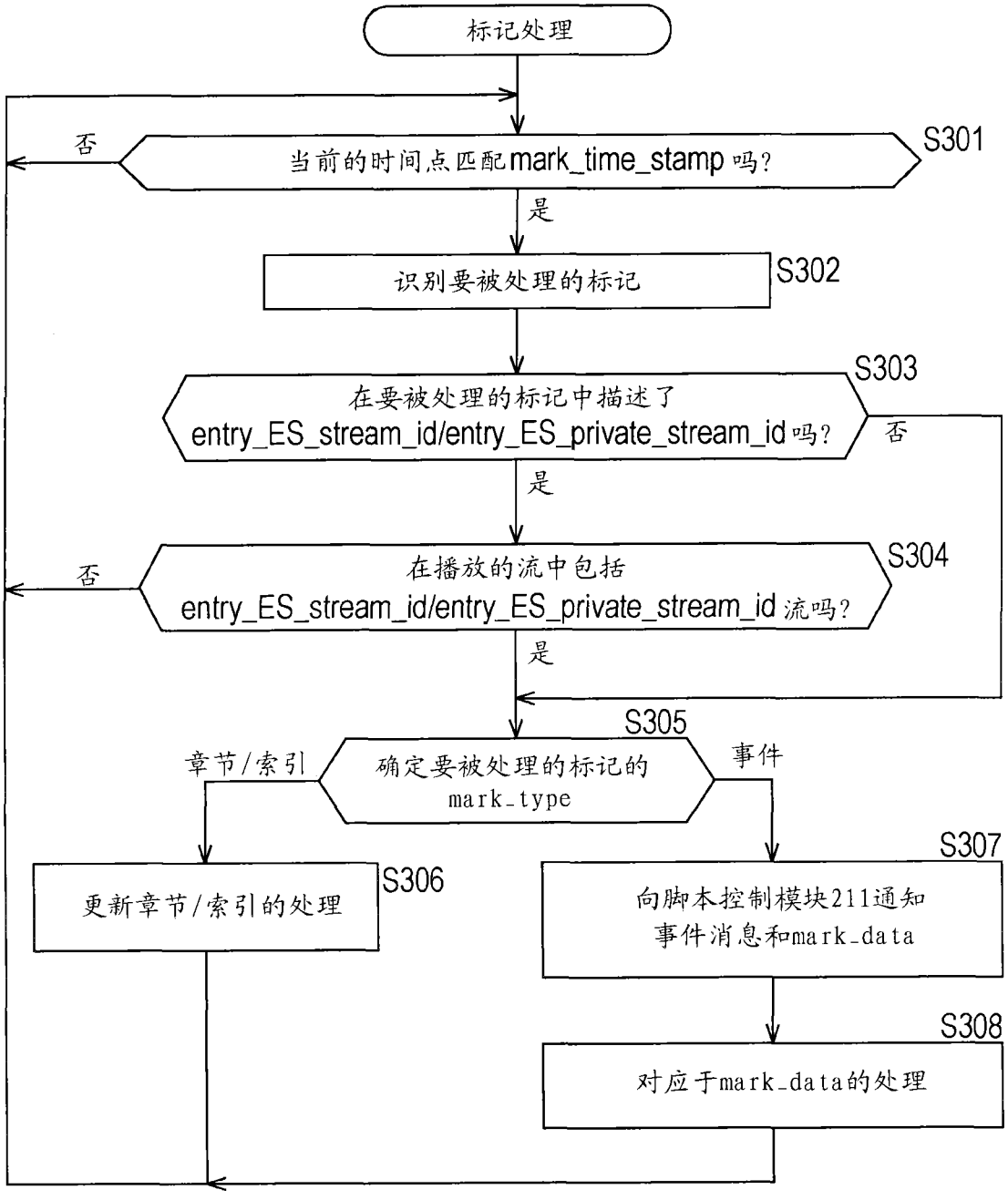

标记处理
当前的时间点匹配mark_time_stamp吗?
S301
否
是
识别要被处理的标记
S302
在要被处理的标记中描述了
entry_ES_stream_id/entry_ES_private_stream_id吗?
S303
否
是
在播放的流中包括
entry_ES_stream_id/entry_ES_private_stream_id流吗?
S304
否
是
确定要被处理的标记的
mark_type
S305
章节/索引
事件
更新章节/索引的处理
S306
向脚本控制模块211通知
事件消息和mark_data
S307
对应于mark_data的处理
S308

图 40

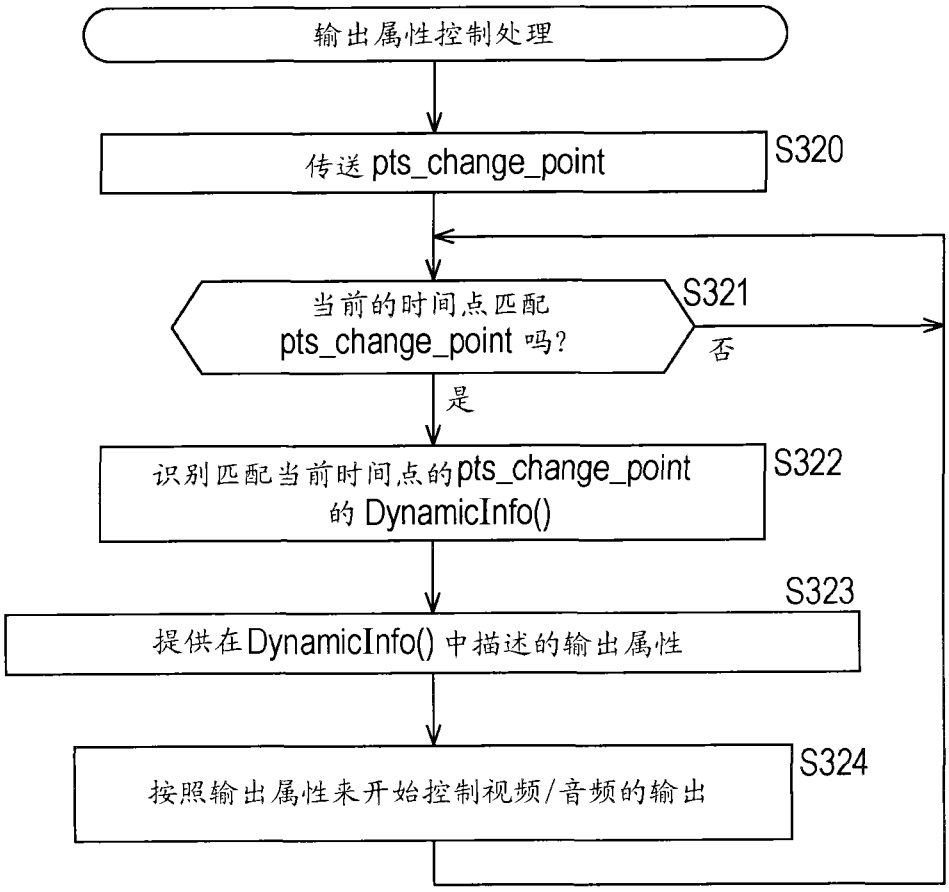

输出属性控制处理
传送 pts_change_point
S320
当前的时间点匹配
pts_change_point 吗?
S321
否
是
识别匹配当前时间点的pts_change_point
的 DynamicInfo()
S322
S323
提供在DynamicInfo() 中描述的输出属性
按照输出属性来开始控制视频/音频的输出
S324

图 41

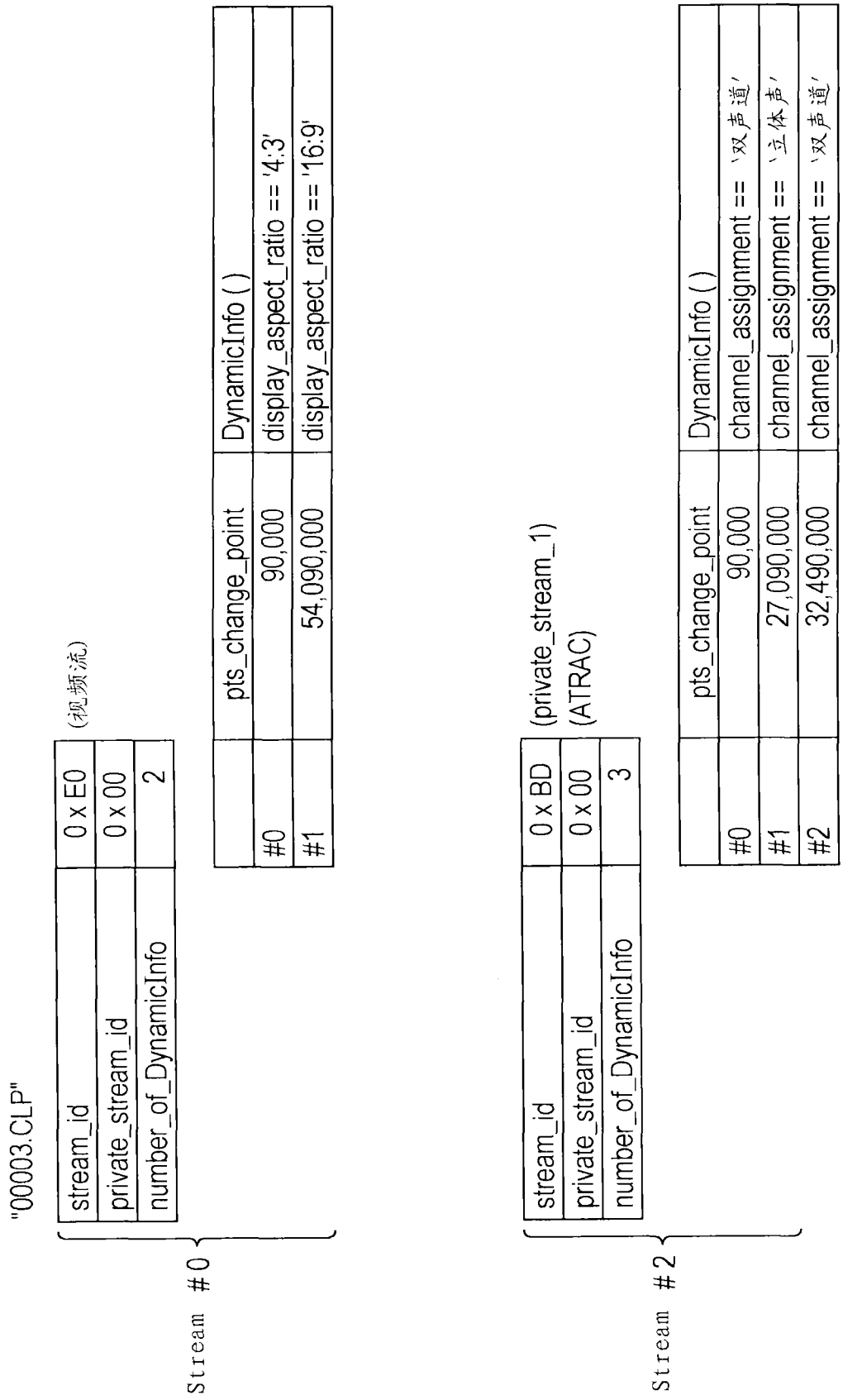

"00003.CLP"
Stream #0
stream_id
0 x E0
(视频流)
private_stream_id
0 x 00
number_of_DynamicInfo
2
pts_change_point
DynamicInfo ( )
#0
90,000
display_aspect_ratio == '4:3'
#1
54,090,000
display_aspect_ratio == '16:9'
Stream #2
stream_id
0 x BD
(private_stream_1)
private_stream_id
0 x 00
(ATRAC)
number_of_DynamicInfo
3
pts_change_point
DynamicInfo ( )
#0
90,000
channel_assignment == '双声道'
#1
27,090,000
channel_assignment == '立体声'
#2
32,490,000
channel_assignment == '双声道'

图 42

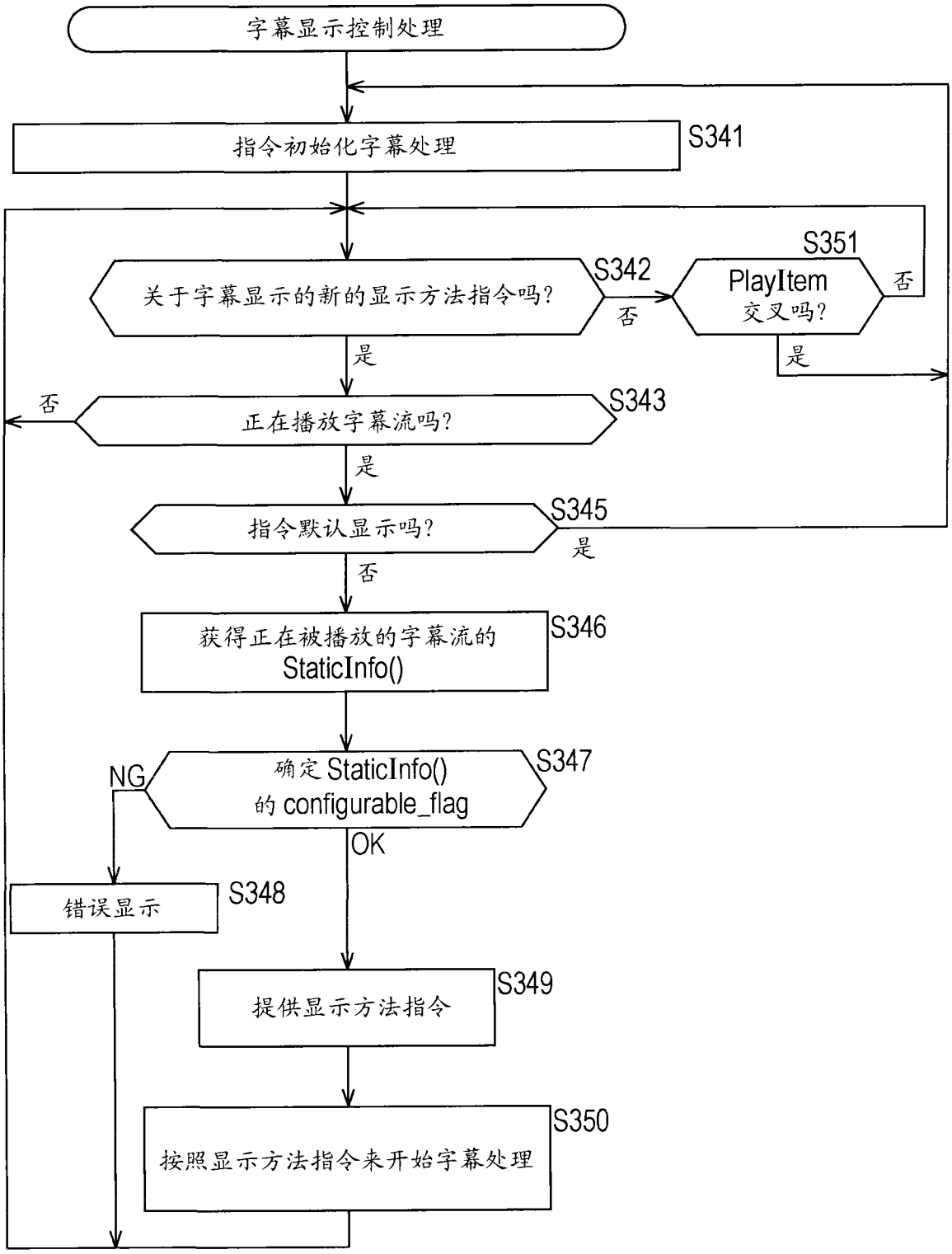

字幕显示控制处理
指令初始化字幕处理
S341
关于字幕显示的新的显示方法指令吗?
S342
否
PlayItem
交叉吗?
S351
否
是
是
否
正在播放字幕流吗?
S343
是
指令默认显示吗?
S345
是
否
获得正在被播放的字幕流的
StaticInfo()
S346
NG
确定StaticInfo()
的configurable_flag
S347
OK
错误显示
S348
提供显示方法指令
S349
按照显示方法指令来开始字幕处理
S350

图 43

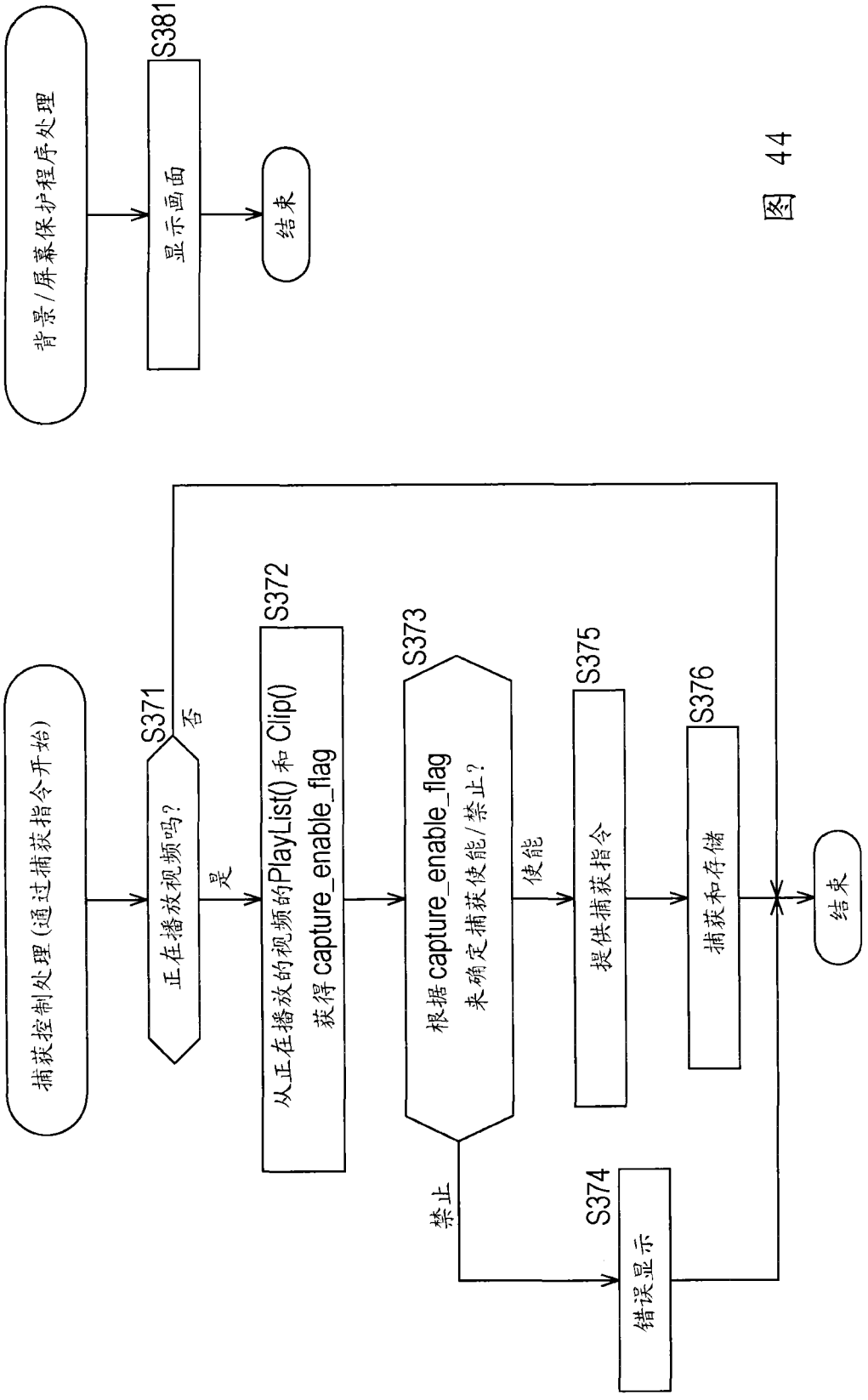

背景/屏幕保护程序处理
S381
显示画面
结束
捕获控制处理(通过捕获指令开始)
S371
正在播放视频吗?
否
是
S372
从正在播放的视频的PlayList()和Clip()
获得capture_enable_flag
S373
根据capture_enable_flag
来确定捕获使能/禁止?
使能
禁止
S375
提供捕获指令
S376
捕获和存储
S374
错误显示
结束

图 44

语法	比特数量	助记符
private_stream2_PES_payload() {		
reserved_for_future_use	7	bslbf
capture_enable_ps2	1	uimsbf
video_stream_id	8	uimsbf
1stRef_picture	16	uimsbf
2ndRef_picture	16	uimsbf
3rdRef_picture	16	uimsbf
4thRef_picture	16	uimsbf
au_information()		
grain_variance_information()		
VBI()		
for (i=0 ; i<N1 ; i++) {		
padding_word	16	bslbf
}		
}		

图 45

语法	比特数量	助记符
au_information() {		
length	16	uimsbf
reserved_for_word_align	8	bslbf
number_of_access_unit	8	uimsbf
for (i=0 ; i<number_of_access_unit ; i++) {		
reserved	3	bslbf
capture_enable_flag_AU	1	uimsbf
pic_struct_copy	4	uimsbf
au_ref_flag	1	uimsbf
reserved	2	bslbf
AU_length	21	uimsbf
}		uimsbf
}		

图 46