

[12] 发明专利申请公开说明书

[21] 申请号 98126734.3

[43]公开日 1999年6月30日

[11]公开号 CN 1221188A

[22]申请日 98.10.21 [21]申请号 98126734.3

[30]优先权

- [32]97.10.21 [33]JP [31]288180/97
- [32]97.10.21 [33]JP [31]288181/97
- [32]98.2.27 [33]JP [31]046857/98
- [32]98.2.27 [33]JP [31]046858/98
- [32]98.4.30 [33]JP [31]120389/98
- [32]98.4.30 [33]JP [31]120391/98

[71]申请人 索尼公司

地址 日本东京都

[72]发明人 浜田俊也 藤波靖

[74]专利代理机构 柳沈知识产权律师事务所

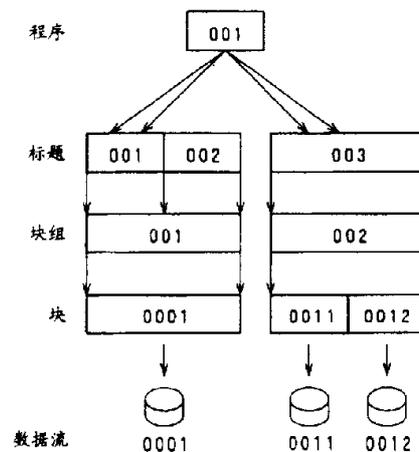
代理人 马莹

权利要求书 1 页 说明书 34 页 附图页数 39 页

[54]发明名称 信息处理装置、信息处理方法、显示介质和记录介质

[57]摘要

一种信息处理装置及其信息处理方法以及显示介质和记录介质,其中该信息处理装置包括:逐个地控制多个数据段的第一控制装置,控制任意数目的第一控制装置的第二控制装置,控制第二控制装置任何任意范围的第三控制装置;该处理方法包括步骤:逐个地控制多个数据段,控制在第一控制步骤任意数目的第一控制步骤的控制状态,和控制所说的第二控制步骤任何任意范围的控制状态。



权 利 要 求 书

1. 一种信息处理装置，包括：
逐个地(1 on 1)控制多个数据段的第一控制装置；
5 控制任意数目的第一控制装置的第二控制装置；
控制所述第二控制装置任何任意范围的第三控制装置。
2. 根据权利要求1的信息处理装置，其特征在于所说的第二控制装置控制所说的任意数目的第一控制装置，以至由所说的第一控制装置控制的数据段至少在时间轴上彼此重叠的所说的数据的部分被重放。
- 10 3. 根据权利要求1的信息处理装置，其特征在于所说的第一控制装置是所说的数据的单元，和所说的第三控制装置是由使用者所看的数据的单元。
4. 根据权利要求1的信息处理装置，其特征在于，在所说的第三控制装置或所说的第二控制装置的范围内，可以做到所说的数据的无缝重放操作。
- 15 5. 一种信息处理方法，包括如下步骤：
逐个地控制多个数据段的第一控制步骤；
控制在所述第一控制步骤的任何任意数目的控制状态的第三控制步骤；和
控制在所说的第二控制步骤的任何任意范围的控制状态的第三控制步骤。
- 20 6. 一种显示介质，用于显示程序以使信息处理装置完成包括如下的处理：
逐个地控制多个数据段的第一控制步骤；
控制在所述第一控制步骤的任意数目的控制状态的第三控制步骤；和
控制在所说的第二控制步骤的任何任意范围的控制状态的第三控制步
25 骤。
7. 一种用于记录数据和控制所说的数据的控制信息的记录介质，所说的记录介质其特征在于所说的控制信息包括：
逐个地控制多个数据段的第一控制装置；
控制任何任意数目的第一控制装置的第二控制装置；
30 控制第二控制装置任何任意范围的第三控制装置。

说明书

信息处理装置、信息处理方法、 显示介质和记录介质

5

总地说来，本发明涉及信息处理装置、信息处理方法、显示介质和记录介质。更具体地说，本发明涉及允许更加容易进行编辑的信息处理装置、信息处理方法、显示介质和记录介质。

可以适当地编辑记录在诸如光盘的记录介质上的视频和音频数据。在编辑工作中，具体地说，一段视频数据的部分区域与另一视频数据的部分结合，或者说部分视频数据的一个区域被擦除。

当进行编辑数据的通常的工作时，也就是数据本身被链接或删除，就出现了编辑包括已经编辑过的再编辑的数据时的既费工又费时的问题。

因此，致力于上述问题的本发明的一个目的是允许既容易又快捷地进行编辑。根据本发明的信息处理装置的特征在于该装置包括：

逐个地(1 on 1)控制多个段的数据的第一控制装置；
控制任何任意数目的第一控制装置的第二控制装置；和
控制在第二控制装置中的任何任意范围的第三控制装置。

根据本发明的信息处理方法的特征在于该方法包括：
20 逐个地控制多个段的数据的第一控制步骤；
控制在第一控制步骤的任何任意数目的控制状态的第二控制步骤；和
控制在第二控制步骤的任何任意范围的控制状态的第三控制步骤。

根据本发明的图像显示介质的特征在于该介质用于表现使得信息处理装置完成处理的程序，该处理包括如下步骤：

25 逐个地控制多个段的数据的第一控制步骤；
控制在第一控制步骤的任何任意数目的控制状态的第二控制步骤；和
控制在第二控制步骤的任何任意范围的控制状态的第三控制步骤。

根据本发明的用于记录数据和控制数据的控制信息的记录介质特征在于，控制信息包括：

30 逐个地控制多个段的数据的第一控制装置；
控制任何任意数目的第一控制装置的第二控制装置；和



控制在第二控制装置中的任何任意范围的第三控制装置。

在本发明的信息处理装置中，多个数据段由第一控制装置逐个地控制，和任何任意数目的第一控制装置由第二控制装置控制，其中任何任意的范围由第三控制装置控制。

5 在根据本发明的信息处理方法和根据本发明的图像显示介质中，在第一控制步骤的任何任意数目的控制状态在第二步被控制，和在第二控制步骤的任何任意范围的控制状态在第三步被控制。

在本发明的记录介质中，记录如控制信息，第一控制装置，第二控制装置和第三控制装置。

10 结合下面的附图描述本发明的优选实施例：

图 1 是用于描述目录编排的解释图；

图 2 是用于描述 VOLUME.TOC 文件的结构的解释图；

图 3 是用于描述 volume_information()(卷_信息())的结构解释图；

图 4 是用于描述 volume_attribute()(卷_属性())的结构解释图；

15 图 5 是用于描述 resume_()(恢复_())的结构解释图；

图 6 是用于描述 volume_rating()(卷_等级())的结构解释图；

图 7 是用于描述 write_protect()(写_保护())的结构解释图；

图 8 是用于描述 play_protect()(播放_保护())的结构解释图；

图 9 是用于描述 recording_timer()(记录_定时器())的结构解释图；

20 图 10 是用于描述 text_block()(文本_块())的结构解释图；

图 11 是用于描述 language_set()(语言_设定())的结构解释图；

图 12 是用于描述 text_item()(文本_项目())的结构解释图；

图 13 是用于描述 ALBUM.STR 的结构解释图；

图 14 是用于描述 album()(册())的结构解释图；

25 图 15 是用于描述 TLTLE_###.VDR 的结构解释图；

图 16 是用于描述 title_info()(标题_信息())的结构解释图；

图 17 是用于描述 PROGRAM_\$\$\$PGI 的结构解释图；

图 18 是用于描述 program()(程序())的结构解释图；

图 19 是用于描述 play_list()(播放_目录())的结构解释图；

30 图 20 是用于描述 play_item()(播放_项目())的结构解释图；

图 21 是用于描述 CHUNKGROUP_###.CGIT 的结构解释图；



图 22 是用于描述 chunk_connection_info() (块_连接_信息()) 的结构解释图；

图 23 是用于描述 chunk_arrangement_info() (块_安排_信息()) 的结构解释图；

5 图 24 是用于描述 CHUNK_%%%_ABST 的结构解释图；

图 25A、25B 是表示本发明应用的光盘装置的典型结构的方框图；

图 26 是用于描述目录编排的解释图；

图 27 是用于描述目录的逻辑编排的解释图；

图 28 是用于描述偏移的解释图；

10 图 29 是用于描述目录的编排的解释图；

图 30 是用于描述目录的编排的解释图；

图 31 是用于描述目录的编排的解释图；

图 32 是用于描述目录的编排的解释图；

图 33 是用于描述目录的编排的解释图；

15 图 34 是用于描述目录的逻辑的编排的解释图；

图 35 是用于描述目录的逻辑的编排的解释图；

图 36 是用于描述 file_type_id(文件_类型_识别符)的解释图；

图 37 是用于描述 mark_type(标记_类型)的解释图；

图 38 是用于描述 chunk_group(块组)的解释图；

20 图 39 是用于描述 chunk_group(块组)的解释图；

图 40 是用于描述 file_type_id(文件_类型_识别符)的解释图；

图 41 是用于描述 chunk_sync_play_flag(块__同步__播放__标记)的解释图；

25 图 42 是用于描述 original_time_count_type(原始_时间_计数_类型)的解释图；

图 43 是用于描述 file_type_id(文件_类型_识别符)的解释图；

图 44 是用于描述 info_type(信息_类型)的解释图；

图 45 是用于描述 slot_unit_type(时隙__单位__类型)的解释图；

图 46 是用于描述 file_type_id(文件_类型_识别符)的解释图；

30 图 47 是用于描述 program_status(程序_状态)的解释图；

图 48 是表示解释分开标题处理的流程图；



图 49 是用于描述程序、标题、块(chunk)组、多个块(chunks)和数据流的分层的解释图；

图 50 是用于解释交换(swap)标题的处理的解释图；

图 51 是用于描述程序、标题、块组，多个块和数据流的分层的解释图；

5 图 52 是用于解释删除标题的处理的流程图；

图 53 是用于描述删除标题的处理的解释图；

图 54 是用于解释合并标题的处理的流程图；

图 55 是表示解释设置操作为重放程序的处理流程图；

图 56 是用于描述程序、标题、块组、多个块和数据流的分层的解释图；

10 图 57 是用于解释重放程序的处理的流程图；

图 58 是用于描述程序结构的解释图；

图 59 是用于描述播放序列的结构解释图；

图 60 是用于描述程序、播放序列和播放项目间的关系的解释图。

15 在描述本发明的实施例之前，在说明书的权利要求中所指出的每个装置是由一个典型的实施装置来举例说明的，在下面描述本发明的特征中，这些典型的实施装置被放置在每个装置后面的括号里，以便说明这些装置与实施装置之间的关系。然而不用说，这些实施装置并不意味着具有限制意义，也就是说，这些装置的例子不限于附加的实施装置。

本发明的信息处理装置的特征在于该装置包括：

20 逐段地控制多个数据段的第一控制装置(通过在图 24 中所示的 CHUNK-%%%.ABST 具体地执行)。

控制任何任意数目的第一控制装置的第二控制装置(通过图 21 中所示的 CHUNK-###.CGIT 具体地执行)，和

25 控制在第二控制装置中的任何任意的范围的第三控制装置(通过图 15 中所示的 TITLE-###, VDR 具体执行)。

利用解释在由本发明提供一个记录介质上的文件的格式来开始下面的描述，其中信息被记录在该记录介质上并且从该记录介质上来播放该信息。在图 1 中所述记录在记录介质上的文件被分类成下列七种形式。

VOLUME.TOC

30 ALBUM.STR

PROGRAM_\$\$\$.PGI



TITLE_###.VDR

CHUNKGROUP_@@@.CGIT

CHUNK_%%%.ABST

CHUNK_%%%.MPEG2

5 VOLUME.TOC 和 ALBUM.STR 文件被放置在一个根目录中。一个称为
 “ PROGRAM ” 的目录被设置的在根目录的下面。该 PROGRAM 目录包括
 PROGRAM_\$\$\$.PGI 文件，其中符号 \$\$\$ 表示一个 program(程序)的
 号。同样地，一个称为 “ TITLE ” 的目录被设置在根目录的下面。该 TITLE
 (标题)目录包括 TITLE_###.VDR 文件,其中符号 ### 表示一个 TITLE
 10 的号。一个称为 “ CHUNKGROUP ” 的目录被设置在根目录的下面。该
 CHUNKGROUP 目录包括 CHUNKGROUP_###.CGIT 文件，其中符号 @
 @@ 表示一个 chunkgroup 的号。一个称为 “ CHUNK ” 的目录被设置在根目
 录的下面。该 CHUNK 目录包括 CHUNK_%%%.ABST 文件，其中符号 %
 %%% 表示一个 chunk (块) 的号。

15 同样地，一个称为 “ MPEGAV ” 的目录被设置在根目录的下面。该
 MPEGAV 目录包括多个子目录，每个子目录包括 CHUNK_%%%.MPEG2
 文件，其中符号 %%% 表示一个 chunk 的号。

 通常在记录介质中存在一个 VOLUME.TOC 文件。然而在具有一种特殊
 结构的记录介质中，例如像具有 ROM 和 RAM 混合结构这样的记录介质中，
 20 可以存在多个 VOLUME.TOC 文件。这个 VOLUME.TOC 文件用于表示记录
 介质整个的特性。

 图 2 是表示 VOLUME.TOC 文件的结构图。如在图中所示，在文件的开
 头设置 file_type_id，以便指示该文件是一个 VOLUME.TOC 文件。跟随着
 file_type_id 是 volume_information(), 最后是 text_block()。

25 图 3 是表示 volume_information() 文件的结构图。如在图中所示，
 volume_information() 包括 volume_attribute()、 resume()、 volume_rating()、
 write_protect()、 play_protect() 和 recording_timer()。

 volume_attribute() 是一个用于记录逻辑卷的属性的区域。图 4 是一个表示
 volume_attribute() 的详细结构图。如在图中所示， volume_attribute() 包括
 30 title_play_mode_flag 和 promgram_playback_mode_flag。

 resume() 是一个区域，该区域被用于在记录介质被再次插入时记录在一



个弹出操作之前用于一个状态恢复的信息。图 5 是表示 resume()的一个详细结构图。

在图 3 中所示的 volume_rating()是一个区域，根据用户的年龄和用户的类型，该区域记录用于执行整个卷的监视器/收听器的年龄限制的信息。图 6 5 是表示 volume_rating()的详细结构图。

图 3 中的 write_protect()是一个区域，该区域记录用于限制改变和删除一个 title 和一个 program 操作的信息。图 7 是表示 write_protect()的详细结构图。

图 3 中的 play_protect()一个区域，该区域记录用于设置一个播放允许功能或一个播放禁止功能的信息和记录用于限制在该卷中记录的一个 title 或一个 program 的播放操作数的信息。图 8 是表示 play_protect()的详细结构图。 10

在图中所示的 recording_timer()是一个区域，该区域用于记录控制一个记录时间的信息。图 9 是表示 recording_timer()的详细结构图。

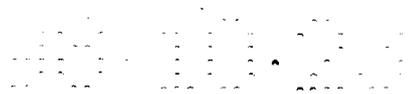
图 10 是一个表示在图 2 中所示的 VOLUME.TOC 文件的 text_block()的详细结构图。如在图 10 中所示的， text_block()包括 language_set 和 text_item。图 11 和 12 分别示出了 language_set 和 text_item 的详细结构图。 15

通常在一个记录介质中存在一个在图 1 中所示的 ALBUM.STR 文件。然而在具有一种特殊结构的记录介质中，例如像具有 ROM 和 RAM 混合结构这样的记录介质中，可以存在多个 ALBUM.STR 文件。这个 ALBUM.STR 文件用于把多个记录介质组合成一个结构，该结构使记录介质看起来好象只有一个单个记录介质。 20

图 13 是表示 ALBUM.STR 文件的结构图。如在图中所示，在文件的开头设置 file_type_id，以便指示该文件是一个 ALBUM.STR 文件。跟随着 file_type_id 是 album()，最后是 text_block()。

album()是一个区域，该区域记录用于处理多个 volume 的信息，也就是，多个记录介质作为一个综合 volume。图 14 是表示 album()的一个详细结构图。 25

在图 1 中存在与多个 title 一样多的 TITLE_###.VDR 文件。一个 title 例如在一个光盘的情况下涉及一个音乐的选择或在一个电视广播的情况下涉及一个节目。图 15 是表示 TITLE_###.VDR 文件的结构图。如在图中所示，在文件的开头设置 file_type_id，以便指示该文件是一个 TITLE_###.VDR 文件。跟随着 file_type_id 是 title_info()，最后是 text_block()。符号# 30



是表示一个 title 号的字符串。

title_info() 是一个区域，该区域记录在一个 chunk group 上的 title 一个开始点和一个结束点和其它的 title 的属性。图 16 是表示 title_info() 的详细结构图。

5 在图 1 中存在与多个 program 一样多的 PROGRAM_\$\$\$.PGI 文件。一个 program 包括多个删节，每个删节规定了一部分区域或所有的 title。一个 program 的删节以一个规定的顺序被播放。图 17 是表示一个 PROGRAM_\$\$\$.PGI 文件的结构图。如在图中所示，在文件的开头设置 file_type_id，以便指示该文件是一个 PROGRAM_\$\$\$.PGI 文件。跟随着 file_type_id 是
10 program()，最后是 text_block()。符号 \$\$\$ 是表示一个 program 号的字符串。

program() 是一个区域，该区域用于记录在一个操作中所需要信息以便收集需要的一部分 title 并且播放它们而不完成该内容的不可逆的编辑。图 18 是表示 program() 的详细结构图。

15 在图 18 中所示的 program() 具有一个 play_list()。图 18 是表示 program() 的详细结构图。

在 play_list() 中设置有多个 play_item()。图 20 是一个表示 play_item() 的详细图。

20 在图 1 中存在与多个 chunk group 一样多的 CHUNKGROUP_### .CGIT 文件。一个 chunk group 是一个用于排列一个位流的数据结构。当用户通常操作一个用于在记录介质上记录信息和从记录介质播放信息的装置时，例如一个 VDR (Video Disc Recorder)，用户不知道这个文件的存在。

图 21 是一个表示一个 CHUNKGROUP_### .CGIT 文件的结构图。如在图中所示，在文件的开头设置 file_type_id，以便指示该文件是一个
25 CHUNKGROUP_### .CGT 文件。跟随着 file_type_id 是 chunkgroup_time_base_flags (块组_时_基_标志) 和 chunkgroup_time_base_offset (块组_时_基_偏移)，其次是 chunk_connection_info()，最后是 text_block()。

30 chunkgroup_time_base_flags 是特征位，每个特征位与一个 chunkgroup 的一个基准计数器相关。chunkgroup_time_base_offset 表示一个 chunk group 的基准时间轴的开始时间。chunkgroup_time_base_offset 是在一个计数器中设置的 32 位值，该计数器以一个 90kHz 的频率来计算。chunk_connection_info()



是一个区域，用于记录像一个视频转换点、以及视频和音频同步这样的特殊信息。图 22 是一个表示 `chunk_connection_info()` 的详细结构图。

`chunk_connection_info()` 包括与属于一个 chunk group 的 chunk 的数量一样多的 `chunk_arrangement_info()` 的循环。图 23 是一个表示 `chunk_arrangement_info()` 的详细结构图。

在图 1 中存在与多个 chunks 一样多的 `CHUNK_%%%.ABST` 文件。一个 chunk 是用于一个数据流文件的信息。图 24 是一个表示 `CHUNK_%%%.ABST` 文件的结构图。如在图中所示，在文件的开头设置 `file_type_id`，以便指示该文件是一个 `CHUNK_%%%.ABST` 文件。

10 在图 1 中所示的 `CHUNK_%%%.MPEG2` 文件是一个数据流文件。与只用于存储信息的其它文件不同，这个文件被用于存储一个 MPEG 位流。

图 25 表示一个光盘装置的典型结构图，它用于把信息记录在作为一个记录介质的光盘上和由该光盘来播放信息，其中记录介质包括上述的文件。在光盘装置中，设置一个单个光头 2 用于一个可重写光盘 1。光头 2 用于从光盘 1 读出信息和把信息写入到光盘 1 中。

15 在一个 RF 和解调/调制电路 3 中一个位流被解调之后，在借助于一个开关 5 被提供给用于缓冲在一个读出率与一个解码处理率之间的差的一个读出通道缓冲器 6 之前，由光头 2 从光盘 1 读出的位流先经过在 ECC 电路 4 中的误差校正。读出通道缓冲器 6 的一个输出被提供给一个解码器 7。读出通道缓冲器 6 这样被设计以致于一个系统控制器 13 能够读和写该读出通道缓冲器 6。

25 利用解码器 7 对由读出通道缓冲器 6 输出的位流进行解码，作为解码的结果，该解码器 7 输出视频和音频信号。由解码器 7 输出的视频信号被提供给一个合成电路 8 以便与一个由 OSD(On Screen Display)控制电路 9 产生的视频信号合成。合成的结果通过一个输出端 P1 提供给一个显示单元，以便在显示单元上被显示，该显示单元在图中没有被示出。同时，由解码器 7 产生的音频信号通过一个输出端 P2 提供给一个扬声器，以便在扬声器中被播放，该扬声器在图中没有被示出。

30 另一方面，由一个输入端 P3 输入的视频信号和由一个输入端 P4 输入的音频信号在它们被提供给一个写入通道缓冲器 11 之前由一个编码器 10 来编码，其中写入通道缓冲器 11 用于缓冲在一个编码处理率与一个写入率之间的



差。写入通道缓冲器 11 这样被设计以致于系统控制器 13 能够读和写该写入通道缓冲器 11。

从写入通道缓冲器 11 中读出在写入通道缓冲器 11 中存储的数据并且借助于开关 5 提供给 ECC 电路 4。在 ECC 电路 4 中，在该数据被提供给 RF & 解调/调制电路 3 以便被调制之前，一个误差校正码被加到该数据上。由 RF & 解调/调制电路 3 输出的一个信号，严格地说，一个 RF 信号利用光头 2 记录到光盘 1 上。

一个地址检测电路 12 检测在经过一个记录或播放操作的光盘 1 的磁道上关于一个地址的信息。系统控制器 13 控制构成光盘装置的部件的操作。该系统控制器 13 包括：一个 CPU 21，用于执行各种控制、一个 ROM 单元 22，用于存储像处理程序这样由 CPU 21 执行的信息、一个 RAM 单元 23，用于存储像由 CPU 21 进行的处理期间获得的数据这样的信息和一个 RAM 单元 24，用于存储待记录到光盘 1 中或待由光盘 1 播放的各种信息文件。CPU 21 根据由地址检测电路 12 输出的检测结果精细地调整光头 2 的位置。CPU 21 也控制开关 5 的转换操作。由用户来操作由各种开关和各种按钮构成的输出单元 14，以便把各种指令输入给光盘装置。

下面来解释从一个信息文件中读数据的基本操作。在从一个 VOLUME.TOC 信息文件中读数据的操作中，例如，在系统控制器 13 中使用的 CPU 21 利用一个文件系统操作指令首先确定 VOLUME.TOC 文件被记录在光盘 1 中的实际地址和文件的长度，该文件系统操作指令被包括在预先的一个处理程序中。然后，CPU 21 根据在 VOLUME.TOC 文件的地址上的信息把光头 2 移动到一个读出位置。接着，CPU 21 把光头 2、FR & 解调/调制电路 3 和 ECC 电路 4 设置到读出方式，并且根据读出通道缓冲器 6 来定位开关 5。此外，在利用光头 2 开始一个读出操作之前，CPU 21 精确地调整光头 2 的位置。在读出操作中，利用光头 2 读出 VOLUME.TOC 文件的内容和利用 FR & 解调/调制电路 3 对其进行解调。在 FR & 解调/调制电路 3 的输出被存储在读出通道缓冲器 6 之前要经过在 ECC 电路 4 中的误差校正。

当存储在读出通道缓冲器 6 中的数据量变为等于或超过 VOLUME.TOC 文件的容量时，CPU 21 停止读出操作。以后，CPU 21 读出在读出通道缓冲器 6 中存储的数据并且把该数据存储在 RAM 单元 24 中。

下面通过以 VOLUME.TOC 信息文件作为例子来解释把数据写入到一个



信息文件中的基本操作。首先，CPU 21 为了把数据被写入到其中的一个自由区域而检索文件系统，也就是光盘 1，该自由区域具有等于或大于一个 VOLUME.TOC 文件的容量，并且利用一个包括在预先的一个处理程序中的文件系统操作指令来确定该自由区域的地址。

5 接着，CPU 21 把在 RAM 单元 24 中准备好并且待重新写入到光盘 1 中的 VOLUME.TOC 文件传送给写入通道缓冲器 11。然后，CPU 21 根据在 VOLUME.TOC 文件上的信息把光头 2 移动到一个写入位置。接着，CPU 21 把光头 2、FR & 解调/调制电路 3 和 ECC 电路 4 设置到写入方式，并且根据写入通道缓冲器 11 来定位开关 5。此外，在利用光头 2 开始一个写入操作之前，CPU 21 精确地调整光头 2 的位置。

15 在写入操作中，从写入通道缓冲器 11 读出重新准备的 VOLUME.TOC 文件的内容和借助于开关 3 把该内容提供给 ECC 电路 4。在 ECC 电路 4 中，在利用 FR & 解调/调制电路 3 对其进行调制之前一个误差校正码被加到该内容上。利用光头 2 把由 FR & 解调/调制电路 3 的输出的一个信号记录到光盘 1 中。

 当从写入通道缓冲器 11 中读出的数据量变为等于或超过 VOLUME.TOC 文件的容量时，CPU 21 停止写入操作。

20 最后，CPU 21 重写一个指向文件系统的 VOLUME.TOC 文件的指示符，以便利用一个包括在预先的一个处理程序中的文件系统操作指令使指示符指向重新被写的位置。

 下面以在图 1 中所示的 CHUNK_0001.MPEG2 作为一个例子来解释播放一个数据流的基本操作。首先，在系统控制器 13 中使用的 CPU 21 利用一个文件系统操作指令确定 CHUNK_0001.MPEG2 文件被记录在光盘 1 中的实际地址和文件的长度，该文件系统操作指令被包括在预先的一个处理程序中。

25 然后，CPU 21 根据在 CHUNK_0001.MPEG2 文件的地址上的信息把光头 2 移动到一个读出位置。接着，CPU 21 把光头 2、FR & 解调/调制电路 3 和 ECC 电路 4 设置到读出方式，并且把开关 5 定位在读出通道缓冲器 6 的一侧上。此外，在利用光头 2 开始一个读出操作之前，CPU 21 精确地调整光头 2 的位置。

30 在读出操作中，利用光头 2 读出的 CHUNK_0001.MPEG2 文件的内容经过 FR & 解调/调制电路 3、ECC 电路 4 和开关 5 被存储在读出通道缓冲器 6



中。在读出通道缓冲器 6 存储的数据被提供给解码器 7，以便利用解码器 7 对由读出通道缓冲器 6 输出的数据进行解码，作为解码的结果，该解码器 7 输出视频和音频信号。由解码器 7 产生的音频信号提供给输出端 P2。同时，由解码器 7 输出的视频信号通过合成电路 8 提供给输出端 P2。

5 当从光盘 1 读出并且由解码器 7 解码以便待精确显示的数据量变为等于 CHUNK_0001.MPEG2 文件的容量时，或当由输出单元 14 接收到一个停止读出操作的指令时，CPU 21 停止读出和解码操作。

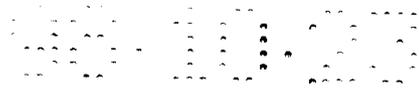
10 下面通过以 CHUNK_0001.MPEG2 信息文件作为例子来解释把数据流记录到一个信息文件中的基本操作。首先，CPU 21 为了把数据流被写入到其中的一个自由区域而检索文件系统，也就是光盘 1，该自由区域具有等于或大于 CHUNK_0001.MPEG2 文件的容量，并且利用一个包括在预先的一个处理程序中的文件系统操作指令来确定该自由区域的地址。

15 由一个输入端 P3 输入的视频信号和由一个输入端 P4 输入的音频信号在它们被提供给一个写入通道缓冲器 11 之前由一个编码器 10 来编码。CPU 21 根据在 CHUNK_0001.MPEG2 文件的地址上的信息把光头 2 移动到一个写入位置。接着，CPU 21 把光头 2、FR & 解调/调制电路 3 和 ECC 电路 4 设置到写入方式，并且根据写入通道缓冲器 11 来定位开关 5。此外，在利用光头 2 开始一个写入操作之前，CPU 21 精确地调整光头 2 的位置。

20 在写入操作中，从写入通道缓冲器 11 读出重新准备的 CHUNK_0001.MPEG2 文件的内容和借助于开关 5、ECC 电路 4、和 FR & 解调/调制电路 3 提供给光头 2。一个由 RF & 解调/调制电路 3 输出的一个信号利用光头 2 记录到光盘 1 上。

25 当由写入通道缓冲器 11 读出并且记录到光盘 1 中的数据量变为等于 CHUNK_0001.MPEG2 文件的容量时，或当由输出单元 14 接收到一个停止写入操作的指令时，CPU 21 停止写入操作。最后，CPU 21 重写一个指向文件系统的 CHUNK_0001.MPEG2 文件的指示符，以便利用一个包括在预先的一个处理程序中的文件系统操作指令使指示符指向重新被写的位置。

30 假设在图 26 中所示的信息和数据流文件已经被存储在光盘 1 中。在这个例子中，光盘 1 包括一个称为文件，该文件用于存储一个 program。此外，光盘 1 也包括三个分别称为 TITLE_001.VDR、TITLE_002.VDR 和 TITLE_003.VDR 的 title 文件。



此外，光盘 1 也包括二个称为 CHUNKGROUP_0.001.CGIT 和 CHUNKGROUP_002.CGIT 的 chunk group 信息文件。在其首位上，光盘 1 包括三个称为 CHUNK_0001.MPEG2 、 CHUNK_0011.MPEG2 和 CHUNK_0012.MPEG2 的流文件以及分别与 CHUNK_0001.MPEG2 、
5 CHUNK_0011.MPEG2 和 CHUNK_0012.MPEG2 的流文件相关的三个称为 CHUNK_0001.ABST 、 CHUNK_0011.ABST 和 CHUNK_0012.ABST 的信息文件。

图 27 是一个表示光盘 1 的逻辑结构图，该光盘 1 包括在图 26 中所示的信息和流文件。在这个例子中，称为 CHUNK_0001.ABST 、
10 CHUNK_0011.ABST 和 CHUNK_0012.ABST 的 chunk 信息文件分别规定了 CHUNK_0001.MPEG2 、 CHUNK_0011.MPEG2 和 CHUNK_0012.MPEG2 的流文件。具体地说，在图 24 中所示的每个 CHUNK_%%%.ABST 的一个 chunk_file_id 字段中，与数据流文件相关的文件 ID 被确定。

此外，在这个例子中，称为 CHUNKGROUP_001.CGIT 的 chunk-group
15 信息文件规定了称为 CHUNK_0001.ABST 的 chunk 信息文件，而称为 CHUNKGROUP_002.CGIT 的 chunk-group 信息文件规定了称为 CHUNK_0011.ABST 和 CHUNK_0012.ABST 的 chunk 信息文件。具体地说，在图 23 中所示的 chunk_arrangement_info() 的 chunk_info_file_id 字段中，规定了一个 chunk 信息的文件 ID 。这个 chunk_arrangement_info() 被包括在一个
20 chunk-group 信息文件中。在该数据结构中存在与属于一个 chunk-group 的 chunks 这样多的 chunk_arrangement_info() 。应该指出的是：在图 22 中所示的 chunk_connection_info() 中描述了 chunk_arrangement_info() 并且在图 21 中所示的 chunkgroup_###.cgit 中描述了这个 chunk_connection_info() 。

在 CHUNKGROUP_001 中只有一个 chunk_arrangement_info() 。这个
25 chunk_arrangement_info() 的 chunk_info_file_id 字段规定了 CHUNK_0001 。另一方面，在 CHUNKGROUP_002 中有二个 chunk_arrangement_info() 。这些 chunk_arrangement_info() 的 chunk_info_file_id 字段分别规定了 CHUNK_0011 和 CHUNK_0012 。因此，一个 chunk group 能够被用于规定一个顺序，在该顺序中，多个 chunks 将被播放。

30 具体地说，首先，利用在图 21 中所示的 chunkgroup_###.cgit 文件中的 chunkgroup_time_base_offset 来确定一个定时器用于 chunk-group 的初始



值。然后，当每个 chunk 被分类时，在图 23 中所示的 `chunk_arrangement_info()` 的 `presentation_start_cg_time_count`（显示_开始_归类_时间_计数）和 `presentation_end_cg_time_count`（显示_结束_归类_时间_计数）被确定。

例如，假设 `CHUNK_0011` 和 `CHUNK_0012` 的时间长度分别是 A 和 B，
5 如在图 28 中所示的。在这种情况下，`CHUNK_0011` 的 `presentation_start_cg_count` 和 `presentation_end_cg_count` 分别等于 `chunkgroup_time_base_offset` 和 `chunk_group_time_base_offset + A`。另一方面，`CHUNK_0012` 的 `presentation_start_cg_count` 和 `presentation_end_cg_count` 分别等于 `chunkgroup_time_base_offset + A` 和 `chunk_group_time_base_offset + A + B`。通过利用这种方式设置字段使 `CHUNKGROUP_002` 被限定，以至于
10 `CHUNK_0011` 和 `CHUNK_0012` 连续地被播放。

应该指出的是：如果 `CHUNK_0011` 的播放时间与 `CHUNK_0012` 播放时间重叠，那么在两个播放时间中的一个播放时间能够被移动以便消除重叠。此外在图 23 中所示的 `chunk_arrangement_info()` 中的 `transition_info()` 被用作作为一个描述性字段，该描述性字段用于规定一种特殊的作用，例如在从一个数据流向另一个数据流过渡中的一个信号渐强、信号渐弱或擦除。
15

在图 26 中所示的例子中，`TITLE_001.VDR` 和 `TITLE_002.VDR` 文件信息文件规定了 `CHUNKGROUP_001.CGIT` 的 `chunk-group` 信息文件，而 `TITLE_003.VDR` 规定了 `CHUNKGROUP_002.CGIT` 的 `chunk-group` 信息文件。具体地说，在图 16 中所示的 `title_info()` 中的 `cgit_file_id` 规定了 `chunk-group` 的文件 ID。此外，称为 `title_start_chunk_group_time_stamp`（标题_开始_块组_时间_标记）和 `title_end_chunk_group_time_stamp`（标题_结束_块组_时间_标记）被用于规定一个时间范围，在该时间范围中该 `title` 被限定在 `chunkgroup` 中。
20

在图 27 所示的例子中，例如，`TITLE_001` 和 `TITLE_002` 规定了 `CHUNKGROUP_001` 第一半部分和第二半部分。应该指出的是：分割与由用户的一个请求相一致并且由用户任意地来确定它的位置，而不是预先地被确定。在这个例子中，利用一个距离 A 使分割成 `TITLE_001` 和 `TITLE_002` 的位置被设置在与 `CHUNKGROUP_001` 的头部分离的位置上。
25

`TITLE_001` 规定了 `CHUNKGROUP_001` 作为一个 `chunk group` 和 `CHUNKGROUP_001` 的开始时间作为一个 `title` 的一个开始时间。由用户确定
30



的一个点的时间被规定作为 title 的结束时间。

具体地说，CHUNKGROUP_001 的 chunkgroup_time_base_offset(头部的
位置)被设置为 TITLE_001 的 title_start_chunk_group_time_stamp，而
CHUNKGROUP_001 的 chunkgroup_time_base_offset 和距离 A 的总和被设置
5 为 TITLE_001 的 title_end_chunk_group_time_stamp。

TITLE_002 规定了 CHUNKGROUP_001 作为一个 chunk group 和用户确
定的一个点的时间被规定作为 title 的开始时间。CHUNKGROUP_001 的结束
时间作为一个 title 的一个结束时间。

具体地说，CHUNKGROUP_001 的 chunkgroup_time_base_offset(头部的
10 位置)和距离 A 的总和被设置为 TITLE_002 的 start_chunk_group_time_stamp，
而 CHUNKGROUP_001 的 chunkgroup_time_base_offset 和
CHUNKGROUP_001 长度的总和被设置为 TITLE_002 的
title_end_chunk_group_time_stamp。

TITLE_003 规定了 CHUNKGROUP_002 作为一个 chunk group 和
15 CHUNKGROUP_002 的开始时间作为一个 title 的一个开始时间。
CHUNKGROUP_002 一个结束时间被规定作为 title 的结束时间。

具体地说，CHUNKGROUP_002 的 chunkgroup_time_base_offset(头部的
位置)被设置为 TITLE_003 的 title_start_chunk_group_time_stamp，而
CHUNKGROUP_002 的 chunkgroup_time_base_offset 和 CHUNKGROUP_002
20 的长度的总和被设置为 TITLE_003 的 title_end_chunk_group_time_stamp。

此外，在这个例子中，称为 PROGRAM_001.PG1 的 program 信息文件规
定列举以一个顺序待播放的部分 TITLE_001 和部分 TITLE_003。具体地说，
在图 20 中所示的 play_item() 中的 title_number 被用于规定一个 title。由一个
title 限定的时间被用于限定开始点和结束点以便抽出一个切割。多个这样的
25 切割集合在一起以便构成一个 program。

下面描述一个附加的记录操作以便附加地把新的信息记录到光盘 1 中。
具体地说，这个附加记录操作典型地作为一个视频记录操作被进行，或由用
户操作输入单元以便把一个进行实时记录的指令输入给光盘装置来进行该附
加记录操作。在后者的情况下，如果不知道视频记录结束时间，那么按下一
30 个记录按钮。然而，对于能够断定记录结束时间的情况，按下用于单触摸记
录功能的按钮。单触摸记录功能是一种用于把视频记录进行一个固定时间周

期的功能。

下面通过以定时器记录作为一个例子来解释附加记录操作。在这种情况下，光盘装置的用户规定了在其它的操作之中：一个记录开始时间、一个记录结束时间、一个位流的位速率和一个频道预先被记录。此外，在视频记录被预约的一个时间点上，光盘 1 预先被检查以便发现是否留有一个适合于位速率和记录时间长度的自由空间。

如果在预约的一个时间与执行预约的视频记录的一个时间之间在光盘 1 上进行另一种记录操作，那么确实在可能的范围之内不能再分配以规定的位速率执行预约的视频记录的确定自由空间。在这种情况下，CPU 21 既可以把位速率减小到比预先规定的值更小的一个值以便在预约的时间周期记录信息，或者通过尽可能长时间的保持位速率不变以便在一个时间周期记录信息。更不用说，接着 CPU 21 记录操作，并且当检测预约的视频记录不方便时，通知用户这个不方便的信息被发出。

当预约的视频记录的开始时间被接近时，CPU 21 利用一个嵌入定时器和一个时钟信号来自动地使光盘装置从休眠状态恢复到操作状态。然后，CPU 21 从开始发出一个包括在处理程序中的文件系统操作指令，以便在光盘 1 上分配一个用于记录一个预约节目的区域。也就是说，首先，CPU 21 从预约记录的结束时间减去开始时间以便找到记录时间的长度，然后计算记录时间的长度与位速率的积以便获得待分配用于记录预约节目的所需区域的容量。除了在预约记录中需要的数据流文件之外，需要在一个信息文件中存储数据。更具体地说，当在一个 title 信息文件中需要存储一个新的 title 时，在光盘 1 上必须分配一个用于记录 title 信息文件的区域。如果不能分配一个具有足够容量的区域，那么需要采取上述的防范措施技术，也就是，减少位速率或仅在与分配的区域对应的一个时间周期进行记录操作。

应该指出的是：由于在这种情况下存储一个新的 title，所以用户给一个新的数据流文件一个名称，严格的说，给在一个新数据流目录中新数据流文件一个名称。使该名称为 ¥ MPEGAV ¥ STREAM_003 ¥ CHUNK_0031。也就是说，在根目录中的 MPEGAV 目录下的 STREAM_003 中数据流文件的名称是 CHUNK_0031.MPEG2，如在图 29 中所示的。

CPU 21 把以记录方式执行的指令发给光盘装置的其它部件。例如，通过输入端 P3 接收的视频信号和通过输入端 P4 从一个在图中没有示出的调谐



器接收的音频信号利用编码器 10 来编码,然后把它们存储在写入通道缓冲器 11 中。接着, CPU 21 把光头 2 移动到一个由关于早先分配区域的一个地址的信息确定的写入位置上。然后, CPU 21 把光头 2、FR & 解调/调制电路 3 和 ECC 电路 4 设置到写入方式,并且根据写入通道缓冲器 11 来定位开关 5。

5 此外,在 CPU 21 精确地调整光头 2 的位置之后,利用光头 2 开始一个写入操作。在此时,借助于开关 5、ECC 电路 4、FR & 解调/调制电路 3 和光头 2,从写入通道缓冲器 11 读出将被记录在一个新设置的名称为 CHUNK_0031.MPEG2 的文件中的数据,以便把它们记录到光盘 1 上。

当在上述的写入操作期间发生下列一种情况时, CPU 21 停止操作。

- 10
1. 达到预约的视频记录的结束时间。
 2. 由于像一个不充足存储容量这样的原因使信息不能再被记录到光盘 1 上。
 3. 接收的一个停止记录操作的指令。

接着,通过利用包括在预先的处理程序中的文件系统操作指令, CPU 21

15 利用指向一个已经新记录信息的地址的值来校正指向在文件系统 CHUNK_0031.MPEG2 的指示符。此外, CPU 21 分别为 chunk 信息、chunk-group 信息和 title 信息准备文件、并且给每个文件一个名称以及把这些信息存储到文件中。应该指出的是:在记录操作期间或在预约时间上需要在光盘 1 上预先分配用于记录这些文件的自由空间。

20 结果,新的信息文件典型地如在图 30 中所示地被产生。在该图中,这些文件通过在其右侧上设置的星号‘*’来命名,它们在上述的操作中新产生文件的名称。

图 31 是一个表示在新产生的信息文件之间的关系图。如在图中所示的, TITLE_004 规定了 CHUNKGROUP_003, CHUNKGROUP_003 规定了

25 CHUNK_0031, 而 CHUNK_0031 规定了 STREAM_0031。

也就是说,在一个信息文件中一个新的数据流被记录为 TITLE_004。通过利用该光盘装置的一个功能来检验一个 title,用户能够知道像 TITLE_004 的属性这样的信息。此外,能够播放 TITLE_004。

下面描述与在图 26(或图 27)所示操作类似的在一个光盘 1 上重写/记录

30 信息的操作。与在一个视频带上记录一个信号的操作非常类似,一个重写-记录操作是在已经记录在光盘 1 上的整个现有节目上擦除该现有节目并且记



录一个新节目的操作。

在重写 - 记录操作中，使操作开始的一个位置是很重要的。假设用户规定 TITLE_001 的头部作为开始一个重写 - 记录操作的位置。在这种情况下，通过重写记录在 TITLE_001、TITLE_002 和 TITLE_003 中现有的信息，以它们列举的顺序来进行重写 - 记录操作。如果即使达到 TITLE_003 结束时间也没有完成重写 - 记录操作，那么通过在光盘 1 分配一个新的自由空间来继续记录操作。如果 TITLE_002 被规定作为开始一个重写 - 记录操作的位置，那么由于 TITLE_001 在操作的开始位置之前，所以利用记录操作将不能重写 TITLE_001 中的信息。

10 假设利用重写现有信息来进行定时视频记录，该现有信息是在 TITLE_003 的头部位置开始的。在这种情况下，该光盘装置的用户规定了在其它的操作之中：一个记录开始时间、一个记录结束时间、一个位流的位速率和一个频道预先被记录。此外，TITLE_003 的头部被规定作为一个记录开始的位置，该开始位置对于重写记录操作来说是重要的。此外，同样在这种情况下，在视频记录被预约的一个时间点上，预先检查在光盘 1 上是否留有一个适合于位速率和记录时间长度的自由空间。在重写记录操作的情况下，在光盘 1 上的由一个规定位置开始的多个可重写 title 的总容量与自由区域之和是一个可记录的空间。具体地说，在这种情况下，由 TITLE_003 控制的 STREAM_0011 和 STREAM_0012 的数据流与光盘 1 上的一个自由空间的总和是一个可记录的空间。

25 在一个重写记录操作中，对于上述可记录空间来说具有一些适合于选择的项目，这些选择的项目是关于视频记录实际被进行的顺序。作为第一个可以想的到的选择项目是能够选择一种操作方法以在 title 中规定了数据流的顺序来记录信息。具体地说，在这种情况下，能够选择一种从 STREAM_0011 的头部开始记录的操作方法，并且当 STREAM_0011 的结束被达到时，该记录持续到 STREAM_0012 的头部。然后，当 STREAM_0012 的结束被达到时，视频记录被持续到光盘 1 上的自由空间。作为另一种方法，首先，在光盘 1 上的自由空间上进行视频记录，并且在自由空间被完全地用完的一个时间点上该记录持续到一个现有的数据流。

30 在模仿一个视频带的意义上前一种操作方法是极好的。也就是说，由于该记录操作与把信息记录到一个视频带上的操作类似，所以该操作的特征在



于用户能够容易地理解该操作。另一方面，特征在于一个已经被记录的数据流被以后删除，在保护记录的信息的意义上后一种操作方法是极好的。

应该指出的是：如果在预约的一个时间与执行预约的视频记录的一个时间之间在光盘 1 上进行另一种记录操作，那么确实在可能的范围之内不能再分配以规定的位速率执行预约的视频记录的确定自由空间。在这种情况下，与前面描述的附加记录操作非常类似，CPU 21 既可以自动地把位速率减小到比预先规定的值更小的一个值以便在预约的时间周期记录信息，或者通过尽可能长时间的保持位速率不变以便在一个时间周期记录信息。

当预约的视频记录的开始时间被接近时，光盘装置从休眠状态恢复到操作状态。然后，CPU 21 在光盘 1 上分配所有的自由空间。不用说，也存在这样一种方法，其中在该时间点上不分配一个自由区域，而在所需区域的一个时间点上分配该自由区域一个用于记录一个预约节目的区域。为了解释简单的目的，在记录开始之前分配一个所需的区域。

应该指出的是：由于规定了一个开始时间、一个结束时间和一个位速率的原因使在定时记录中预先已知一个所需区域的长度，所以只能分配一个具有一个所需长度或具有所需长度加上一定的附加余量的区域。在需要记录信息文件的情况下，例如在记录期间需要把一个 title 信息文件记录为一个新 title 的情况下，需要分配一个具有足够长度来记录该信息文件的区域。

给一个新的数据流文件一个名称，严格的说，给在一个新数据流目录中新数据流文件一个名称。使该名称为 ¥ MPEGAV ¥ STREAM_002 ¥ CHUNK_0031。也就是说，在根目录中的 MPEGAV 目录下的 STREAM_002 中数据流文件的名称是 CHUNK_0031.MPEG2，如在图 32 中所示的。

通过输入端 P3 接收的视频信号和通过输入端 P4 从一个在图中没有示出的调谐器接收的音频信号利用编码器 10 来编码，然后把它们存储在写入通道缓冲器 11 中。接着，CPU 21 把光头 2 移动到一个由关于早先分配区域的一个地址的信息确定的写入位置上。然后，CPU 21 把光头 2、FR & 解调/调制电路 3 和 ECC 电路 4 设置到写入方式，并且把开关 5 定位在写入通道缓冲器 11 侧上。此外，在 CPU 21 精确地调整光头 2 的位置之后，利用光头 2 开始一个写入操作。在此时，借助于开关 5、ECC 电路 4、FR & 解调/调制电路 3 和光头 2，从写入通道缓冲器 11 读出将被记录在一个新设置的名称为 CHUNK_0031.MPEG2 文件中的数据，以便把它们记录到光盘 1 上。



在此时，首先重写称为 CHUNK_0011.MPEG2 的数据流。在记录已经达到称为 CHUNK_0011.MPEG2 数据流的结束之后，在持续到称为 CHUNK_0031.MPEG2 的数据流之前该操作持续到称为 CHUNK_0012.MPEG2 的一个数据流。

5 当上述的处理正在被进行时，在上述 3 个条件中的任一个条件被满足时 CPU 21 停止写入操作。

接着，CPU 21 执行包括在预先的处理程序中的文件系统操作指令，以便校正数据流文件、chunk 信息、chunk-group 信息和 title 信息准备文件。

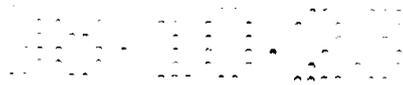
顺便说一下，利用与写入操作结束同步的定时来改变这些文件的结构。

10 例如，当在称为 CHUNK_0011.MPEG2 和称为 CHUNK_0012.MPEG2 的两个数据流文件写入操作已经被完成之后在称为 CHUNK_0013.MPEG2 的数据流文件上进行记录时，在光盘 1 上的文件结构被改变在图 33 中所示的一种结构。一个在其右侧上设置有星号 ‘*’ 的文件名称是在这时新产生的一个文件的名称。

15 图 34 是一个表示利用这种方法新产生的文件之间的关系图，也就是，如在图 33 中所示的文件。当与在图 31 中所示的文件相比时，显然加入了 CHUNK_0031 作为包括在 CHUNKGROUP_002 中的一个 chunk，由 TITLE_003 和 CHUNK_0031 规定的 CHUNKGROUP_002 规定了 STREAM_0031。

20 另一方面，如果当数据正在被写入到一个现有数据流文件中时重写记录操作被完成，也就是，如果当数据正在被写入到例如称为 CHUNK_0011.MPEG2 的数据流文件中时重写记录操作被完成，那么由于没有数据被写入到该文件中，所以在称为 CHUNK_0031.MPEG2 的数据流文件上分配用于重写操作的自由区域被释放。在这种情况下，进行特殊的 title 处理。具体地说，当在 TITLE_003 的头部开始一个重写记录操作和在 TITLE_003 的中部完成该操作时，该 title 被分离。更具体地说，如在图 35 中所示的，新的 TITLE_003 被指定到在重写记录操作的开始位置与结束位置之间的一个区域，而 TITLE_004 被指定到在该区域之后的一个区域，也就是，该区域的其余部分，假设 TITLE_003 原始被指定的该区域。

30 下面解释播放一个 title 的操作。假设一个具有在图 26 中所示文件的光盘 1 被插入到光盘装置中，然后从光盘 1 来播放一个 title。当光盘 1 被插入



到该装置中时，首先，CPU 21 从光盘 1 上的信息文件中读出数据，并且把该数据存储在 RAM 单元 24 中。通过重复该基本操作来进行数据处理以便从上述的一个信息文件中读入数据。

具体地说，首先，CPU 21 从 VOLUME.TOC 和 ALBUM.STR 中读出数据。然后，CPU 21 检查称为 TITLE 的目录以便找出在该目录中存在多少具有扩展名为 “.VDR ” 的文件。具有这样一个扩展名的文件是一个具有 title 信息的文件。该文件数等于 title 数。在图 26 中所示的例子中，title 数是三。接着，CPU 21 从三个文件中读出 title 信息并且把该信息存储在 RAM 单元 24 中。

10 CPU 21 控制 OSD 控制电路 9 以便使 OSD 控制电路 9 产生字符信息，也就是，关于在光盘 1 上记录的 title 的信息。利用合成电路 8 把该字符信息与一个视频信号合成。然后通过输入端 P1 输出合成的结果以便在显示单元上显示该结果。具体地说，显示在该例子中存在的三个现有 title 的每个 title 的长度和属性。该属性包括 title 的名称和 title 被记录的日期。

15 假设用户把 TITLE_002 作为例子规定为待播放的一个 title 。在 TITLE_002 的信息文件中，具体地说，在图 16 中所示的 title_info() 的 cgit_file_id 中，规定了 CHUNKGROUP_001 的一个文件 ID 被记录。CPU 21 记录该文件 ID 并且把该文件 ID 存储在 RAM 单元 24 中的 CHUNKGROUP_001 中。

20 然后，CPU 21 检查与 CHUNK 对应的 TITLE_002 的开始和结束时间。该开始和结束时间分别地被记录在如图 16 中所示的 title_info() 的 title_start_chunk_group_time_stamp 和 title_end_chunk_group_time_stamp 字段中。通过与包括在关于一个 CHUNKGROUP 的信息中的信息比较来进行该检查，其中对应的 chunks 已经被记录在 CHUNKGROUP 中。更具体地说，通过
25 与在图 23 中所示的 chunk_arrangement_info() 的 presentation_start_cg_time_count 和 presentation_end_cg_time_count 字段中记录的信息比较来进行该检测。在这个例子中，已知 TITLE_02 的开始时间是 CHUNK_0001 的中间，如在图 27 中所示的。也就是说，显然为了从头部来播放 TITLE_002，播放操作需要从 CHUNK_0001.MPEG2 数据流文件的中间
30 开始。

接着，为了确定数据流的哪部分与 TITLE_002 的头部相对应，CPU 21



检查该数据流。也就是说，CPU 21 计算在与 TITLE_002 的头部相对应的数据流中一个偏移时间(一个时间特征)的大小。然后，通过利用在 CHUNK 文件中的特征点信息，一个播放开始点与开始时间被确认之前的一个点相对应。利用该方法能够确定播放开始点与该文件的头部的偏移距离。

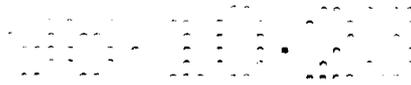
5 接着，通过利用一个包括预先的处理程序中的文件系统操作指令，CPU 21 在已经记录了 CHUNK_0001.MPEG2 的光盘 1 确定一个实际地址和该地址的长度。此外把早先发现的播放开始点的偏移地址加的这个实际地址上，以便精确地确定 TITLE_002 的播放开始点的地址。

10 接着，CPU 21 把光头 2 移动到一个由关于 CHUNK_0001.MPEG2 文件的地址的信息确定的读出位置上。然后，CPU 21 把光头 2、FR & 解调/调制电路 3 和 ECC 电路 4 设置到读出方式，并且把开关 5 定位在读出通道缓冲器 6 侧上。此外，在 CPU 21 精确地调整光头 2 的位置之后，利用光头 2 开始一个读出操作。在此时，从名称为 CHUNK_0001.MPEG2 文件中读出的数据被存储到读出通道缓冲器 6 中。

15 在读出通道缓冲器 6 中存储的数据被输出给解码器 7 以便由解码器 7 解码。作为解码的结果，解码器 7 输出视频信号和音频信号。在由光盘 1 读出的、由解码器 7 解码的和在显示单元上显示的数据量等于 CHUNK_0001.MPEG2 文件大小的时间点上，CPU 21 使播放操作过渡到 TITLE_003。利用与 TITLE_002 相同的方式来进行播放 TITLE_003 的信息的操作。

20 当播放由记录的 title 来数据的操作被完成时或当接收到一个停止读出操作的指令时，读出处理和解码处理被结束。

应该指出的是：当一个新盘或一个具有不同格式的盘被插入到光盘装置中作为光盘 1 时，CPU 21 试图从插入的盘中读出 VOLUME.TOC 和
25 ALBUM.STR。然而在该新插入的盘中通常不存在这些文件。在这种情况下，也就是，在 VOLUME.TOC 和 ALBUM.STR 不能被读出的情况下，CPU 21 发出一个信息以便向用户请求一个指令。响应该信息，用户给 CPU 21 一个指令，以便在新插入的盘具有不同的格式的情况下弹出或在新插入的盘即使具有相同格式但是一个新盘的情况下使光盘 1 初始化。作为另一种方法，在
30 数据已经从具有相同格式的盘中被破坏的情况下，利用一些方法该指令可以使新插入盘上的数据被恢复。



下面，进一步解释 title(标题)。图 15 所示的 TITLE_###.VDR 是存储 title 的信息的一个文件。在标题上的信息是记录在一个 title __ information() 字段中。存在于 TITLE_###.VDR 中的 title __ information()字段的数目是 1。因此，许多 TITLE_###.VDR 文件作为标题存在于一 volume(卷)中。

5 title 数不限定于图 16 所示的 title_info() 中。而是该 title 数由文件的名字或文件的 id 确定。也就是说，在文件名 TITLE_###.VDR 中的 ### 符号是用于标题数码的正整数。标题不是信息结构。更确切地说，标题是与块组(chunk group)中的范围有关。该范围从表示开始点的标题索引开始，结束于表示下一个标题的开头的另一个标题索引，或者该范围可以是在该块组的
10 结束点结束的范围的一部分。

如图 36 中所示，在图 15 所示的 TITLE_###.VDR 的 file_type_id 是用于记录用于表示该文件是包括在 title __ informatio()中的文件的识别的 16 字符串的区。text_block() 是用于存储各种文本的区域。在其上只记录允许使用 text_block 的文本项。

15 如图 16 所示，title __ information()是记录在块组上的开始点和结束点和标题的其它属性的区域。另外，title __ information()可以包括指示是否在以标题数码为序的重放标题的操作中能够保证标题间的无缝重放操作的标记。这个标记允许光盘装置预先得到是否标题间的无缝重放操作可以完成的信息，和当合并标题时是否必须改变设计。

20 标题或块组(chunk group)期间的无缝重放操作被保证。由于标题间的边界也是文件间的边界，然而，在一些情况下，标题间的无缝重放操作可以保证。应当注意到，光盘装置一般具有改变该设计成为允许完成无缝重放操作状态的功能。

图 16 表示的 title __ information()的 title __ information __ length 是用以记录由字节表示的 title_information 的长度的区域。flags_for_title(标题标志)是记录对应如写属性(也就是，改变标题的操作是否允许)、标题可以重放次数的限制和等级水平的信息的字段。cigt_file_id (cgit_文件__识别符)是用于记录作为标题基础的 CHUNKGROUP_###.CGIT 的识别信息文件。

30 title_start_chunk_group_time_stamp(标题__开始__块__组__时间__标记)是用于记录在块组中定义的本地时间轴上的标题的重放的开始时间。title_start_chunk_group_time_stamp 显示是由标题的标题索引指示的图像的时



间。另一方面， `title_end_chunk_group_time_stamp`(标题__结束__块__组__时间__标记)是用于记录在块组中定义的本地时间轴上的标题的重放结束时间的
的时间。 `title_start_chunk_group_time_stamp` 是 `chunk group`(块组)的重放结束
时间或者是表示紧接在时间轴上的该标题后面的另一个标题开始点的标题索引
5 索引的值。

`title_playback_time`(标题__重放__时间)是用于记录标题的重放时间的区域，也就是时间码值，或帧或场的数码。标志__数(`number_of_marks`)是用于
记录在除去了标题索引的标题中的所有标志组的总数的区域。如图 37 所示，标志__类型(`mark_type`)是用于记录在标题中的任何位置的标志的类型的
10 区域。标志是用作在标题中的随机存取点。 `mark_chunk_group_time_stamp`(标志__块__组__时间__标记)是用作记录在块组的时间轴上的时间标记(`time stamp`)的区域。时间标记对应设置在标题中的标志的位置。时间标记被安排成用 1 的一个最小值顺序开始的。表示标题的相同的开始和结束点的时间标记可能存在。填充__字节(`stuffing_bytes`)是用于记录填充字节的区域。它的长度是 $8 \times n$ 比特，这里的 n 大于等于 0。
15

然后，参照图 21 至 24 解释块组和块(`chunk group` 和 `chunk`)。
`CHUNKGROUP_###.CGIT` 是存储标题的时间轴的定义，块(`chunks`)的结构和包括在标题中的不连续点的处理的文件的名字。

标题是由诸如包括非视频数据和 DV(数字视频)比特流的数据流的各种
20 比特流构成的。在 DV 格式中时间轴是以帧单位表示的。如果 MPEG 2 视频的 STC(系统时间时钟)用来作为参考，格式变化使得控制 DV 比特流不可能。

由于这个原因，在标题中的局部时间轴被设定。该时间轴不取决于由标题构成的数据流。标题间的边界设定独立于块(`chunk`)间的边界的设定。因此，不是为每个块(`chunk`)设定局部时间轴(也就是通过逐个地(1-on-1)与比特
25 流关联)，或者为每个标题设定局部时间轴，为包括多个标题的块组(`set of chunks`)设定局部时间轴是适合的，也就是，任意的标题数目。一组块(`set of chunks`)称为块组(`chunk group`)。

在块组中，定义单个时间轴，在该时间轴上，块(`chunks`)被放置以确定
显示该块的时间。也就是说，块组是以一种状态对块的安排，在该状态中比
30 特__数据流文件(一串字节)沿着时间轴延伸。沿时间轴对比特数据流中包括的所有的块(`chunks`)的安排称之为路径。在一个块组中，可以安排多个路径。



描述块组的重放开始时间和重放结束时间的路径称之为主路径。其它的路径是作为子路径知道的。子路径主要是表示作为附加信息记录在后面的音频块的信息。

5 块间的联接点不必与标题间的边界相符合。因此，块间的联接的点不是标题的属性。然而，如果包括作为块的属性的块间的关系，将出现在分层处的矛盾。在不连续点处的信息位于块与标题之间，因此考虑适当地置于块组分级。

10 至此如上所述是综合地描述，块组的信息包括如如何沿时间轴布局块，重放块的顺序和在第一个块的结束与第一个块以后下一个重放块的开始间的联接点处的不连续点。

应当注意到，块组还用于规定在相同的时间重放的数据流。例如，在图 38 中所示的例子中，TITLE_001 规定了 CHUNKGROUP_001，CHUNKGROUP_001 规定了 CHUNK_0001 和 CHUNK_0002。至少由 CHUNK_0001 和 CHUNK_0002 分别规定的 STREAM_0001 和 STREAM_0002 15 的部分在时间轴上彼此重叠，和因此在相同的时间重放。

图 39 是表示块组被产生的情形的示图。在这种情况下，控制比特流 A 的块 A 和控制比特流 B 的块结合在一起。在产生的块组中，块 A 和 B 被分别控制作为主路径和子路径。至少比特流 A 和 B 的部分彼此重叠，和因此在相同的时间重放。

20 如图 40 所示，在图 21 中所示的 CHUNKGROUP_###.CGIT 的 file_type_id 是由遵守 ISO 646 的 16 个字符串表示的区域和用于指示该文件是 CHUNKGROUP_CGIT 文件的标识符。chunkgroup_time_base_flag(块组__时__基__标志)是记录有关块组(chunk group)的参考计数器的区域。chunkgroup_time_base_offset(块组__时__基__偏移)是记录在块组中的参考时 25 间轴的开始时间的 64 比特区域。该开始时间设定在计算 90KHz 时钟脉冲的数的计数器中。text_block() (文本__数据块()) 是用于存储各种文本的区域。只有允许使用 text__block() 的文本项目记录在这里。

30 如图 22 中所示，chunk_connection_info() [块__联接__信息()] 是用于记录在诸如视频变化点和音频 - 视频同步点的单独点上的信息。chunk_connection_info() 是描述块间的联接的状态。在作为编辑结果得到的 2 个块间联接处的单独的点上，必须从一个块转变到在 GOP 中间的另一个块。



在那个编辑点的相邻处的信息描述在 `chunk_connection_info()` 中。一个块决不适用于 2 个或更多的块组(chunk group)。

`chunk_connection_info_length`(块__联接__信息__长度)是用于记录用字节表示的 `chunk_connection_info()` 的长度的区域。`number_of_chunks`(块的数目)是用于记录用于块组中的块的总数。如图 41 所示, `chunk_sync_play_flag`(块__同步__播放__标记)是表示在相同的时间是否必须重放 2 个或更多的块的标记。在这个标记中的设定的 0 值表示只有一个块重放。另一方面, 1 值表示在相同的时间有多个块被重放。

图 23 中所示的 `chunk_arrangement_info()`(块__安排__信息)的 `chunk_arrangement_info_length` (块__安排__信息__长度)是用于记录用字节表示的长度的信息的区域。更详细地说, 该长度是开始于 `chunk_arrangement_info_length` 的第一个字节, 结束于 `transition_info()`(过渡__信息())的最后一个字节的字节计数值。`chunk_info_file_id`(块__信息__文件__识别符)是用于记录表示文件是块信息文件的文件识别的区域。

`chunk_switch_stream_id`(块__开关__数据流__识别符)是用于表示记录在两个块互相联接在一起的情况下连续重放数据流的数据流识别区域。具体地说, 在识别视频或音频数据的 MPEG 2 数据包开头中记录的识别用于这个数据流的标识。`presentation_start_cg_time_count`(图像显示__开始__时间__计算)是用于记录表示作为块组期间的一个时间的块显示开始时间。该块的显示开始时间是由在块组中定义的总时间标记(global time stamp)表示的。在块组期间显示块的操作开始于显示开始时间。另一方面, `presentation_end_time_count`(图像显示__结束__时间__计算)是用于记录表示作为块组期间的一个时间的块显示结束时间。该块的显示结束时间是由在块组中定义的 global time stamp 表示的。

如图 42 中所示, `original_time_count_type`(初始__时间__计算__类型)是用于记录在数据流中计算时间的类型。例如, 在 MPEG 2 视频数据流的情况下, `original_time_count_type` 具有 “ 0000 ” 的值。`number_of_start_original_time_count_extension`(初始__时间__计算__延伸__开始__数)是用于记录每一个表示在需要多个时间计算的情况下新需要的开始时间的区域。另一方面, `number_of_end_original_time_count_extension`(初始__时间__计算__延伸__结



束__数)是用于记录每一个表示在需要多个时间计算的情况下新需要的结束时间的数字区域。presentation_start_original_time_count(显示__开始__初始__时间__计算)是用于记录数据流期间的时间或对应 presentation_start_cg_time_count(显示__开始__cg_时间__计算)的计算值。另一方面, presentation_end_original_time_count(显示__结束__初始__时间__计算)是用于记录数据流期间的时间或对应 presentation_end_cg_time_count(显示__结束__cg_时间__计算)的计算值。

tc_ext_attributes(tc_ext_属性)是用于记录 time_count_extension (时间__计算__延伸)属性的区域。time_count_extension 可以包括在其它数据中表示应用哪个数据流 time_count_extension 的信息。start_original_time_count_extension(开始__初始__时间__计算__延伸)是用于记录开始计算值或从一个块转换到另一个块的必要的开始时间。这个信息是选择项,和当必须记录多个时间或计算值时使用。另一方面, end_original_time_count_extension(结束__初始__时间__计算__延伸)是用于记录结束计算值或从一个块转换到另一个块所必要的结束时间。这个信息是选择项,和当必须记录多个时间或计算值时使用。transition_info() 是用于记录当开关从一个块转换到另一个块时应用特别效果所需要的信息的区域。该信息可以规定一个块,转换时间和所述的几个特技的类型。

在图 24 中所示的 CHUNK_%%%.ABST 是用于记录由用%%%.ABST 标记表示的 sub_file (子__文件)数识别的块构成的数据流中抽取的特征点的文件的名字。该文件包括诸如开始字节位置,构成如 GOP 和音频帧的比特流的每个单位的长度和属性的信息。GOP 信息和音频帧信息作为每个块(sub_file)的 CHUNK_%%%.ABST 集中在一起的。

如图 43 所示, CHUNK_%%%.ABST 的 file_type_id(文件__类型__识别符)是用于记录表示该文件包括数据 stream_info() [数据流__信息()] 的识别符的区域。file_type_id 是适合 ISO 646 的 16 字符串。

如图 44 所示,图 24 所示的 info_type(信息__类型)是用于记录随后的数据 stream_info() 的类型的区域。info_type 识别数据流的类型。number_of_programs(程序数)是用于记录包括在 MPEG 2 TS(传输数据流)中包括的程序数码。为了得到这个程序的数码,必须去取 PSI(程序规定信息)。在不是 TS 的 MPEG 数据流的情况下,程序的数码是 1。number_of_streams(数

据流数)是用于记录在这个程序中所用的数据流的数码区域。该数据流的数码等于在 TS 情况下的不同的 PID 的数码(数据包识别符)。在不是 TS 的 MPEG 数据流的情况下, 数据流的数码等于具有不同于其它的数据流识别符的数据流的数码。

5 `stream_identifier`(数据流__识别符)是用于记录数据流的识别符或数据流的扩展识别符的区域。在 TS 的情况下, 利用了 PID 。

`slot_unit_type`(时隙__单位__类型)是用于记录在图 45 中所示的定界固定的间隙的数据流的情况下如何界定一个数据流的区域。在如帧和场的定界符的时间定界符索引的情况下, 使用时间标记值。 `slot_time_length`(时隙__长度)是用于记录对应一个时隙的时间的区域。 `slot_time_length`(时隙__长度)是使用 90KHz 时钟脉冲计数的计数器的时间标记的值。 `number_of_slots`(时隙__数)是用于记录包括在 `CHUNK_%%.ABST` 中的 `slot__info()` 字段数目的区域。 `number_of_I_picture_in_a_slot`(在一个时隙中 I 图像的数目)是用于记录包括在一个时隙中的 I 图像的数目。 `number_of_I_picture_in_a_slot`(在一个时隙中 I 图像的数目)是在 1-15 范围内的一个整数。然而, 应当注意到, 包括在作为它的开头的在具有 GOP 开头(header)的时隙紧接着前面的时隙中的 I-图像的数目可以小于在一个时隙中的 I - 图像的数目。当设置作为不紧随 GOP 开头之后的开头的具有 I - 图像的图像标题的时隙时, 使用了 `number_of_I_picture_in_a_slot`(在一个时隙中 I 图像的数目)。

20 接下来, 在图 17 和 18 中所示的程序中的信息将进一步解释。在 `PROGRAM_$$$PGI` 中存在仅一个 `program()` 字段。如在一卷中存在许多作为程序的 `PROGRAM_$$$PGI` 文件。程序数目不限定于 `program()` (程序()) 中。而是由文件的名字或文件的 id (标识) 确定程序的数目。

 如图 46 所示, 图 17 中所示的 `PROGRAM_$$$PGI` 的 `file_type_id` (文件__类型__id) 是用于记录用于指示该文件是包括在 `program()` 中的文件的识别符的 16 个字符串的区域。 `text_block()` 是为存储各种文本而形成的。只有允许使用 `text_block()` 的文本项目记录在这里。

 图 18 中所示的 `program()` 的程序的标记[`flags_for_program()`]是用于记录如写属性(是否改变程序的操作被允许)的各种标记, 程序重放次数的限定和等级水平的区域。

30 如图 47 所示, `program_status`(程序__状态)是用于记录程序属性的区

域。这个字段是选择地设定。然而，如果不希望在该字段中设定什么，在这里必须描述“none”(什么也没有)。

program_playback_time() [程序__重放__时间()] 是用于记录程序的重放时间的区域。 number_of_play_sequences [播放__次序的数码] 是用于记录用于程序中 play __ sequence 的数码。在这个格式的例子中，播放顺序的数码设定为固定的值 1。也就是说，由于在 1 程序 = 1 信道重放操作设定在这个格式的例子中，为了在相同时间实施重放 2 信道的操作，需要能够同时重放 2 程序的说明。如果没有限制 1 程序 = 1 信道重放操作，用 1 程序， 2-信道同时重放操作是可能的。在使用多信道 I/O 以相同的时间重放 2 播放顺序的操作中，光盘装置确定每个播放顺序被指定的那个输出信道。

number_of_play_lists (播放__目录__数码) 是用于记录在这个播放顺序中使用的播放目录的数码。在这个例子中播放目录的数码设定为 1。 play_list_start_time_stamp_offste (播放__目录__开始__时间__标记__偏移) 是用于记录在播放顺序中得到的由计时器从播放顺序开始时间开始计算的结果的区域。这个值是播放目录的开始时间。在程序中，只有 1 播放目录允许存在存在播放顺序中。时间单位系统是基于 90KHz 的频率。也就是说，最小的时间单位是 1/90,000 秒。 stuffing_byte (填充__字节) 是记录填充字节的区域。它的长度是 $8 \times n$ 比特，这里 n 大于等于 0。

其次，解释分开和移动标题的编辑处理。在标题分开操作中，由使用者规定的位置分开存在的标题，以产生新的标题。在标题移动操作中，标题的顺序被改变。上述的标题信息文件的格式结构是允许标题分开的操作和容易完成的标题移动操作的结构。也就是说，根据上述的格式，块组(chunk group)，也就是由比特-流文件集合成的结构体，和标题，也就是作为用分开或移动的操作的方法将标题分裂成信息片由使用者识别的结构体，不必在块组(chunk group)下改变信息。

下面，结合图 48 所示的流程解释分开标题的处理。如图所示，处理开始于步骤 S1，在该步骤，使用者规定分开点。例如，如图 27 中所示的 TITLE_002 中的预定的位置是作为分开点规定的。于是，处理的流程进行到步骤 S2，在该步骤，CPU 21 产生作为具有被分开的标题的开始点的第一标题的名为 TITLE_002. VDR 的信息文件，也就是，在分开之前，TITLE_002 作为它的开始点和该分开点作为它的结束点。



于是，处理的流程进行到步骤 S3，在该步骤，CPU 21 为具有分开点是它的开始点和分开的标题的结束点，也就是分开前的 TITLE_002 的结束点作为它的结束点的第二标题产生名为 TITLE_003.VDR 的信息文件。

因此，处理流程继续到步骤 S4，在该步骤 CPU 21 改变分开的标题随后
5 的标题的名字。更具体地说，图 27 中所示的 TITLE_003 变到 TITLE_004 以产生图 49 中所示的标题。

如上所述，在块组，块和数据流(chunk group, chunk 和 streams)上的信息即使标题被分开了也完全不变。

然后，通过参照图 50 的流程解释调换标题的处理。更具体地说，在图
10 49 中所示的 TITLE_002 如在相同的图中所示，为 TITLE_003 所替换。

如图 50 所示，该处理开始于步骤 S11，在该步骤，CPU 21 改变具有数码为 002 的标题的名字 TITLE_002 为用数码 X 表示的标题 TITLE_X。于是处理流程进行到步骤 S12，在该步骤，CPU 21 改变具有 003 数码的标题的名字 TITLE_003 为用数码 002 表示的标题 TITLE_002。于是处理流程进行到
15 步骤 S13，在该步骤，CPU 21 改变具有数码为 X 的标题的名字 TITLE_X 为用数码 003 表示的标题 TITLE_003。

如上所述，CPU 21 在改变具有数码 X 的标题名字 TITLE_X 为具有数码 003 的标题名字 TITLE_003 以前，首先改变具有数码 002 的标题的名字 TITLE_002 为具有数码 X 的标题 TITLE_X。这是为了避免如果 CPU 21 立即
20 改变具有数码 002 的标题的名字 TITLE_002 为具有数码 003 的标题 TITLE_003，这里将在 CPU 21 改变具有数码 003 的 2 文件中的一个名字 TITLE_003 为具有数码 002 的标题 TITLE_002 以前会有 2 个具有相同的数码 003 的标题。

在图 50 中所示的处理中，TITLE_002，该具有数码 002 的标题为具有
25 数码 003 的 TITLE_003 所调换，如图 51 所示。

图 52 表示了表示删除标题的处理的流程。如图中所示，该处理开始于步骤 S21，在该步骤，使用者规定要删除的标题。处理的流程于是进行到步骤 S22，在该步骤，CPU 21 作出是否与规定的标题有关的块组，块和数据流(chunk group, chunks 和 streams)与另外的标题有关的判断。如果它们不与另
30 外的标题有关，处理的流程进行到步骤 S23，在该步骤 CPU 21 删除与规定的标题有关的块组，块和数据流。

如果在步骤 S22 作出的判断结果指示与规定的标题有关的块组，块和数据流还与另外的标题有关，另一方面，块组，块和数据流的删除将使它们不再存在。因此，在这种情况下，在步骤 23 完成的操作将跳过。

5 在步骤 S23 的操作完成以后，或如果在步骤 S22 作出的判断结果指示与规定的标题有关的块组，块和数据流还与另外的标题有关，处理流程继续到步骤 S24，在该步骤，CPU 21 处理剩余的信息文件，也就是，未删除的信息文件。详细地说，由于标题被删除，所以在删除的一个标题后面的标题被赋予的数增加 1。然而，应当注意，如果在步骤 S22 作出的判断结果指示与规定的标题有关的块组，块和数据流还与另外的标题有关，该规定的标题本质上是不删除的。在这种情况下，OSD 控制电路 9 输出表示本质上不被删除的规定的标题将到显示单元的消息。

15 如上所述在 TITLE_002 已经删除以后，如图 53 所示，前面的 TITLE_003 变成 TITLE_002，和前面的 TKITLE_004 文件变成 TITLE_003。在这同时，前面的 CHUNKGROUP_001 块组被分成 CHUNKGROUP_001 和 CHUNKGROUP_002，和前面的 CHUNKGROUP_002 块组变成 CHUNKGROUP_003。另外，前面的 STREAMS_001 比特流分成 STREAM_001 和 STREAM_002。

20 上述表明，在删除标题的典型的操作中，一个数据流被删除以增加空闲区域的大小。另一方面，还有一种通过只擦除在用作控制信息的标题上的信息以删除标题的技术。在这种情况下，占有实际区域的数据流是不能实际删除的，以至与该标题有关的块组，块和数据流即使标题被删除也未完全改变。

25 其次，参照图 54 所示的流程解释标题合并的处理，如图所示，该处理开始于步骤 S31，在该步骤，使用者规定合并的标题。例如，被合并的标题是图 49 所示的 TITLE_002 和 TITLE_0033。处理的流程于是进行到步骤 S32，在该步骤，CPU 21 产生具有该标题开头的开始点，也就是 TITLE_002，作为它的开始点和尾标题的结束点，也就是 TITLE_003，作为它的结束点的标题。在这个例子中，产生的名字是 TITLE_002。

30 于是，处理流程进行到步骤 S33 以在合并以前完成标题的删除处理。在这种情况下，前面的 TITLE_002 和 TITLE_003 被删除。因此，处理流程进行到步骤 S34，在该步骤，CPU 21 处理随在合并的标题后的标题的信息文件。更具体地说，在这种情况下，图 49 中所示的 TITLE_004 变到 TITLE_003。

结果，得到图 27 中所示的标题。

应当注意到，在上面给出的例子中，2 标题适合相同块组和彼此相邻。在另一情况下，合并的 2 标题具有它们自己的块组，块和数据流。在合并那两个标题的处理中，块组，块和数据流按一个顺序安排，用这个顺序合并标题和形成单个块组。

接下来解释程序。重放程序的操作意味着一种对集中必要的标题部分和对未加工的材料没有进行不可逆的编辑的集中的部分也就是数据流的重放的操作。实施这个操作的结构称之为程序。重放程序的操作也称之为指示符重放操作。

10 程序包括多个重放序列，每一个是由多个播放目录构成的。播放序列控制输出信道。那个光盘装置的输出信道被指定以播放根据光盘装置的规定确定的序列。

15 播放序列的播放目录用于表示记录的重叠和沿时间轴的播放项目间的显示操作。更详细地说，播放项目是按播放目录安排的，用那种方法播放项目的处理部分在时间轴上是不重叠的。因此，在播放目录中的播放项目可以顺序地处理。

在这种格式的例子中，程序只有 1 个播放序列，该序列只有一个播放目录。

20 播放目录包括播放项目，该项目被安排得以至在时间轴上没有重叠。在程序中，播放项目是由一对入点(in-point)和出点(out-point)表示的，它们指向标题的区域。在播放项目间的连接处，重放操作可以是无缝或有时也可以不是无缝的操作。也就是说无缝重放操作是不保障的。

25 通过参照图 55 所示的流程图解释设置重放程序的处理。现在，假设图 27 所示的存在 PROGRAM_001 的状态的例子，另一个程序 PROGRAM_002 被产生。

如图 55 所示，处理开始于步骤 S41，在该步骤使用者规定标题，和在重放标题中入点和出点以重放之。例如，在 TITLE_003 中的预定的第一和第二位置被规定如图 56 所示分别作为入点和出点。

30 处理的流程于是进行到步骤 S42，在该步骤，CPU 21 设置在步骤 S41 规定的标题，也就是 TITLE_003，在 program()的 play_list(播放__目录)的 play_item(播放__项目())的标题__数(title_number)中，在步骤 S41 在



item_start_time_stamp(项目__开始__时间__标记)中规定的入点和在步骤 S41 在 item_end_time_stamp(项目__结束__时间__标记)中规定的出点。应当注意到, play_item, play_list 和 program()是分别在图 20, 19 和 18 中表示的。

于是处理流程进行到步骤 S43, 在该步骤, CPU 21 产生作为存储
5 program()的文件的 PROGRAM_&&&.PGI 和记录该文件到光盘 1。应当注
意到, 在该文件的名称中的标记&&&是产生的程序的数码。因此, 在这个
情况标记\$\$\$是 002。如前所述, 产生图 56 中所示的 PROGRAM_002。

接下来, 参照图 57 中所示的流程图解释重放程序的处理。如图所示,
10 该处理开始于步骤 S51, 在该步骤 CPU 21 从光盘 1 读出信息文件, 该光盘
包括图 27 或 56 中所示的类似的一些文件, 这是等使用者在光盘装置上装上
光盘 1 即刻通过重复基本操作以读入早先描述过的信息文件。该 CPU 21 首
先存储信息文件在 RAM 单元 24 中。

更详细地说, 首先, 该 CPU 21 读出 VOLUME.TOC 和 ALBUM.STR,
和然后对文件系统作出有关位于在命名“program”的目录下面每一个具有
15 “.PIG”的扩展的文件的数码的查询。具有该扩展“.PIG”的文件是在程序
重放文件上存储信息的文件。每个具有扩展“.PIG”的文件的数码是能够重
放的程序的数码。在图 27 所示的例子中, 每具有扩展“.PIG”的文件数码
是 1。在图 56 中所示的例子中, 另一方面, 每个具有扩展“.PIG”的文件
数码是 2。

20 处理流程于是进行到步骤 S52, 在该步骤 CPU 21 控制 OSD 控制电路 9
以读取诸如可以重放的程序的数码, 它们的长度和来自 RAM 单元 24 的它们
的属性的信息和输出它们到显示单元。该标记包括每个程序的名字和每个程
序记录日期和时间。

于是处理流程进行到步骤 S53, 在该步骤, 使用者规定了要重放的程序。
25 由于在图 56 所示的例子中可以重放的是 2 个程序, 使用者通过操作输入单元
14 规定它们中的一个。由于使用者规定了重放的程序, 处理流程继续到步骤
S54, 在该步骤, CPU 21 完成重放规定的程序的操作。

在重放步骤 S54 完成规定的程序的操作中, 该程序从由规定的入点指示
的位置到由规定的出点指示的位置重放。例如, 在重放图 56 中所示的例子的
30 PROGRAM_001 的操作中, 在 TITLE_001 中的从入点到出点的范围和在
TITLE_003 中的从入点到出点的范围被重放在重放 PROGRAM_002 的操作

的情况下，另一方面，在 TITLE_003 中的从入点到出点的范围被重放。

于是，处理流程进行到步骤 S55，在该步骤，CPU 21 作出是否重放程序的操作已经完成的判断。如果该操作仍没有完成，该处理流程返回到步骤 S54 以重复在那个步骤的处理和以后的处理。如果在步骤 S55 作出的判断结果表示重放程序的操作已经完成，另一方面，处理流程进行到步骤 S56，在该步骤 CPU21 作出是否没有另外的程序要重放的判断。如果有另外的程序要重放，处理流程进行到步骤 S53，以重复执行在该步骤的处理和以后的处理。如果在步骤 S56 作出的判断结果指示所有的程序已经重放了，另一方面，该处理被终止。

10 在图 58 到 60 中示出了程序间的关系，播放的顺序和播放的项目。

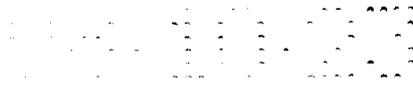
如图 58 中所示，程序包括多个以预定的顺序组合的播放顺序。如图 59 中所示，播放序列包括任意数目的播放目录。播放目录描述任意数目的对应预定时间(时间标记)播放项目。

在图 59 所示的例子中，播放目录 1 描述了一个项目在另一个项目后面顺序重放的播放项目 1，播放项目 2，播放项目 3，播放项目 4，播放项目 5，播放项目 6，和播放项目 7。另一方面，播放目录 2 描述开始于播放项目 2 的中间和结束于播放项目 3 的中间定时的重放的播放项目 4。作为播放目录 3，播放项目 5 的播放开始于播放项目 3 和 4 的中间，结束于播放项目 6 的中间。因此，在重放这个播放序列的操作中，播放项目 1，播放项目 2，播放项目 3，播放项目 6 和播放项目 7 是一个项目跟在另一个后面地顺序地重放，在相同的时间，播放项目 4 用从播放项目 2 的中间开始和结束于播放项目 3 的中间的定时重放，同时播放项目 5 用从播放项目 3 和 4 的中间开始到播放项目 6 的中间结束的定时重放。

程序间的关系，播放序列和播放项目归纳入图 60 中所示的单个图中。总而言之，程序包括任意数目的播放序列，每个序列包括任意数目的播放目录，在每个播放目录中包括任意数目的播放项目。

到目前所述，本发明已经举例说明将它们应用到光盘装置上。值得注意的是，本发明还可以应用在信息可以记录在或从其它的类型记录介质重放的情况。

30 应当注意，作为表现执行以为使用者完成前述的处理的计算机程序的显示介质，除了象磁盘，CD-ROM 和固-态存储器件的记录介质以外，还可以



利用诸如网络和卫星的通信介质。

5 如上所述，在根据本发明的信息处理装置中，多个数据段由第一种控制装置逐个(1-on-1)控制，并且任意数目的第一种控制装置由第二种控制装置控制。第二种控制中的任何任意范围由第三种控制装置控制。另外，在本发明的信息处理方法和显示介质中，在第一控制步骤中的任何任意数目的控制状态在第二控制步骤被控制和在第二控制步骤中的任何任意范围的控制状态在第三控制步骤被控制。此外，在本发明的记录介质中，作为控制信息，第一控制装置，第二控制装置和第三控制装置被记录。结果实现了用短的时间周期容易地完成编辑。

说明书附图

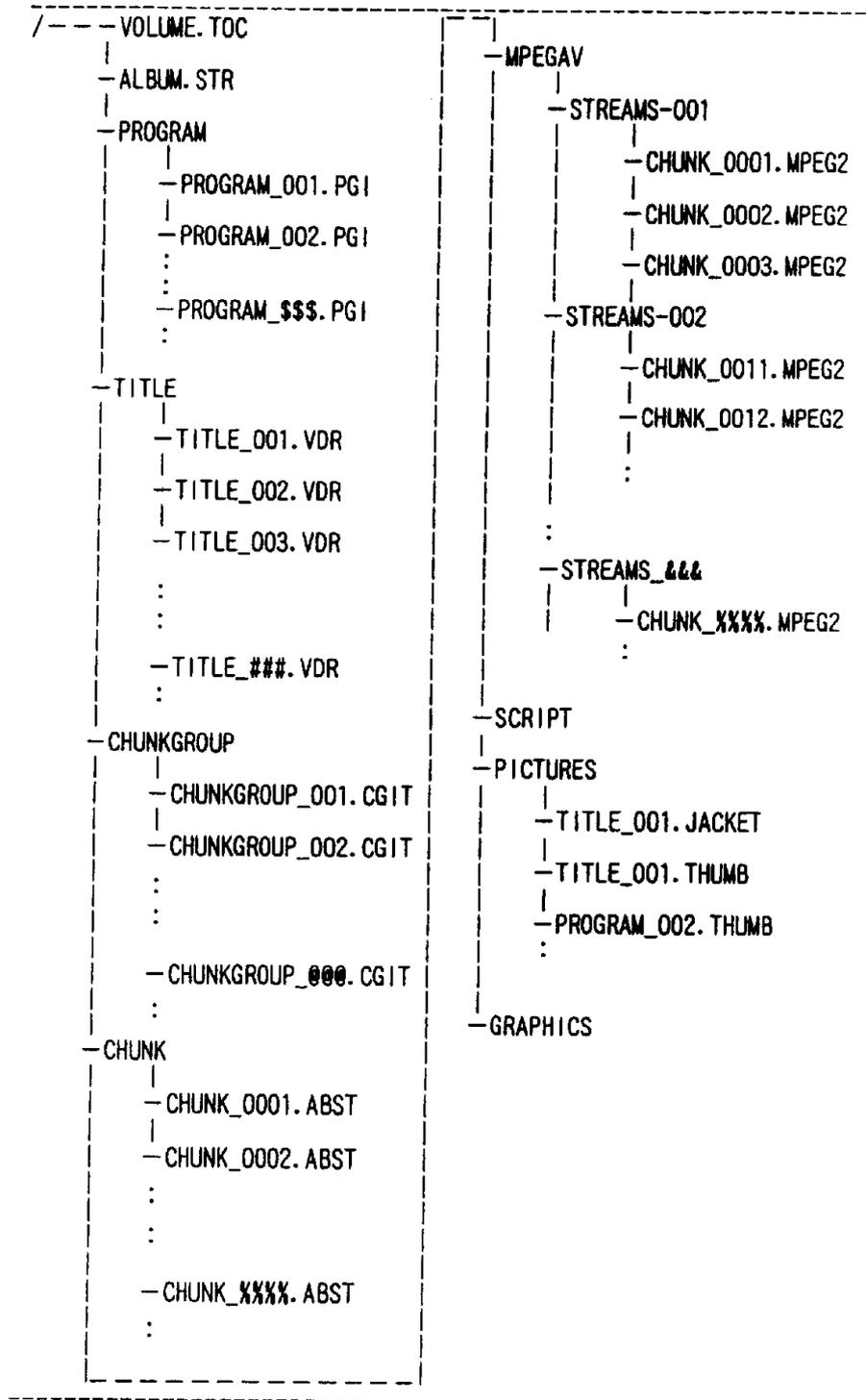


图 1

语法	比特数	助记符
VOLUME.TOC{ file_type_id volume_information() text_block() }	8*16	char[16]

图 2

语法	比特数	助记符
volume_information(){ volume_attribute() resume() volume_rating() write_protect() play_protect() recording_timer() }		

图 3

语法	比特数	助记符
volume_attribute(){ volume_attribute_length vdr_version reserved title_playback_mode_flag program_playback_mode_flag volume_play_time() update_time_count() maker_id model_code POSID }	32 4*4 6 1 1 4*8 32 8*16 8*16 32	uimsbf bcd bslbf bslbf bslbf bcd uimsbf char[16] char[16] bslbf

图 4

语法	比特数	助记符
resume() {		
resume_length	32	uimsbf
reserved // for byte alignment	3	bslbf
resume_switch	1	bit
reserved	4	bslbf
number_of_records	4	uimsbf
reserved // for byte alignment	7	bslbf
resume_auto_execute_time_flag	1	bit
resume_auto_execute_time()	4*14	bcd
reserved	4	bslbf
resume_auto_execute_record_number	4	uimsbf
for(i=0;i<number_of_records;i++) {		
resume_mode_flag	4	bslbf
object_type	4	bslbf
linked_record_number	4	uimsbf
number_of_times	16	uimsbf
resume_updated_time()	4*14	bcd
switch(object_type) {		
case title:		
title_number	16	uimsbf
title_local_time_stamp	64	uimsbf
break;		
case program:		
program_number	16	uimsbf
program_local_time_stamp	64	uimsbf
break;		
case program_bind:		
program_bind_number	16	uimsbf
program_order	16	uimsbf
program_number	16	uimsbf
program_local_time_stamp	64	uimsbf
break;		
case play_item:		
play_item_number	16	uimsbf
play_item_local_time_stamp	64	uimsbf
break;		
}		
}		
}		

图 5

语法	比特数	助记符
volume_rating() {		
volume_rating_length	32	uimsbf
reserved	6	bslbf
volume_rating_type	2	bslbf
volume_rating_password	128	bslbf
switch (rating_type) {		
case age_limited:		
country_code_for_rating	32	bslbf
for(i=0; i<32; i++){		
rating_bit_for_age_limited	1	bslbf
break;		
case CARA:		
CARA_category	4	bslbf
reserved	4	bslbf
reserved	16	bslbf
break;		
case RSAC:		
RSAC_category	4	bslbf
level	4	bslbf
reserved	16	bslbf
break;		
}		
}		

图 6

语法	比特数	助记符
<code>write_protect() {</code>		
<code>write_protect_length</code>	32	uimsbf
<code>volume_write_protect_level</code>	4	uimsbf
<code>password_enable_flag</code>	1	bslbf
<code>append_only_flag</code>	1	bslbf
<code>expiration_time_enable_flag</code>	1	bslbf
<code>number_of_times_enable_flag</code>	1	bslbf
<code>password_for_volume_write_protect</code>	128	bslbf
<code>reserved</code>	8	bslbf
<code>write_protect_set_time()</code>	56	bcd
<code>reserved</code>	8	bslbf
<code>write_protect_expiration_time()</code>	56	bcd
<code>number_of_times</code>	16	uimsbf
<code>}</code>		

图 7

语法	比特数	助记符
play_protect() {		
play_protect_length	32	uimsbf
volume_play_protect_flag	2	bslbf
reserved	2	bslbf
password_enable_flag	1	bslbf
reserved	1	bslbf
expiration_time_enable_flag	1	bslbf
number_of_times_enable_flag	1	bslbf
password_for_volume_play_protect	128	bslbf
reserved	8	bslbf
play_protect_set_time()	56	bcd
reserved	8	bslbf
play_protect_expiration_time()	56	bcd
number_of_times	16	uimsbf
}		

图 8

语法	比特数	助记符
recording_timer() {		
recording_timer_length		
recording_timer_flag		
number_of_entry		
for(i=0; i<number_of_entry; i++) {		
date_and_time		
channel		
program		
:		
}		
}		

图 9

语法	比特数	助记符
text_block() {		
text_block_length	32	uimsbf
number_of_language_sets	8	uimsbf
number_of_text_items	16	uimsbf
for(i=0;i<number_of_language_sets;i++){		
language_set()		
}		
for(i=0;i<number_of_text_items;i++){		
text_item()		
}		
}		

图 10

语法	比特数	助记符
language_set() {		
reserved	8	bslbf
language_code	24	bslbf
character_set_type	8	bslbf
number_of_language_set_names	8	uimsbf
for(i=0;i<number_of_language_set_names;i++){		
character_set_type_for_name	8	bslbf
language_set_name_length	8	uimsbf
language_set_name 8*language_set_name_length		bslbf
}		
}		

图 11

语法	比特数	助记符
text_item() {		
text_item_length	16	uimsbf
text_item_id	16	uimsbf
text_item_sub_id	16	uimsbf
flags	8	bslbf
number_of_used_language_sets	8	uimsbf
//loop for each language set		
for(i=0;i<number_of_used_language_sets;i++){		
language_set_id	8	uimsbf
reserved	4	bslbf
text_string_length	16	uimsbf
text_string	8*text_string_length	bslbf
bitmap()		
}		
stuffing_bytes	8*n	bslbf
}		

图 12

语法	比特数	助记符
ALBUM. STR {		
file_type_id	8*16	char[16]
album()		
text_block()		
}		

图 13

语法	比特数	助记符
album() {		
album_length	32	uimsbf
reserved	6	bslbf
volume_status	1	bslbf
if (volume_status=="1b") {		
chief_volume_flag	1	bslbf
} else {		
reserved	1	"0"
}		
if (volume_status=="1b") {		
if (chief_volume_flag=="1b") {		
reserved	6	bslbf
album_type	2	bslbf
albim_id	128	bslbf
number_of_discs_in_album	16	uimsbf
number_of_volumes_in_album	16	uimsbf
for (i=0;i<number_of_volumes_in_album;i++) {		
disc_id_for_album_member	128	bslbf
volume_id_for_album_member	128	bslbf
title_offset_number	16	uimsbf
}		
reserved_for_program_bind	8	bslbf
number_of_program_binds	8	uimsbf
for (i=0;i<number_of_program_binds;i++) {		
number_of_program_in_this_program_bind	16	uimsbf
for (i=0;i<number_of_programs_in_this_program_bind;i++) {		
disc_id_for_program_bind_member	128	uimsbf
volume_id_for_program_bind_member	128	uimsbf
program_number	16	uimsbf
}		
}		
} else {		
//chief_volume_flag=="0b"		
chief_disc_id	128	uimsbf
chief_volume_id	128	uimsbf
(album_id	128	bslbf
}		
}		
}		

图 14

语法	比特数	助记符
TITLE_###. VDR{ file_type_id title_info() text_block() }	8*16	char [16]

图 15

语法	比特数	助记符
title_info() { title_info_length flags_for_title cgit_file_id title_start_chunk_group_time_stamp title_end_chunk_group_time_stamp title_playback_time() reserved number_of_marks for (i=0;i<number_of_marks;i++) { reserved mark_type mark_chunk_group_time_stamp } stuffing_bytes }	32 32 16 64 64 32 32 16 4 4 64 8*n	uimsbf bslbf uimsbf uimsbf uimsbf bcd bslbf uimsbf bslbf bslbf uimsbf bslbf

图 16

语法	比特数	助记符
PROGRAM_\$\$\$ PGI { file_type_id program() text_block() }	8*16	char[16]

图 17

语法	比特数	助记符
program() {		
program_length	32	uimsbf
flags_for_program	32	bslbf
program_status	4	bslbf
program_playback_time()	32	bcd
reserved	32	bslbf
number_of_play_sequences	16	uimsbf
for (j=0; j<number_of_play_sequence; j++) {		
number_of_play_lists	16	uimsbf
for (k=0; k<number_of_play_lists; k++) {		
play_list_start_time_stamp_offset	64	uimsbf
play_list(k)		
}		
}		
stuffing_bytes	8*n	bslbf
}		

图 18

语法	比特数	助记符
play_list() {		
//playback sequence of play items in this play list		
number_of_play_items	16	uimsbf
for (k=0; k<number_of_play_items; k++) {		
play_item_number	16	uimsbf
reserved	31	bslbf
seamless_connection_flag	1	bslbf
}		
//play_item_table		
for (PIN=1; PIN<=number_of_play_items_in_program; PIN++) {		
play_item()		
}		
}		

图 19

语法	比特数	助记符
play_item() {		
play_item_length	32	uimsbf
play_item_type	8	bslbf
play_mode	8	bslbf
total_playback_time()	32	bcd
menu_item_number	16	uimsbf
return_item_number	16	uimsbf
next_item_number	16	uimsbf
prev_item_number	16	uimsbf
if(play_item_type= "0000b") {		
//play item for one "cut"		
title_number	16	uimsbf
//IN point		
item_start_time_stamp	64	uimsbf
//OUT point		
item_end_time_stamp	64	uimsbf
}		
}		

图 20

语法	比特数	助记符
CHUNKGROUP_###.CGIT{		
file_type_id	8*16	char[16]
chunkgroup_time_base_flags	32	bslbf
chunkgroup_time_base_offset	64	uimsbf
chunk_connection_info()		
text_block()		
}		

图 21

语法	比特数	助记符
chunk_connection_info() {		
chunk_connection_info_length	32	uimsbf
reserved	16	bslbf
number_of_chunks	16	uimsbf
chunk_sync_play_flag	8	bslbf
// chunk info file list		
for(i=0; i<number_of_chunks; i++){		
chunk_arrangement_info()		
}		
}		

图 22

语法	比特数	助记符
chunk_arrangement_info() {		
chunk_arrangement_info_length	32	uimsbf
chunk_info_file_id	16	bslbf
reserved	5	bslbf
chunk_switch_stream_id	16	bslbf
presentation_start_cg_time_count	64	uimsbf
presentation_end_cg_time_count	64	uimsbf
reserved	4	bslbf
chunk_time_count_type	4	bslbf
number_of_start_original_time_count_extension	8	uimsbf
number_of_end_original_time_count_extension	8	uimsbf
// presentation start position and time		
presentation_start_original_time_count	64	uimsbf
presentation_end_original_time_count	64	uimsbf
for(i=0; j<number_of_start_original_time_count_extension; j++){		
tc_ext_attributes	16	bslbf
start_original_time_count_extension	64	uimsbf
}		
// presentation end position and time		
for(k=0; k<number_of_end_original_time_count_extension; k++){		
tc_ext_attributes	16	bslbf
end_original_time_count_extension	64	uimsbf
}		
transition_info()		
}		

图 23

语法	比特数	助记符
CHUNK_%%%.ABST(file_type_id	8*16	char[16]
reserved	4	bslbf
chunk_file_id	16	uimsbf
info_type	4	bslbf
//stream_info()		
if (info_type== "MPEG2_System_TS") { number_of_programs	8	uimsbf
else { number_of_programs	8	"0000 0001"
}		
for(i=0;i<number_of_programs;i++){ number_of_streams	8	uimsbf
for (i=0;i<number_of_streams;i++) { stream_identifier	16	bslbf
//slot type information	4	
reserved	4	bslbf
slot_unit_type		bslbf
if (slot_unit_type== "time_stamp") { slot_time_length	32	uimsbf
} else { reserved	32	bslbf
}	32	
number_of_slots	4	uimsbf
reserved		bslbf
switch(info_type) case MPEG1_System: case MPEG2_System_PS: case MPEG2_System_TS: case video_elementary_stream	4	uimsbf
number_of_l_pictures_in_slot break;		
default: reserved break;	4	bslbf
}		
//stream attribute ES_attribute()		
}		
//loop of slot info for (i=0;i<number_of_streams;i++) { for (i=0;i<number_of_slots;i++) { slot_info()		
}		
}		
}		

图 24

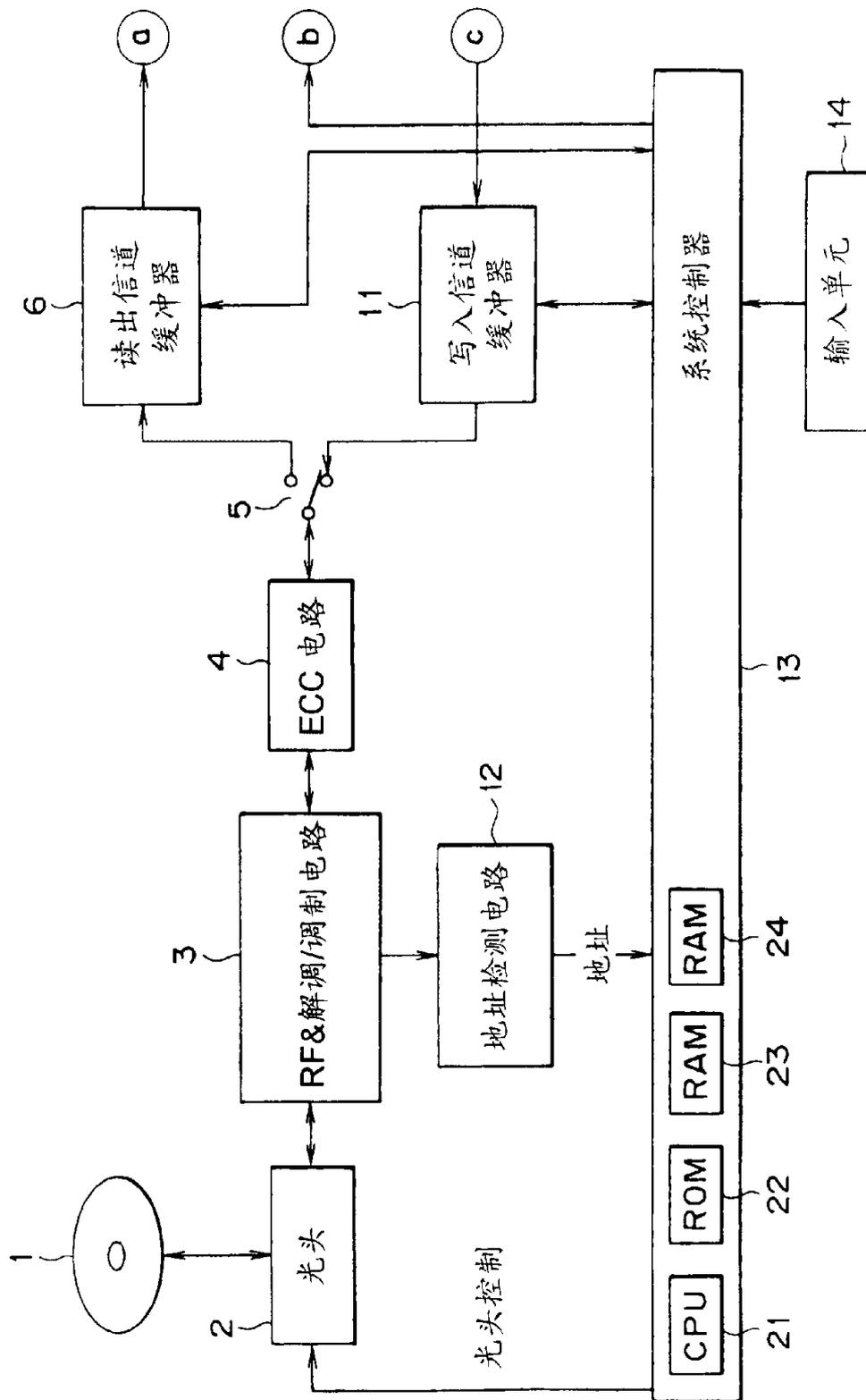


图 25A

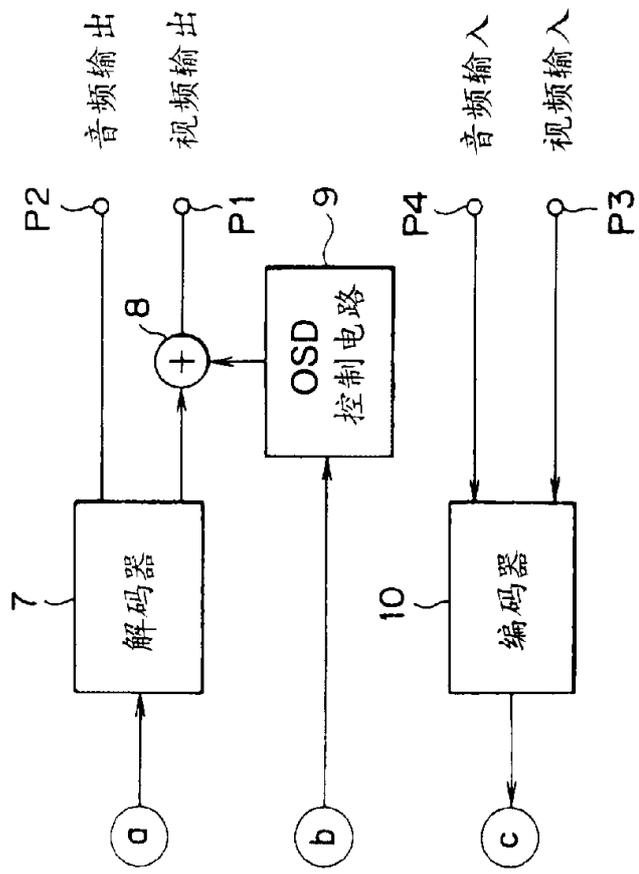


图 25B

```

| --- VOLUME.TOC
| --- ALBUM.STR
| --- PROGRAM
| | --- PROGRAM__001.PGI
| --- TITLE
| | --- TITLE__001.VDR
| | --- TITLE__002.VDR
| | --- TITLE__003.VDR
| --- CHUNKGROUP
| | --- CHUNKGROUP__001.CGIT
| | --- CHUNKGROUP__002.CGIT
| --- CHUNK
| | --- CHUNK__0001.ABST
| | --- CHUNK__0011.ABST
| | --- CHUNK__0012.ABST
| --- MPEGAV
| | --- STREAMS__001
| | | --- CHUNK__0001.MPEG2
| | --- STREAMS__002
| | | --- CHUNK__0011.MPEG2
| | | --- CHUNK__0012.MPEG2

```

图 26

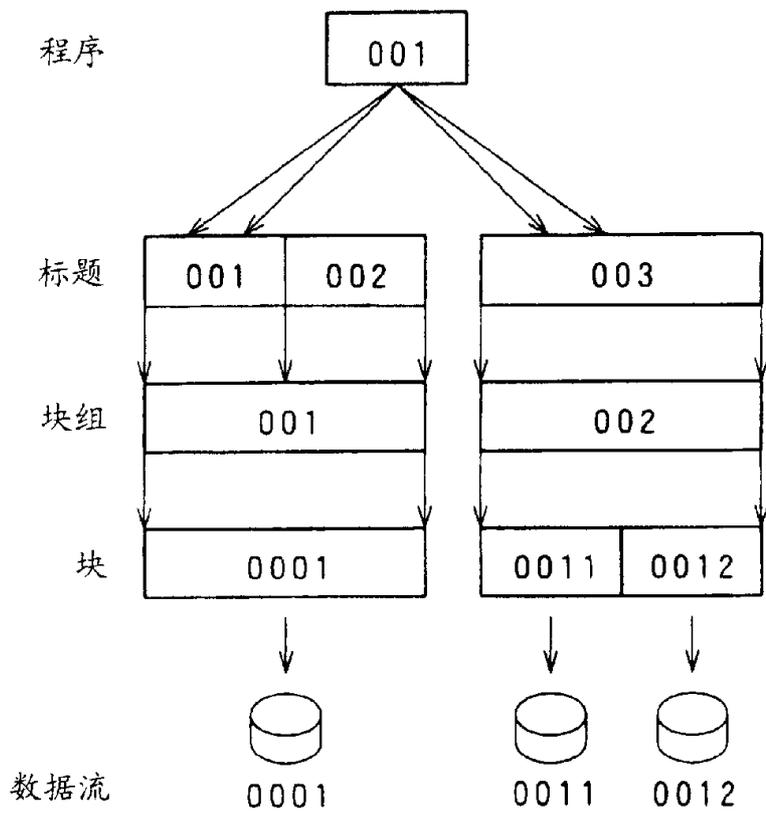


图 27

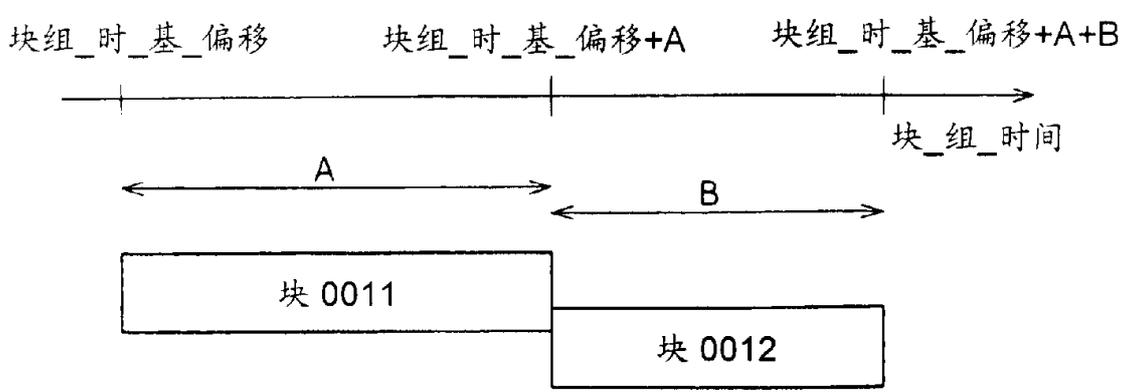


图 28

```

/-----MPEGAV
|   -STREAMS_003
|   |   -CHUNK_0031.MPEG2

```

图 29

```

/---VOLUME.TOC
-ALBUM.STR
-PROGRAM
|   -PROGRAM_001.PGI
-TITLE
|   -TITLE_001.VDR
|   -TITLE_002.VDR
|   -TITLE_003.VDR
|   -TITLE_004.VDR*
|
-CHUNKGROUP
|   -CHUNKGROUP_001.CGIT
|   -CHUNKGROUP_002.CGIT
|   -CHUNKGROUP_003.CGIT*
|
-CHUNK
|   -CHUNK_0001.ABST
|   -CHUNK_0011.ABST
|   -CHUNK_0012.ABST
|   -CHUNK_0031.ABST*
|
-MPEGAV
|   -STREAMS_001
|   |   -CHUNK_0001.MPEG2
|   |
|   -STREAMS_002
|   |   -CHUNK_0011.MPEG2
|   |   -CHUNK_0012.MPEG2
|   |
|   -STREAMS_003*
|   |   -CHUNK_0031.MPEG2*
|

```

图 30

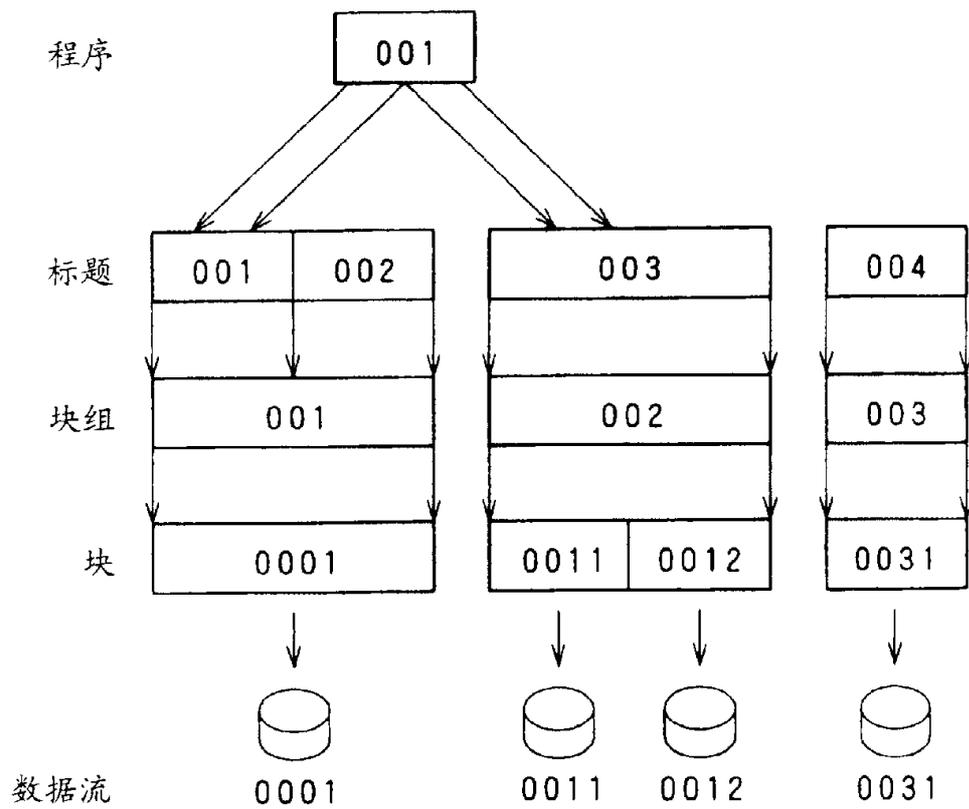


图 31

```

/-----MPEGAV
|       -STREAMS_002
|       |       -CHUNK_0031.MPEG2

```

图 32

```

/---VOLUME.TOC
-ALBUM.STR
-PROGRAM
|   -PROGRAM_001.PGI
-TITLE
|   -TITLE_001.VDR
|   -TITLE_002.VDR
|   -TITLE_003.VDR
|   -TITLE_004.VDR*
-CHUNKGROUP
|   -CHUNKGROUP_001.CGIT
|   -CHUNKGROUP_002.CGIT
-CHUNK
|   -CHUNK_0001.ABST
|   -CHUNK_0011.ABST
|   -CHUNK_0012.ABST
|   -CHUNK_0031.ABST*
-MPEGAV
|   -STREAMS_001
|       |   -CHUNK_0001.MPEG2
|   -STREAMS_002
|       |   -CHUNK_0011.MPEG2
|       |   -CHUNK_0012.MPEG2
|       |   -CHUNK_0031.MPEG2*

```

图 33

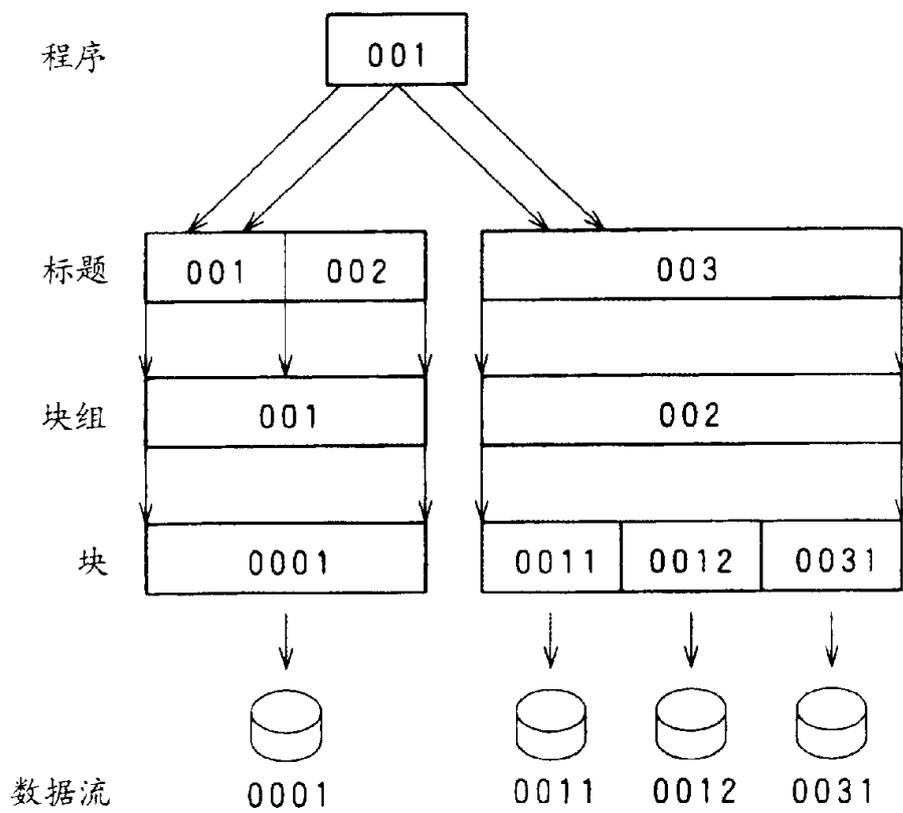


图 34

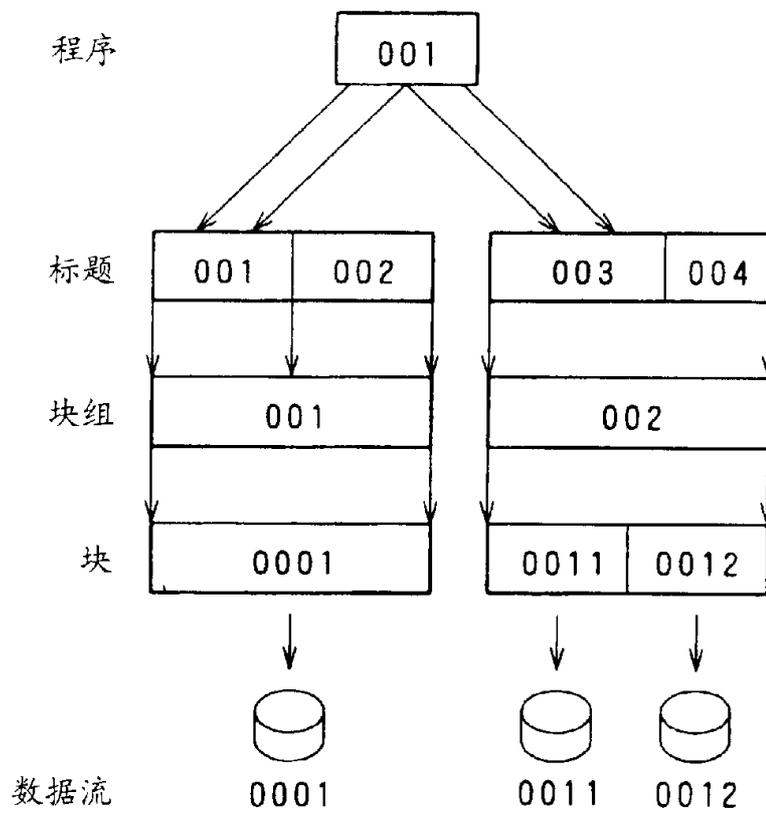


图 35

字段名称	值
文件_类型_识别符	"标题_信息_00000"

图 36

标志_类型	意义
0000b	索引类型 1
0001b	索引类型 2
0010b..0111b	索引类型 3...索引类型 8
1000b	跳入
1001b	跳出
1010b	跳到标题结束或下一个标题
1011b	景物变化
1100b	音频无声
1101b	音频峰值
1110b	静止图象
1111b	保留

索引: 在标题中的直接进入点

跳入: 标题跳过的开始点

跳出: 标题跳过的结束点

图 37

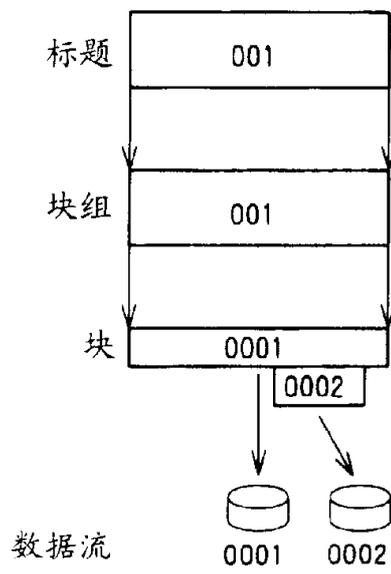


图 38

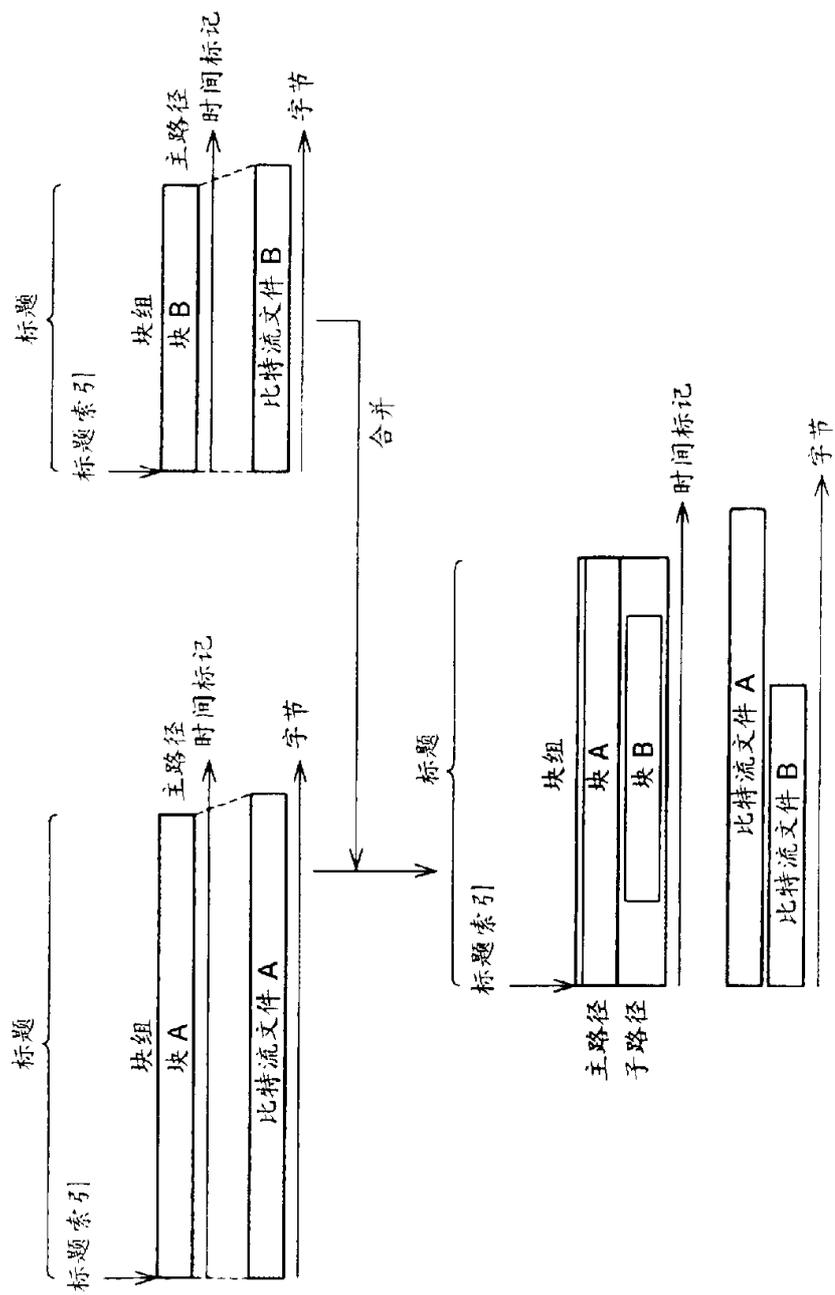


图 39

字段名称	值
文件_类型_识别符	“块组_0000”

图 40

块_同步_播放_标记	意义
0b	在相同时间仅播放一个块
1b	需要同时播放块

图 41

初始_时间_计算_类型	意义
0000	MPEG2 系统时间标记 (90kHz)
0001	SMPTE 时间代码
0010	字段 (525/60) (59.94Hz)
0011	字段 (625/50)
0100	帧 (525/60) (29.97Hz)
0101	帧 (625/50)
0110	引入帧时间代码 (525/60)
0111	非引入帧时间代码 (525/60)
1000	60Hz
1001	50Hz
1010	24Hz
1011	秒
1100..1111	保留

图 42

字段名称	值
文件_类型_识别符	“数据流_信息_0000”

图 43

文件_类型	意义
0000	MPEG2_系统_PS
0001	MPEG2_系统_TS
0010	MPEG2_系统_PES
0011	视频_基本_流
0100	除_视频的_基本_流
0101	MPEG1_系统_流
0110	保留
0111	保留
1000	用户_DVC
1001 .. 1111	保留

图 44

时隙_单位_类型	意义
0000b	'时间_标记': 时间标记值
0001b	'GOP': 1个GOP (图象组)
0010b	'音频_帧': 1个音频帧
0011b..1111b	保留

图 45

字段名称	值
文件_类型_识别符	"程序_信息_000"

图 46

程序_状态	意义
0000b	无
0001b	初始
0010b	复制
0011b	预检
0100b	排练
0101b	暂时
0110b	完成
0111b	中止
1000b	编辑
1001b	备用
1001b .. 1111b	保留

图 47

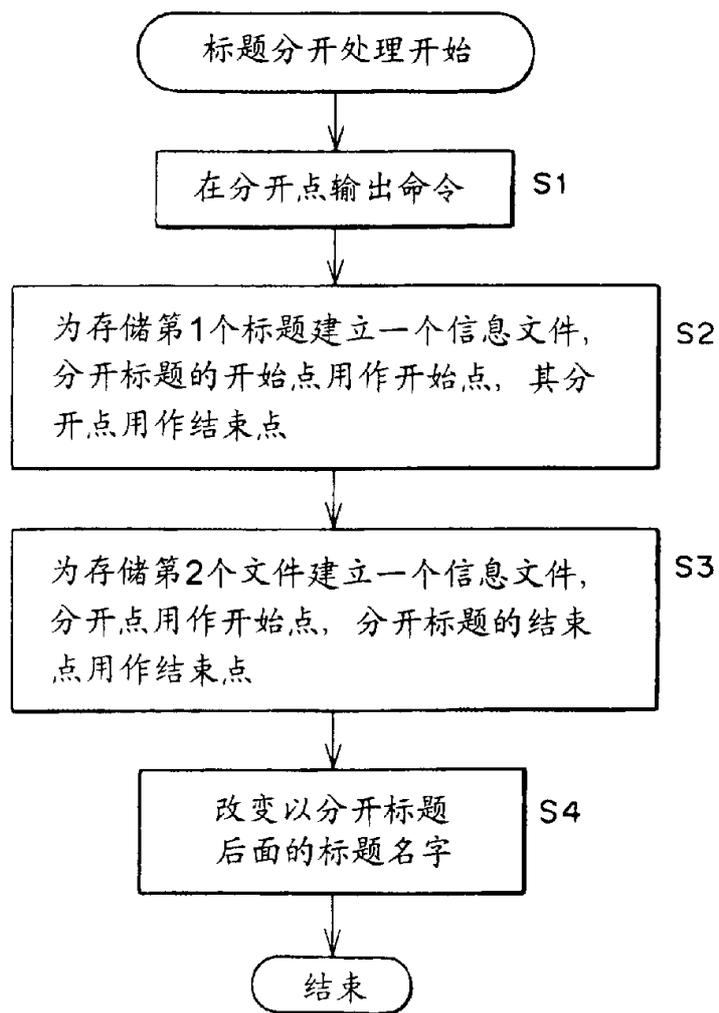


图 48

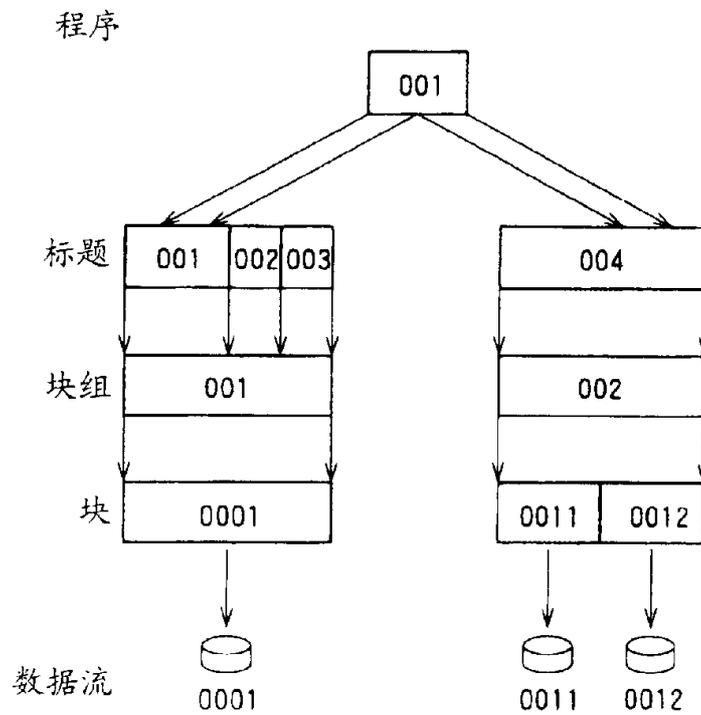


图 49

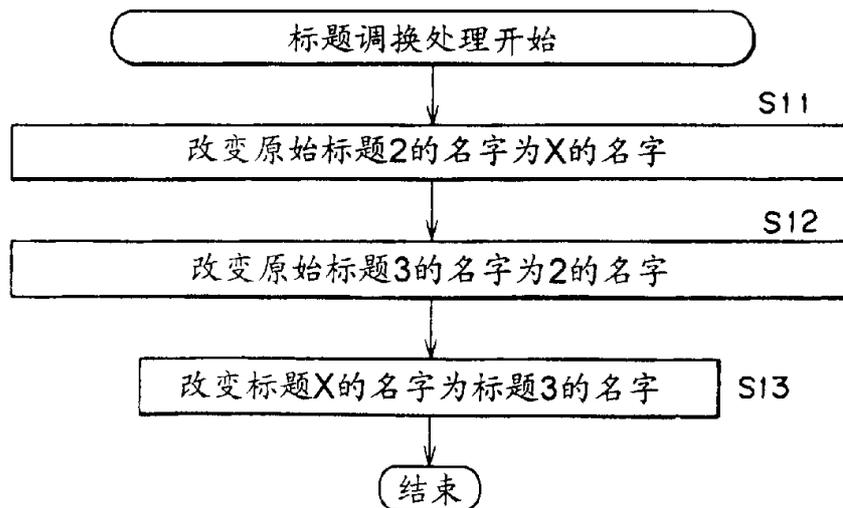


图 50

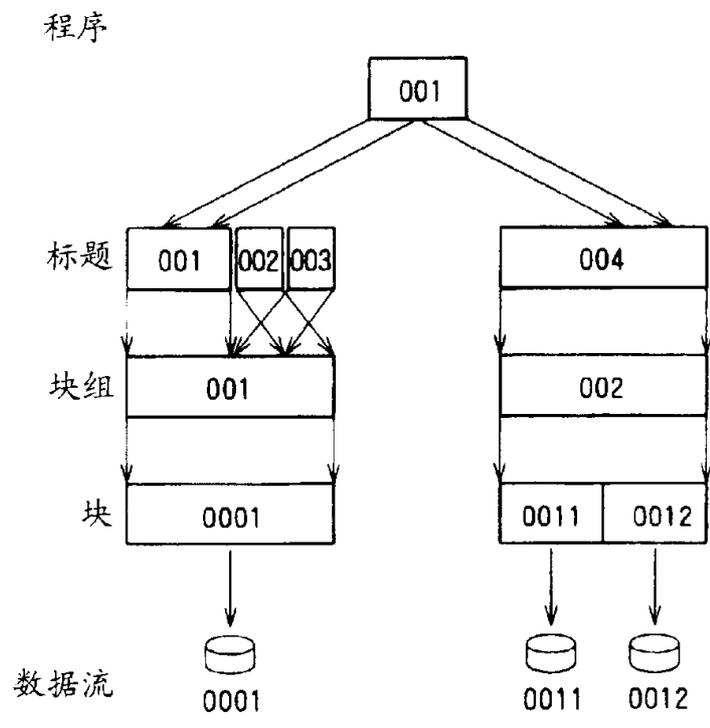


图 51

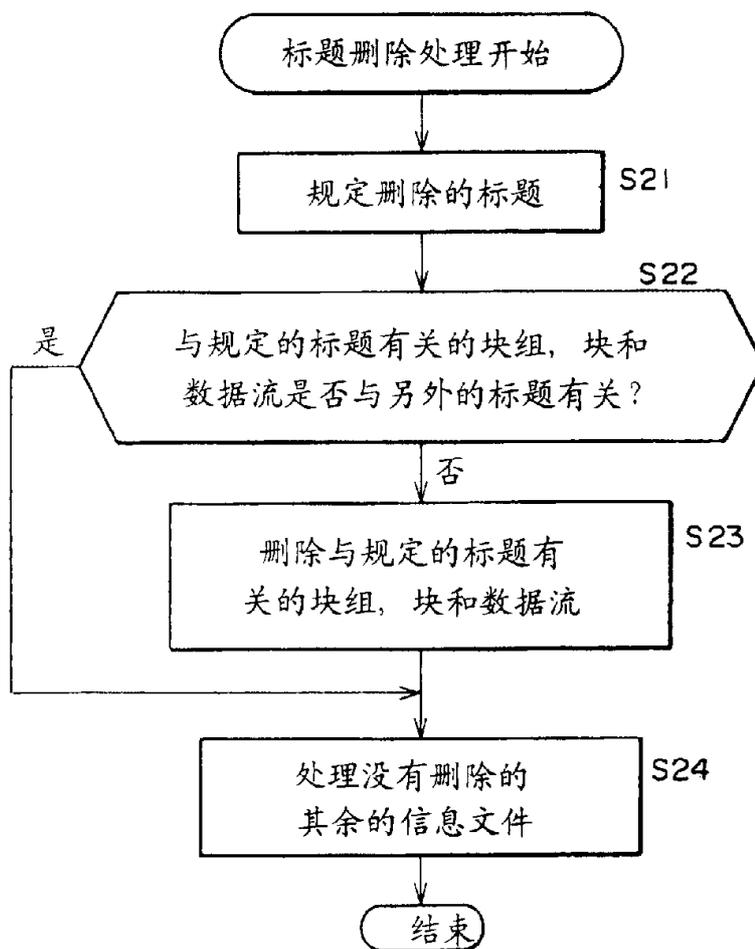


图 52

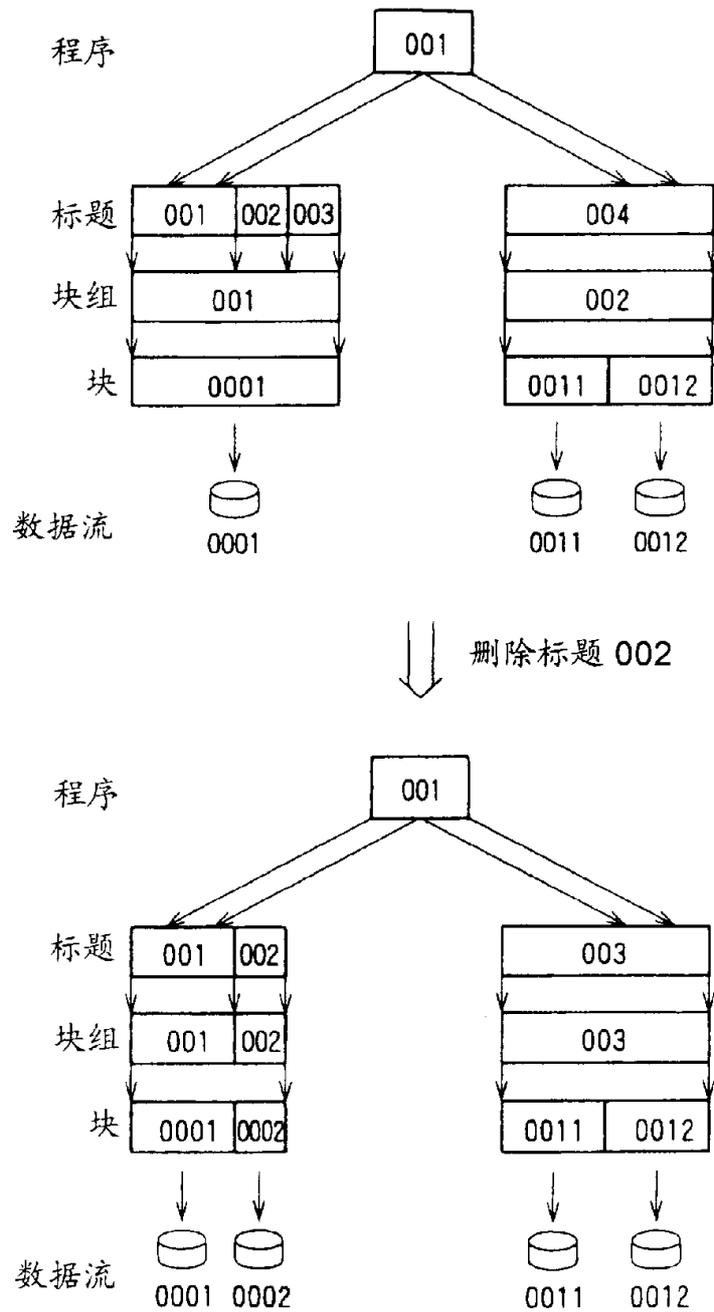


图 53

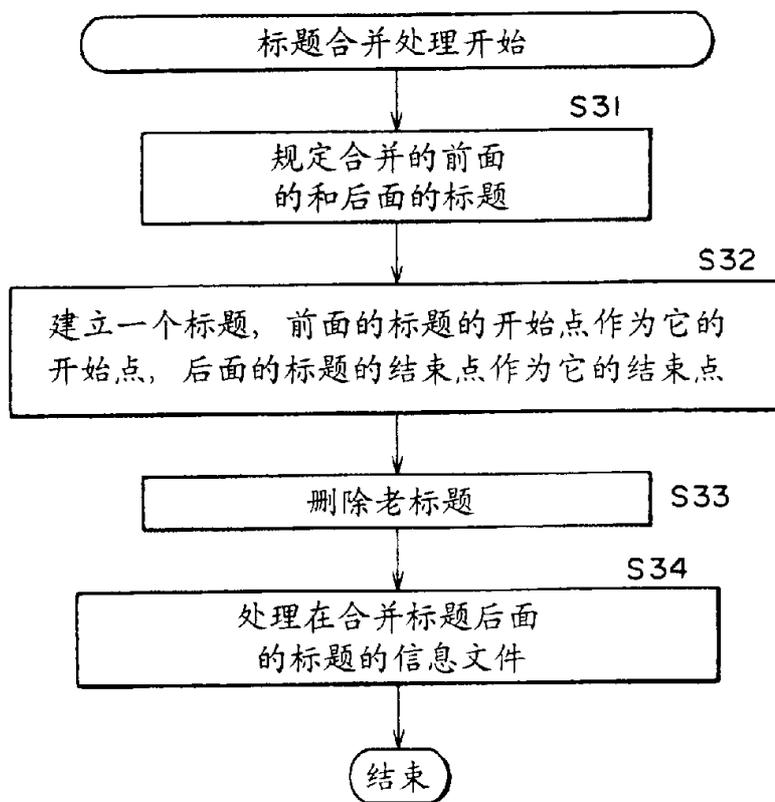


图 54

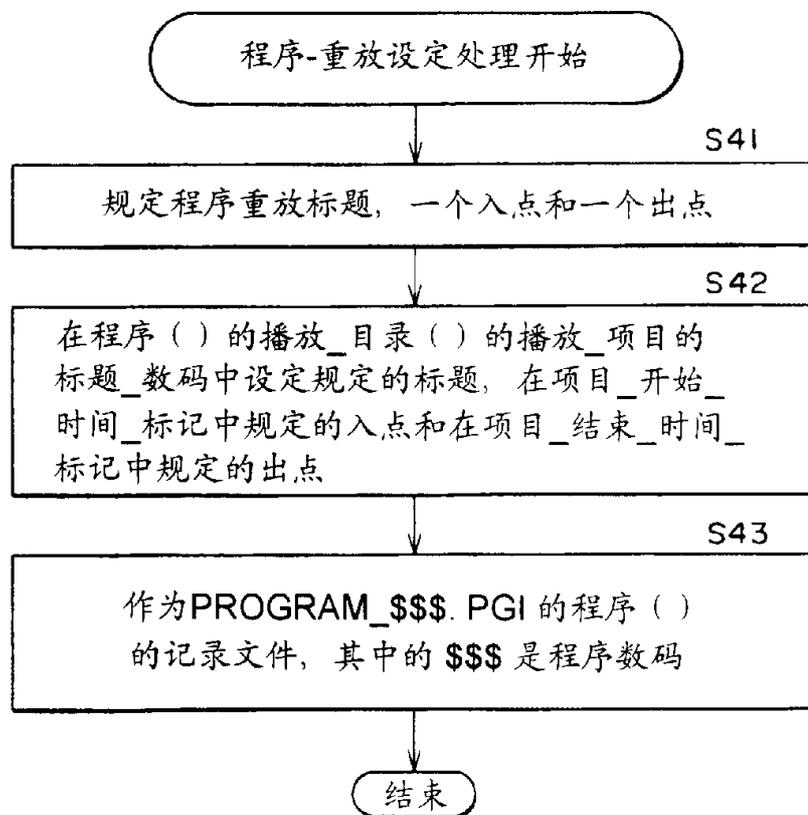


图 55

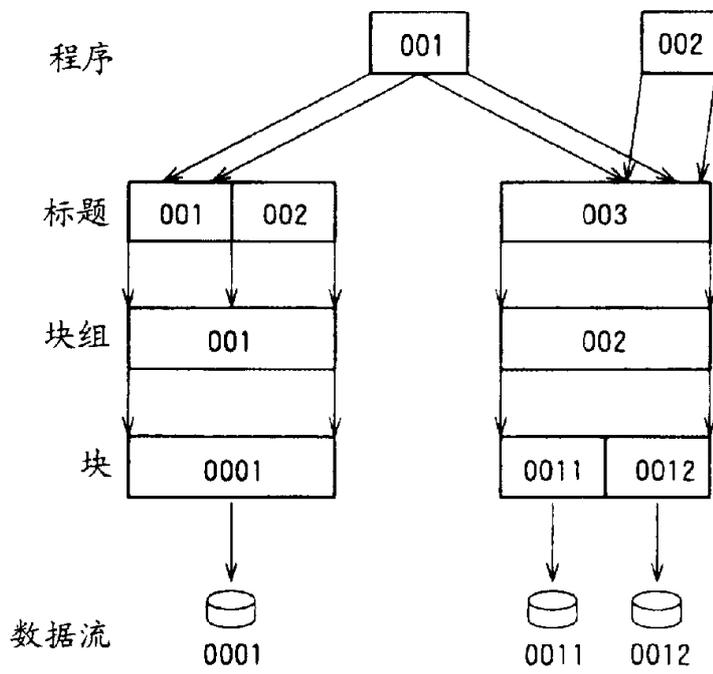


图 56

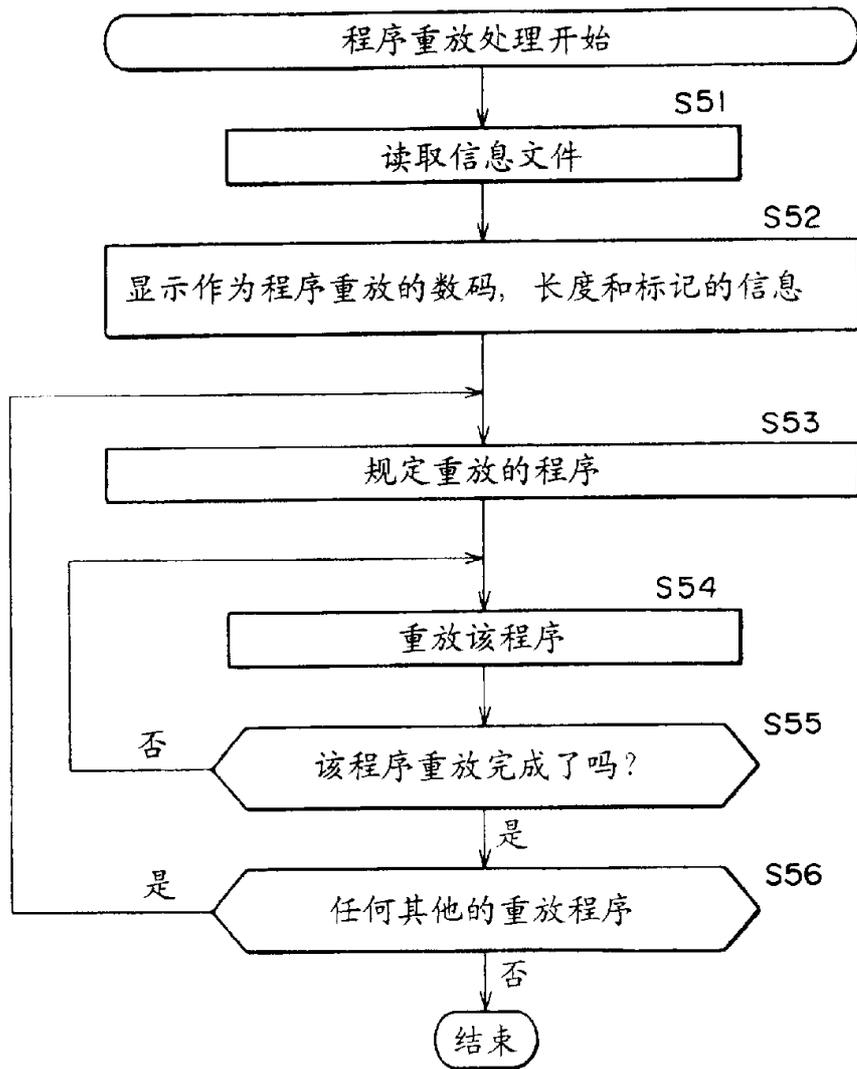


图 57

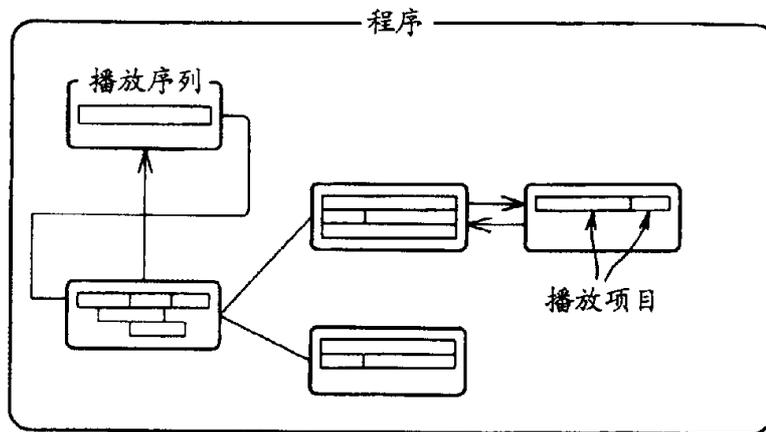


图 58

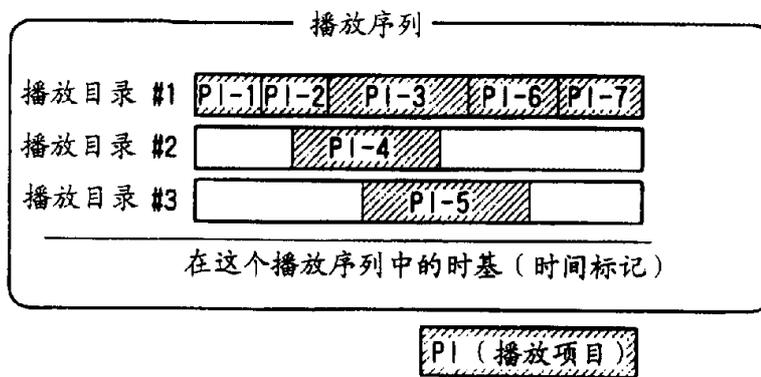


图 59

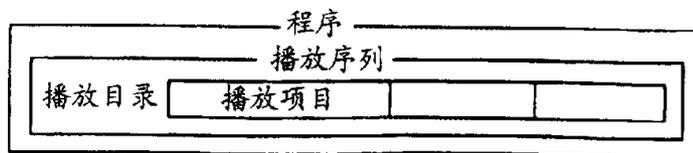


图 60