РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 335 856** (13) **C1**

(51) МПК
H04N 5/92 (2006.01)
G11B 20/12 (2006.01)
H04N 7/32 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ,
ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21), (22) Заявка: **2007120508/09, 17.11.2005**

(24) Дата начала отсчета срока действия патента:
17.11.2005

(30) Конвенционный приоритет:
02.12.2004 JP 2004-350487

(45) Опубликовано: **10.10.2008 Бюл. № 28**

(56) Список документов, цитированных в отчете о поиске: **JP 11261962 A, 24.09.1999. RU 2228546 C2, 10.05.2004. RU 2158969 C2, 10.11.2000. JP 2004328774 A1, 18.11.2004. EP 0614186 A2, 07.09.1994.**

(85) Дата перевода заявки PCT на национальную фазу:
01.06.2007

(86) Заявка PCT:
JP 2005/021500 (17.11.2005)

(87) Публикация PCT:
WO 2006/059520 (08.06.2006)

Адрес для переписки:
103735, Москва, ул.Ильинка, 5/2, ООО "Союзпатент", пат.пов. С.В.Истомину

(72) Автор(ы):
ФУДЗИНАМИ Ясуси (JP),
ХАМАДА Тосия (JP),
КАКУМУ Тацуя (JP),
УЕДА Акихико (JP)

(73) Патентообладатель(и):
СОНИ КОРПОРЕЙШН (JP),
СОНИ КОМПЬЮТЕР ЭНТЕРТЕЙНМЕНТ ИНК. (JP)

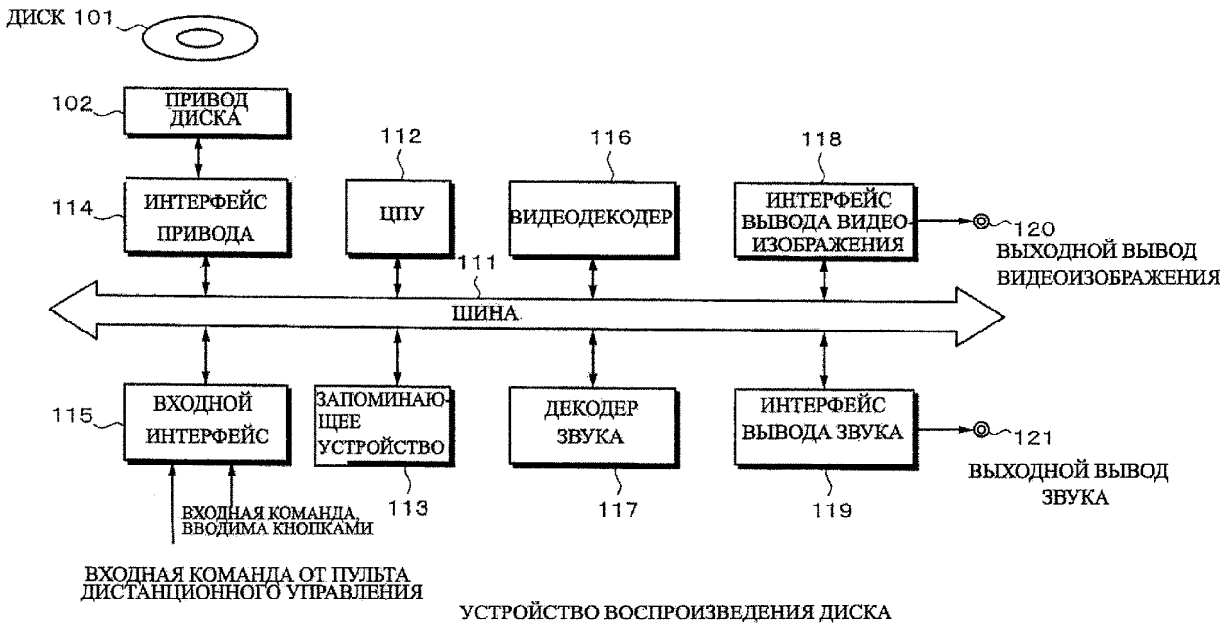
RU 2335856 C1

RU 2335856 C1

(54) УСТРОЙСТВО ЗАПИСИ ДАННЫХ, СПОСОБ ЗАПИСИ ДАННЫХ, УСТРОЙСТВО ОБРАБОТКИ ДАННЫХ, СПОСОБ ОБРАБОТКИ ДАННЫХ, ПРОГРАММА, НОСИТЕЛЬ ЗАПИСИ ПРОГРАММЫ, НОСИТЕЛЬ ЗАПИСИ ДАННЫХ И СТРУКТУРА ДАННЫХ

(57) Реферат:

Изобретение относится к системам записи и обработки данных. Техническим результатом является собственно создание способа обработки данных, в котором процессы могут быть эффективно выполнены с постоянным временем считывания данных. Воспроизведение с переменной скоростью выполняется без ухудшения качества изображения. Контроллер 425 создает EP_map () в файле информации клипа с адресом RAPI, выделенным модулем 423 выделения информации RAPI, PTS изображения с внутренним кодированием, которому непосредственно предшествует RAPI, и одним из конечных положений изображения с внутренним кодированием и второго, третьего и четвертого эталонных изображений, которому предшествует изображение с внутренним кодированием, и сохраняет EP_map () в выходном сервере 426, т.е контроллер 425 копирует значение, близкое к заданной величине подсчета секторов (количество секторов, которые можно считать за один раз во время процесса кодирования) конечных положений четырех эталонных изображений (1stRef_picture, 2ndRef_picture, 3rdRef_picture и 4thRef_picture) до N-th_Ref_picture_copy, определяет index_minus1 на основе N-th_Ref_picture_copy и записывает его на диск. 6 н. и 2 з.п. ф-лы, 68 ил.



Фиг. 1

RU 2335856 C1

RU 2335856 C1

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY,
PATENTS AND TRADEMARKS

(19) **RU** (11) **2 335 856** (13) **C1**

(51) Int. Cl.
H04N 5/92 (2006.01)
G11B 20/12 (2006.01)
H04N 7/32 (2006.01)

(12) ABSTRACT OF INVENTION

(21), (22) Application: **2007120508/09, 17.11.2005**

(24) Effective date for property rights: **17.11.2005**

(30) Priority:
02.12.2004 JP 2004-350487

(45) Date of publication: **10.10.2008 Bull. 28**

(85) Commencement of national phase: **01.06.2007**

(86) PCT application:
JP 2005/021500 (17.11.2005)

(87) PCT publication:
WO 2006/059520 (08.06.2006)

Mail address:
103735, Moskva, ul.Il'inka, 5/2, OOO "Sojuzpatent", pat.pov. S.V.Istominu

(72) Inventor(s):
FUDZINAMI Jasusi (JP),
KhAMADA Tosija (JP),
KAKUMU Tatsuja (JP),
UEDA Akikhiko (JP)

(73) Proprietor(s):
SONI KORPOREJShN (JP),
SONI KOMP'JuTER EhNTERTEJNMENT INK. (JP)

(54) DATA RECORDING DEVICE, DATA RECORDING METHOD, DATA PROCESSING DEVICE, DATA PROCESSING METHOD, SOFTWARE, SOFTWARE MEDIA, DATA MEDIA AND STRUCTURE

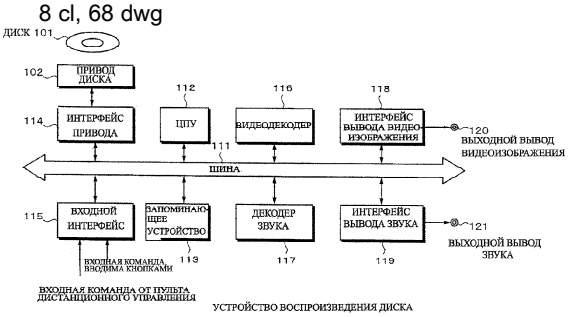
(57) Abstract:

FIELD: information technology.

SUBSTANCE: playback with variable speed is performed without picture quality deterioration. Controller 425 creates EP_map () with RAPI address in videoclip information file, dedicated information selection module 423 RAPI, image PTS with internal encoding, which is immediately preceded by RAPI, one of final positions of the picture with internal encoding, as well as the second, the third and the fourth reference pictures, which are preceded by the picture with internal encoding. The controller saves EP_map () in output server 426, i.e. controller 425 copies the value, close to given number of sectors (quantity of sectors, which can be read at one time during encoding process) of final positions for the four reference pictures (1stRef_picture, 2ndRef_picture, 3rdRef_picture and 4thRef_picture) to N-th_Ref_picture_copy, defines value of index_minus 1 on the basis of N-th_Ref_picture_copy and records it to disc.

EFFECT: effective process performance with constant data reading time.

8 cl, 68 dwg



Фиг. 1

RU 2335856 C1

RU 2335856 C1

Область техники, к которой относится изобретение

Настоящее изобретение относится к устройству записи данных, способу записи данных, устройству обработки данных, способу обработки данных, программе, носителю записи программы, носителю записи данных и структуре данных, в частности к таким, которые позволяют чрезвычайно удобно обрабатывать данные.

Уровень техники

В последние годы, например, цифровой универсальный диск (ЦУД, DVD) получил распространение в качестве носителя записи большой емкости и с возможностью оперативного доступа. Кроме того, также широкое распространение получили устройства DVD, которые выполняют различную обработку DVD.

Устройства DVD включают в себя устройства записи DVD, которые выполняют запись данных широковещательных телевизионных программ и т.д. на DVD и воспроизводят эти программы с DVD, автомобильные навигационные системы, в которых используются DVD, на которых записана информация карты и т.д., которые воспроизводят информацию карты с диска DVD и отображают ее, игровые устройства, в которых используются DVD, на которых записаны игровые программы и т.д., которые считывают программу с диска DVD и выполняют ее, и другие устройства.

Подробно диск DVD описан, например, в непатентном документе 1 "DVD Specifications for Read-Only Disc Part 3; Version 1.1 December 1997".

Устройство, такое как устройство DVD, которое может обрабатывать большое количество данных, должно позволять чрезвычайно удобно обрабатывать эти данные.

Сущность изобретения

Настоящее изобретение было разработано с учетом указанной выше проблемы, и цель настоящего изобретения состоит в разработке чрезвычайно удобной обработки данных.

Настоящее изобретение направлено на устройство записи данных, которое включает в себя блок детектирования, который выбирает информацию об изображении с внутренним кодированием, содержащемся в видеоданных, и которое становится точкой оперативного доступа и детектирует информацию положения множества изображений, перед которыми расположено выбранное изображение с внутренним кодированием; блок обозначения, который обозначает информацию положения изображения, расположенного на заданном расстоянии от изображения с внутренним кодированием в информации положения множества изображений, детектируемых блоком детектирования; и блок записи, который записывает информацию положения множества изображений, детектированных блоком детектирования, и информацию об изображении, обозначенном блоком обозначения, на носителе записи данных, вместе с видеоданными.

Устройство записи данных может дополнительно включать в себя блок выделения, который выделяет информацию положения для изображения, соответствующую заданному количеству секторов, в информации положения множества изображений, детектированных блоком детектирования. Блок обозначения обозначает информацию положения изображения, расположенного на заданном расстоянии от изображения с внутренним кодированием в информации положения множества изображений, детектированных блоком детектирования, в соответствии с информацией положения изображения, соответствующей заданному количеству секторов в информации положения множества изображений, выделенных блоком выделения. Блок записи записывает информацию о положении изображения, соответствующую заданному количеству секторов, в информации положения множества изображений, выделенных блоком выделения.

Заданное количество секторов устанавливается в соответствии с емкостью, которую устройство, воспроизводящее видеоданные с носителя записи данных, использует как буфер для видеоданных.

Настоящее изобретение направлено на способ записи данных, включающий в себя этапы выбора информации об изображении с внутренним кодированием, которая содержится в видеоданных и становится точкой оперативного доступа, и детектирования информации положения множества изображений, перед которыми расположено выбранное

изображение с внутренним кодированием; обозначения информации положения изображения, расположенной на заданном расстоянии от изображения с внутренним кодированием в информации положения множества изображений, детектированной на этапе детектирования; и записи информации положения множества изображений, детектированной на этапе детектирования и информации об изображении, обозначенной на этапе обозначения, на носителе записи данных вместе с видеоданными.

Настоящее изобретение направлено на первую программу, которая обеспечивает выполнение компьютером обработки, включающей в себя следующие этапы: выбора информации об изображении с внутренним кодированием, которое содержится в видеоданных и становится точкой оперативного доступа, и детектирования информации положения для множества изображений, перед которыми расположено выбранное изображение с внутренним кодированием; обозначения информации положения изображения, расположенного на заданном расстоянии от изображения с внутренним кодированием, в информации положения множества изображений, детектированной на этапе детектирования; и записи информации положения множества изображений, детектированных на этапе детектирования, и информации изображения, обозначенной на этапе обозначения, на носителе записи данных вместе с видеоданными.

Настоящее изобретение направлено на программу носителя записи первой программы, причем выполнение программы компьютером приводит к осуществлению следующих этапов способа: информации об изображении с внутренним кодированием, которая содержится в видеоданных и становится точкой оперативного доступа, и детектирования информации положения множества изображений, перед которыми расположено выбранное изображение с внутренним кодированием; обозначения информации положения изображения, расположенного на заданном расстоянии от изображения с внутренним кодированием, в информации положения множества изображений, детектированных на этапе детектирования; и записи информации положения множества изображений, детектированных на этапе детектирования, и информации изображения, обозначенной на этапе обозначения, на носителе записи данных вместе с видеоданными.

Настоящее изобретение направлено на устройство обработки данных, включающее в себя блок считывания, который считывает информацию положения множества изображений с внутренним кодированием и множества изображений, перед которыми расположено множество изображений с внутренним кодированием, и информацию обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, перед которыми расположено множество изображений с внутренним кодированием, с носителя записи данных, вместе с видеоданными, причем множество изображений с внутренним кодированием содержится в видеоданных и становится точками оперативного доступа; блок выбора, который выбирает изображения с внутренним кодированием и множество изображений, которым предшествуют изображения с внутренним кодированием, в соответствии с информацией обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием во множестве изображений с внутренним кодированием, которые становятся точками оперативного доступа видеоданных, считанных блоком считывания и записанных на носитель записи данных, и множество изображений, которым предшествуют изображения с внутренним кодированием; и блок воспроизведения, который выполняет воспроизведение с быстрой перемоткой вперед видеоданных со всеми изображениями от изображений с внутренним кодированием, до изображений, расположенных на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, которым предшествуют изображения с внутренним кодированием, и которые были выбраны блоком выбора.

Изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, могут представлять собой изображения, которые расположены от изображений с внутренним кодированием приблизительно на расстоянии заданного

количества секторов.

Заданное количество секторов может быть установлено в соответствии с емкостью, используемой в качестве буфера для видеоданных, когда видеоданные воспроизводят с носителя записи данных.

Настоящее изобретение направлено на способ обработки данных, включающий в себя следующие этапы: считывания информации положения множества изображений с внутренним кодированием и множества изображений, перед которыми расположено множество изображений с внутренним кодированием, и информации обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, перед которыми расположено множество изображений с внутренним кодированием, с носителя записи данных вместе с видеоданными, причем множество изображений с внутренним кодированием содержится в видеоданных и становится точками оперативного доступа; выбора изображений с внутренним кодированием и множества изображений, которым предшествуют изображения с внутренним кодированием, в соответствии с информацией обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений с внутренним кодированием, которые становятся точками оперативного доступа видеоданных, считанных на этапе считывания и записанных на носителе записи данных, и множества изображений, которым предшествуют изображения с внутренним кодированием; и воспроизведения с быстрой перемоткой вперед видеоданных со всеми изображениями от изображений с внутренним кодированием до изображений, расположенных на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, которым предшествуют изображения с внутренним кодированием и которые были выбраны на этапе выбора.

Настоящее изобретение направлено на вторую программу, причем выполнение программы компьютером приводит к осуществлению следующих этапов способа: считывания информации положения множества изображений с внутренним кодированием и множества изображений, перед которыми расположено множество изображений с внутренним кодированием, и информации обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, перед которыми расположено множество изображений с внутренним кодированием, с носителя записи данных вместе с видеоданными, причем множество изображений с внутренним кодированием содержатся в видеоданных и становятся точками оперативного доступа; выбора изображений с внутренним кодированием и множества изображений, которым предшествуют изображения с внутренним кодированием, в соответствии с информацией обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений с внутренним кодированием, которые становятся точками оперативного доступа видеоданных, считанных на этапе считывания и записанных на носителе записи данных, и множество изображений, которым предшествуют изображения с внутренним кодированием; и воспроизведения с быстрой перемоткой вперед видеоданных со всеми изображениями, от изображений с внутренним кодированием, до изображений, расположенных на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, которым предшествуют изображения с внутренним кодированием, и которые были выбраны на этапе выбора.

Настоящее изобретение направлено на программу носителя записи второй программы, причем выполнение программы компьютером приводит к осуществлению следующих этапов способа: считывания информации положения множества изображений с внутренним кодированием и множества изображений, перед которыми расположено множество изображений с внутренним кодированием, и информации обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, перед которыми расположено множество изображений с внутренним кодированием, с носителя записи данных вместе с

видеоданными, причем множество изображений с внутренним кодированием содержатся в видеоданных и становятся точками оперативного доступа; выбора изображений с внутренним кодированием и множества изображений, которым предшествуют изображения с внутренним кодированием, в соответствии с информацией обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений с внутренним кодированием, которые становятся точками оперативного доступа видеоданных, считанных на этапе считывания и записанных на носителе записи данных, и множества изображений, перед которыми расположены изображения с внутренним кодированием; и воспроизведения с быстрой перемоткой вперед видеоданных со всеми изображениями, от изображений с внутренним кодированием до изображений, расположенных на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, которым предшествуют изображения с внутренним кодированием и которые были выбраны на этапе выбора.

Настоящее изобретение представляет собой носитель записи данных, на котором записаны данные, включающие в себя информацию положения множества изображений, перед которыми расположено изображение с внутренним кодированием, выбранное из информации об изображении с внутренним кодированием, которое содержится в видеоданных и становится точкой оперативного доступа, и информацию, которая обозначает изображение, расположенное на заданном расстоянии от изображения с внутренним кодированием, в информации положения множества изображений.

Настоящее изобретение направлено на структуру данных, включающую в себя информацию положения множества изображений, перед которыми расположено изображение с внутренним кодированием, выбранное из информации об изображении с внутренним кодированием, которая содержится в видеоданных и становится точкой оперативного доступа, и информации, которая обозначает изображение, расположенное на заданном расстоянии от изображения с внутренним кодированием, в информации положения множества изображений.

В устройстве записи данных выбирают способ записи данных и первую программу в соответствии с настоящим изобретением, информацию об изображении с внутренним кодированием, которая содержится в видеоданных и становится точкой оперативного доступа. Детектируют информацию положения множества изображений, перед которыми расположено выбранное изображение с внутренним кодированием. Обозначают информацию положения изображения, расположенного на заданном расстоянии от изображения с внутренним кодированием, в информации положения множества изображений, которые были детектированы. Информацию положения множества изображений, которые были детектированы, и информацию изображения, которая была обозначена, записывают на носитель записи данных вместе с видеоданными.

В устройстве обработки данных считывают способ обработки данных и вторую программу в соответствии с настоящим изобретением, информацию положения множества изображений с внутренним кодированием и множества изображений, перед которыми расположено множество изображений с внутренним кодированием, и информацию обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, перед которыми расположено множество изображений с внутренним кодированием, с носителя записи данных вместе с видеоданными. Множество изображений с внутренним кодированием содержатся в видеоданных и становятся точками оперативного доступа. Выбирают изображения с внутренним кодированием и множество изображений, которым предшествуют изображения с внутренним кодированием, в соответствии с информацией обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, из множества изображений с внутренним кодированием, которые становятся точками оперативного доступа видеоданных, которые были считаны и записаны на носителе записи данных, и множество изображений, которым

предшествуют изображения с внутренним кодированием. Видеоданные воспроизводят с быстрой перемоткой вперед со всеми изображениями от изображений с внутренним кодированием до изображений, расположенных на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, которым предшествуют изображения с внутренним кодированием, которые были выбраны.

Носитель записи данных и структура данных в соответствии с настоящим изобретением включают в себя информацию положения множества изображений, перед которыми расположено изображение с внутренним кодированием, выбранное из информации об изображении с внутренним кодированием, которое содержится в видеоданных и становится точкой оперативного доступа, и информация, которая обозначает изображение, расположенное на заданном расстоянии от изображения с внутренним кодированием, записана в информации положения множества изображений.

Устройство записи данных и устройство обработки данных в соответствии с настоящим изобретением могут быть выполнены как независимые устройства или в виде блока, который выполняет обработку данных.

В соответствии с настоящим изобретением обработка данных может быть выполнена с большим удобством. В частности, производственные процессы могут быть эффективно выполнены с постоянным временем считывания данных.

Краткое описание чертежей

На фиг.1 представлена блок-схема, на которой изображен пример структуры аппаратных средств устройства воспроизведения диска в соответствии с вариантом выполнения настоящего изобретения; на фиг.2A и на фиг.2B представлена блок-схема, на которой изображен пример структуры группы программного модуля, которую выполняет центральный процессор 112; на фиг.3 показан график, изображающий зависимость между реально прошедшим временем и количеством импульсов синхронизации при частоте синхронизации 90 кГц; на фиг.4 показан график, изображающий зависимость между реально прошедшим временем и временем, подсчитанным на основе импульсов синхронизации тактового генератора, который обновляет время на основе выходных видеоданных видеодекодера; на фиг.5 представлена блок-схема, на которой изображен пример структуры модуля 215 управления буфером; на фиг.6 представлена блок-схема, изображающая пример структуры каталогов диска 101; на фиг.7 представлена блок-схема, изображающая синтаксис файла "PLAYLIST.DAT"; на фиг.8 представлена блок-схема, изображающая синтаксис PlayItem(); на фиг.9 представлена блок-схема, изображающая синтаксис PlayListMark(); на фиг.10 представлена блок-схема, изображающая взаимозависимость значения mark_type и типа Mark(); на фиг.11 представлена блок-схема, изображающая зависимость PlayList(), PlayItem(), клипов и потоков программы, сохраненных в файле потока клипа; на фиг.12 представлена блок-схема, изображающая синтаксис файла информации клипа Clip(); на фиг.13 представлена блок-схема, изображающая взаимозависимость stream_id, private_stream_id и элементарных потоков, идентифицированных таким образом; на фиг.14 представлена блок-схема, изображающая синтаксис StaticInfo(); на фиг.15 представлена блок-схема, изображающая синтаксис DynamicInfo(); на фиг.16 представлена блок-схема, изображающая синтаксис EP_map(); на фиг.17 представлена блок-схема, изображающая зависимость значения index_N_minus1 и 1stRef_Picture к 4thRef_Picture, показанных на фиг.16; на фиг.18A и на фиг.18B представлены схемы, изображающие синтаксис потока программы, пакета потока программы и заголовка пакета потока программы по системе MPEG-2; на фиг.19A и на фиг.19B представлены схемы, изображающие синтаксис пакета PES по системе MPEG-2; на фиг.20A, на фиг.20 B и на фиг.20C представлены схемы, изображающие синтаксис пакета PES по системе MPEG-2; на фиг.21A и на фиг.21B представлены схемы, изображающие синтаксис пакета PES по системе MPEG-2; на фиг.22A и на фиг.22B представлены схемы, изображающие взаимозависимость значения stream_id PES_packet() и атрибута элементарного потока системы MPEG-2; на фиг.23 представлена блок-схема, изображающая stream_id, который используется устройством воспроизведения диска; на

фиг.24 представлена блок-схема, изображающая синтаксис private_stream1_PES_payload(); на фиг.25 представлена блок-схема, изображающая взаимозависимость значения private_stream_id и атрибута элементарного потока, сохраненного в private_payload(); на фиг.26 представлена блок-схема, изображающая синтаксис private_stream2_PES_payload(); на фиг.27 представлена блок-схема, изображающая синтаксис au_information(); на фиг.28 представлена схема, изображающая pic_struct; на фиг.29 представлена блок-схема, изображающая конкретный пример файла "PLAYLIST.DAT"; на фиг.30A и на фиг.30B представлены схемы, изображающие конкретные примеры файлов информации клипа "00001.CLP", "00002.CLP" и "00003.CLP"; на фиг.31 представлена блок-схема, изображающая конкретный пример EP_map()файла информации клипа "00001.CLP"; на фиг.32 представлена блок-схема, изображающая конкретные примеры PlayList#0 и PlayList#1 для PlayListMark(); на фиг.33 показана блок-схема последовательности операций, описывающая обработку перед воспроизведением; на фиг.34 показана блок-схема последовательности операций, описывающая процесс воспроизведения; на фиг.35 показана блок-схема последовательности операций, описывающая зависимость между порядком декодирования и порядком вывода; на фиг.36 представлена схема, изображающая структуру видеодекодера; на фиг.37 представлена схема, изображающая структуру DPB, показанного на фиг.36; на фиг.38 показана блок-схема последовательности операций, описывающая процесс обновления времени; на фиг.39 представлена схема, изображающая процесс обновления времени, соответствующий значению pic_stnict; на фиг.40 показана блок-схема последовательности операций, описывающая процесс изменения PlayItem; на фиг.41 показана блок-схема последовательности операций, описывающая процесс отображения кода времени; на фиг.42 показана блок-схема последовательности операций, описывающая обработку изменения потока; на фиг.43 показана блок-схема последовательности операций, описывающая обработку, выполняемую модулем 215 управления буфером; на фиг.44 показана блок-схема последовательности операций, описывающая обработку, выполняемую модулем 215 управления буфером; на фиг.45 показана блок-схема последовательности операций, описывающая обработку считывания видеопотока; на фиг.46 показана блок-схема последовательности операций, описывающая обработку считывания потока звука; на фиг.47 показана блок-схема последовательности операций, описывающая обработку считывания потока субтитров; на фиг.48 показана блок-схема последовательности операций, описывающая обработку повторной синхронизации; на фиг.49 показана блок-схема последовательности операций, описывающая обработку метки; на фиг.50 представлена схема, изображающая определение соответствия при обработке метки; на фиг.51 представлена схема, изображающая синхронизацию playListEnd; на фиг.52 представлена схема, изображающая синхронизацию playListEnd; на фиг.53 представлена схема, изображающая интервалы событий; на фиг.54 показана блок-схема последовательности операций, описывающая обработку управления атрибутом выхода; на фиг.55 представлена блок-схема, изображающая конкретный пример набора pts_change_point и DynamicInfo() описанного в файле информации клипа "00003.CLP"; на фиг.56 показана блок-схема последовательности операций, описывающая обработку управления отображением субтитров; на фиг.57 показана блок-схема последовательности операций, описывающая обработку управления захватом и обработку фонового изображения/скринсейвера; на фиг.58 представлена блок-схема, изображающая другой синтаксис private_stream2_PES_payload(); на фиг.59 представлена блок-схема, изображающая еще один синтаксис au_information(); на фиг.60 представлена блок-схема, на которой изображен пример структуры аппаратных средств устройства записи диска; на фиг.61 представлена схема, изображающая функцию, выполняемую устройством записи диска, показанным на фиг.60; на фиг.62 показана блок-схема последовательности операций, описывающая обработку создания Ep_map; на фиг.63 показана блок-схема последовательности операций, описывающая обработку воспроизведения при быстрой перемотке вперед; на фиг.64 показана блок-схема последовательности операций,

описывающая обработку выбора точки входа; на фиг.65 представлена блок-схема, изображающая другую функцию, выполняемую устройством записи диска, показанным на фиг.60; на фиг.66 представлена блок-схема, изображающая другой синтаксис EP_map(); на фиг.67 представлена блок-схема, изображающая priority_flag показанного на фиг.65; и на фиг.68 показана блок-схема последовательности операций, описывающая обработку установки приоритета.

Подробное описание изобретения

Далее будут описаны варианты выполнения настоящего изобретения. Взаимозависимость между элементами, указанными в формуле изобретения и в вариантах выполнения, является следующей. Взаимозависимость обозначает, что конкретные примеры, которые поддерживают изобретение, представленные в формуле изобретения, описаны в вариантах выполнения. Таким образом, даже если некоторые конкретные примеры не описаны в виде элементов в формуле изобретения, это не подразумевает, что конкретные примеры не соответствуют элементам, представленным в формуле изобретения. И, наоборот, даже если конкретные примеры описаны в данном разделе как аналоги элементов, указанных в формуле изобретения, это не подразумевает, что эти конкретные примеры не соответствуют другим элементам, кроме элементов, указанных в формуле изобретения.

В данном разделе не подразумевается, что все аспекты изобретения, соответствующего конкретным примерам, описанным в вариантах выполнения настоящего изобретения, представлены в формуле изобретения. Другими словами, описание данного раздела соответствует конкретным примерам, описанным в вариантах выполнения настоящего изобретения. Таким образом, описание данного раздела не отрицает, что существуют аспекты настоящего изобретения, которые не указаны в формуле изобретения настоящей заявки на патент и, что могут быть оформлены заявки на выделенный патент, и/или дополнительные аспекты настоящего изобретения могут быть добавлены как изменения.

Другими словами, устройство записи данных в соответствии с настоящим изобретением включает в себя блок детектирования (например, модуль 421 мультиплексирования, показанный на фиг.61), который выбирает информацию об изображении с внутренним кодированием, которое содержится в видеоданных и становится точкой оперативного доступа, и детектирует информацию положения множества изображений, перед которыми расположено выбранное изображение с внутренним кодированием; блок обозначения (например, контроллер 425, показанный на фиг.61), который обозначает информацию положения изображения, расположенную на заданном расстоянии от изображения с внутренним кодированием, в информации положения множества изображений, детектируемых блоком детектирования; и блок записи (например, привод 409 диска, показанный на фиг.60), который записывает информацию положения для множества изображений, детектированных блоком детектирования, и информацию об изображении, обозначенном блоком обозначения на носителе записи данных вместе с видеоданными.

Способ записи данных в соответствии с настоящим изобретением включает в себя этапы выбора информации об изображении с внутренним кодированием, которое содержится в видеоданных и становится точкой оперативного доступа, и детектирования информации положения о множестве изображений, перед которыми расположено выбранное изображение с внутренним кодированием (например, на этапе S383 блок-схемы последовательности операций, показанной на фиг.62); обозначения информации положения для изображения, расположенного на заданном расстоянии от изображения с внутренним кодированием, в информации положения множества изображений, детектированных в этапе детектирования (например, на этапе S386 блок-схемы последовательности операций, показанной на фиг.62); и записи информации положения для множества изображений, детектированных на этапе детектирования, и информации об изображении, обозначенном на этапе обозначения, на носителе записи данных вместе с видеоданными (например, на этапе S386 блок-схемы последовательности операций, показанной на фиг.62).

Устройство обработки данных в соответствии с настоящим изобретением включает в себя блок считывания (например, привод 102 диска, показанный на фиг.1), который считывает информацию положения множества изображений с внутренним кодированием и множества изображений, перед которыми расположено множество изображений с внутренним кодированием, и информацию обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, перед которыми расположено множество изображений с внутренним кодированием, с носителя записи данных, вместе с видеоданными, причем множество изображений с внутренним кодированием содержатся в видеоданных и становятся точками оперативного доступа; блок выбора (например, модуль 212 управления проигрывателем, показанный на фиг.2A и фиг.2B), который выбирает изображения с внутренним кодированием и множество изображений, которым предшествуют изображения с внутренним кодированием в соответствии с информацией обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений с внутренним кодированием, которые становятся точками оперативного доступа видеоданных, считанных блоком считывания и записанных на носителе записи данных, и множество изображений, которым предшествуют изображения с внутренним кодированием; и блок воспроизведения (например, модуль 214 управления декодером, показанный на фиг.2A и фиг.2B), который выполняет воспроизведение с быстрой перемоткой вперед видеоданных со всеми изображениями, от изображений с внутренним кодированием, до изображений, расположенных на заданном расстоянии от изображений с внутренним кодированием во множестве изображений, которым предшествуют изображения с внутренним кодированием и выбранных блоком выбора.

Способ обработки данных в соответствии с настоящим изобретением включает в себя этапы считывания информации положения множества изображений с внутренним кодированием и множества изображений, перед которыми расположено множество изображений с внутренним кодированием, и информацию обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, перед которыми расположено множество изображений с внутренним кодированием, с носителя записи данных вместе с видеоданными, причем множество изображений с внутренним кодированием содержатся в видеоданных и становятся точками оперативного доступа (например, на этапе S391 блок-схемы последовательности операций, показанной на фиг.62); выбора изображений с внутренним кодированием и множества изображений, которым предшествуют изображения с внутренним кодированием, в соответствии с информацией обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений с внутренним кодированием, которые становятся точками оперативного доступа видеоданных, считанных на этапе считывания и записанных на носителе записи данных, и множества изображений, которым предшествуют изображения с внутренним кодированием (например, на этапе S395 блок-схемы последовательности операций, показанной на фиг.62); и воспроизведения с быстрой перемоткой вперед видеоданных со всеми изображениями, от изображений с внутренним кодированием до изображений, расположенных на заданном расстоянии от внутренних изображений во множестве изображений, которым предшествуют изображения с внутренним кодированием и которые были выбраны на этапе выбора (например, на этапе S396 блок-схемы последовательности операций, показанной на фиг.62).

Поскольку носитель записи программы и программа в соответствии с настоящим изобретением аналогичны способу записи данных, их описание здесь не приведено.

Далее, со ссылкой на прилагаемые чертежи будут описаны варианты выполнения настоящего изобретения.

[Структура аппаратных средств]

На фиг.1 показана блок-схема, представляющая пример структуры аппаратных средств

устройства воспроизведения диска в соответствии с вариантом выполнения настоящего изобретения.

Устройство воспроизведения диска, показанное на фиг.1, можно применять, например, в проигрывателе диска, игровом устройстве, автомобильной навигационной системе и т.д.

В устройстве воспроизведения диска, показанном на фиг.1, диск 101 представляет собой оптический диск, такой как DVD, магнитооптический диск, магнитный диск или тому подобное. Данные содержания, такие как видеоданные, данные звука и данные субтитров, а также дополнительные данные, необходимые для воспроизведения этих данных, записаны на диск 101.

В случае необходимости данные, записанные на диск 101, включают в себя программу, которая может выполняться компьютером. В соответствии с данным вариантом выполнения настоящего изобретения в качестве носителя записи используется диск 101, который представляет собой носитель записи в форме диска. В качестве альтернативы носитель записи может представлять собой, например, полупроводниковую память или носитель записи в форме ленты. Данные, которые считывают с диска в удаленном местоположении, могут быть переданы и введены в устройство воспроизведения диска, показанное на фиг.1. Другими словами, данные могут быть считаны с диска 101 другим устройством, подключенным к устройству воспроизведения диска. Данные, которые считывает другое устройство, могут быть приняты и обработаны устройством воспроизведения диска. Кроме того, устройство воспроизведения диска может принимать данные из сервера, или тому подобное, в котором сохранены данные, аналогичные данным, записанным на диске 101, через сеть, такую как Интернет, и обрабатывать принятые данные. Кроме того, устройство воспроизведения диска также может принимать данные из другого устройства, такого как сервер или тому подобное, записывать принятые данные на диск 101 и затем обрабатывать данные, записанные на диск 101.

Диск 101 может быть загружен в привод 102 диска и выгружен из него. Привод 102 диска имеет встроенный интерфейс (не показан). Привод 102 диска соединен с интерфейсом 114 привода через встроенный интерфейс. Привод 102 диска выполняет привод диска 101, считывает данные с диска 101 в соответствии с, например, командой считывания и передает эти данные в интерфейс 114 привода.

С шиной 111 соединены ЦПУ (центральное процессорное устройство) 112, запоминающее устройство 113, интерфейс 114 привода, входной интерфейс 115, видеодекодер 116, декодер 117 звука, интерфейс 118 вывода видеоизображения и интерфейс 119 вывода звука.

ЦПУ 112 и запоминающее устройство 113 составляют компьютерную систему. Другими словами, ЦПУ 112 выполняет группу программных модулей, которая представляет собой программу, сохраненную в запоминающем устройстве 113, для управления всем устройством воспроизведения диска и выполняет различные процессы, которые будут описаны ниже. В запоминающем устройстве 113 также сохранена группа программных модулей, которую выполняет ЦПУ 112. Кроме того, в запоминающем устройстве 113 временно сохранены данные, необходимые для работы ЦПУ 112. Запоминающее устройство 113 может состоять только из энергонезависимой памяти или из комбинации энергозависимой и энергонезависимой памяти. Когда устройство воспроизведения диска, показанное на фиг.1, содержит жесткий диск, на котором записана (установлена) группа программных модулей, которые выполняет ЦПУ 112, запоминающее устройство 113 может состоять только из энергонезависимой памяти.

Программа (группа программных модулей), которую выполняет ЦПУ 112, может быть заранее сохранена в запоминающем устройстве 113, используемом как носитель записи, который встроен в устройство воспроизведения диска.

В качестве альтернативы программа может быть временно или постоянно сохранена на диске 101 или на съемном носителе записи, таком как гибкий диск, CD-ROM (КД-ПЗУ, постоянное запоминающее устройство на компакт-диске, предназначенное только для считывания), MO (МО, магнитооптическом) диске, магнитном диске или на карте памяти.

Съемный носитель записи может быть предоставлен в виде так называемого упакованного программного обеспечения.

Программа может быть предварительно сохранена в запоминающем устройстве 113 или может быть установлена с описанного выше съемного носителя записи в устройство воспроизведения диска. В качестве альтернативы программа может быть передана по беспроводному каналу передачи данных с сайта загрузки в устройство воспроизведения диска через спутник, при выполнении цифровой широковещательной передачи со спутника или по кабелю может быть передана в устройство воспроизведения диска через локальную вычислительную сеть (ЛВС) или такую сеть, как Интернет. Устройство воспроизведения диска принимает программу через входной интерфейс 115 и устанавливает эту программу во встроенном запоминающем устройстве 113.

Программа может выполняться с помощью одного ЦПУ или может выполняться распределенно с использованием множества ЦПУ.

Интерфейс 114 привода управляет приводом 102 диска под управлением ЦПУ 112. Привод 102 диска передает данные, считываемые с диска 101, в ЦПУ 112, запоминающее устройство 113, видеодекодер 116, декодер 117 звука и т.д. через шину 111.

Входной интерфейс 115 принимает сигналы в соответствии с операциями пользователя, выполняемыми с клавишами (кнопками) и в соответствии с командами, передаваемыми через пульт дистанционного управления (не показан), и передает сигналы в ЦПУ 112 через шину 111. Входной интерфейс 115 также выполняет функции интерфейса передачи данных для модема (включая модем ADSL (АЦАЛ, асимметричная цифровая абонентская линия)), NIC (КСИ, карта сетевого интерфейса) или тому подобное.

Видеодекодер 116 декодирует кодированные видеоданные, которые были считаны с диска 101 с помощью привода 102 диска, передает их в видеодекодер 116 через интерфейс 114 привода и в шину 111 и передает декодированные видеоданные в ЦПУ 112 и в интерфейс 118 вывода видеоизображения через шину 111.

Декодер 117 звука декодирует кодированные данные звука, которые были считаны с диска 101, с помощью привода 102 и передает их в декодер 117 звука через интерфейс 114 привода и шину 111, передает декодированные звуковые данные в ЦПУ 112 и в интерфейс 119 вывода звука через шину 111.

Интерфейс 118 вывода видеоизображения выполняет заданную обработку видеоданных, передаваемых через шину 111, и выводит обработанные видеоданные на выходной вывод 120 видеоизображения. Интерфейс 119 вывода звука выполняет заданную обработку данных звука, переданных через шину 111, и выводит обработанные данные звука на выходной вывод 121 звука.

Выходной вывод 120 видеоизображения подключен к устройству вывода видеоизображения, такому как ЭЛТ (CRT, электронно-лучевая трубка) или жидкокристаллическая панель (не показана). Таким образом, видеоданные, выводимые с выходного вывода 120 видеоизображения, поступают в устройство вывода видеоизображения и отображаются с помощью него. Выходной вывод 121 звука соединен с устройствами вывода звука, такими как громкоговоритель и усилитель (не показаны). Таким образом, данные звука, выводимые с выходного вывода 121 звука, подают в устройства вывода звука и, таким образом, выводят.

Данные видеоизображения и данные звука могут быть переданы по беспроводному каналу или по кабелю, из устройства воспроизведения диска на вывод видеоизображения и в устройство вывода звука.

[Структура группы программных модулей]

На фиг.2А и фиг.2В показан пример структуры группы программных модулей, которые выполняет ЦПУ 112, показанное на фиг.1.

Группа программных модулей, которые выполняет ЦПУ 112, в основном, состоит из операционной системы (ОС, OS) 201 и программы 210 воспроизведения видеосодержания, используемой как прикладная программа.

[Операционная система 201]

Когда устройство воспроизведения диска включают, начинается работа операционной системы 201, которая выполняет заданную обработку, такую как исходные установки, и вызывает программу 210 воспроизведения видеосодержания, которая представляет собой прикладную программу.

Операционная система 201 предоставляет услуги для инфраструктуры, такие как услуга считывания файлов для программы 210 воспроизведения содержания. Другими словами, операционная система 201 обеспечивает услугу, которая обеспечивает работу привода 102 диска через интерфейс 114 привода, по запросу считывания файла, принимаемому из программы 210 воспроизведения видеосодержания, считывает данные с диска 101 и передает эти данные в программу 210 воспроизведения видеосодержания. Кроме того, операционная система 201 также интерпретирует файловую систему.

Операционная система 201 имеет функцию многозадачной обработки. Другими словами, операционная система 201 может одновременно обеспечивать работу множества программных модулей на основе разделения времени. Другими словами, хотя программа 210 воспроизведения видеосодержания состоит из нескольких программных модулей, они могут работать параллельно.

[Программа 210 воспроизведения видеосодержания]

Программа 210 воспроизведения видеосодержания состоит из модуля 211 управления сценариями, модуля 212 управления проигрывателем, модуля 213 подачи данных содержания, модуля 214 управления декодированием, модуля 215 управления буфером, модуля 216 управления видеодекодером, модуля 217 управления декодером звука, модуля 218 управления декодером субтитров, модуля 219 обработки графических изображений, модуля 220 вывода видеоизображения и модуля 221 вывода звука.

Программа 210 воспроизведения видеосодержания представляет собой программное обеспечение, которое выполняет ключевую роль при воспроизведении данных с диска 101. Когда диск 101 загружают (вставляют) в привод 102 диска, программа 210 воспроизведения видеосодержания проверяет, является ли диск 101 диском, на котором было записано содержание в заданном формате (который будет описан ниже). Программа 210 воспроизведения видеосодержания считывает файл сценария (который будет описан ниже) с диска 101, выполняет сценарий, считывает файл метаданных, необходимых для воспроизведения содержания с диска 101, и управляет воспроизведением содержания в соответствии с метаданными.

Далее будет описан программный модуль, который составляет программу 210 воспроизведения видеосодержания, показанную на фиг.2A и фиг.2B. На фиг.2A и фиг.2B, в общем, стрелкой в виде сплошной линии обозначены данные содержания, тогда как стрелкой из пунктирной линии обозначены данные управления.

[Модуль 211 управления сценарием]

Модуль 211 управления сценарием интерпретирует и выполняет программу сценария (сценарий), записанную на диске 101. Программа сценария может описывать операции, такие как "обеспечить работу модуля 219 обработки графических изображений для создания изображения, такого как меню, и отобразить его", "изменить отображение меню в соответствии с сигналом, переданным из интерфейса пользователя (ИП, UI), такого как пульт дистанционного управления (например, для перемещения курсора по меню)", и "управлять модулем 212 управления проигрывателем".

[Модуль 212 управления проигрывателем]

Модуль 212 управления проигрывателем обращается к метаданным и т.д., записанным на диске 101, для управления воспроизведением содержания, записанного на диске 101. Другими словами, модуль 212 управления проигрывателем анализирует PlayList() и Clip(), записанные на диске 101, и управляет модулем 213 подачи данных содержания, модулем 214 управления декодированием и модулем 215 управления буфером в соответствии с результатами анализа. Кроме того, модуль 212 управления проигрывателем выполняет управление изменением потока, которое изменяет поток, воспроизводимый в соответствии с командами, полученными из модуля 211 управления сценарием и из

входного интерфейса 115, как будет описано ниже. Кроме того, модуль 212 управления проигрывателем получает значение времени из модуля 214 управления декодированием, отображает это значение времени и выполняет обработку метки (Mark()) (которая будет описана ниже).

[Модуль 213 подачи данных содержания]

Модуль 213 подачи данных содержания запрашивает операционную систему 201 считать данные содержания, метаданные и т.д. с диска 101 под управлением модуля 212 управления проигрывателем или в соответствии с количеством данных, сохраненным в модуле 215 управления буфером.

Метаданные и т.д., которые операционная система 201 считала с диска 101 в соответствии с запросом, полученным из модуля 213 подачи данных содержания, передают в заданные модули. С другой стороны, данные содержания, которые операционная система 201 считала с диска 101 в соответствии с запросом, принятым с модуля 213 подачи данных содержания, подают в модуль 215 управления буфером.

[Модуль 214 управления декодированием]

Модуль 214 управления декодированием выполняет управление операциями модуля 216 управления видеодекодером, модуля 217 управления декодером звука и модуля 218 управления декодером субтитров под управлением модуля 212 управления воспроизведением. Модуль 214 управления декодированием имеет блок 214A подсчета времени, который подсчитывает время. Модуль 214 управления декодированием управляет синхронизацией вывода видеоданных, которые выводят под управлением модуля 216 управления видеодекодером, и вывода данных, которые синхронизированы с видеоданными. В этом случае вывод данных, которые должны быть синхронизированы с выводом видеоданных, представляют собой данные звука, которые выводят под управлением модуля 217 управления декодером звука.

Блок 214A подсчета времени может автономно подсчитывать время путем подсчета опорных импульсов тактовой частоты, передаваемых извне или из внутренних часов, которые синхронизированы с декодером или тому подобное.

Однако в декодере, построенном на основе программных средств, выполняемых путем управления модулями различного типа, показанными на фиг.2A и фиг.2B, если процесс подсчета времени выполняется с помощью программных средств, помимо этих процессов, нагрузка на обработку ЦПУ 112 увеличивается. Таким образом, блок 214A подсчета времени должен обновлять значение времени на основе выводимых видеоданных из декодера.

На фиг.3 показана взаимозависимость между временем независимых часов блока 214A подсчета времени и действительно прошедшим временем. На фиг.3, поскольку блок 214A подсчета времени подсчитывает импульсы тактовой частоты 90 кГц, время линейно увеличивается вправо от значения 0 времени. В момент времени 33,3 мс значение подсчета тактовых импульсов составляет 3003.

На фиг.4 показан пример времени, которое обновляет блок 214A подсчета времени на основе вывода видеоданных видео декодером. В случае, показанном на фиг.4, когда проходят 33 мс, выходное время тактовой частоты обновляют до значения 3003. Когда проходят 83,3 мс, выходное время обновляют до 7507. Когда проходит 116 мс, выходное время обновляют до значения 10510. В этом примере выходное время составляет 16,66 мс.

На фиг.3, хотя отсчет времени выполняется в ступенчатой форме с разрешающей способностью 1/90 кГц, течение времени представлено прямой линией для сравнения с фиг.4. В следующем описании предполагается, что блок 214A подсчета времени обновляет время на основе вывода видеоданных, как описано со ссылкой на фиг.4.

[Модуль 215 управления буфером]

Модуль 215 управления буфером имеет буфер 215A как часть пространства сохранения в запоминающем устройстве 113, показанном на фиг.1. Модуль 213 подачи данных содержания временно сохраняет данные содержания, считываемые с диска 101, в буфере 215A, в соответствии с запросом, полученным из операционной системы 201.

Кроме того, модуль 215 управления буфером передает данные, сохраненные в буфере 215A, в модуль 216 управления видеодекодером, модуль 217 управления декодером звука или в модуль 218 управления декодером субтитров, в соответствии с запросом, полученным из модуля 216 управления видеодекодером, модуля 217 управления декодером звука или модуля 218 управления декодером субтитров соответственно.

Другими словами, модуль 215 управления буфером имеет блок 233 функции считывания видеоданных, блок 234 функции считывания звука блок 235 функции считывания субтитров, которые будут описаны ниже со ссылкой на фиг.5. Блок 233 функции считывания видеоданных модуля 215 управления буфером обрабатывает запрос на данные, полученные из модуля 216 управления декодером видеоизображения, для передачи данных, сохраненных в буфере 215A в модуль 216 управления декодером видеоизображения. Аналогично, блок 234 функции считывания звука в модуле 215 управления буфером обрабатывает запрос, полученный из модуля 217 управления декодером звука, для передачи данных, сохраненных в буфере 215A, в модуль 217 управления декодером звука. Блок 233 функции считывания видеоданных в модуле 215 управления буфером обрабатывает запрос, полученный из модуля 218 управления декодером субтитров, для передачи данных, сохраненных в буфере 215A, в модуль 218 управления декодером субтитров.

[Модуль 216 управления видеодекодером]

Модуль 216 управления видеодекодером обеспечивает работу блока 233 функции считывания видеоданных (фиг.5) модуля 215 управления буфером для считывания кодированных видеоданных для одного модуля доступа видеоизображения в один момент времени из буфера 215A управления модуля 215 буфером и передает видеоданные в видеодекодер 116, показанный на фиг.1. Кроме того, модуль 216 управления видеодекодером управляет видеодекодером 116 для декодирования данных для одного модуля доступа видеоизображения в один момент времени. Кроме того, модуль 216 управления видеодекодером передает видеоданные, декодированные видеодекодером 116, в модуль 219 обработки графических изображений.

Модуль обработки видеоизображения представляет собой, например, одно изображение (один кадр или одно поле) видеоданных.

[Модуль 217 управления декодером звука]

Модуль 217 управления декодером звука обеспечивает работу блока 234 функции считывания звука (фиг.5) модуля 215 управления буфером для считывания кодированных данных звука для одного модуля доступа к звуку в один момент времени из буфера 215A модуля 215 управления буфером и передает кодированные данные звука в декодер 117 звука, показанный на фиг.1. Модуль 217 управления декодером звука управляет декодером 117 звука для декодирования кодированных данных звука, для модуля доступа звука в один момент времени. Кроме того, модуль 217 управления декодером звука передает данные звука, декодированные декодером 117 звука, в модуль 221 вывода звука.

Модуль доступа к звуку представляет собой заданное количество данных звука (например, некоторое количество данных, которые выводят синхронно с одним изображением). В соответствии с данным вариантом выполнения предполагается, что один модуль доступа к звуку имеет заданную фиксированную длину.

[Модуль 218 управления декодером субтитров]

Модуль 218 управления декодером субтитров обеспечивает работу блока 235 функции считывания субтитров (фиг.5) модуля 215 управления буфером для считывания кодированных данных субтитров для одного модуля доступа к субтитрам в один момент времени из буфера 215A модуля 215 управления буфером. Кроме того, модуль 218 управления декодером субтитров имеет программное обеспечение декодирования субтитров (не показано). Программное обеспечение декодирования субтитров декодирует данные, считываемые из буфера 215A. Модуль 218 управления декодером субтитров передает декодированные данные субтитров (данные изображения субтитров) в модуль 219 обработки графических изображений.

Модуль доступа к субтитрам представляет собой заданное количество данных субтитров (например, некоторое количество данных, которые выводят синхронно с одним изображением). В соответствии с данным вариантом выполнения предполагается, что размер одного элемента доступа к субтитрам описан в его начале.

[Модуль 219 обработки графических изображений]

Модуль 219 обработки графических изображений увеличивает или уменьшает данные субтитров, принятые из модуля 218 управления декодером субтитров, в соответствии с командой модуля 212 управления проигрывателем и накладывает увеличенные или уменьшенные данные субтитров на видеоданные, полученные из модуля 216 управления видеодекодером. Модуль 219 обработки графических изображений увеличивает или уменьшает кадр изображения видеоданных, на которые были наложены данные субтитров, в результате чего размер кадра добавленных видеоданных соответствует экрану устройства вывода видеоизображения, соединенного с выходным выводом 120 видеоизображения, показанным на фиг.1. Добавленные видеоданные выводят в модуль 220 вывода видеоданных.

Кроме того, модуль 219 обработки графических изображений генерирует меню, сообщение и т.д. в соответствии с командами модуля 211 управления сценарием и модуля 212 управления проигрывателем и накладывает это меню, сообщение и т.д. на выходные видеоданные.

Кроме того, модуль 219 обработки графических изображений преобразует соотношение размера видеоданных, выводимых в модуль 220 вывода видеоизображения, в соответствии с соотношением размеров устройства вывода видеоизображения, подключенного к выходному выводу 120 видеоизображения, показанному на фиг.1, и информацией, которая представляет соотношение размеров видеоданных, записанной на диске 101.

Другими словами, когда соотношение размеров устройства вывода видеоизображения составляет 16:9, если информация, которая представляет соотношение размеров видеоданных, представляет собой 4:3, модуль 219 обработки графических изображений выполняет обработку сжатия, которая уменьшает видеоданные, выводимые в модуль 220 вывода видеоизображения, в поперечном (горизонтальном) направлении, обеспечивает представление левого и правого краев видеоданных в виде черного цвета и выводит полученные в результате видеоданные. Когда соотношение размеров устройства вывода видеоизображения составляет 4:3, если информация, которая представляет соотношение размеров видеоданных, составляет 16:9, модуль 219 обработки графических изображений выполняет обработку сжатия видеоданных, которые выводят в модуль 220 вывода видеоизображения, в продольном (вертикальном) направлении, обеспечивает отображение верхнего и нижнего краев видеоданных в виде черного цвета и выводит полученные в результате видеоданные.

Когда соотношение размеров устройства вывода видеоизображения и соотношение размеров, которое представляет информация для видеоданных, совпадают, например 4:3 или 16:9, модуль 219 обработки графических изображений выводит видеоданные без сжатия в модуль 220 вывода видеоизображения.

Кроме того, модуль 219 обработки графических изображений выполняет захват видеоданных, которые обрабатывают в соответствии с запросом, полученным, например, из модуля 212 управления проигрывателем. Кроме того, модуль 219 обработки графических изображений сохраняет захваченные видеоданные или передает эти видеоданные в модуль 212 управления проигрывателем.

[Модуль 220 вывода видеоизображения]

Модуль 220 вывода видеоизображения исключительно занимает часть запоминающего устройства 113, показанного на фиг.1, как буфер 220A ППППО (FIFO, первым прибыл - первым обработан) и временно сохраняет видеоданные, принятые из модуля 219 обработки графических изображений. Кроме того, модуль 220 вывода видеоизображения часто считывает видеоданные из ППППО 220A и выводит эти видеоданные на выходной

вывод 120 видеоизображения (фиг.1).

[Модуль 221 вывода звука]

Модуль 221 вывода звука исключительно занимает часть запоминающего устройства 113, показанного на фиг.1, как буфер 221A ППО и временно сохраняет данные звука, принятые из модуля 217 управления декодером звука (декодер 117 звука). Кроме того, модуль 221 вывода звука часто считывает звуковые данные из буфера 221A и выводит эти данные звука на выходной вывод 121 звука (фиг.1).

Кроме того, когда данные звука, принятые из модуля 217 управления декодером звука, представляют собой данные звука в двойном (двуязычном) режиме, которые имеют левый и правый каналы данных "основного звука" и данных "вспомогательного звука", модуль 221 вывода звука выводит данные звука, принятые из модуля 217 управления декодером звука, на выходной вывод 121 звука, в соответствии с заранее установленным режимом вывода звука.

Другими словами, если "основной звук" был обозначен как режим вывода звука, модуль 221 вывода звука копирует левый канал данных звука, полученный из модуля 217 управления декодером звука, как правый канал данных звука, и выводит левый и правый каналы данных звука (данных "основного звука") на выходной вывод 121 звука. Если "вспомогательный звук" был обозначен как режим вывода звука, модуль 221 вывода звука копирует правый канал данных звука, полученный из модуля 217 управления декодером звука, как левый канал, и выводит левый и правый каналы (данные "вспомогательного звука") на выходной вывод 121 звука. Если оба "основной и вспомогательный звуки" были обозначены как режим вывода звука, модуль 221 вывода звука непосредственно выводит данные звука, полученные из модуля 217 управления декодером звука на выходной вывод 121 звука.

Если данные звука, принятые из модуля 217 управления декодером звука, представляют собой данные звука, записанные в стереофоническом режиме, модуль 221 вывода звука непосредственно выводит данные звука, полученные из модуля 217 управления декодером звука, на выходной вывод 121 звука, независимо от того, какой режим вывода звука был обозначен.

Пользователь может в интерактивном режиме обозначить режим вывода звука на экране меню, генерируемого программой 210 воспроизведения видеосодержания, с помощью пульта дистанционного управления.

[Структура модуля 215 управления буфером]

На фиг.5 показан пример структуры модуля 215 управления буфером, представленного на фиг.2A и фиг.2B.

Модуль 215 управления буфером исключительно использует часть запоминающего устройства 113, показанного на фиг.1, в качестве буфера 215A, и временно сохраняет данные, считанные с диска 101 в буфере 215A. Кроме того, модуль 215 управления буфером считывает данные из буфера 215A и передает эти данные в модуль 216 управления видеодекодером, модуль 217 управления декодером звука и в модуль 218 управления декодером субтитров, показанные на фиг.2A и фиг.2B.

Помимо буфера 215A, модуль 215 управления буфером имеет блок 231 сохранения указателя начала данных, и блок 232 сохранения указателя записи данных, которые представляют собой часть запоминающего устройства 113. Кроме того, модуль 215 управления буфером имеет блок 233 функции считывания видеоданных и 234 функции считывания звука, а также блок 235 функции считывания субтитров в качестве внутренних модулей.

Буфер 215A представляет собой, например, кольцевой буфер, который последовательно сохраняет данные, считываемые с диска 101. После того как буфер 215A считает данные, составляющие объем его сохранения, буфер 215A продолжит сохранять данные в режиме так называемой бесконечной петли, в результате чего буфер 215A перезаписывает новые данные поверх старых данных.

На блоке 231 сохранения указателя начала данных записан указатель начала данных,

который представляет положение (адрес) самых последних данных, которые не были считаны с буфера 215A, среди данных, сохраненных в буфере 215A.

На блоке 232 сохранения указателя записи данных записан указатель, который представляет положение (адрес) в буфере 215A самых новых данных, которые были считаны с диска 101.

Всякий раз, когда данные считывают с диска 101, их сохраняют в буфере 215A, при этом положение, которое представляет указатель записи данных, обновляют в направлении по часовой стрелке, показанном на фиг.5. Всякий раз, когда данные считывают из буфера 215A, положение, которое представляет указатель начала данных, обновляют в направлении по часовой стрелке, показанном на фиг.5. Таким образом, достоверные данные, сохраненные в буфере 215A, начинаются от положения, которое представляет указатель начала данных, до положения, которое представляет указатель записи данных в направлении по часовой стрелке, показанном на фиг.5.

Блок 233 функции считывания видеоданных считывает видеопоток (элементарный поток видеоданных) из буфера 215A в соответствии с запросом, полученным из модуля 216 управления видеодекодером, показанным на фиг.2A и фиг.2B, и передает этот видеопоток в модуль 216 управления видеодекодером. Блок 234 функции считывания звука считывает звуковой поток (элементарный поток данных звука) из буфера 215A в соответствии с запросом, полученным из модуля 217 управления декодером звука, показанным на фиг.2A и фиг.2B, и передает этот звуковой поток в модуль 217 управления декодером звуком. Аналогично, блок 235 функции считывания субтитров считывает поток субтитров (элементарный поток данных субтитров) из буфера 215A в соответствии с запросом, полученным из модуля 218 управления декодером субтитров, показанным на фиг.2A и на фиг.2B, и передает поток субтитров в модуль 218 управления декодером субтитров.

Другими словами, программный поток в соответствии, например, со стандартом MPEG 2 (Группа экспертов движущегося изображения) был записан на диск 101, при этом программный поток называется программным потоком MPEG2-system. В этом программном потоке, по меньшей мере, один элементарный поток из видеопотока, потока звука и потока субтитров был мультиплексирован на основе разделения времени. Блок 233 функции считывания видеоданных имеет функцию демультиплексирования для программного потока. Блок 233 функции считывания видеоданных демультиплексирует видеопоток из программного потока, сохраненного в буфере 215A, и считывает этот видеопоток.

Аналогично, блок 234 функции считывания звука имеет функцию демультиплексирования программного потока. Блок 234 функции считывания звука демультиплексирует поток звука из программного потока, сохраненного в буфере 215A, и считывает этот поток звука. Аналогично, блок 235 функции считывания субтитров имеет функцию демультиплексирования программного потока. Блок 235 функции считывания субтитров демультиплексирует поток субтитров из программного потока, сохраненного в буфере 215A, и считывает этот поток субтитров.

Блок 233 функции считывания видеоданных имеет блок 241 сохранения указателя считывания видеоданных, регистр 242 stream_id и регистр 243 au_information(), которые представляют собой часть запоминающего устройства 113, показанного на фиг.1.

На блоке 241 сохранения указателя считывания видеоданных сохранен указатель считывания видеоданных, который представляет положение (адрес) видеопотока в буфере 215A. Блок 233 функции считывания видеоданных считывает данные, как видеопоток, из положения указателя считывания видеоданных в буфере 215A. Регистр 242 stream_id сохраняет stream_id, который используется для анализа потока программы, сохраненного в буфере 215A и для идентификации видеопотока, который считывают из потока программы. В регистре 243 au_information() сохранена au_information(), которая представляет собой данные, необходимые для считывания видеопотока из буфера 215A.

Блок 234 функции считывания звука имеет блок 251 сохранения указателя считывания звука, регистр 252 stream_id и регистр 253 private_stream_id, которые представляют

собой часть запоминающего устройства 113, показанного на фиг.1.

На блоке 251 сохранения указателя считывания звука сохранен указатель считывания звука, который представляет положение (адрес) потока звука, сохраненного в буфере 215A. Блок 234 функции считывания звука считывает данные в виде потока звука из положения указателя считывания звука в буфере 215A. В регистре 252 stream_id и в регистре 253 private_stream_id сохранены stream_id и private_stream_id (которые будут описаны ниже) соответственно, используемые для анализа программного потока, сохраненного в буфере 215A, и для идентификации потока звука, считываемого из потока программы.

Блок 235 функции считывания субтитров имеет блок 261 сохранения флага функции считывания субтитров, блок 262 сохранения указателя считывания субтитров, регистр 263 stream_id и регистр 264 private_stream_id, которые составляют часть запоминающего устройства 113, показанного на фиг.1.

На блоке 261 сохранения флага функции считывания субтитров сохраняют флаг функции считывания субтитров. Когда флаг функции считывания субтитров, сохраненный на блоке 261 сохранения флага функции считывания субтитров, равен, например, "0", блок 235 функции считывания субтитров не работает. Когда флаг функции считывания субтитров, сохраненный на блоке 261 сохранения функции считывания субтитров равен, например, "1", блок 235 функции считывания субтитров работает.

На блоке 262 сохранения указателя считывания субтитров сохранен указатель считывания субтитров, который представляет положение (адрес) потока субтитров, сохраненного в буфере 215A. Блок 235 функции считывания субтитров считывает данные в виде потока субтитров из положения указателя считывания субтитров в буфере 215A. В регистре 263 stream_id и в регистре 264 private_stream_id сохранены stream_id и private_stream_id (которые будут описаны ниже) соответственно, используемые для анализа потока программы, сохраненного в буфере 215A, и для идентификации потока субтитров, который считывают из потока программы.

[Описание формата данных для данных, записанных на диске 101]

Далее будет описан формат данных для данных, записанных на диске 101.

На фиг.6 схематично представлена структура директорий диска 101.

Файловая система, используемая для диска 101, представляет собой, например, одну из систем, определенных МОС (ISO Международная организация по стандартизации) - 9660 и UDF (универсальный формат диска) (http://www.osta.org/specs/). Файлами данных, записанными на диске 101, управляют иерархически в структуре директории. Файловая система, которую можно использовать для диска 101, не ограничивается этими файловыми системами.

На фиг.6 представлена директория "VIDEO", расположенная в корневой директории, которая представляет собой основание файловой системы. В директории "VIDEO" имеется две директории: директория "CLIP" и директория "STREAM".

Помимо этих двух директорий, которые представляют собой директорию "CLIP" и директорию "STREAM", имеются два файла данных, которые представляют собой файл "SCRIPT.DAT" и файл "PLAYLIST.DAT" в директории "VIDEO".

Файл "SCRIPT.DAT" представляет собой файл сценария, который описывает программу сценария. Другими словами, файл "SCRIPT.DAT" описывает программу сценария, которая позволяет интерактивно воспроизводить данные, записанные на диске 101. Программа сценария, сохраненная в файле "SCRIPT.DAT", интерпретируется и выполняется модулем 211 управления сценарием, показанным на фиг.2A и фиг.2B.

В файле "PLAYLIST.DAT" сохранен, по меньшей мере, один список воспроизведения (PlayList(), который будет описан ниже со ссылкой на фиг.7). Список воспроизведения описывает процедуру воспроизведения содержания, такого как видеоданные, записанные на диске 101.

В директории "CLIP" имеется, по меньшей мере, один файл информации клипа. В директории "STREAM" имеется, по меньшей мере, один файл потока клипа. Другими

словами, на фиг.6 показаны три файла информации клипа "00001.CLP", "00002.CLP" и "00003.CLP" в директории "CLIP". В директории "STREAM" имеется три файла потока клипа "00001.PS", "00002.PS" и "00003.PS".

В файле потока клипа сохранен поток программы, который представляет собой, по меньшей мере, один поток из видеоданных, данных звука и данных субтитров, которые были сжаты и кодированы, мультиплексированных на основании разделения времени.

В файле информации клипа сохранены метаданные о потоке клипа, например его характеристике.

Другими словами, файл потока клипа и файл информации клипа скоррелированы во взаимозависимости 1 к 1. На Фиг.6 файл потока клипа называется в соответствии с правилом присвоения названий, состоящим из пятизначного числа + точка + "PS", когда название файла информации клипа в соответствии с правилом присвоения названий состоит из того же пятизначного числа, которое соответствует потоку клипа + точка + "CLP".

Таким образом, файл потока клипа и файл информации клипа могут быть идентифицированы по расширению названия файла (с правой стороны после точки). Кроме того, можно определить, является ли файл потока клипа и файл информации клипа скоррелированными по своим названиям файла, помимо их расширения (с левой стороны от точки).

Далее будут подробно описаны файлы, записанные на диске 101.

[PLAYLIST.DAT]

На фиг.7 показана внутренняя структура файла "PLAYLIST.DAT" в директории "VIDEO", показанной на фиг.6.

На фиг.7 файл "PLAYLIST.DAT" имеет поле Syntax", в котором описана структура данных файла "PLAYLIST.DAT"; и поле "No. of bits", которое описывает длину в битах каждого из входа данных в поле "Syntax"; а также поле "Mnemonic", в котором "bslbf" (первым идет бит слева от строки битов) и "uimsbf" (первым идет старший значимый целочисленный бит без знака) представляют, что вход данных в поле "Syntax" сдвинут от левого бита и что вход данных в поле "Syntax" представляет собой целое число без знака и сдвинут от старшего значащего бита. Такие условия применяют для других списков, аналогичных представленным на фиг.7.

Файл "PLAYLIST.DAT" начинается с name_length (8 битов) и name_string (255 байтов), которые описывают информацию, такую как название (имя файла).

Другими словами, name_length представляет размер name_string, который следует непосредственно после нее, в байтах. name_string представляет название (название файла) для файла "PLAYLIST.DAT".

Байты для name_length, начиная от name_string, используются как фактическое название. Когда значение name_length равно 10, 10 байт от начала name_string интерпретируют как фактическое название.

После name_string следует number_of_PlayLists (16 бит). number_of_PlayLists представляет количество PlayList(), перед которыми следуют name_string. После number_of_PlayLists следует PlayList(), представленная number_of_Play Lists.

PlayList() представляет собой список воспроизведения, который описывает процедуру воспроизведения файла потока клипа, записанного на диске 101. PlayList() имеет следующую внутреннюю структуру.

PlayList() начинается с PlayList_data_length (32 бита). PlayList_data_length представляет размер PlayList().

После PlayList_data_length следует reserved_for_word_aligninent (15 бит) и capture_enable_flag_PlayList (1 бит) последовательно. После reserved_for_word_alignment размером 15 бит следует capture_enable_flag_PlayList размером 1 бит для 16-битового совмещения положения в положении capture_enable_flag_PlayList, для размещения его в положении 16 битов. capture_enable_flag_PlayList представляют собой флаг размером 1 бит, который обозначает, разрешено использовать во вторую очередь видеоданные (видеоданные,

которые принадлежат PlayList()), соответствующие видеопотоку, воспроизводимому в соответствии с PlayList(), в устройстве воспроизведения диска, которое воспроизводит данные с диска 101. Когда значение capture_enable_flag_PlayList равно, например, 1 (0 или 1), это обозначает, что разрешено вторичное использование видеоданных, которые принадлежат PlayList(). Когда значение capture_enable_flag_PlayList равно, например, 0 (0 или 1), это обозначает, что не разрешено вторичное использование видеоданных, которые принадлежат PlayList().

На фиг.7 показан capture_enable_flag_PlayList, состоящий из одного бита. В качестве альтернативы, capture_enable_flag_PlayList может состоять из множества битов. В этом случае может быть разрешено пошаговое использование видеоданных, принадлежащих PlayList(), в качестве вторичного использования. Другими словами, capture_enable_flag_PlayList может состоять из двух битов. Когда значение capture_enable_flag_PlayList равно 00B (где B обозначает, что предыдущее число представляет собой двоичное число), видеоданные запрещено использовать вторично. Когда значение capture_enable_flag_PlayList равно 01B, видеоданные, которые уменьшены до размера 64×64 пикселей или меньше, разрешено использовать для вторичного использования. Когда значение capture_enable_flag_PlayList равно 10B, видеоданные разрешено использовать вторично, без какого-либо уменьшения размера.

Вторичное использование видеоданных может быть ограничено в отношении прикладных программ, а не по размерам. Другими словами, когда значение captwe_enable_flag_PlayList равно 01B, только программе 210 воспроизведения видеосодержания (фиг.2A и фиг.2B) может быть разрешено вторичное использование видеоданных. Когда значение capture_enable_flag_PlayList равно 10B, любой прикладной программе, включая видеопрограмму 210 воспроизведения содержания в устройстве воспроизведения диска, показанном на фиг.1, может быть разрешено вторичное использование видеоданных. В данном примере другая прикладная программа, кроме программы 210 воспроизведения видеосодержания в устройстве воспроизведения диска, показанном на фиг.1, представляет собой прикладную программу, которая отображает фоновое изображение или скринсейвер.

Когда capture_enable_flag_PlayList состоит из 2 битов, reserved_for_word_alignment, после которого он следует, таким образом, состоит из 14 битов и используется для выравнивания слова.

В качестве альтернативы при capture_enable_flag_PlayList видеоданные могут быть разрешены для вторичного использования за пределами устройства воспроизведения диска. В случае, когда с помощью capture_enable_flag_PlayList, разрешено вторичное использование видеоданных за пределами устройства воспроизведения диска, такие видеоданные записывают, например, на носитель записи, который может быть загружен в устройство воспроизведения диска или который может быть соединен с устройством воспроизведения диска, или они могут быть переданы в другое устройство через сеть, такую как Интернет. В этом случае информация, которая представляет количество раз, которое видеоданные могут быть записаны на носитель записи, или количество раз, которое видеоданные могут быть переданы, может быть добавлена к видеоданным.

После capture_enable_flag_PlayList последовательно следует PlayList_name_length (8 бит) и PlayList_name_string (255 байт). PlayList_name_length представляет размер PlayList_name_string в байтах. PlayList_name_string представляет название PlayList().

После PlayList_name_string следует namber_of_PlayItems (16 битов). number_of_Play Items обозначает количество PlayItem().

После number_pf_PlayItems следует PlayItem(), представленный number_of_PlayItems.

Один PlayList() может описывать процедуру воспроизведения содержания в виде модуля PlayItem().

Коды идентификации (ИД, ID), которые являются уникальными в PlayList(), добавлены к PlayItem(), представленным number_pf_PlayItems. Другими словами, первый PlayItem() в PlayList() идентифицирован номером 0. Другие PlayItem() последовательно

идентифицированы числами 1, 2 ... и т.д.

После PlayItem(), который представлен number_of_PlayItems, следует один PlayListMark(). PlayListMark() представляет собой набор Mark() в виде меток на временной оси при воспроизведении в соответствии с PlayList(). PlayListMark() будет подробно описан ниже со ссылкой на фиг.9.

[Описание PlayItem()]

На фиг.8 показана внутренняя структура PlayItem(), содержащегося в PlayList(), показанном на фиг.7.

PlayItem() начинается с длины (16 битов). Здесь поле length представляет собой размер PlayItem(), включая размер этого поля length.

После поля length последовательно следует Clip_Information_file_name_length (16 бит) и Clip_Information_file_name (с переменной длиной). Clip_Information_file_name_length представляет размер Clip_Information_file_name в байтах. Clip_Information_file_name представляет название файла для файла информации клипа (файл, имеющий расширение CLP, показанный на фиг.6), которое соответствует файлу потока клипа (файл, имеющий расширение PS, показанный на фиг.6), воспроизводимому в соответствии с PlayItem(). В соответствии с указанными выше правилами для файлов потока клипа и файлов информации клипа название файла в файле информации клипа, воспроизводимом в соответствии с PlayItem(), может быть распознано Clip_Information_file_name, и файл потока клипа может быть идентифицирован.

После Clip_Information_file_name последовательно следуют IN_time (32 бита) и OUT_time (32 бита).

IN_time и OUT_time представляют информацию времени и представляют положение начала воспроизведения и положение окончания воспроизведения файла потока клипа, идентифицированного с помощью Clip_Information_file_name.

IN_time может обозначать среднее положение (включая начало) файла потока клипа, в качестве положения начала воспроизведения. OUT_time может обозначать среднее положение (включая конец) файла потока клипа в качестве положения окончания воспроизведения.

PlayItem() воспроизводит содержание от IN_time до OUT_time файла потока клипа, идентифицированного Clip_hubrmation_file_name. Содержание, воспроизводимое в соответствии с PlayItem(), иногда называется клипом.

[Описание PlayListMark()]

На фиг.9 показана внутренняя структура PlayListMark(), содержащегося в PlayList(), показанном на фиг.7.

Как описано выше, PlayListMark() представляет собой набор меток Mark(), которые представляют собой метки на оси времени воспроизведения в соответствии с PlayList() (фиг.7). Количество Mark() равно 0 или больше. Одна метка Mark() имеет, по меньшей мере, информацию времени, которая представляет один момент времени на временной оси воспроизведения, выполняемого в соответствии с PlayList(), информацию типа, которая представляет тип Mark(), и информацию аргумента, представляющую аргумент события, когда информация типа представляет тип события, которое происходит.

Другими словами, PlayListMark() начинается с поля length (32 бита). Поле length представляет размер PlayListMark(), включая размер длины.

После поля length следует number_of_PlayList_marks (16 битов). number_of_PlayList_marks представляет количество Mark(), перед которыми следует number_of_PlayList_marks. После number_of_PlayList_marks следует Mark(), представленное полем number_of_PlayList_marks.

Mark() начинается с mark_type (8 битов). mark_type представляет собой информацию предыдущего типа и представляет тип Mark(), которому принадлежит mark_type.

В соответствии с этим вариантом выполнения Mark() имеет три типа, такие как, например, раздел, индекс и события.

Когда тип Mark() представляет собой раздел (иногда называется меткой раздела), она

представляет собой метку начального положения раздела, который представляет собой модуль поиска в качестве раздела PlayList(). Когда тип Метки() представляет собой индекс (иногда называется индексной меткой), она является меткой начального положения индекса, который представляет собой модуль подразделения раздела. Когда тип Mark() представляет собой событие (иногда называется меткой события), Mark() представляет собой метку положения, в котором происходит событие, во время воспроизведения содержания в соответствии с PlayList(). Модуль 211 управления сценарием информирует о том, что произошло событие в соответствии с меткой события.

На фиг.10 показана взаимозависимость между значением mark_type и типом Mark(). На фиг.10 mark_type метки раздела равен 1; mark_type метки индекса равен 2; и mark_type метки события равен 3. На фиг.10 другие значения, представленные 8 битами mark_type, а именно, 0 и 4-255, зарезервированы для расширения в будущем.

Как снова показано на фиг.9, после mark_type следует mark_name_length (8 битов). Mark() заканчивается mark_name_string. mark_name_length и mark_name_string используются для описания названия Mark(). mark_name_length представляет действительный размер mark_name_string. mark_name_string представляет название Mark(). Таким образом, байты в поле mark_name_length с начала mark_name_string представляют действительное название Mark().

После mark_name_length следуют четыре элемента ref_to_PlayItem_id (16 битов), mark_time_stamp (32 бита), entry_ES_stream_id (8 битов) и entry_ES_private_stream_id (8 битов), которые коррелируют Mark(), определенную в PlayList(), с файлом потока клипа.

ref_to_PlayItem_id описывает ИД как последовательный номер, установленный для PlayItem(), которому принадлежит Mark(). ref_to_PlayItem_id идентифицирует PlayItem() (фиг.8), которому принадлежит Mark(). Таким образом, как описано на фиг.8, файл информации клипа и файл потока клипа идентифицированы.

mark_time_stamp представляет положение, которое представляет Mark() в файле потока клипа, идентифицированном по ref_to_PlayItem_id.

На фиг.11 представлена взаимозависимость PlayList(), PlayItem(), клипов и программных потоков, сохраненных в файле потока клипа.

На фиг.11 PlayList() состоит из трех PlayItem(), которые последовательно пронумерованы как ИД#0, ИД#1 и ИД#2. В следующем описании PlayItem(), пронумерованный как ИД#i, представлен PlayItem#i.

На фиг.11 клипы, как содержание, воспроизводимое в соответствии с PlayItem#0, PlayItem#1 и PlayItem#2, представлены клипом A, клипом B и клипом C соответственно.

Сам модуль клипа располагается от IN_time до OUT_time программного потока, сохраненного в файле потока клипа, идентифицированном Clip_Information_file_name в PlayItem(), показанном на фиг.8. На фиг.11 программные потоки, представленные как клип A, клип B и клип C, представлены как программный поток A, программный поток B и программный поток C соответственно.

На фиг.11 в Mark() в качестве метки времени t0 на временной оси воспроизведения в соответствии с PlayList(), ref_to_PlayItem_id и mark_time_stamp описаны следующим образом.

На фиг.11, поскольку время t0 представляет собой момент времени, в который воспроизводят PlayItem#1, ref_to_PlayItem_id описывает 1 как ИД для PlayItem#1. Поскольку в момент времени t0 воспроизводится программный поток B как объект клипа B, mark_time_stamp описывает время файла потока клипа, в котором сохранен программный поток B, соответствующий моменту времени t0.

И снова, как показано на фиг.9, когда Mark() скоррелирован с конкретным элементарным потоком, entry_ES_stream_id и entry_ES_private_stream_id используются для идентификации элементарного потока. Другими словами, entry_ES_stream_id описывает stream_id элементарного потока, который скоррелирован с Mark(). С другой стороны, когда это требуется, entry_ES_private_stream_id описывает private_stream_id элементарного потока, скоррелированного с Mark().

Когда воспроизводят video stream#1 клипа, в котором были мультиплексированы video stream#1 и video stream#2, и когда время раздела требуется изменить, во время воспроизведения video stream#2, stream_id и private_stream_id video stream#1 описаны в entry_ES_stream_id и entry_ES_private_stream_id Mark(), в момент времени метки раздела, в то время как воспроизводится video stream#2. Кроме того, stream_id и private_stream_id для video stream#2 описаны в entry_ES_stream_id и entry_ES_private_stream_id Mark() в момент времени метки раздела, во время воспроизведения video stream#2.

entry_ES_stream_id и entry_ES_private_stream_id Mark(), которая не скоррелирована с конкретным элементарным потоком, равны, например, 0.

После entry_ES_private_stream_id следует mark_data (32 бита). Когда Mark() представляет собой метку события, mark_data представляет собой информацию аргумента, используемую в качестве аргумента события, которое происходит по метке события. Когда Mark() представляет собой метку раздела или индексную метку, mark_data могут использоваться как номер раздела или номер индекса, которые представляет метка раздела или метка индекса.

[Описание Clip()]

Далее будет описана внутренняя структура файла информации клипа, имеющего расширение CLP и сохраненного в директории "CLIP", показанной на фиг.6.

На фиг.6 показаны три файла информации клипа "00001.CLP", "00002.CLP" и "00003.CLP" в директории "CLIP". Эти файлы информации клипа содержат метаданные, которые представляют характеристики файлов потока клипа "00001.PS", "00002.PS" и "00003.PS", сохраненные в директории "STREAM".

На фиг.12 показана внутренняя структура файла информации клипа Clip().

Файл Clip() информации клипа начинается с presentation_start_time и presentation_end_time (32 бита каждый). presentation_start_time и presentation_end_time представляют время начала и время окончания файла потока клипа, соответствующего файлу Clip() информации клипа. Время файла потока клипа описано как множество импульсов с частотой следования 90 кГц, используемых как время в системе MPEG2.

После presentation_end_time следует reserved_for_word_alignment (7 бит) и capture_enable_flag_Clip (1 бит). reserved_for_word_alignment размером 7 бит используется для выравнивания слова. capture_enable_flag_Clip представляет собой флаг, который обозначает, разрешено ли вторичное использование видеоданных, так же, как и capture_enable_flag_PlayList, показанный на фиг.7.

Однако capture_enable_flag_PlayList, показанный на фиг.7, обозначает, разрешено ли вторичное использование видеоданных, которые принадлежат PlayList(), соответствующему видеопотоку, воспроизводимому в соответствии с PlayList(). В отличие от этого capture_enable_flag_Clip, показанный на фиг.12, обозначает, разрешено ли вторичное использование видеоданных, соответствующих элементарному потоку видеоданных, сохраненному в файле потока клипа, соответствующему файлу Clip() информации клипа. Таким образом, capture_enable_flag_PlayList, показанный на фиг.7, отличается от capture_enable_flag_Clip, показанного на фиг.12, по количеству модулей видеоданных, которые разрешается использовать вторично.

Так же, как и capture_enable_flag_PlayList, описанный со ссылкой на фиг.7, capture_enable_flag_Clip, описанный со ссылкой на фиг.12, может состоять из множества битов, а не из одного бита.

После capture_enable_flag_Clip следует number_of_streams (8 бит). number_of_streams описывает количество StreamInfo(). Таким образом, после number_of_streams следует StreamInfo(), представленный number_of_streams.

StreamInfo() начинается с поля length (16 битов), length представляет размер StreamInfo(), включающий в себя размер в длину. После поля length следует streamed (8 битов) и private_stream_id (8 битов). stream_id и private_stream_id идентифицируют

элементарный поток, который скоррелирован с StreamInfo().

На фиг.13 показана взаимозависимость stream_id, private_stream_id и элементарные потоки, идентифицируемые с их помощью.

stream_id представляет собой то же поле, что определено в соответствии со стандартом MPEG2-System. Стандарт MPEG2-System определяет значение поля stream_id для каждого атрибута элементарного потока. Таким образом, атрибут элементарного потока, определенный в стандарте MPEG2-System, может быть идентифицирован только по stream_id.

Данный вариант выполнения может работать с атрибутами элементарных потоков, которые не определены в стандарте MPEG2-System. private_stream_id представляет собой информацию, которая идентифицирует атрибут элементарного потока, который не определен в стандарте MPEG2-System.

На фиг.13 показана взаимозависимость stream_id и private_stream_id элементарных потоков, имеющих четыре атрибута, которые представляют собой элементарный видеопоток, кодированный в соответствии с системой кодирования, определенной в MPEG, элементарный поток звука, кодированный в соответствии с системой ATRAC (АКАП, акустическое кодирование с адаптивным преобразованием) (ниже иногда называется звуковым потоком ATRAC), элементарным потоком звука, кодированным в соответствии с системой LPCM (ЛИКМ, линейная импульсно-кодовая модуляция) (ниже иногда называется звуковым потоком LPCM), и элементарным потоком субтитров (ниже иногда называется потоком субтитров).

Стандарт MPEG2-System определяет, что элементарный поток видеоданных, кодированный в соответствии с системой кодирования, определенной в MPEG, мультиплексирован с использованием значения в диапазоне от 0xE0 до 0xEF (где 0x обозначает, что строка знаков, следующая после него, представлена с использованием обозначения в шестнадцатеричной системе). Таким образом, 16 элементарных потоков видеоданных, кодированных в соответствии с системой кодирования, определенной в MPEG и идентифицированной stream_id в диапазоне от 0xE0 до 0xEF, могут быть мультиплексированы с программным потоком.

Поскольку элементарные потоки видеоданных, кодированные в соответствии с системой кодирования, определенной в MPEG, могут быть идентифицированы по stream_id в диапазоне от 0xE0 до 0xEF, private_stream_id не требуется.

С другой стороны, в MPEG2-System stream_id не определен для звукового потока ATRAC, звукового потока LPCM и потока субтитров.

Таким образом, в соответствии с данным вариантом выполнения для элементарных потоков, stream_id которых не определен в MPEG2-System, используется 0xBD, который представляет собой значение, представляющее атрибут private_stream_1 в MPEG2-System. Кроме того, как показано на фиг.13, эти элементарные потоки идентифицированы private_stream_id.

Другими словами, звуковой поток ATRAC идентифицирован по private_stream_id в диапазоне от 0x00 до 0x0F. Таким образом, 16 звуковых потоков ATRAC могут быть мультиплексированы с программным потоком. Звуковой поток LPCM идентифицирован private_stream_id в диапазоне от 0x10 до 0x1F. Таким образом, 16 звуковых потоков LPCM могут быть мультиплексированы с программным потоком. Поток субтитров идентифицирован private_stream_id в диапазоне от 0x80 до 0x9F. Таким образом, 32 потока субтитров могут быть мультиплексированы с программным потоком.

stream_id и private_stream_id будут подробно описаны ниже.

Возвращаясь к фиг.12, после private_stream_id последовательно следует StaticInfo() и reserved_for_word_alignment (8 битов). StaticInfo() описывает информацию, которая не изменяется, в то время как воспроизводится элементарный поток, идентифицированный stream_id и private_stream_id, описанный в StreamInfo(), включающий StaticInfo(). StaticInfo() будет описан ниже со ссылкой на фиг.14.

reserved_for_word_alignment используется для выравнивания слова.

После reserved_for_word_alignment следует number_of_DynamicInfo (8 битов). number_of_DynamicInfo представляет количество наборов pts_change_point (32 бита каждый) и DynamicInfo(), которые следуют после number_of_DynamicInfo.

Таким образом, после number_of_DynamicInfo следуют наборы pts_change_point и DynamicInfo(), представленные number_of_DynamicInfo.

pts_change_point представляет время, в которое информация DynamicInfo(), в паре с pts_change_point становится действительной. pts_change_point, которое представляет время начала элементарного потока, равно presentation_start_time, описанному в начале файла Clip() информации клипа, который соответствует файлу потока клипа, в котором сохранен элементарный поток.

DynamicInfo() описывает так называемую динамическую информацию, которая изменяется, в то время как воспроизводится элементарный поток, идентифицированный stream_id и private_stream_id. Информация, описанная в DynamicInfo(), становится действительной во время воспроизведения, которое представлено pts_change_point, в паре с DynamicInfo(). DynamicInfo() будет описано ниже со ссылкой на фиг.15.

После наборов pts_change_point и DynamicInfo(), представленных number_of_DynamicInfo, следует EP_map(). EP_map() будет описан ниже со ссылкой на фиг.16.

[Описание StaticInfo()]

Далее, со ссылкой на фиг.14, будет подробно описан StaticInfo(), представленное на фиг.12.

На фиг.14 представлен синтаксис StaticInfo().

Содержание StaticInfo() изменяется в зависимости от атрибута соответствующего элементарного потока. Атрибут элементарного потока, соответствующий StaticInfo(), определен по stream_id и private_stream_id, содержащимся в StreamInfo(), показанным на фиг.12, включая StaticInfo().

Когда элементарный поток, соответствующий StaticInfo(), представляет собой видеопоток (stream == VIDEO), StaticInfo() состоит из picture_size (4 бита), frame_rate (4 бита), cc_flag (1 бит) и reserved_for_word_alignment для выравнивания слова.

picture_size представляет размер изображения, отображаемого видеоданными, соответствующими видеопотоку. frame_rate представляет частоту следования кадров видеоданных, соответствующих видеопотоку. cc_flag обозначает, содержит ли видеопоток данные субтитров. Когда видеопоток содержит данные субтитров, cc_flag установлен равным 1. Когда видеопоток не содержит данные субтитров, cc_flag установлен равным 0.

Когда элементарный поток, соответствующий Staticmfo(), представляет собой поток звука (stream == AUDIO), StaticInfo() состоит из audio_language_code (16 битов), channel_configuration (8 битов), lfe_existence (1 бит), sampling_frequency (4 бита) и reserved_for_word_alignment для выравнивания слова.

audio_language_code описывает код, который представляет язык звуковых данных, содержащихся в звуковом потоке. channel_configuration представляет атрибут, такой как монофонические (моно), стереофонические, многоканальные и т.д. данные звука, содержащиеся в потоке звука. lfe_existence обозначает, содержит ли поток звука канал эффекта низкой частоты. Когда поток звука содержит канал эффекта низкой частоты, lfe_existence установлен равным 1. Когда поток звука не содержит канал эффекта низкой частоты, lfe_existence равен 0. samplmg_frequency представляет собой информацию, которая представляет частоту выборки данных звука, содержащихся в потоке звука.

Когда элементарный поток, соответствующий StaticInfo(), представляет собой поток субтитров (stream == SUBTITLE), StaticInfo() состоит из subtitle_language_code (16 битов), configurable_flag (1 бит) и reserved_for_word_alignment для выравнивания слова.

subtitle_language_code описывает код, который представляет язык данных субтитров, содержащихся в потоке субтитров. configurable_flag представляет собой информацию, которая обозначает, разрешено ли изменение режима визуального отображения данных субтитров из режима визуального отображения, принятого по умолчанию. Когда разрешено

изменение режима визуального отображения, configurable_flag установлен равным 1. Когда изменение режима визуального отображения не разрешено, configurable_flag установлен равным 0. Режим визуального отображения данных субтитров включает в себя размер отображения данных субтитров, положения отображения, цвет отображения, структуру отображения (например, мигание), направление отображения (вертикальное или горизонтальное) и т.д.

[Описание DynamicInfo()]

Далее, со ссылкой на фиг.15 будет подробно описан DynamicInfo(), показанный на фиг.12.

На фиг.15 показан синтаксис DynamicInfo().

DynamicInfo() начинается с reserved_for_word_alignment (8 битов) для выравнивания слова. Элементы, которым предшествует reserved_for_word_alignment, зависят от атрибута элементарного потока, соответствующего DynamicInfo(). Атрибут элементарного потока, соответствующего DynamicInfo(), определен stream_ID и private_stream_id, содержащимся в StreamInfo(), показанным на фиг.12, который включает в себя DynamicInfo(), так же, как и StaticInfo(), описанный со ссылкой на фиг.14.

Как показано на фиг.12, DynamicInfo() описывает динамическую информацию, которая изменяется при воспроизведении элементарного потока. Динамическая информация не является специфической. Однако в варианте выполнения, показанном на фиг.15, данные элементарного потока, соответствующего Dynamichifo(), а именно выходной атрибут данных, которые выводят при обработке элементарного потока, описан в DynamicInfo().

В частности, когда элементарный поток, соответствующий DynamicInfo(), представляет собой видеопоток (stream == VIDEO), DynamicInfo() состоит из display_aspect_ratio (4 бита) и reserved_for_word_alignment для выравнивания слова. display_aspect_ratio описывает режим вывода потока видеоданных, например соотношение размеров видеоданных. Другими словами, display_aspect_ratio описывает информацию, которая представляет соотношение размеров 16:9 или 4:3. DynamicInfo() потока видеоданных может описывать, например, размер изображения видеоданных (X пикселей $\times$ Y пикселей), а также соотношение размеров.

Когда элементарный поток, соответствующий DynamicInfo(), представляет собой поток звука (stream == AUDIO), DynamicInfo() состоит из channel_assignment (4 бита) и reserved_for_word_alignment для выравнивания слова. Когда поток звука содержит два канала данных звука, channel_assignment описывает режим вывода по двум каналам. Другими словами, channel_assignment описывает информацию, которая представляет назначение стерео каналов или вывод двойного звука (на двух языках).

Когда элементарный поток, соответствующий DynamicInfo(), представляет собой поток субтитров (stream == SUBTITLE), DynamicInfo() состоит из reserved_for_word_alignment для выравнивания слова. Другими словами, в соответствии с вариантом выполнения, показанным на фиг.15, выходной атрибут в качестве динамической информации не определен для потока субтитров.

[Описание EP_map()]

Далее, со ссылкой на фиг.16 будет подробно описано EP_map(), представленное на фиг.12.

На фиг.16 показан синтаксис EP_map().

EP_map() описывает информацию точки, с которой может начинаться декодирование (точка входа), от которой может быть декодирован каждый из элементарных потоков, мультиплексированных с программным потоком, сохраненным в файле потока клипа, соответствующем файлу Clip() информации клипа (показан на фиг.12), который включает в себя EP_map().

Точка, с которой может начаться декодирование потока, имеющего фиксированную скорость, может быть получена с использованием расчетов. Однако для потока, размер которого изменяется в каждом из модулей доступа видеоданных, такого как поток видеоданных, кодированный в соответствии со стандартом MPEG, точка возможного

начала декодирования не может быть получена с помощью расчетов. Точка возможного начала декодирования не может быть получена до тех пор, пока поток не будет проанализирован. Для обеспечения оперативного доступа к данным необходимо быстро распознавать точку возможного начала декодирования. При использовании EP_map() точка возможного начала декодирования не может быть быстро распознана.

В соответствии со стандартом MPEG2-video начало изображения с внутренним кодированием, включающее Sequence_header() и т.д., представляет собой точку возможного начала декодирования.

EP_map() начинается с reserved_for_word_alignment (8 бит) для выравнивания слова. После reserved_for_word_augnment следует number_oi_stream_id_entries (8 бит). number_of_stream_id_entries представляет количество элементарных потоков, которые описывают информацию точки возможного начала декодирования в EP_map().

После number_of_stream_id_entries следуют наборы информации, которые идентифицируют элементарный поток и информацию точки возможного начала декодирования элементарного потока. Они повторяются определенное количество раз, представленное number_of_stream_id_entries.

Другими словами, после number_of_stream_id_entries следует stream_id (8 бит) и private_stream_id (8 бит) как информация, которая идентифицирует элементарный поток. После private_stream_id следует number_of_EP_entries (32 бита). number_of_EP_entries представляет количество точек возможного начала декодирования, идентифицированное stream_id и private_stream_id, после которых следует number_of_EP_entries.

После number_of_EP_entries следует информация точек возможного начала декодирования элементарного потока, идентифицированного stream_id и private_stream_id. Информация повторяется определенное количество раз, которое представлено number_of_EP_entries.

Другими словами, когда текущий элементарный поток представляет собой видеопоток, установлены index_N_minus1 и N-th_Ref_picture_copy. После этого размещены PTS_EP_start (32 бита) и RPN_EP_start (32 бита) как информация точки возможного начала декодирования. Эта последовательность повторяется. index_N_minus1 и N-th_Ref_picture_copy будут описаны ниже.

Когда текущий элементарный поток представляет собой другой поток, кроме потока видеоданных, после reserved_for_future_use величиной 16 битов следует PTS_EP_start (32 бита) и RPN_EP_start (32 бита) как информация точки разрешенного начала декодирования. Эта последовательность повторяется.

Один из 1stRef_picture, 2ndRef_picture, 3ndRef_picture и 4thRef_picture, записанных в private_stream_2, представленном RPN_EP_start, который будет описан ниже, копируют в N-th_Ref_picture_copy. Кроме того, информация, обозначающая, какое поле было копировано, записывается в index_N_minus1 со значением, показанным на фиг.17. Другими словами, как показано на фиг.17, когда lstRef_picture копируется в index_N_minus1 записывают 0. Когда копируют 2ndRef_picture, в index_N_minus1 записывают 1. Когда копируют 3ndRef_picture, в index_N_minus1 записывается 2. Когда копируется 4thRef_picture, в index_N_minus1 записывается 4.

PTS_EP_start как один элемент информации о точках возможного начала декодирования представляет время (время воспроизведения) возможного начала декодирования в файле потока клипа, в котором сохранен программный поток, мультиплексированный с элементарным потоком, идентифицированным stream_id и private_stream_id.

RPN_EP_start, который представляет собой другой элемент информации о точках возможного начала декодирования, описывает положение точки возможного начала декодирования в файле потока клипа, в котором сохранен программный поток, мультиплексированный с элементарным потоком, идентифицированным stream_id и private_stream_id как значение, такое же, как количество pack() программного потока. В соответствии с этим вариантом выполнения размер pack() составляет 2048 байт и является фиксированным. Кроме того, в соответствии с данным вариантом выполнения

один сектор диска 101 (фиг.1) составляет 2048 байт.

Непосредственно перед точкой возможного начала декодирования (точкой входа) потока видеоданных расположен пакет private_stream_2 (PES_packet() в качестве атрибута private_stream_2). В пакете private_stream_2 сохранена информация, используемая для декодирования потока видеоданных, сохраненного между двумя соседними пакетами private_stream_2. Таким образом, для потока видеоданных RPN_EP_start как информация точки возможного начала декодирования описывает начальное положение пакета private_stream_2, после которого непосредственно следует реальная точка возможного начала декодирования.

Наборы PTS_EP_start и RPN_EP_start как информацию о точках возможного начала декодирования заранее сортируют в порядке возрастания каждого элементарного потока, идентифицированного stream_id и private_stream_id в EP_map(). Таким образом, в наборах PTS_EP_start и RPN_EP_start как в информации о точках возможного начала декодирования можно выполнять двоичный поиск.

Способ оперативного доступа для потоков с переменной скоростью передачи и потоков, размер которых отличается в модулях доступа видеоизображения, описан, например, в выложенной публикации №2000-341640 японского патента (Заявка №HEI 11-317738 на японский патент).

[Описание файла потока клипа]

Далее будет описана внутренняя структура файлов потока клипа, имеющих расширение PS и сохраненных в директории "STREAM", показанной на фиг.6 ("00001.CLP", "00002.PS" и "00003.PS" на фиг.6).

Файл потока клипа построен на основе MPEG2_Program_Stream(), который определен в MPEG2-System (ISO/IEC 13818-1).

На фиг.18A и фиг.18B показаны Таблица 2-31, Таблица 2-32 и Таблица 2-33, описанные в стандарте MPEG2-System (ISO/IEC 13818-1:20000).

Программный поток, сохраненный в файле потока клипа, представляет собой MPEG2_Program_Stream(), определенный в Таблице 2-31 стандарта MPEG2 System. Этот программный поток состоит, по меньшей мере, из одного pack() и одного MPEG_program_end_code. MPEG2_Program_Stream() описан в японском патенте №2785220.

Один pack() состоит из одного Pack_header() и любого количества PES_packet(), как определено в Таблице 2-32 стандарта MPEG2-System. Pack_header() подробно описан в Таблице 2-33 стандарта MPEG2-System.

В стандарте MPEG2-System pack() имеет размер переменной длины. Однако, как представлено на фиг.16, предполагается, что размер pack() составляет 2048 байт и является фиксированным. В этом примере количество PES_packet() для одного pack() составляет 1, 2 или 3. Когда Pack() начинается с private_stream_2 packet, после него обычно немедленно следует PES_packet() соответствующего потока видеоданных. Кроме того, в качестве третьего PES_packet() могут использоваться padding_packet (пакеты данных заполнения). Пакет private_stream_2 обычно представлен в начале Pack().

Когда Pack() не начинается с пакета private_stream_2, Pack() начинается с PES_packet(), который содержит данные содержания видеоданных, данных звука, данных субтитров или тому подобное. Второй PES_packet() может представлять собой padding_packet (пакет данных заполнения).

На фиг.19A и на фиг.19B - фиг.21A - фиг.21B показан PES_packet(), определенный в Таблице 2-17 стандарта MPEG2-System.

PES_packet(), в основном, состоит из packet_start_code_prefix, stream_id, PES_packet_length (они показаны на фиг.19A и фиг.19B), блоков заголовка (включающих в себя stuffing_byte), которые изменяются в соответствии со stream_id или тому подобное (эти блоки показаны на фиг.19A и фиг.19B - фиг.21A и фиг.21B), и PES_packet_data_byte (показан на фиг.21A и фиг.21B). Когда PES_packet() представляет собой padding_packet (stream_id=padding_stream), при этом требуемое количество padding_byte (0xFF)

(фиг.21A и фиг.21B) повторяют вместо PES_packet_data_byte требуемое количество раз.

Как показано на фиг.19A и фиг.19B - фиг.20A, фиг.20B и на фиг.20C, блоки заголовка PES_packet() могут описывать информацию, которая представляет синхронизацию отображения, называемую PTS (ВМП, временная метка презентации), и информация, которая представляет моменты времени декодирования, называется DTS (ВМД, временная метка декодирования). В соответствии с этим вариантом выполнения PTS добавляют к каждому из всех модулей доступа (модулей декодирования, которые составляют элементарный поток, определенный в MPEG2- System). В случае, когда это указано в MPEG2-System, добавляется DTS.

Элементарный поток, мультиплексированный с программным потоком, сохранен в PES_packet_data_byte (фиг.21A и фиг.21B) PES_packet(). streamJd в PES_packet() описывает значение в соответствии с атрибутом элементарного потока для идентификации элементарного потока, сохраненного в PES_packet_data_byte.

Взаимозависимость значений, описанных в stream_id PES_packet(), и атрибутов (типов) элементарных потоков, определена в Таблице 2-18 стандарта MPEG2-System. На фиг.22A и на фиг.22B показана Таблица 2-18 стандарта MPEG2-System.

В соответствии с вариантом выполнения настоящего изобретения, например, значения, показанные на фиг.23, используются как stream_id, определенный в стандарте MPEG2-System, как показано на фиг.22A и фиг.22B.

Другими словами, в соответствии с данным вариантом выполнения пять структур 10111101B, 10111110B, 10111111B, 110xxxxxB и 1110xxxxB используются как значения stream_id, где "x" представляет любое из значений 0 и 1.

В соответствии с таблицей, показанной на фиг.23, stream_id в PES_packet() элементарного потока, имеющего атрибут private_stream_1, составляет 10111101B. В соответствии с таблицей, показанной на фиг.23, stream_id для PES_packet() для padding_packet составляет 10111110 B. В соответствии с таблицей, показанной на фиг.23, stream_id PES_packet() элементарного потока, имеющего атрибут private_stream_2, составляет 10111111B.

stream_id в PES_packet() потока звука (элементарный поток звука), определенного в MPEG, составляет 110xxxxxB. Пять битов младшего разряда xxxxx из 110xxxxxB представляют собой номер потока звука, который идентифицирует поток звука. 32 ($=2^5$) потока звука (потоки звука, определенные в MPEG), которые могут быть идентифицированы по номеру потока звука, могут быть мультиплексированы с программным потоком.

streamed в PES_packet() потока видеоданных (элементарного потока видеоданных), определенного в MPEG, составляет 1110xxxxB. Четыре бита младших разрядов xxxx из 110xxxxB представляет собой номер потока видеоданных, который идентифицирует видеопоток. 16 ($=2^4$) потока видеоданных (потоки видеоданных, определенные в MPEG), могут быть мультиплексированы с программным потоком.

PES_packet(), stream_id которого составляет 110xxxxB, используется для сохранения потока видеоданных, определенного в MPEG. PES_packet(), stream_id которого равен 110xxxxxB, используется для сохранения потока звука, определенного в MPEG. С другой стороны, stream_id PES_packet() для элементарного потока в соответствии с системой кодирования (например, системой ATRAC) не определен в MPEG. Таким образом, так же, как и видеопоток и поток звука, определенные в MPEG, элементарный поток в соответствии с системой кодирования, который не определен в MPEG, не может быть сохранен в PES_packet() со stream_id.

Таким образом, в соответствии с данным вариантом выполнения PES_packet_data_byte для PES_packet() private_stream_1 расширен так, что он сохраняет элементарный поток в соответствии с системой кодирования, которая не определена в MPEG.

Расширенный PES_packet_data_byte для PES_packet() private_stream_1 описан как private_stream1_PES_payload().

[Описание private_stream1_PES_payload()]

На фиг.24 показан синтаксис private_stream1_PES_payload().

private_stream1_PES_payload() состоит из private_header() и private_payload(). В private_payload() сохранен элементарный поток, такой как поток звука ATRAC, в потоке звука LPCM, поток субтитров или тому подобное, кодированный в соответствии с системой кодирования, не определенной в системе MPEG.

private_header() начинается с private_stream_id (8 битов). private_stream_id представляет собой информацию идентификации, которая идентифицирует элементарный поток, сохраненный в private_payload(). private_stream_id имеет следующее значение в соответствии с атрибутом элементарного потока.

На фиг.25 показана взаимозависимость значения private_stream_id и атрибута элементарного потока, сохраненного в private_payload().

На фиг.25 показано три структуры 0000xxxxB, 0001xxxxB и 100xxxxxB, в качестве значения private_stream_id, где "x" представляет собой любое значение из 0 и 1, аналогично случаю, показанному на фиг.23.

В соответствии с таблицей, показанной на фиг.25, private_stream_id для private_stream1_PES_payload() в private_payload() потока ATRAC представляет собой 0000xxxxB. Четыре бита младших разрядов xxxx в 0000xxxxB представляют собой номер потока звука, который идентифицирует поток звука ATRAC. 16 (=2^4) потоков звука ATRAC, которые могут быть идентифицированы по номеру потока звука, могут быть мультиплексированы с программным потоком (MPEG2_Program_Stream()).

В соответствии с таблицей, показанной на фиг.25, private_stream_id для private_stream1_PES_payload() в private_payload() потока звука LPCM составляет 0001xxxxB. Четыре бита младших разрядов xxxx в 0001xxxxB представляют собой номер потока звука, который идентифицирует поток звука LPCM. 16 (=2^4), которое может быть идентифицировано по количеству потока звука, может быть мультиплексировано с программным потоком.

В соответствии с таблицей, показанной на фиг.25, private_stream_id для private_stream1_PES_payload() private_payload() потока субтитров составляет 100xxxxB. Пять битов младших разрядов xxxxx в 1000xxxxxB представляют собой номер потока субтитров, который идентифицирует поток субтитров. 32 (=2^5) потоков субтитров могут быть мультиплексированы с программным потоком.

На фиг.13 показана взаимозависимость фиг.23 и фиг.25.

Возвращаясь к фиг.24, элементы, которым предшествует private_stream_id в private_stream1_PES_payload(), изменяются в зависимости от атрибута элементарного потока, сохраненного в private_payload(). Атрибут элементарного потока, сохраненного в private_payload(), определяется private_stream_id в начале private_header().

Когда элементарный поток, сохраненный в private_payload(), представляет собой поток звука ATRAC (private_stream_id == ATRAC), reserved_for_future_use (8 бит) описан для будущего расширения. После reserved_for_future_use следует AU_locator (16 бит). AU_locator представляет начальное положение модуля доступа к звуку в потоке звука ATRAC, сохраненном в private_payload(), на основе положения, перед которым непосредственно расположен AU_locator. Когда в private_payload() не сохранен модуль доступа к звуку, 0xFFFF описан, например, в AU_locator.

Когда элементарный поток, сохраненный в private_payload(), представляет собой поток звука LPCM (private_stream_id == LPCM), fs_flag (1 бит), reserved_for_future_use (3 бита), ch_flag (4 бита) и AU_locator (16 бит) описаны в указанном порядке.

fs_flag представляет частоту выборки потока звука LPCM, сохраненного в private_payload(). Когда частота выборки потока звука LPCM составляет 48 кГц, fs_flag установлен в 0. Когда частота выборки потока звука LPCM равна 44,1 кГц, fs_flag установлен в 1.

ch_flag представляет количество каналов потока звука LPCM, сохраненного в private_payload(). Когда поток звука LPCM является монофоническим, ch_flag установлен равным 1. Когда поток звука LPCM представляет собой стереофонический поток, ch_flag

установлен равным 2.

AU_locator представляет начальное положение модуля доступа к звуку потока звука LPCM, сохраненного в private_payload() на основе положения, перед которым непосредственно расположен AU_locator. Когда в private_payload() не сохранен модуль доступа к звуку, в AU_locator описан, например, 0xFFFF.

Когда элементарный поток, сохраненный в private_payload(), представляет собой поток субтитров (private_stream_id == SUBTITLE), reserved_for_future_use (8 бит) описан для будущего расширения. После reserved_for_future_use непосредственно следует AU_locator (16 бит). AU_locator представляет начальное положение модуля доступа субтитров в потоке субтитров, сохраненном в private_payload(), на основе положения, следующего непосредственно после AU_locator. Когда в private_payload() не содержится модуль доступа субтитров, в AU_locator описан, например, 0xFFFF.

[Описание private_stream2_PES_payload()]

На фиг.26 показан синтаксис private_stream2_PES_payload().

private_stream2_PES_payload() представляет собой расширение PES_packet_data_byte (фиг.21A и фиг.21B) private_payload() для private_stream_2, а именно расширение PES_packet_data_byte для PES_packet() private_stream_2. private_stream2_PES_payload() описывает информацию, используемую для декодирования потока видеоданных.

В соответствии с данным вариантом выполнения перед PES_packet() private_stream_2 непосредственно расположена точка возможного начала декодирования потока видеоданных. Таким образом, в соответствии с данным вариантом выполнения, когда PES_packet() private_stream_2 детектируют в программном потоке, потоки видеоданных, непосредственно предшествующие PES_packet(), могут быть декодированы.

RPN_EP_start в EP_map(), показанном на фиг.16, представляет начальное положение PES_packet()private_stream_2 для потока видеоданных.

private_stream2_PES_payload() начинается с reserved_for_future_use (8 бит) для будущего расширения. После reserved_for_future_use следует video_stream_id (8 бит), 1stRef_picture (16 бит), 2ndRef_picture (16 бит), 3rdRef_picture (16 бит), 4thRef_picture (16 бит), au_information() и VBI(), в указанной последовательности.

video_stream_id описывает то же значение, что и stream-id PES_packet() потока видеоданных, который следует непосредственно после PES_packet() в private_stream_2. video_stream_id идентифицирует PES_packet(), в котором сохранен видеопоток, декодированный с помощью информации, сохраненной в private_stream2_PES_payload() в PES_packet() private_stream_2.

1stRef_picture, 2stRef_picture, 3rdRef_picture и 4thRef_picture представляют относительные значения положений в последнем pack(), включающем первый, второй, третий и четвертый эталонные изображения из PES_packet() для private_stream_2 в PES_packet() следующего private_stream_2 потока видеоданных, идентифицированного video_stream_id, соответственно. Подробно 1stRef_picture, 2ndRef_picture, 3rdRef_picture и 4stRef_picture описаны как bytes_to_first_P_pic и bytes_to_second_P_pic в выложенной публикации № HEI 09-46712 японского патента (Заявка на японский патент № HEI 07-47211420).

au_information() описывает информацию о модуле видеодоступа потока видеоданных из PES_packet() private_stream_2 в PES_packet() private stream_2. au_information() будет описано подробно со ссылкой на фиг.27.

VBI() используется для описания информации о закрытом заголовке (??субтитрах).

PES_packet() private_stream_2, который содержит private_stream2_PES_payload() описано для точки возможного начала декодирования каждого потока видеоданных.

На фиг.27 показан синтаксис au_information(), представленный на фиг.26.

au_information() начинается с поля length (16 битов), length представляет размер au_information(), включающий в себя размер в длину. После length следует reserved_for_word_alignment (8 бит) и number_oi_access_unit (8 бит) в указанной последовательности. reserved_for_word_alignment используется для выравнивания слова.

number_of_access_unit представляет количество модулей видеодоступа, сохраненных от PES_packet() private_stream2 для PES_packet() следующего private_stream2.

Другими словами, number_of_access_unit представляет собой количество модулей доступа (изображений), содержащихся в потоке видеоданных, представленном video_stream_id из au_information() для следующего au_information, а именно, до конца файла потока клипа, когда au_information() представляет собой последний для него au_information(), в PES_packet() private_stream_2, video_stream_id которого в private_stream2_PES_payload(), показанном на фиг.26, является тем же.

После number_of_access_unit следует содержание цикла в соответствии с number_of_access_unit. Другими словами, описана информация о, по меньшей мере, одном модуле видеодоступа из PES_packet() private_stream_2, включающая в себя number_of_access_unit для PES_packet() следующего private_stream_2.

Информация, описанная для цикла (информация о модулях видеодоступа), представляет собой следующее.

Цикл содержит pic_struct_copy (4 бита), au_ref_flag (1 бит), AU_length (21 бит) и зарезервировано.

pic_struct_copy описывает копию pic_struct(), определенную в ISO/IEC 14496-10, D.2.2. для модуля видеодоступа, соответствующего MPEG4-AVC (ISO/IEC 14496-10). pic_struct() представляет собой информацию, которая обозначает, что, например, изображение отображается как рамка или после того, как будет отображено верхнее поле изображения, отображается нижнее его поле.

На фиг.28 показана таблица pic_stract.

pic_struct используется как информация обозначения режима отображения, которая обозначает, как изображение отображается.

В таблице pic_struct, показанной на фиг.28, когда обозначено изображение для отображения одной рамки, как перечислено в самом левом поле значения, показанном на фиг.28, 0 устанавливается для pic_struct изображения. Аналогично, когда изображение обозначено так, что отображается верхнее поле или нижнее поле, 1 или 2 устанавливаются в качестве значений pic_struct изображения соответственно. Когда изображение обозначено так, что оно последовательно отображает верхнее поле и нижнее поле, 3 установлено в качестве значения pic_struct изображения. Когда изображение обозначено так, что последовательно отображается нижнее поле и верхнее поле, 4 установлено в качестве значения pic_struct изображения. Когда изображение обозначено так, что последовательно отображается верхнее поле, нижнее поле и верхнее поле, в качестве значения pic_struct для изображения установлено 5. Когда изображение обозначено так, что последовательно отображается нижнее поле, верхнее поле и нижнее поле, в качестве значения pic_struct для изображения установлено 6. Когда изображение обозначено так, что повторно отображается один кадр два раза или три раза, в качестве значения pic_struct изображения установлено 7 или 8 соответственно.

au_ref_flag обозначает, является ли соответствующий модуль доступа эталонным изображением, на которое ссылаются, когда декодируют другой модуль доступа. Когда соответствующий модуль доступа представляет собой эталонное изображение, au_ref_flag установлен в 1. Когда соответствующий модуль доступа не является эталонным изображением, au_ref_flag установлен в 0.

AU_length представляет размер соответствующего модуля доступа в байтах.

[Конкретный пример данных, записанных на диск 101]

На фиг.29 - фиг.32 показаны конкретные примеры данных, которые имеют описанный выше формат и которые были записаны на диск 101, показанный на фиг.1.

На фиг.29 - фиг.32 используются видеопоток в соответствии с MPEG2-video и поток звука в соответствии с ATRAC. Однако поток видеоданных и поток звука, используемые в настоящем изобретении, не ограничиваются этими потоками. Другими словами, можно использовать поток видеоданных в соответствии с MPEG4-Visual, поток видеоданных в соответствии с MPEG4-AVC или тому подобное. С другой стороны, также можно

использовать поток звука в соответствии с MPEG1/2/4 audio, поток звука в соответствии со звуком LPCM или тому подобное.

В отличие от потока видеоданных и потока звука поток субтитров может не быть последовательно декодирован и может не отображаться в тех же интервалах. Другими словами, поток субтитров поступает с перерывами из модуля 215 управления буфером, показанного на фиг.2A и фиг.2B, в модуль 218 управления декодером субтитров. Модуль 218 управления декодером субтитров декодирует поток субтитров.

На фиг.29 - фиг.32 показаны конкретные примеры файла "PLAYLIST.DAT", три файла информации клипа "00001.CLP", "00002.CLP" и "00003.CLP" и т.д. в случае, когда эти три файла информации клипа "00001.CLP", "00002.CLP" и "00003.CLP" сохранены в директории "CLIP" и три файла потока клипа "00001.PS", "00002.PS" и "00003.PS", соответствующие трем файлам информации клипа "00001.CLP", "00002.CLP" и "00003.CLP", сохранены в директории "STREAM" на диске 101, как показано на фиг.6. Однако на фиг.29 - фиг.32 не показана часть данных, таких как файл "PLAYLIST.DAT" и т.д.

Другими словами, на фиг.29 показан конкретный пример файла "PLAYLIST.DAT", представленного на фиг.7.

На фиг.29 number_of_PlayLists установлено равным 2. Таким образом, number of PlayList(), сохраненное в файле "PLAYLIST.DAT", равно 2. На фиг.29 первый и второй PlayList() представляют собой PlayList#0 и PlayList#1 соответственно.

capture_enable_flag_PlayList первого PlayList(), а именно PlayList#0, равен 1. Таким образом, разрешено повторное использование видеоданных, воспроизводимых в соответствии с PlayList#0. С другой стороны, number_of_PlayItems PlayList#0 равно 2. Таким образом, number of PlayItem(), содержащийся в PlayList#0, равно 2. На фиг.29 конкретные примеры PlayItem#0 и PlayItem#1 как два PlayItem() описаны ниже поля "PlayList#0".

В PlayItem#0 в качестве первого PlayItem(), содержащегося в PlayList#0, Clip_Information_file_name, описанный со ссылкой на фиг.8, составляет "00001.CLP", IN_time представляет собой 180090, OUT_time составляет 27180090. Таким образом, клип, воспроизводимый в соответствии с PlayItem#0 из PlayList#0, начинается от момента времени 180090 до момента времени 27180090 файла потока клипа "00001.PS", который соответствует файлу информации клипа "00001.CLP".

В PlayItem#1 в качестве второго PlayItem(), который содержится в PlayList#0, Clip_Information_file_name, описанный со ссылкой на фиг.8, составляет "00002.CLP", In-Time составляет 90000, OUT_time составляет 27090000. Таким образом, клип, воспроизводимый в соответствии с PlayItem#1 из PlayList#0, начинается от момента времени 90000 до момента времени 27090000 файла потока клипа "00002.PS", который соответствует файлу информации клипа "00002.CLP".

На фиг.29, в PlayList#1 в качестве второго PlayList(), capture_enable_flag_PlayList установлен равным 0. Таким образом, не разрешено вторично использовать видеоданные, воспроизводимые в соответствии с PlayList#1. В PlayList#1 number_of_PlayItems равно 1. Таким образом, количество PlayItem(), содержащихся в PlayList#1, равно 1. На фиг.29 конкретный пример PlayItem#0 в качестве одного PlayItem() описан ниже поля "PlayList#1".

В PlayItem#0 в качестве одного из PlayItem(), содержащегося в PlayList#1, Clip_Information_file_name, описанный со ссылкой на фиг.8, составляет "00003.CLP", IN_time равно 90000, OUT_time равно 81090000. Таким образом, клип, воспроизводимый в соответствии с PlayItem#0 из PlayList#1, начинается от момента времени 90000 до момента времени 81,090,000 файла потока клипа "00003.PS", соответствующего файлу информации клипа "00003.CLP".

Далее, на фиг.30A и фиг.30B показан конкретный пример файла информации клипа Clip(), который описан со ссылкой на фиг.12. Другими словами, на фиг.30A и фиг.30B показаны конкретные примеры файлов информации клипа "00001.CLP", "00002.CLP" и "00003.CLP", показанных на фиг.6.

В файле информации клипа "00001.CLP", presentation_start_time равно 90000 и presentation_end_time равно 27,990,000. Таким образом, программный поток, сохраненный в файле потока клипа "00001.PS", соответствующий файлу информации клипа "00001.CLP", может использовать содержание для 310 секунд (27990000-90000/90 кГц).

В файле информации клипа "00001.CLP", capture_enable_flag_Clip установлен равным 1. Таким образом, разрешено повторное использование потока видеоданных, мультиплексированного с программным потоком, сохраненным в файле "00001.PS" потока клипа, соответствующего файлу информации клипа "00001.CLP".

Кроме того, на фиг.30A и фиг.30B в файле информации клипа "00001.CLP" number_of_streams равно 4. Таким образом, четыре элементарных потока мультиплексированы с программным потоком, сохраненным в файле потока клипа "00001.PS".

Если предположить, что четыре элементарных потока представлены stream#0, stream#1, stream#2 и stream#3, на фиг.30A и фиг.30B, конкретные примеры StreamInfo() (фиг.12) четырех элементарных потоков, которые представляют собой stream#0, stream#1, stream#2 и stream#3, представляют собой описанное ниже поле "00001.CLP".

В первом элементарном потоке stream#0 файла потока клипа "00001.PS" stream_id представляет собой 0xE0. Таким образом, как описано со ссылкой на фиг.23 и фиг.25 (или фиг.13), элементарный поток stream#0 представляет собой поток видеоданных. В соответствии с данным вариантом выполнения private_stream_id не скоррелирован с потоком видеоданных. На фиг.30A и фиг.30B private_stream_id равен 0x00.

В потоке видеоданных stream#0 как первый элементарный поток файла потока клипа "00001.PS", picture_size для StaticInfo() (фиг.14), содержащийся в StreamInfo(), равен "720×480", frame_rate составляет "29,97 Гц", cc_flag установлен в значение "Да". Таким образом, поток видеоданных stream#0 представляет собой видеоданные, имеющие 720×480 пикселей и период кадра 29,97 Гц. Кроме того, видеопоток stream#0 содержит данные закрытого заголовка (??субтитров).

В потоке видеоданных stream#0, который используется как первый элементарный поток файла потока клипа "00001.PS", number_of_DynamicInfo для StreamInfo() (фиг.12) равно 0. При этом отсутствует пара из pts_change_point и DynamicInfo().

Во втором элементарном потоке stream#1 в файле потока "00001.PS", stream_id равен OxBD, private_stream_id составляет 0x00. Таким образом, как описано со ссылкой на фиг.23 и фиг.25, элементарный поток stream#1 представляет собой поток звука ATRAC.

В потоке звука ATRAC stream#1, который представляет собой второй элементарный поток файла потока клипа "00001.PS", audio_language_code для StaticInfo() (фиг.14), содержащийся в StreamInfo(), составляет "японский", channel_configuration представляет собой "СТЕРЕО", lfe_existence составляет "НЕТ", sampling_frequency составляет "48 кГц". Таким образом, поток звука ATRAC stream#1 представляет собой стереофонические данные звука на японском языке. Кроме того, поток звука ATRAC stream#1 не содержит канал эффекта низкой частоты, и частота выборки составляет 48 кГц.

Кроме того, в потоке звука ATRAC stream#1 в качестве второго элементарного потока файла потока клипа "00001.PS", поскольку number_of_DynamicInfo для StreamInfo() (фиг.12) равно 0, отсутствует пара pts_change_point и DynamicInfo().

В третьем элементарном потоке stream#2 файла потока клипа "00001.PS" stream_id составляет 0xBD, private_stream_id составляет 0x80. Таким образом, как описано на фиг.23 и фиг.25, элементарный поток stream#2 представляет собой поток субтитров.

В потоке субтитров stream#2 в качестве третьего элементарного потока файла потока клипа "00001.PS" subtitle_language_code для StaticInfo() (фиг.14), содержащийся в StreamInfo(), представляет собой "японский", configurable_flag установлен в 0. Таким образом, поток субтитров stream#2 представляет собой данные субтитров на японском языке. Кроме того, режим его отображения не разрешено изменять.

В потоке субтитров stream#2 в качестве третьего элементарного потока файла потока клипа "00001.PS", поскольку number_of_DynamicInfo для StreamInfo() (фиг.12)

установлен в 0, отсутствует пара из pts_change_point и DynamicInfo().

В четвертом элементарном потоке stream#3 файла потока клипа "00001.PS" stream_id установлен в 0xBD, private_stream_id составляет 0x81. Таким образом, как описано со ссылкой на фиг.23 и фиг.25, элементарный поток stream#3 представляет собой поток субтитров.

Для того чтобы различать поток субтитров stream#2 как третий элементарный поток файла потока клипа "00001.PS" от потока субтитров stream#3 как четвертый элементарный поток, их private_stream_id установлены 0x80 и 0x81 соответственно.

В потоке субтитров stream#2, как в четвертом элементарном потоке файла потока клипа "00001.PS", subtitle_language_code для StaticInfo() (фиг.14), содержащийся в StreamInfo(), составляет "японский", при этом configurable_flag установлен в 1. Таким образом, поток субтитров stream#3 представляет собой данные субтитров на японском языке. При этом разрешено изменение режима отображения потока субтитров stream#3.

В потоке субтитров stream#3, как в четвертом элементарном потоке файла потока клипа "00001.PS", поскольку number_of_DynamicInfo для StreamInfo() (фиг.12) равно 0, отсутствует пара из pts_change_point и DynamicInfo().

На фиг.30A и фиг.30B, в файле информации клипа "00002.CLP", presentation_start_time составляет 90000, presentation_end_time составляет 27090000. Таким образом, программный поток, сохраненный в файле потока клипа "00002.PS", соответствующем файлу информации клипа "00002.CLP", может использовать содержание в течение 300 секунд ((27090000-90000)/90 кГц).

В файле информации клипа "00002.CLP" capture_enable_flag_Clip равен 0. Таким образом, не разрешено повторное использование потока видеоданных, мультиплексированного с программным потоком, сохраненным в файле потока клипа "00002.PS", соответствующем файлу информации клипа "00002.CLP".

На фиг.30A и фиг.30B, в файле информации клипа "00002.CLP" number_of_streams равно 4. Таким образом, так же, как и в предыдущем файле потока клипа "00001.PS", четыре элементарных потока мультиплексированы с программным потоком, сохраненным в соответствующем файле потока клипа "00002.PS".

Если предположить, что четыре элементарных потока представлены как stream#0, stream#1, stream#2 и stream#3, на фиг.30A и фиг.30B конкретные примеры StreamInfo() (фиг.12) четырех элементарных потоков, которые представляют собой stream#0, stream#1, stream#2 и stream#3, представляют собой описанное ниже поле "00002.CLP".

На фиг.30A и фиг.30B, содержание StreamInfo() с первого по четвертый элементарные потоки stream#0-#3 файла потока клипа "00002.PS" являются теми же, что и элементарные потоки с первого по четвертый, stream#0-#3, файла потока клипа "00001.PS". Таким образом, их описание здесь не будет приведено.

Как описано выше, содержание StreamInfo() с первого по четвертый элементарных потоков, stream#0-#3, файла потока клипа "00002.PS" являются теми же, что и содержание с первого по четвертый элементарных потоков, stream#0-#3 файла потока клипа "00001.PS". Таким образом, первый элементарный поток stream#0 файла потока клипа "00002.PS" представляет собой поток видеоданных. Второй элементарный поток stream#1 представляет собой поток звука ATRAC. Третий и четвертый элементарные потоки stream#2 и stream#3 представляют собой потоки субтитров.

Далее, на фиг.30A и фиг.30B, в файле информации клипа "00003.CLP", presentation_start_time равно 90000, presentation_end_time составляет 81090000. Таким образом, программный поток, сохраненный в файле потока клипа "00003.PS", соответствующий файлу информации клипа "00003.CLP", может использовать содержание в течение 900 секунд ((81090000-90000)/90 кГц).

В файле информации клипа "00003.CLP" capture_enable_flag_Clip равен 1. Таким образом, для потока видеоданных, мультиплексированного с программным потоком, сохраненным в файле потока клипа "00003.PS", соответствующем файлу информации клипа "00003.CLP", разрешено вторичное использование.

Кроме того, на фиг.30A и фиг.30B, в файле информации клипа "00003.CLP" number_of_streams равно 3. Таким образом, три элементарных потока мультиплексированы с программным потоком, сохраненным в файле потока клипа "00003.PS".

Если предположить, что три элементарных потока представлены как stream#0, stream#1 и stream#2, на фиг.30A и фиг.30B конкретные примеры StreamInfo() (фиг.12) этих трех потоков, которые представляют собой stream#0, stream#1 и stream#2, представляют собой описанное ниже поле "00003.CLP".

В первом элементарном потоке stream#0 файла потока клипа "00003.PS" stream_id равен 0xE0. Таким образом, как описано со ссылкой на фиг.23 и фиг.25 (или фиг.13), элементарный поток stream#0 представляет собой видеопоток. Так же, как и первый элементарный поток stream#0 файла потока клипа "00001.PS", private_stream_id равен 0x00.

В потоке видеоданных stream#0, так же, как и в первом элементарном потоке файла потока клипа "00003.PS", picture_size для StaticInfo() (фиг.14), содержащегося в StreamInfo(), составляет "720×480", frame_rate составляет "29,97 Гц", cc_flag установлен в "Нет". Таким образом, видеопоток stream#0 представляет собой видеоданные, имеющие 720×480 пикселей и период кадра 29,97 Гц. Видеопоток stream#0 не содержит данные закрытого титра.

В потоке видеоданных stream#0, как и в первом элементарном потоке файла потока клипа "00003.PS", number_of_DynamicInfo для StreamInfo() (фиг.12) равно 2. Таким образом, два набора pts_change_point и DynamicInfo() описаны в StreamInfo().

Во втором элементарном потоке stream#1 файла потока клипа "00003.PS" streamed составляет OxE1. Таким образом, как описано со ссылкой на фиг.23 и фиг.25 (или фиг.13), элементарный поток stream#1 представляет собой видеопоток. Для различия потока видеоданных stream#0 как первого элементарного потока файла потока клипа "00003.PS" от потока видеоданных stream#1 как второго элементарного потока их stream_id установлены равными OxEO и OxE1 соответственно. Так же, как и первый элементарный поток stream#0 файла потока клипа "00001.PS", private_stream_id равен 0x00.

В потоке видеоданных stream#1, который используется как второй элементарный поток файла потока клипа "00003.PS", picture_size, frame_rate и cc_flag для StaticInfo() (фиг.14), содержащегося в StreamInfo(), являются тем же, что и в потоке видеоданных stream#0 в первом элементарном потоке. Таким образом, видеопоток stream#1 как второй элементарный поток файлов потока клипа "00003.PS" представляет собой видеоданные, имеющие 720 x 480 пикселей и период кадра 29,97 Гц. Поток видеоданных stream#1 не содержит данные закрытого титра.

В потоке видеоданных stream#1, как во втором элементарном потоке потока файла клипа "00003.PS", поскольку number_of_Dynamidnfo для StreamInfo() (фиг.12) равно 0, отсутствует пара pts_change_point и DynamicInfo().

В третьем элементарном потоке stream#2 файла потока клипа "00003.PS" stream_id равен 0xBD, private_stream_id составляет 0x00. Таким образом, как описано со ссылкой на фиг.23 и фиг.25, элементарный поток stream#2 представляет собой поток звука ATRAC.

В потоке звука ATRAC stream#2 в качестве третьего элементарного потока файла потока клипа "00003.PS", audio_language_code, channel_configuration, lfe_existence и sampling_frequency для StaticInfo() (фиг.14), содержащегося в StreamInfo(), являются теми же, что и соответствующие параметры потока звука ATRAC для stream#1 как второго элементарного потока файла потока клипа "00001.PS". Таким образом, звуковой поток ATRAC stream#2 как третий элементарный поток файла потока клипа "00003.PS" представляет собой стереофонические данные звука на японском языке. Кроме того, звуковой поток ATRAC stream#2 не содержит канал низкочастотного эффекта. Звуковой поток ATRAC stream#2 имеет частоту выборки 48 кГц.

В звуковом потоке ATRAC stream#2 в качестве третьего элементарного потока файла потока клипа "00003.PS" number_of_DynamicInfo для StreamInfo() (фиг.12) равно 3.

Таким образом, StreamInfo() описывает три набора pts_change_point и DynamicInfo().

На фиг.31 показан конкретный пример EP_map() файла информации клипа Clip(), описанного со ссылкой на фиг.12. Другими словами, на фиг.31 показан конкретный пример EP_map(), показанный на фиг.16, файла информации клипа "00001.CLP", показанного на фиг.6.

На фиг.31, в EP_map(), number_of_stream_id_entries равно 1. Таким образом, EP_map() описывает информацию точки возможного начала декодирования одного элементарного потока.

В EP_map(), показанном на фиг.31, stream_id составляет 0xE0. Таким образом, как описано на фиг.23 и фиг.25, EP_map() описывает PTS_EP_start и RPN_EP_start (фиг.16), как информацию RAPI (Информация точки оперативного доступа), которая представляет собой точку возможного начала декодирования потока видеоданных, идентифицированного по stream_id, который составляет OxEO. Другими словами, на фиг.31, EP_map() представляет собой файл информации клипа "00001.CLP". Как описано со ссылкой на фиг.30A и фиг.30B, в файле потока клипа "00001.CLP", который соответствует файлу информации клипа "00001.CLP", элементарный поток, stream_id которого равен OxEO, представляет собой первый поток видеоданных, stream#0 файла потока клипа "00001.CLP". Таким образом, информация, описанная в EP_map(), показанном на фиг.31, представляет PTS_EP_start и RPN_EP_start точки возможного начала декодирования потока видеоданных stream#0.

На фиг.31 первые пять PTS_EP_start и RPN_EP_start точек возможного начала декодирования первого потока видеоданных stream#0 файла потока клипа "00001.CLP" описаны, но шестой и более последний PTS_EP_start и RPN_EP_start не представлены.

RPN_EP_start, PTS_EP_start, 1stRef_picture, 2ndRef_picture, 3rdRef_picture и 4thRef_picture представляют начальное положение всех RAPI, мультиплексированных с мультиплексированным потоком и конечные положения всех изображений с внутренним кодированием, перед которыми непосредственно следует каждый из RAPI, и второе, третье и четвертое эталонные изображения, перед которыми следует изображение с внутренним кодированием.

Положение верхнего RAPI равно 0 (сектор). PTS изображения с внутренним кодированием, перед которым непосредственно следует верхний RAPI, составляет 90000. Оконечные положения изображения с внутренним кодированием второго, третьего и четвертого эталонных изображений составляет 28, 37, 48 и 58 как относительные значения суммы подсчета секторов с начала RAPI соответственно.

Положение второго RAPI составляет 244 (сектора). PTS изображения с внутренним кодированием, перед которым непосредственно следует второй RAPI, составляет 135045. Оконечные положения изображения с внутренним кодированием второго, третьего и четвертого эталонных изображений составляют 10, 18, 25 и 31 как относительные значения подсчета сектора от начала RAPI соответственно.

Положение третьего RAPI составляет 305 (секторов). PTS изображения с внутренним кодированием, перед которым непосредственно расположен третий RAPI, составляет 180090. Оконечные положения изображения с внутренним кодированием второго, третьего и четвертого изображений представляют собой 25, 44, 50 и 54, которые представляют относительные величины подсчета секторов от начала RAPI соответственно.

Положение четвертого RAPI составляет 427 (секторов). PTS изображения с внутренним кодированием, перед которым непосредственно расположен четвертый RAPI, составляет 225135. Оконечные положения изображения с внутренним кодированием второго, третьего и четвертого эталонных изображений составляют 8, 15, 22 и 29 как относительные величины подсчета секторов с начала RAPI соответственно.

Положение пятого RAPI равно 701 (секторов). PTS изображения с внутренним кодированием, перед которым непосредственно распложен пятый RAPI, равно 270180. Оконечные положения изображения с внутренним кодированием второго, третьего и четвертого эталонных изображений составляют 26, 32, 41 и 48 как относительные

величины подсчета секторов с начала RAPI соответственно.

Значение, близкое к заданному значению величины подсчета секторов (величине подсчета секторов, которые можно совместно считывать в процессе кодирования оконечных положений четырех эталонных изображений (1stRef_Picture, 2ndRef_Picture, 3rdRef_Picture и 4thRef_Picture), сохранено в N-th_Ref_picture_copy. На фиг.31 выбрано значение, ближайшее к величине подсчета секторов "30".

Например, для верхней точки входа PTS_EP_start=90000 и RPN_EP_start=0. Для N-th_Ref_picture_copy, 28, которое представляет собой ближайшее значение к "30", выбрано из 28, 37, 48 и 58. Таким образом, "0", которое представляет 1stRef_picture, сохранено в index_N_minys1.

Далее, для второй точки входа PTS_EP_start=135045 и RPN_EP_start=244. Для N-th_Ref_picture_copy значение 31, которое является ближайшим к "30", выбрано из 10, 18, 25 и 31. Таким образом, значение "3", которое представляет 4thRef_picture, сохранено в index_N_minus1.

Таким образом, значения (0, 28), (3, 31), (0, 25), (3, 29) и (1, 32), сохраненные в index_N_minus1 и N-th_Ref_picture_copy пяти точек входов, представлены в качестве примеров на фиг.31.

Этот выбранный алгоритм представлен исключительно с учетом качества воспроизведения устройства воспроизведения. Таким образом, в данном варианте выполнения выбрано значение, ближайшее к относительно малой величине подсчета секторов "30". Вместо этого может быть выбрано другое значение подсчета секторов. Когда значение index_N_minus1 представляет собой малую величину, если данные считывают из положения RAPI для размера N-th_Ref_picture_copy, количество эталонных изображений, которые содержатся, является малой величиной. В отличие от этого, когда значение index_N_minusl велико, количество эталонных изображений, которые содержатся, велико.

В данном примере, хотя оконечные положения четырех эталонных изображений описаны для данных пяти точек, существует вероятность, в соответствии с которой количество эталонных изображений становится равным четырем или меньше, в зависимости от способа кодирования видеоизображения или интервалов внутри изображений. В таком случае могут быть описаны конечные положения для максимального количества эталонных изображений, самое большее, четырех эталонных изображений.

В EP_map(), показанном на фиг.31, private_stream_id составляет 0x00. Когда stream_id представляет видеопоток, как описано выше, private_stream_id не учитывается.

На фиг.32 показаны конкретные примеры PlayListMark() для PlayList#0 и PlayList#1, описанных со ссылкой на фиг.29 (PlayList(), показанный на фиг.7).

Верхняя представленная на фиг.32 таблица представляет PlayListMark() (фиг.9) для PlayList#0.

В верхней таблице, показанной на фиг.32, number_of_PlayList_marks для PlayListMark() в PlayList#0, равно 7. Таким образом, количество Mark(), содержащихся в PlayListMark() PlayList#0, равно 7.

В верхней таблице, показанной на фиг.32, mark_type (фиг.9) для Mark#0, в качестве первой Mark() для семи Mark(), содержащихся в PlayList#0, составляет "Chapter". Таким образом, Mark#0 представляет собой метку раздела. Кроме того, поскольку ref_to_PlayItem_id (фиг.9) равно 0, Mark#0 принадлежит PlayItem#0 для двух PlayItem#0 и #1, показанных на фиг.29. Кроме того, mark_time_stamp в Mark#0 составляет 180090. Таким образом, Mark#0 представляет собой метку времени (времени воспроизведения) 180090 файла потока клипа, воспроизводимого в соответствии с PlayItem#0, содержащимся в PlayList#0. Как entry_ES_stream_id, так и entry_ES_private_stream_id для Mark#0 равно 0. Таким образом, Mark#0 не скоррелирован ни с одним из элементарных потоков. Кроме того, mark_data для Mark#0 равно 1. Таким образом, Mark#0 представляет раздел с номером 1.

Файл потока клипа, воспроизводимый в соответствии с PlayItem#0, содержащийся в

PlayList#0, представляет собой файл потока клипа "00001.PS", идентифицированный по "00001.CLP", описанном в Clip_Information_file_name в PlayItem#0 (фиг.29). Таким образом, время 180090, представленное mark_time_stamp в Mark#0, представляет собой момент времени файла потока клипа "00001.PS".

В верхней таблице, показанной на фиг.32, Mark#4 в качестве пятой Mark() из семи Mark(), содержащихся в PlayList#0, представляет собой метку раздела, которая совпадает с тем же первым Mark#0.

Другими словами, mark_type (фиг.9) для Mark#4 в качестве пятой Mark(), представляет собой "Chapter". Таким образом, Mark#4 представляет собой метку раздела. Кроме того, ref_to_PlayItem_id (фиг.9) для Mark#4 равно 1. Таким образом, Mark#4 принадлежит PlayItem#1 для двух PlayItem#0 и #1, показанных на фиг.29, содержащихся в PlayList#0. mark_time_stamp в Mark#4 равно 90000. Таким образом, Mark#4 представляет собой метку времени 90000 файла потока клипа, воспроизводимого в соответствии с PlayItem#1, содержащимся в PlayList#0. Кроме того, как entry_ES_stream_id, так и entry_ES_private_stream_id для Mark#4, равно 0. Таким образом, Mark#4 не находится в корреляции ни с одним из элементарных потоков. Кроме того, mark_data для Mark#4 равно 2. Таким образом, Mark#4 представляет раздел с номером, равным 2.

В данном примере воспроизводимый файл потока клипа в соответствии с PlayItem#1, содержащимся в PlayList#0, представляет собой файл потока клипа "00002.PS", идентифицированный "00002.CLP", описанном в Clip_Information_file_name для PlayItem#1, описанном со ссылкой на фиг.29. Таким образом, момент времени 90000, представленный mark_time_stamp в Mark#4, представляет собой время файла потока клипа "00002.PS".

В верхней таблице, показанной на фиг, 32, mark_type (фиг.9) для Mark#1, вторая Mark() из семи меток Mark(), содержащихся в PlayList#0, представляет собой "Index". Таким образом, Mark#1 представляет собой метку индекса. Кроме того, ref_to_PlayItem_id (фиг.9) для Mark#1 равен 0. Таким образом, Mark#1 принадлежит PlayItem#0 двух PlayItem#0 и #1, показанных на фиг.29, содержащихся в PlayList#0. Кроме того, mark_time_stamp для Mark#1 составляет 5 580090. Таким образом, Mark#1 представляет собой метку времени 5580090 файла потока клипа, воспроизводимого в соответствии с PlayItem#0, содержащимся в PlayList#0. Кроме того, как entry_ES_stream_id, так и entry_ES_private_stream_id для Mark#1 равны 0. Таким образом, Mark#1 не скоррелирован ни одним из элементарных потоков. Кроме того, mark_data для Mark#1 равно 1. Таким образом, Mark#1 представляет индекс с номером 1.

В данном примере файл потока клипа, воспроизводимый в соответствии с PlayItem#0, содержащимся в PlayList#0, представляет собой файл потока клипа "00001.PS", как описано выше. Таким образом, время 5 580 090, представленное mark_time_stamp в Mark#1, представляет собой время файла потока клипа "00001.PS".

В верхней таблице, показанной на фиг.32, Mark#2, Mark#5 и Mark#6, такие как третья, шестая и седьмая Mark() из семи Mark(), содержащихся в PlayList#0, представляют собой метки индекса, такие как вторая Mark#1.

В верхней таблице, показанной на фиг.32, mark_type (фиг.9) для Mark#3 как четвертая Mark() из семи Mark() содержащихся в PlayList#0 представляет собой "Событие". Таким образом, Mark#3 представляет собой метку события. Кроме того, ref_to_PlayItem_id (фиг.9) Mark#3 равно 0. Таким образом, Mark#3 принадлежит PlayItem#0 из двух PlayItem#0 и #1, показанных на фиг.29, содержащихся в PlayList#0. Кроме того, mark_time_stamp для Mark#3 составляет 16380090. Таким образом, Mark#3 представляет собой метку времени, 16380090 файла потока клипа, воспроизводимого в соответствии с PlayItem#0, содержащимся в PlayList#0. entry_ES_stream_id и entry_ES_private_stream_id для Mark#3 равны 0. Таким образом, Mark#3 не скоррелирован ни с одним из элементарных потоков. Кроме того, mark_data для Mark#3 равно 0. Таким образом, Mark#3 обеспечивает то, что происходит событие, аргумент которого равен 0.

Как описано выше, файл потока клипа, воспроизводимый в соответствии с PlayItem#0,

содержащимся в PlayList#0, представляет собой файл потока клипа "00001.PS". Время 16380090, представленное mark_time_stamp из Mark#3, представляет собой время файла потока клипа "00001.PS".

В верхней таблице, показанной на фиг.32, моменты времени PlayItem(), которым принадлежит Mark(), описаны в левом поле с правой стороны таблицы PlayListMark() для PlayList#0. Моменты времени PlayList#0 описаны в правом поле с правой стороны в таблице.

Нижняя таблица, показанная на фиг.32, представляет PlayListMark() для PlayList#1 (фиг.9).

В нижней таблице, показанной на фиг.32, number_of_PlayList_marks для PlayListMark() PlayList#1 равно 3. Таким образом, количество Mark(), содержащихся в PlayListMark() для PlayList#1, равно 3.

В нижней таблице, показанной на фиг.32, mark_type (фиг.9) для Mark#0 в качестве первой Mark() из трех Mark(), содержащихся в PlayList#1, представляет собой "Раздел". Таким образом, Mark#0 представляет собой метку раздела. Кроме того, ref_to_PlayItem_id (фиг.9) для Mark#0 равно 0. Таким образом, Mark#0 принадлежит одному из PlayItem#0, показанному на фиг.29, содержащемуся в PlayList#1. mark_time_stamp для Mark#0 составляет 90000. Таким образом, Mark#0 представляет собой метку времени 90000 файла потока клипа, воспроизводимого в соответствии с PlayItem#0, содержащимся в PlayList#1. Как entry_ES_stream_id, так и entry_ES_private_stream_id в Mark#0 равны 0. Таким образом, Mark#0 не скоррелирована ни с одним из элементарных потоков. Кроме того, mark_data для Mark#0 равно 0. Таким образом, Mark#0 представляет раздел с номером 0.

Файл потока клипа, воспроизводимый в соответствии с PlayItem#0, содержащимся в PlayList#1, представляет собой файл потока клипа "00003.PS", идентифицированный по "00003.CLP", описанному в Clip_Information_file_name для PlayItem#0, описанного со ссылкой на фиг.29. Таким образом, время 90 000, представленное mark_time_stamp в Mark#0, представляет собой время файла потока клипа "00003.PS".

В нижней таблице, показанной на фиг.32, mark_type (фиг.9) для Mark#1 как вторая Mark() из трех Mark(), содержащихся в PlayList#1, представляет собой "Событие". Таким образом, Mark#1 представляет собой метку события. Кроме того, ref_to_PlayItem_id (фиг.9) для Mark#1 равно 0. Таким образом, Mark#1 принадлежит PlayItem#0, показанному на фиг.29, содержащемуся в PlayList#1. Кроме того, mark_time_stamp для Mark#1 составляет 27090000. Таким образом, Mark#1 представляет собой метку времени 27090000 файла потока клипа, воспроизводимого в соответствии с PlayItem#0, содержащимся в PlayList#1. Кроме того, в Mark#1 entry_ES_stream_id равен 0xE0 и entry_ES_private_stream_id равен 0. Таким образом, Mark#1 скоррелирован с элементарным потоком, stream_id которого равен 0xE0, а именно Mark#1 скоррелирован с потоком видеоданных, который описан со ссылкой на фиг.23 и фиг.25. Кроме того, mark_data для Mark#1 равно 1. Таким образом, Mark#1 обеспечивает выполнение события с атрибутом, равным 1.

Как описано выше, файл потока клипа, воспроизводимый в соответствии с PlayItem#0, содержащимся в PlayList#1, составляет "00003.PS". Таким образом, время 27090000, представленное mark_time_stamp в Mark#1, представляет собой время файла потока клипа "00003.PS".

Видеопоток, stream_id которого равен 0xE0, скоррелированный с Mark#1, представляет собой видеопоток, stream_id которого составляет 0xE0, описанный в "00003.CLP", описанном в Clip_Information_file_name для PlayItem#0, содержащимся в PlayList#1 (фиг.29), которому принадлежит Mark#1, а именно первому элементарному потоку (потоку видеоданных) stream#0 для трех элементарных потоков stream#0-#2, мультиплексированных с файлом потока клипа "00003.PS", идентифицированным по файлу информации клипа "00003.CLP", показанному на фиг.30A и фиг.30B.

В нижней таблице, показанной на фиг.32, mark_type (фиг.9) для Mark#2, в качестве

третьей Mark() из трех Mark(), содержащихся в PlayList#1, представляет собой "Событие". Таким образом, Mark#2 представляет собой метку события. Кроме того, ref_to_PlayItem_id (фиг.9) для Mark#2 равно 0. Таким образом, Mark#2 принадлежит PlayItem#0, который представляет собой один из PlayItem, показанный на фиг.29, содержащийся в PlayList#1. Кроме того, mark_time_stamp для Mark#2 составляет 27540000. Таким образом, Mark#1 представляет собой метку времени 27540000 файла потока клипа, воспроизводимую в соответствии с PlayItem#0, содержащимся в PlayList#1. Кроме того, в Mark#2 entry_ES_stream_id составляет OxEl и entry_ES_private_stream_id равен 0. Таким образом, Mark#2 представляет собой поток элемента, stream_id которого равен 0xE1, а именно скоррелирован с потоком видеоданных, как описано со ссылкой на фиг.23 и фиг.25. Кроме того, mark_data для Mark#2 равна 2. Таким образом, Mark#2 обеспечивает выполнение события, аргумент которого равен 2.

В этом примере, как описано выше, файл потока клипа, воспроизводимый в соответствии с PlayItem#0, содержащимся в PlayList#1, представляет собой файл потока клипа "00003.PS". Таким образом, время 27540000, представленное Mark#2, представляет собой время файла потока клипа "00003.PS".

Видеопоток, stream_id которого равен 0xE1, скоррелированный с Mark#2, представляет собой видеопоток, stream_id которого равно 0xE1, описанный в "00003.CLP", описанном в Clip_Information_file_name для PlayItem#0, содержащемся в PlayList#1, показанном на фиг.29, а именно во втором элементарном потоке (потоке видеоданных) stream#1 из трех элементарных потоков stream#0-#2, мультиплексированных с файлом потоком клипа "00003.PS", распознаваемых по файлу информации клипа "00003.CLP", показанном на фиг.30A и фиг.30B.

В нижней таблице, показанной на фиг.32, время для PlayItem(), которым принадлежат Mark(), описаны с правой стороны в таблице PlayListMark() для PlayList#1.

На фиг.32, хотя mark_data описывает номера раздела и индекса, которые представляют метки раздела и индекса, они не обязательно могут быть описаны в mark_data. Вместо этого, путем подсчета меток раздела и индекса PlayListMark() можно распознавать номера раздела и индекса.

[Описание работы устройства воспроизведения диска]

Далее будет описана работа устройства воспроизведения диска, показанного на фиг.1, предполагая, что данные, описанные на фиг.29 - фиг.32, были записаны на диск 101, показанный на фиг.1.

В соответствии с определением MPEG2-System для системы мультиплексирования не обязательно добавлять временную метку ко всем модулям доступа. Вместо этого, такое определение указывает, что временные метки могут быть добавлены через интервал 0,7 секунды или меньше. Другими словами, существуют модули доступа, имеющие временную метку, и модули доступа, не имеющие временной метки.

В данном примере предполагается, что модуль доступа в положении начала декодирования потока видеоданных обычно имеет временную метку. Другими словами, как будет описано в разделе "Процесс подготовки к воспроизведению" максимум PTS_EP_start, который удовлетворяет условию PTS_EP_start $\leq$ IN_time с EP-map(), получают как положение начала декодирования с использованием способа двоичного поиска. Модуль доступа, немедленно следующий после положения начала воспроизведения видеоданных, зарегистрированных в EP_map(), обычно имеет временную метку.

Кроме того, определение указывает, что существуют не спаренные поля. Другими словами, немедленно после модуля доступа pic_struct=1, размещен модуль доступа pic_struct=2. Кроме того, немедленно после модуля доступа pic_struct=2, размещен модуль доступа pic_struct=1.

В этом примере предполагается, что модули доступа pic_struct=7 и 8 не используются.

Когда диск 101 загружают в привод 102 диска, соответствующее сообщение передают через интерфейс 114 привода и операционную систему 201, показанные на фиг.2A и фиг.2B, в программу 210 воспроизведения видеосодержания. Когда программа 210

воспроизведения содержания получает из операционной системы 201 сообщение, которое обозначает, что диск 101 был загружен в привод 102 диска, программа 210 воспроизведения видеосодержания начинает процесс предварительной обработки воспроизведения, показанный на фиг.33.

[Процесс предварительной обработки воспроизведения]

На фиг.33 показана блок-схема последовательности операций, описывающая процесс предварительной обработки воспроизведения, которую выполняет программа 210 воспроизведения видеосодержания.

Следует отметить, что устройство воспроизведения диска не обязательно должно выполнять операции или обработку во временной последовательности, соответствующей блок-схеме последовательности операций. В качестве альтернативы, устройство воспроизведения диска может выполнять эти операции или обработку параллельно или по отдельности. Однако в описании для удобства операции или обработка, выполняемые устройством воспроизведения диска, будут описаны в соответствии с блок-схемой последовательности операций.

В процессе предварительной обработки воспроизведения на этапе S101 программа 210 воспроизведения видеосодержания проверяет диск 101, используя функцию файловой системы операционной системы 201, и определяет, является ли диск 101 нормальным диском для программы 210 воспроизведения видеосодержания.

Как указано выше, хотя здесь к диску 101 обращаются (с него считывают файлы) с использованием функции файловой системы операционной системы 201, ее описание здесь не будет приведено.

Когда результат, определенный на этапе S101, представляет, что диск 101 не является нормальным диском, а именно файловая система, используемая на диске 101, не соответствует операционной системе 201, или в корневой директории диска 101 не содержится директория "VIDEO", программа 210 воспроизведения видеосодержания определяет, что программа 210 воспроизведения видеосодержания не соответствует диску 101, обработка переходит на этап S 102. На этапе S102 модуль 219 обработки графических изображений выполняет обработку ошибки и заканчивает предварительную обработку воспроизведения.

Другими словами, модуль 219 обработки графических изображений генерирует сообщение об ошибке, которое обозначает, что диск 101 не является нормальным диском, в связи с обработкой ошибки, и обеспечивает вывод модулем 220 вывода видеоизображения сообщения об ошибке, в результате чего отображается сообщение об ошибке. Обработка ошибки может выполняться, например, путем вывода звукового сигнала предупреждения через модуль 221 вывода звука или путем выгрузки диска 101 из привода 102 диска.

Когда результат, определенный на этапе S101, обозначает, что диск 101 является нормальным диском, обработка переходит на этап S103. На этапе S103 программа 210 воспроизведения видеосодержания обеспечивает передачу запроса из модуля 213 подачи данных содержания в операционную систему 201 на считывание двух файлов данных "SCRIPT.DAT" и "PLAYLIST.DAT", сохраненных в директории "VIDEO" диска 101 (фиг.6). После этого обработка переходит на этап S104. На этапе S104 файл "SCRIPT.DAT" подают в модуль 211 управления сценарием. Кроме того, файл "PLAYLIST.DAT" подают в модуль 212 управления проигрывателем.

После этого поток переходит с этапа S104 на этап S105-S107. На этапах S105-S107 модуль 212 управления проигрывателем выполняет процесс инициализации. Модуль 211 управления сценарием ожидает, пока модуль 212 управления проигрывателем не закончит процесс инициализации.

[Процесс инициализации модуля 212 управления проигрывателем]

В процессе инициализации, на этапе S105, модуль 212 управления проигрывателем анализирует файл "PLAYLIST.DAT" и проверяет количество файлов информации клипа, описанных в файле "PLAYLIST.DAT", и их названия файлов.

Другими словами, поскольку файл "PLAYLIST.DAT" представляет собой такой файл, как показан на фиг.29, и number_of_PlayLists в файле "PLAYLIST.DAT", показанное на фиг.29, равно 2, модуль 212 управления проигрывателем распознает, что существуют два PlayList(), которые представляют собой PlayList#0 и PlayList#1. Кроме того, поскольку number_of_PlayItems первого PlayList#0 в файле "PLAYLIST.DAT", показанном на фиг.29, равно 2, модуль 212 управления проигрывателем распознает, что PlayList#0 содержит два PlayItem(), которые представляют собой PlayItem#0 и PlayItem#1. После этого модуль 212 управления проигрывателем обращается к Clip_Information_file_name первого PlayItem#0 и второго PlayItem#1, содержащихся в PlayList#0 файла "PLAYLIST.DAT", показанного на фиг.29 и, распознают, что файл информации клипа первого PlayItem#0, содержащегося в PlayList#0, представляет собой "00001.CLP", и файл информации клипа второго PlayItem#1 представляет собой "00002.CLP".

Аналогично, модуль 212 управления проигрывателем распознает, что второй PlayList#1 содержит один PlayItem() (PlayItem#0), поскольку number_of_Play Items равно 1 и, что файл информации клипа PlayItem#0 представляет собой "00003.CLP" из-за Clip_Information_file_name PlayItem#0.

После этого поток переходит с этапа S105 на этап S106. На этапе S106 модуль 212 управления проигрывателем считывает файлы информации клипа, распознанные на этапе S105, а именно три файла информации клипа "00001.CLP", "00002.CLP" и "00003.CLP" из директории "CLIP", находящейся в директории "VIDEO" на диске 101.

На этапе S106 требуется считать только файл информации клипа PlayItem в PlayList(), который первым воспроизводят. В соответствии с данным вариантом выполнения, однако, как описано выше, все файлы информации клипа PlayItem() из PlayList() предварительно считываются.

После этапа S106 поток обработки переходит на этап S107. На этапе S107 модуль 212 управления проигрывателем определяет, были ли файлы информации клипа, распознанные на этапе S105, успешно считаны. Кроме того, модуль 212 управления проигрывателем определяет, присутствуют ли файлы потока клипа, соответствующие файлам информации клипа, на диске 101. Другими словами, на этапе S107 модуль 212 управления проигрывателем определяет, были ли файлы информации клипа "00001.CLP", "00002.CLP" и "00003.CLP" успешно считаны и файлы потока клипа "00001.PS", "00002.PS" и "00003.PS", которые соответствуют файлам информации клипа "00001.CLP", "00002.CLP" и "00003.CLP", которые присутствует в директории "STREAM", находящиеся в директории "VIDEO" на диске 101.

Когда результат, определенный на этапе S107, обозначает, что файлы информации клипа, распознанные на этапе S105, не были успешно считаны или что файлы потока клипа, соответствующие файлам информации клипа, не присутствуют на диске 101, а именно файлы информации клипа и файлы потока клипа в соответствии с файлом "PLAYLIST.DAT" не были записаны на диске 101, определяется, что программа 210 воспроизведения видеосодержания не соответствует диску 101. После этого поток обработки переходит на этап S102. На этапе S102 выполняется указанная выше обработка ошибки, и затем процесс предварительной обработки воспроизведения заканчивается.

В отличие от этого, когда результат, определенный на этапе S107, показывает, что файлы информации клипа, распознанные на этапе S105, были успешно считаны и что файлы потока клипа, соответствующие файлам информации клипа, присутствуют на диске 101, модуль 212 управления проигрывателем заканчивает процесс инициализации. После этого поток обработки переходит на этап S108.

На этапе S108 модуль 211 управления сценарием проводит анализ и выполняет файл "SCRIPT.DAT".

Когда модуль 211 управления сценарием выполняет файл "SCRIPT.DAT", предполагая, что обеспечивается выполнение модулем 212 управления проигрывателем воспроизведения первого PlayList() в (PlayList#0). В этой точке выполняется процесс воспроизведения, показанный на фиг.34.

[Процесс воспроизведения]

На фиг.34 показана блок-схема процесса обработки воспроизведения, которую выполняет программа 210 воспроизведения видеосодержания.

[Обработка подготовки к воспроизведению]

На этапах S121 и S122 модуль 212 управления проигрывателем выполняет обработку подготовки к воспроизведению для PlayList(), воспроизведение которой обеспечивается с помощью модуля 211 управления сценарием, а именно первого PlayList() (PlayList#0).

Другими словами, на этапе S121 модуль 212 управления проигрывателем проверяет IN_time (фиг.8) первого PlayItem#0, содержащегося в первом PlayList#0. После этого поток обработки переходит на этапу S122. На этапе S122 модуль 212 управления проигрывателем проверяет положение начала воспроизведения, соответствующее IN_time для PlayItem#0 файла потока клипа "00001.PS", воспроизводимого первым PlayItem#0, содержащимся в первом PlayList#0.

Когда IN_time (фиг.8) PlayItem() представляет начало файла потока клипа, программный поток считывают с начала файла потока клипа. Однако, когда IN_time представляет другое место, кроме начала файла потока клипа, модуль 211 управления проигрывателем должен детектировать положение, соответствующее IN_time, и считывать файл потока клипа из этого положения.

В частности, на фиг.29, IN_time первого PlayItem#0, содержащегося в первом PlayList#0, составляет 180090. Модуль 212 управления проигрывателем выполняет поиск в EP_map(), показанном на фиг.31, файла потока клипа "00001.CLP", воспроизводимого первым PlayItem#0, содержащимся в первом PlayList#0, для положения начала воспроизведения, где IN_time PlayItem#0 составляет 180090.

Другими словами, модуль 212 управления проигрывателем выполняет поиск максимального PTS_EP_start, которое удовлетворяет PTS_EP_start $\leq$ IN_time, где PTS_EP_start представляет собой точку возможного начала декодирования, описанную в EP_map(), с помощью способа двоичного поиска или тому подобное. Это связано с тем, что положение, представленное IN_time, обычно не является точкой возможного начала декодирования.

В этом случае, как описано выше, IN_time составляет 180090. Кроме того, в EP_map(), показанном на фиг.31 файла потока клипа "00001.CLP", воспроизводимого первым PlayItem#0, содержащимся в первом PlayList#0, максимальное значение PTS_EP_start, которое удовлетворяет PTS_EP_start $\leq$ IN_time, равно 180090. Таким образом, модуль 212 управления проигрывателем выполняет поиск в EP_map(), показанном на фиг.31, для PTS_EP_start, которое равно 180090.

Кроме того, модуль 212 управления проигрывателем считывает 305 (секторов), поиск которых выполняется для RPN_EP_start и определяет положение, представленное RPN_EP_start в файле потока клипа "00001.PS", как положение начала воспроизведения.

После того как модуль 212 управления проигрывателем определит положение начала воспроизведения, поток обработки переходит с этапа S122 на этап S123. На этапе S123 модуль 212 управления проигрывателем выполняет управление модулем 219 обработки графических изображений для отображения временного кода. Модуль 219 обработки графических изображений генерирует временной код под управлением модуля 212 управления проигрывателем и выводит временной код в модуль 220 вывода видеоизображений. В результате отображается временной код.

Временной код, отображаемый на этапе S123, представляет собой, например, значение, на основе которого начало PlayList() преобразуется в 00:00:00 (часы:минуты:секунды). В дополнение к или вместо временного кода могут отображаться номер раздела и номер индекса.

[Обработка анализа PlayListMark()]

После отображения временного кода на этапе S123 поток обработки переходит на этап S124. На этапе S124 модуль 212 управления проигрывателем выполняет процесс анализа, который анализирует PlayList(), воспроизведение которого обеспечивается с помощью

модуля 211 управления сценарием, чтобы, а именно, PlayListMark() (фиг.9), описанный в первом PlayList() (PlayList#0).

В частности, в верхней таблице, показанной на фиг.32, number_of_PlayList_marks для PlayListMark() первого PlayList#0 файла "PLAYLIST.DAT", который был предварительно считан, равно 7. Таким образом, модуль 212 управления проигрывателем распознает, что количество Mark(), содержащихся в PlayList#0, равно 7.

Кроме того, модуль 212 управления проигрывателем анализирует семь Mark() верхней таблицы, показанной на фиг.32, и распознает, что четыре Mark() с первой по четвертую Mark() из семи Mark() принадлежат первому PlayItem() (PlayItem#0) PlayList#0.

После этого модуль 212 управления проигрывателем получает mark_time_stamp для четырех Mark(), которые принадлежат первому PlayItem#0 PlayList#0, и передает их как четырехэлементную матрицу в модуль 214 управления декодированием. Таким образом, четыре значения времени {180090}, {5580090}, {10980090} и {16380090} в виде mark_time_stamp четырех Mark() с первой по четвертую Mark() из семи Mark() в верхней таблице, показанной на фиг.32, передают из модуля 212 управления проигрывателем в модуль 214 управления декодированием. В этой точке атрибут "обработка метки" этих значений времени также передают из модуля 212 управления проигрывателем в модуль 214 управления декодированием. Когда время, подсчитанное к этому времени блоком 214A подсчета, соответствует времени, имеющему атрибут "обработка метки", модуль 214 управления декодированием передает сообщение, которое представляет эту ситуацию, время, сопоставленное со временем, имеющим этот атрибут "обработка метки", и атрибут "обработка метки" в модуль 212 управления проигрывателем.

[Обработка принятия решения в отношении элементарного потока, предназначенного для воспроизведения]

После этого поток обработки переходит с этапа S124 на этап S125. На этапе S125 модуль 212 управления проигрывателем принимает решение в отношении элементарного потока, предназначенного для воспроизведения.

Другими словами, в файле информации клипа "00001.CLP", показанном на фиг.30A и фиг.30B, название файла которого описано в Clip_Information_file_name первого PlayItem#0 (фиг.29) первого PlayList#0 как PlayList(), воспроизведение которого обеспечивается модулем 211 управления сценарием, number_of_streams равно 4. Таким образом, модуль 212 управления проигрывателем распознает, что четыре элементарных потока были мультиплексированы с соответствующим файлом потока клипа "00001.PS". Кроме того, модуль 212 управления проигрывателем проверяет stream_id и private_stream_id для StaticInfo() файла информации клипа "00001.CLP", показанного на фиг.30A и фиг.30B, четырех элементарных потоков, и распознает, что четыре элементарных потока представляют собой один видеопоток, один поток звука ATRAC и два потока субтитров. Другими словами, модуль 212 управления проигрывателем распознает количество элементарных потоков, имеющих индивидуальные атрибуты, мультиплексированные с файлом потока клипа "00001.PS".

Информация о количестве элементарных потоков, имеющих индивидуальные атрибуты, мультиплексированные с файлом потока клипа, используется для изменения одного элементарного потока на другой элементарный поток, предназначенный для воспроизведения (из одного режима звука на другой режим звука или из одного режима субтитров на другой режим субтитров). Когда файл потока клипа не содержит файл потока субтитров (а именно, содержание не включает в себя данные субтитров), определяется, присутствует ли поток субтитров с информацией о количестве элементарных потоков, имеющих атрибут "поток субтитров".

Модуль 212 управления проигрывателем выбирает и принимает решение в отношении воспроизводимого элементарного потока в соответствии с результатом проверки StaticInfo(). В этом случае четыре элементарных потока, мультиплексированных с файлом потока клипа "00001.PS", содержат один элементарный поток, имеющий атрибут "поток видеоданных", и один элементарный поток, имеющий атрибут "поток звука". Таким

образом, элементарный поток, имеющий атрибут "поток видеоданных", и элементарный поток, имеющий атрибут "поток звука" (поток звука ATRAC), безусловно определяются как элементарные потоки, предназначенные для воспроизведения.

С другой стороны, четыре элементарных потока, мультиплексированных с файлом потока клипа "00001.PS", содержат два элементарных потока, имеющих атрибут "поток субтитра". Таким образом, один из этих двух потоков субтитров выбирают и определяют как элементарный поток, предназначенный для воспроизведения. В этом примере выбирают поток субтитров, который первым поступает в двух потоках субтитров в файле информации клипа "00001.CLP".

Когда распознают атрибуты и количество элементарных потоков, мультиплексированных с файлом потока клипа "00001.PS", эти четыре элементарных потока должны быть идентифицированы. Модуль 212 управления проигрывателем идентифицирует эти четыре элементарных потока, мультиплексированных с файлом потока клипа "00001.PS" с использованием stream_id и private_stream_id.

Другими словами, модуль 212 управления проигрывателем идентифицирует элементарный поток, имеющий атрибут "потока видеоданных" из четырех элементарных потоков, мультиплексированных с файлом потока клипа "00001.PS" с stream_id, который представляет собой 0xE0, как описано в файле информации клипа "00001.CLP", показанном на фиг.30A и фиг.30B.

Кроме того, модуль 212 управления проигрывателем идентифицирует поток звука ATRAC, который представляет собой элементарный поток, имеющий атрибут "потока звука", из четырех элементарных потоков, мультиплексированных с файлом потока клипа "00001.PS" со stream_id, который равен 0xBD, и private_stream_id, который составляет 0x00, как описано в файле информации клипа "00001.CLP", показанном на фиг.30A и фиг.30B.

Кроме того, модуль 212 управления проигрывателем идентифицирует два потока субтитров, которые представляют собой элементарные потоки, имеющие атрибут "потока субтитров", из четырех элементарных потоков, мультиплексированных с файлом потока клипа "00001.PS" со stream_id, который представляет собой 0xBD, и private_stream_id, который представляет собой 0x80, а также со stream_id, который равен 0xBD, и private_stream_id, который равен 0x81, как описано для файла информации клипа "00001.CLP" на фиг.30A и фиг.30B соответственно.

Как описано выше, элементарный поток, мультиплексированный с файлом потока клипа, может быть идентифицирован по stream_id и private_stream_id, описанным как метаданные файла информации клипа, соответствующего файлу потока клипа.

Комбинация stream_id и private_stream_id представляет собой механизм, предусмотренный для расширения мультиплексирования MPEG2-System. Когда комбинация stream_id и private_stream_id используется как метаданные, элементарный поток может быть надежно идентифицирован. Кроме того, когда private_stream_id расширен так, что в нем увеличено количество атрибутов соответствующих элементарных потоков, текущий механизм можно использовать без каких-либо изменений. Таким образом, комбинация stream_id и private_stream_id обладает высокой возможностью расширения.

Другими словами, например, в стандарте диска типа blu-ray (BD) использует PID (ИДП, ИД пакета) транспортного потока стандарта MPEG2 для идентификации данных. Таким образом, стандарт BD ограничен стандартом MPEG2. С другой стороны, стандарт DVD-Video определяет sub_stream_id, который аналогичен private_stream_id. Однако sub_stream_id не может быть описан в базе данных для идентификации потока. sub_stream_id описан в фиксированной области для информации, состоящей только из от восьми до 32 потоков (см. VI4-49, Таблица 4.2.1-2 (VTS_AST_ATRT) и VI4-52, Таблица 4.2.1-3 (VTS_SPST_ATRT)). Таким образом, sub_stream_id не обладает высокой возможностью расширения.

С другой стороны, комбинация stream_id и private_stream_id может быть описана с использованием метаданных. Например, в файле информации клипа, Clip(), показанном на

фиг.12, комбинация из stream_id и private_stream_id, может быть описана определенное количество раз, представленное number_of_streams. Таким образом, элементарные потоки, мультиплексированные файлом потока клипа, могут быть идентифицированы по комбинации stream_id и private_stream_id как метаданные, описанные в файле информации клипа Clip(), независимо от количества элементарных потоков (в диапазоне, представленном number_of_streams).

В соответствии с этим вариантом выполнения комбинация из stream_id и private_stream_id используется для идентификации элементарного потока, мультиплексированного с файлом потока клипа, соответствующим файлу информации клипа, показанному на фиг.12. Кроме того, эта комбинация может использоваться для идентификации элементарного потока, который коррелирует Mark() как комбинацию entry_ES_stream_id и entry_ES_private_stream_id для PlayListMark(), показанного на фиг.9. Кроме того, комбинация stream_id и private_stream_id используется для идентификации элементарного потока, который описывает информацию точки возможного начала декодирования в EP_map(), показанном на фиг.16.

[Обработка управления выходным атрибутом]

Далее поток обработки переходит с этапа S125 на этап S126. На этапе S126 модуль 212 управления проигрывателем выполняет обработку управления выходным атрибутом для элементарного потока, определенного для воспроизведения на этапе S125.

В частности, модуль 212 управления проигрывателем проверяет number_of_DynamicInfo (фиг.12), которое представляет количество DynamicInfo() (фиг.15), которые описывают выходные атрибуты потока видеоданных, поток звука ATRAC и поток субтитров, определенные для воспроизведения на этапе S125.

В этом случае, видеопоток, поток звука ATRAC и поток субтитров, предназначенных для воспроизведения, представляет собой элементарные потоки, мультиплексированные с файлом потока клипа "00001.PS". В файле информации клипа "00001.CLP", показанном на фиг.30A и фиг.30B, их number_of_DynamicInfo's все равны 0. Когда number_of_DynamicInfo все равны 0, модуль 212 управления проигрывателем не выполняет процесс управления выходного атрибута для выходных атрибутов элементарных потоков, предназначенных для воспроизведения.

Когда number_of_DynamicInfo элементарных потоков, предназначенных для воспроизведения, не равно 0, выполняется процесс управления выходного атрибута для элементарных потоков. Процесс управления выходных атрибутов будет описан ниже.

[Процесс обработки начала воспроизведения]

После этапа S126 поток обработки переходит на этап S127. На этапе S127 модуль 212 управления проигрывателем выполняет процесс обработки начала воспроизведения для элементарных потоков, предназначенных для воспроизведения.

Другими словами, модуль 212 управления проигрывателем передает название файла для файла потока клипа "00001.PS", с помощью которого элементарный поток, предназначенный для воспроизведения, был мультиплексирован, и RPN_EP_start (=305), описанный в EP_map(), как положение начала воспроизведения, определенное на этапе S122, в модуль 213 подачи данных содержания.

Кроме того, модуль 212 управления проигрывателем инициализирует модуль 215 управления буфером перед тем, как программный поток, сохраненный в файле потока клипа "00001.PS", с помощью которого элементарный поток, предназначенный для воспроизведения, был мультиплексирован, передают в модуль 215 управления буфером.

В частности, модуль 215 управления буфером (фиг.5) устанавливает то же самое для указателя начала данных, сохраненного на блоке 231 накопителя указателя начала данных, указателя записи данных, сохраненного на блоке 232 накопителя указателя записи данных, указателя считывания видеоданных, сохраненного на блоке 241 накопителя указателя считывания видеоданных, указателя считывания звука, сохраненного на блоке 251 накопителя указателя считывания звука и указателя считывания субтитров, сохраненного на блоке 262 накопителя указателя считывания субтитров.

Таким образом, указатель начала данных, сохраненный на блоке 231 накопителя указателя начала данных, и указатель записи данных, сохраненный на блоке 232 накопителя указателя записи данных, содержат одно и то же положение в буфере 215A модуля 215 управления буфером. Это обозначает, что в буфере 215A не были сохранены действительные данные.

Кроме того, модуль 212 управления проигрывателем передает stream_id и, в случае необходимости, private_stream_id как информацию идентификации для элементарного потока, предназначенного для воспроизведения, в модуль 215 управления буфером.

Другими словами, как описано выше, видеопоток, имеющий атрибут "потока видеоданных" в элементарных потоках, предназначенных для воспроизведения, идентифицируется по stream_id, который равен 0xEO. Поток звука ATRAC, имеющий атрибут "потока звука", идентифицируется по stream_id, который имеет значение 0xBD, и private_stream_id, который имеет значение 0x00. Поток субтитров, имеющий атрибут "потока субтитров", идентифицируется по stream_id, который равен 0xBD, и private_stream_id, который составляет 0x80. Модуль 212 управления проигрывателем передает эти stream_id и private_stream_id в модуль 215 управления буфером.

В модуле 215 управления буфером (фиг.5) блок 233 функции считывания видеоданных сохраняет stream_id, который составляет 0xEO для потока видеоданных, принятого из модуля 212 управления проигрывателем, в регистре 242 stream_id. Кроме того, блок 234 функции считывания звука сохраняет stream_id, который составляет 0xBD, и private_stream_id, который составляет 0x00, полученные из модуля 212 управления проигрывателем, в регистре 252 stream_id и в регистре 253 private_stream_id соответственно. Кроме того, блок 235 функции считывания субтитров сохраняет stream_id, который составляет 0xBD и private_stream_id, который составляет 0x80, принятые из модуля 212 управления проигрывателем, в регистре 263 stream_id и в регистре 264 private_stream_id соответственно.

Модуль 212 управления проигрывателем сохраняет stream_id и private_stream_id для элементарного потока, предназначенного для воспроизведения, переданного в модуль 215 управления буфером для последующей обработки. В модуле 212 управления проигрывателем используются stream_id и private_stream_id, когда идентифицируется поступление сообщения запроса на изменение потока или воспроизводимого потока, во время обработки метки, которая будет описана ниже.

Для инициализации модуля 215 управления буфером (фиг.5) модуль 212 управления проигрывателем устанавливает флаг функции считывания субтитров, имеющий значение в соответствии с файлом потока клипа, мультиплексированным с элементарным потоком, предназначенным для воспроизведения, для блока 261 сохранения флага функции считывания субтитров.

Другими словами, в данном случае, поскольку файл потока клипа "00001.PS", с которым были мультиплексированы элементарные потоки, предназначенные для воспроизведения, содержит поток субтитров, флаг функции считывания субтитров, значение которого равно 1, установлен на блоке 261 накопителя флага функции считывания субтитров для активирования блока 235 функции считывания субтитров. Когда файл потока клипа, с помощью которого был мультиплексирован элементарный поток, предназначенный для воспроизведения, не содержит поток субтитров, флаг функции считывания субтитров, значение которого равно 0, устанавливается для блока 261 накопителя флага функции считывания субтитров. В этом случае блок 235 функции считывания субтитров не выполняет какую-либо обработку.

Кроме того, модуль 212 управления проигрывателем передает IN_time, значение которого равно 180 090 и OUT_time, значение которого составляет 27 180 090 первого PlayItem#0 (фиг.29), содержащегося в первом PlayList#0, воспроизведение которого модулем 212 управления проигрывателем было обеспечено модулем 211 управления сценарием для модуля 214 управления декодированием, модуль 214 управления декодированием использует IN_time для начала декодирования клипа, воспроизводимого в

соответствии с PlayItem(), и OUT_time для остановки декодирования клипа и для управления процессом изменения Play Item, который будет описан ниже.

Модуль 212 управления проигрывателем инициализирует режим отображения потока субтитров, в котором модуль 219 обработки графических изображений отображает поток субтитров. Другими словами, модуль 212 управления проигрывателем управляет модулем 219 обработки графических изображений, для отображения потока субтитров в принятом по умолчанию режиме отображения.

[Начало считывания данных]

После этого поток обработки переходит с этапа S127 на этап S128. Модуль 212 управления проигрывателем управляет модулем 213 подачи данных содержания для считывания файла потока клипа, который содержит программный поток, с которым был мультиплексирован элементарный поток, предназначенный для воспроизведения, используя функцию операционной системы 201. Другими словами, модуль 213 подачи данных содержания обозначает файл потока клипа "00001.PS" в директории "STREAM", которая находится в директории "VIDEO" диска 101 (фиг.6), обозначает сектор 305, который представляет собой положение начала воспроизведения, которое было определено на этапе S122, и обеспечивает считывание этого файла операционной системой 201. Модуль 213 подачи данных содержания обеспечивает подачу данных с помощью операционной системы 201 для подачи данных, которые были считаны с диска 101, в модуль 215 управления буфером.

Таким образом, программный поток файла потока клипа "00001.PS" считывается с диска 101. Программный поток подают в модуль 215 управления буфером.

Модуль 215 управления буфером (фиг.5) записывает программный поток, который был считан с диска 101, в положение, представленное указателем записи данных блока 232 накопителя указателя записи данных буфера 215A, и выполняет последовательное увеличение указателя записи данных на размер записываемых данных.

Если только не будет указано другое, когда буфер 215A модуля 215 управления буфером имеет свободное пространство, модуль 213 подачи данных содержания считывает данные с диска 101, передает и сохраняет эти данные в буфере 215A модуля 215 управления буфером. Таким образом, в буфере 215A обычно сохраняется достаточное количество данных.

[Начало управления декодером]

Когда данные считывают с диска 101 и данные сохраняют в буфере 215A модуля 215 управления буфером, поток переходит с этапа S128 на этап S129. На этапе S129 модуль 214 управления декодированием управляет модулем 216 управления видеодекодером, модулем 217 управления декодером звука и модулем 218 управления декодером субтитров, для начала считывания данных из буфера 215A в качестве предварительной операции декодирования.

Таким образом, модуль 216 управления видеодекодером запрашивает данные на блоке 233 функции считывания видеоданных модуля 215 управления буфером (фиг.5). Модуль 216 управления видеодекодером получает один модуль доступа к видео данным, сохраненным в буфере 215A, PTS и DTS (иногда называются временными метками) добавлены к модулю видеодоступа, pic_struct_copy, au_ref_flag и AU_length, которые представляют собой информацию (иногда называемую дополнительной информацией), описанную в PES_packet() private_stream_2, которому непосредственно предшествует точка возможного начала декодирования, и т.д. из модуля 215 управления буфером в соответствии с запросом. Временную метку передают из модуля 216 управления видеодекодером в модуль 214 управления декодированием всякий раз, когда модуль 216 управления видеодекодером получает модуль доступа к видеоданным.

pict_struct_copy, используемый для обновления времени, передают из блока 233 функции считывания видеоданных. Вместо этого может использоваться pic_struct, содержащийся в bit stream, полученный в результате анализа.

С другой стороны, модуль 217 управления декодером звука запрашивает блок 234

функции считывания звука модуля 215 управления буфером (фиг.5) для получения данных. Модуль 217 управления декодером звука получает один модуль доступа к звуку (ATRAC), сохраненный в буфере 215A, и временную метку (PTS, DTS), добавленную к модулю доступа к звуку, из модуля 215 управления буфером, в соответствии с запросом. Временную метку передают из модуля 217 управления декодером звука в модуль 214 управления декодированием всякий раз, когда модуль 217 управления декодером звука получает модуль доступа к звуку.

Кроме того, модуль 218 управления декодером субтитров запрашивает блок 235 функции считывания субтитров из модуля 215 управления буфером (фиг.5) для получения данных. Модуль 218 управления декодером субтитров получает один модуль доступа к субтитрам, сохраненный в буфере 215A, и временную метку, добавленную к модулю доступа к субтитрам из модуля 215 управления буфером, в соответствии с запросом. Временную метку передают из модуля 218 управления декодером субтитров в модуль 214 управления декодированием всякий раз, когда модуль 218 управления декодером субтитров получает модуль доступа к субтитрам. Когда элементарный поток, предназначенный для воспроизведения, не содержит поток субтитров, или в буфере 215A не сохранен модуль доступа к субтитрам, данные не будут передаваться из модуля 215 управления буфером в модуль 218 управления декодером субтитров.

Всякий раз, когда модуль 216 управления видеодекодером, модуль 217 управления декодером звука и модуль 218 управления декодером субтитров запрашивает модуль 215 управления буфером для получения данных, они передают результаты в соответствии со своими запросами данных в модуль 214 управления декодированием.

Подробно данные, считываемые из буфера 215A, когда модуль 215 управления буфером передает данные в модуль 216 управления видеодекодером, модуль 217 управления декодером звука и модуль 218 управления декодером субтитров, будут описаны ниже.

[Начало декодирования данных]

Когда модуль 216 управления видеодекодером, модуль 217 управления декодером звука и модуль 218 управления декодером субтитров начинают считывать данные из буфера 215A модуля 215 управления буфером, поток переходит с этапа S129 на этап S130. На этапе S130 эти модули начинают декодировать данные, которые были считаны.

Другими словами, модуль 214 управления декодированием обеспечивает начало декодирования модулем 216 управления видеодекодером, модулем 217 управления декодером звука и модулем 218 управления декодером субтитров в соответствии с IN_time, которое представляет собой 180090, первого PlayItem#0, содержащегося в PlayList#0, переданного из модуля 212 управления проигрывателем на этапе S127 в соответствии с временными метками, переданными из модуля 216 управления видеодекодером, модуля 217 управления декодером звука и модуля 218 управления декодером субтитров на этапе S129 и, в случае необходимости, с измененным временем для обеспечения синхронизации.

Способ начала декодирования данных с изменением времени для обеспечения синхронизации описан, например, в японском патенте №3496725. Вкратце, минимальное значение временных меток, переданных из видеомодуля 216 управления видеодекодером, модуля 217 управления декодером звука и модуля 218 управления декодером субтитров, установлено как исходное значение времени, которое подсчитывает блок 214A подсчета времени. Блок 214A подсчета времени начинает подсчет времени от этого установленного времени. Когда время, подсчитанное блоком 214A подсчета времени, соответствует временной метке, модуль 214 управления декодированием обеспечивает начало декодирования данных этими модулями.

Модуль 216 управления видеодекодером принимает команду начала декодирования из модуля 214 управления декодированием, передает один модуль доступа к видеоданным, полученный из блока 233 функции считывания видеоданных модуля 215 управления буфером (фиг.5), в видеодекодер 116 (фиг.1) и обеспечивает декодирование

видеодекодером 116 модуля доступа к видеоданным. Кроме того, модуль 216 управления видеодекодером передает видеоданные, декодированные видеодекодером 116, в модуль 219 обработки графических изображений.

После этого модуль 216 управления видеодекодером обеспечивает последующее декодирование видеодекодером 116 модулей доступа к видеоданным, полученным из блока 233 функции считывания модуля 215 управления буфером, по одному и передает декодированный модуль доступа к видеоданным как видеоданные в модуль 219 обработки графических изображений.

В этот момент в видеодекодере 116 изменяют порядок декодированных видеоданных и выходных видеоданных. Например, как показано на фиг.35, видеоданные декодируют в порядке I1, B0, P3, B2, P5 и B4, тогда как видеоданные могут быть выведены в порядке B0, I1, B2, P3, B4 и P5. Таким образом, видеодекодер 116 содержит буфер декодированного изображения, в котором сохраняют декодированные изображения. На фиг.35 In представляет n-е изображение I, Bn представляет n-е изображение B, и Pn представляет собой n-е изображение P.

Модуль 217 управления декодером звука принимает команду начала декодирования из модуля 214 управления декодированием, передает один модуль доступа к звуку, полученный из блока 234 функции считывания звука модуля 215 управления буфером (фиг.5), в декодер 117 звука (фиг.1) и обеспечивает декодирование декодером 117 звука модуля доступа к звуку. Модуль 217 управления декодером звука передает данные звука, декодированные декодером 117 звука, в модуль 221 вывода звука.

После этого модуль 217 управления декодером звука обеспечивает последовательное декодирование декодером 117 звука модулей доступа к звуку, полученных из блока 234 функции считывания звука модуля 215 управления буфером по одному, и передает декодированные модули доступа к звуку как данные звука в модуль 221 вывода звука.

Кроме того, модуль 218 управления декодером субтитров принимает команду начала декодирования из модуля 214 управления декодированием, обеспечивает декодирование с помощью внутреннего программного обеспечения декодирования субтитров одного модуля доступа к субтитрам, полученного из блока 235 функции считывания субтитров, модуля 215 управления буфером (фиг.5) в соответствии с командой, и передает декодированный модуль доступа к субтитрам как данные субтитров (данные изображения субтитров) в модуль 219 обработки графических изображений.

После этого модуль 218 управления декодером субтитров обеспечивает последовательное декодирование с помощью внутреннего программного обеспечения декодирования модулей доступа к субтитрам, полученных из блока 235 функции считывания субтитров, модуля 215 управления буфером, по одному и передает декодированный модуль доступа к субтитрам в виде данных субтитров в модуль 219 обработки графических изображений.

[Обработка графических изображений]

После этого поток обработки переходит с этапа S130 на этап S131. На этапе S131 модуль 219 обработки графических изображений выполняет обработку графических изображений для видеоданных, переданных из модуля 216 управления видеодекодером, и, в случае необходимости, данных субтитров, передаваемых из модуля 218 управления декодером субтитров.

Другими словами, модуль 219 обработки графических изображений выполняет обработку субтитров, которая, например, увеличивает или уменьшает данные субтитров, переданные из модуля 218 управления декодером субтитров, в соответствии с командой режима отображения, полученной из модуля 212 управления проигрывателем. Когда модуль 219 обработки графических изображений не принял команду режима отображения из модуля 212 управления проигрывателем или принял от него команду, устанавливающую принятый по умолчанию режим отображения, модуль 219 обработки графических изображений сохраняет без изменений данные субтитров, полученные из модуля 218 управления декодером субтитров.

Кроме того, модуль 219 обработки графических изображений добавляет видеоданные, принятые из модуля 216 управления видеодекодером, и данные субтитров, принятые из модуля 218 управления декодером субтитров, или данные субтитров, которые были обработаны, получает выходные видеоданные, на которые были наложены данные субтитров, и передает видеоданные с наложением в модуль 220 вывода видеоизображения.

Когда модуль 219 обработки графических изображений принимает команду на отображение в виде меню информации, содержащей сообщение, код времени, номер раздела или номер индекса, из модуля 211 управления сценарием или модуля 212 управления проигрывателем, модуль 219 обработки графических изображений генерирует информацию, накладывает ее на выходные видеоданные и передает данные с наложением в модуль 220 вывода видеоданных.

[Обработка вывода]

После этапа S131 поток обработки переходит на этап S132. На этапе S132 модуль 220 вывода видеоданных последовательно сохраняет выходные видеоданные, передаваемые из модуля 219 обработки графических изображений в ППППО 220A, и последовательно выводит видеоданные, сохраненные в ППППО 220A с заданной скоростью вывода.

Когда ППППО 220A имеет достаточную емкость для сохранения данных (свободное пространство), модуль 220 вывода видеоизображения принимает выходные видеоданные из модуля 219 обработки графических изображений. Когда ППППО 220A не имеет достаточной емкости для хранения, модуль 220 вывода видеоизображения обеспечивает остановку приема модулем 219 обработки графических изображений выходных видеоданных. В результате, модуль 219 обработки графических изображений прекращает прием выходных данных. Кроме того, модуль 219 обработки графических изображений обеспечивает остановку обработки модуля 216 управления видеодекодером и модуля 218 управления декодером субтитров. Таким образом, модуль 216 управления видеодекодером и модуль 218 управления декодером субтитров останавливают свою обработку.

После того как модуль 220 вывода видеоданных обеспечивает остановку приема модулем 219 обработки графических изображений выходных видеоданных, и ППППО 220A выведет видеоданные, когда ППППО 220A имеет достаточную емкость для сохранения данных, модуль 220 вывода видеоизображения обеспечивает прием с помощью модуля 219 обработки графических изображений выходных видеоданных. Так же, как и запрос на прекращение приема выходных видеоданных, модуль 219 обработки графических изображений информирует модуль 216 управления видеодекодером и модуль 218 управления декодером субтитров об этом запросе. Таким образом, модуль 219 обработки графических изображений, модуль 216 управления видеодекодером и модуль 218 управления декодером субтитров возобновляют остановленную обработку.

С другой стороны, модуль 221 вывода звука также обеспечивает последовательное сохранение в ППППО 221A данных звука, передаваемых из модуля 217 управления декодером звука, описанным на этапе S130, и последовательный вывод данных звука с заданной выходной скоростью (частотой выборки).

Пока ППППО 221A имеет достаточную емкость для сохранения данных (пустое пространство), модуль 221 вывода звука принимает данные звука из модуля 217 управления декодером звука. Однако когда ППППО 221A не имеет достаточной емкости для сохранения данных, модуль 221 вывода звука обеспечивает остановку приема модулем 217 управления декодером звука данных звука. Таким образом, модуль 217 управления декодером звука останавливает свою обработку.

После того как модуль 221 вывода звука обеспечивает остановку приема модулем 217 управления декодером звука данных звука и ППППО 21A выведет данные звука, когда ППППО 221A имеет достаточную емкость для сохранения данных, модуль 221 вывода звука обеспечивает прием модулем 217 управления декодером звука данных звука. Таким образом, модуль 217 управления декодером звука возобновляет остановленный процесс.

Так же, как описано выше, как модуль 220 вывода видеоизображения и модуль 221

вывода звука выводят данные, декодируют элементарные потоки.

[Описание внутренней структуры видеодекодера 116]

На фиг.36 показана внутренняя структура видеодекодера 116. В этом примере видеодекодер 116 состоит из механизм 116A декодирования видеоизображения и DPB (БДИ, буфер декодированного изображения) 116B. DPB 116 B также состоит из DPB 116B-1-DPB 116B-n (ниже они совместно называются DPB 116B, если только не будет указано другое). Кроме того, как показано на фиг.37, DPB 116B состоит из буфера 301 видеоданных и буфера 302 дополнительной информации.

В механизме 116A декодирования видеоизображения используется буфер 301 видеоданных DPB 116B, в котором временно содержатся декодируемые видеоданные, и содержатся видеоданные как будущие эталонные изображения. В этот момент дополнительная информация, полученная из блока 233 функции считывания видеоизображений, и параметр (например, pic_struct), полученный путем анализа модуля доступа, записывают в буфер 302 дополнительной информации, который соответствует видео данным, сохраненным в буфере 301 видеоданных.

Хотя общие процессы и их потоки обработки, в которых устройство воспроизведения диска, показанное на фиг.1, воспроизводит данные с диска 101, показанного на фиг.33 и на фиг.34, будут описаны другие процессы или операции, которые выполняет устройство воспроизведения диска, в то время как оно воспроизводит данные с диска 101.

[Передача информации времени в модуль 214 управления декодированием]

Далее будет описано обновление тактовой частоты (блок 214A подсчета времени). Модуль 216 управления видеодекодером обеспечивает декодирование, с помощью видеодекодера 116, входного модуля доступа к видеоданным. После того как видеодекодер 116 выполнит обработку декодирования и изменения порядка, видеоданные для одного кадра (два поля) выводят в модуль 219 обработки графических изображений. Кроме того, временную метку (PTS/DTS) и информацию pic_struct видеоданных передают из модуля 216 управления видеодекодером в модуль 214 управления декодированием.

Если значение pic_struct модуля доступа равно 1 или 2, модуль доступа имеет одно поле. Таким образом, когда выводят два модуля доступа, pic_struct поступившего ранее поля и временную метку, если модуль доступа поступившего ранее поля имеет временную метку, передают из модуля 216 управления видеодекодером в модуль 214 управления декодированием так, что эти два модуля доступа обрабатываются как один. Если поступившее ранее поле не имеет временную метку, информацию, которая обозначает, что поступившее ранее поле имеет временную метку, передают в модуль 214 управления декодированием. Поскольку изолированное поле не разрешено, немедленно после поля, для которого pic_struct равно 1 или 2, размещается поле, для которого значение pic_struct равно 2 или 1, соответственно. Когда эти поля обрабатывают как одно, временная метка поступившего ранее поля используется как представительное значение.

Если pic_struct модуля доступа равно 0, 3, 4, 5 или 6, когда выводят один модуль доступа, pic_struct и временную метку, если модуль доступа ее имеет, передают из модуля 216 управления видеодекодером в модуль 214 управления декодированием. Когда модуль доступа не имеет временную метку, информацию, которая обозначает, что модуль доступа не имеет временную метку, передают в модуль 214 управления декодированием.

Модуль 214 управления декодированием обновляет блок 214A подсчета времени, используя полученную временную метку и информацию pic_struct.

Далее, со ссылкой на блок-схему последовательности операций, показанную на фиг.38, будет описан способ обновления блока 214A подсчета времени.

Модуль 214 управления декодированием определяет, имеет ли полученный модуль доступа временную метку (на этапе S141). Если модуль доступа имеет временную метку, модуль 214 управления декодированием устанавливает значение временной метки (PTS) для блока 214A подсчета времени (на этапе S142). Как описано выше, сразу же после того, как процесс декодирования начнется для модуля доступа, поскольку он обычно имеет временную метку, никакие ненормальные события не могут возникнуть в исходном

положении. Если модуль доступа не имеет временную метку, значение в соответствии с предшествующим pic_struct добавляют к текущему времени (на этапе S144). После этого текущее значение pic_struct сохраняют для следующей обработки и затем текущую обработку заканчивают (на этапе S143).

Как показано на фиг.39, значение, соответствующее сохраненному pic_struct, равно 0, 3 или 4, блок 214A подсчета времени суммирует значения времени этих двух полей. Если pic_struct равно 5 или 6, блок 214A подсчета времени суммирует значение времени трех полей. Если сохраненная pic_struct имеет значения 1 или 2, блок 214A подсчета времени суммирует значения времени двух полей.

Когда выполняется этот процесс изменения времени, значение времени блока 214A подсчета времени представляет время начала отображения модуля доступа (для одного кадра), который был выведен из модуля 216 управления видеодекодером в модуль 219 обработки графических изображений. Другими словами, если видеоданные имеют временную метку, PTS заменяют блоком 214A подсчета времени. Если видеоданные не имеют временную метку, добавляют интервал отображения непосредственно предшествующих видеоданных в порядке отображения.

В этом примере AVC используется как система кодирования видеоизображения. В системе MPEG-2 Video, например, с помощью repeat_first_field, может быть получена длительность отображения модуля доступа.

Как описано выше, в этом случае, если емкость сохранения данных в ПППО 220A не достаточна, вывод видеоданных из модуля 216 управления видеодекодером прекращается. В этом случае обновление блока 214A подсчета времени автоматически останавливается. Когда обновление видеоданных в ПППО 220A возобновляется, обновление блока 214A подсчета времени автоматически возобновляется.

Другими словами, когда режим воспроизведения изменяют на состояние паузы в соответствии с командой, переданной пользователем, поскольку обновление модуля 220 вывода видеоизображения остановлено, модуль 216 управления видеодекодером останавливает свою работу, и, таким образом, останавливается подсчет времени (блок 214A подсчета времени). Когда состояние паузы переключают в состояние нормального воспроизведения, обновление модуля вывода видеоизображения разрешается. В результате, работа модуля 216 управления видеодекодером и вывод видеоданных возобновляются. Кроме того, обновление времени (блок 214A подсчета времени) также возобновляется.

Ожидается, что такие операции выполняются при медленном воспроизведении. Другими словами, медленное воспроизведение представляет собой состояние, в котором состояние паузы и состояние нормального воспроизведения разрешены поочередно. В этой точке время (блок 214A подсчета времени) обновляют в синхронизации с выводимыми видеоданными.

В данном примере блок 214A подсчета времени обновляют синхронно с видеоданными, выводимыми модулем 216 управления видеодекодером. Однако если задержка, которая возникает в модуле 216 управления видеодекодером и после него, в данном случае в модуле 219 обработки графических изображений и в модуле 220 вывода видеоизображения, будет большой, существует вероятность, что взаимозависимость между видеоданными, переданными пользователем, и временем (блок 214A подсчета времени) будет отклоняться. В этом случае, когда обновление времени (блок 214A подсчета времени) выполняется синхронно с видеоданными, выводимыми из модуля 220 вывода видеоизображения, можно исключить отклонение взаимозависимости.

В частности, буфер 302 дополнительной информации, описанный со ссылкой на фиг.37, добавляют к блоку обработки видеоданных в модуле 219 обработки графических изображений модуля 220 вывода видеоизображения и ПППО 220A, в результате чего видеоданные и дополнительную информацию обрабатывают в паре, до тех пор, пока видеоданные не будут выведены. Кроме того, когда видеоданные выводят из модуля 220 вывода видеоизображения, соответствующую дополнительную информацию передают в

модуль 214 управления декодированием. Модуль управления декодированием обновляет время (блок 214A подсчета времени), используя описанный выше алгоритм.

В данном способе независимо от того, является ли задержка, которая возникает в видеодекодере или после него, большой или малой, видеоданные, которые отображаются, и время (блок 214A подсчета времени) могут быть синхронизированы.

В результате, даже если устройство, которое воспроизводит потоковые данные, не содержит часы, осуществляющие независимый подсчет времени, потоковые данные могут быть точно воспроизведены. Таким образом, нагрузка, прикладываемая к ЦПУ 112, которое выполняет обработку, может быть уменьшена.

[Изменение PlayItem]

Как описано на фиг.33 и на фиг.34, воспроизводится первый PlayItem#0 первого PlayList#0, показанного на фиг.29. В соответствии с PlayList#0 после воспроизведения первого PlayItem#0 воспроизводится второй PlayItem#1. Другими словами, выполняется процесс изменения Play Item, который заменяет PlayItem с PlayItem#0 на PlayItem#1.

Далее, со ссылкой на блок-схему последовательности операций, показанную на фиг.40, будет описана обработка изменения PlayItem.

Как описано на фиг.33 и фиг.34, после воспроизведения клипа первого PlayItem#0 PlayList#0, показанного на фиг.29, во время воспроизведения первого PlayItem#0 модуль 214 управления декодированием (фиг.2A и фиг.2 B) проверяет время, подсчитываемое блоком 214A подсчета времени.

[Конец воспроизведения PlayItem#0]

Когда время, которое подсчитал блок 214A подсчета времени, становится равным 27 180 090 (фиг.29), которое представляет собой OUT_time первого PlayItem#0, переданное из модуля 212 управления проигрывателем на этапе S127, показанном на фиг.34, модуль 214 управления декодированием выполняет управление отменой декодирования, для завершения воспроизведения PlayItem#0 на этапе S151.

Если блок 214A подсчета времени не работает на частоте 90 кГц, а именно время обновляют в соответствии с выходом видеоданных, время, которое подсчитывает выходной блок 214A, может неточно соответствовать OUT_time для PlayItem#0. В таком случае в момент времени, когда время OUT_time PlayItem#0 становится близким ко времени, которое подсчитывает блок 214A подсчета времени, процесс декодирования останавливают и, таким образом, воспроизведение PlayItem#0 завершают. Эта обработка будет описана ниже со ссылкой на фиг.51 и на фиг.52.

Другими словами, модуль 214 управления декодированием обеспечивает работу модуля 216 управления видеодекодером, модуля 217 управления декодером звука и модуля 218 управления декодером субтитров, так что они останавливают свою работу при декодировании. Кроме того, модуль 214 управления декодированием управляет модулем 220 вывода видеоизображения, для последовательного вывода видеоданных.

Кроме того, модуль 214 управления декодированием передает сообщение, которое обозначает, что первый PlayItem#0 был воспроизведен, в модуль 212 управления проигрывателем.

[Начало воспроизведения PlayItem#1]

Как описано выше, модуль 212 управления проигрывателем распознает, что первый PlayList#0 содержит первый PlayItem#0 и второй PlayItem#1, на этапе S105, показанном на фиг.33. Когда модуль 214 управления декодированием получает сообщение, обозначающее, что первый PlayItem#0 был воспроизведен, из модуля 212 управления проигрывателем поток переходит с этапа S151 на этап S152. На этапе S152 так же, как первый PlayItem#0, модуль 212 управления проигрывателем начинает воспроизводить второй PlayItem#1.

Так же, как и в случае, возникающем на этапе S122, показанном на фиг.34, как при обработке воспроизведения второго PlayItem#1, модуль 212 управления проигрывателем определяет один из RPN_EP_start, описанных в EP_map(), как положение начала воспроизведения второго PlayItem#1.

Кроме того, модуль 212 управления проигрывателем распознает Mark(), которые принадлежат второму PlayItem#1, описанному на этапе S124, показанном на фиг.34, и количество элементарных потоков, имеющих атрибуты, мультиплексированные с файлом потока клипа "00002.PS", воспроизводимые в соответствии с PlayItem#1, описанным на этапе S125, показанном на фиг.34, и определяет элементарный поток для воспроизведения.

Модуль 212 управления проигрывателем выполняет ту же обработку, что и на этапе S127, показанном на фиг.34.

Другими словами, модуль 212 управления проигрывателем передает RPN_EP_start EP_map(), определенный как положение начала воспроизведения, и название файла потока клипа, мультиплексированного с элементарным потоком, предназначенным для воспроизведения, а именно название файла потока клипа "00002.PS", соответствующее "00002.CLP", описанному в Clip_Information_file_name второго PlayItem#1 (фиг.29), в модуль 213 подачи данных содержания.

Кроме того, в файле потока клипа "00002.PS", мультиплексированном с элементарным потоком, предназначенным для воспроизведения, сохранен программный поток, который поступает в модуль 215 управления буфером, модуль 212 управления проигрывателем инициализирует модуль 215 управления буфером.

Другими словами, модуль 215 управления буфером (фиг.5) устанавливает то же значение для указателя начала данных, сохраненное на блоке 231 сохранения указателя начала данных, указателей записи данных, сохраненных на блоке 232 сохранения указателя записи данных, указателя считывания видеоданных, сохраненного на блоке 241 сохранения указателя считывания видеоданных, указателей считывания звука, сохраненных на блоке 251 сохранения указателя считывания звука, и указателей считывания субтитров, сохраненных на блоке 262 сохранения указателя считывания субтитров.

Кроме того, модуль 212 управления проигрывателем передает stream_id и в случае необходимости private_stream_id как информацию идентификации, которая идентифицирует элементарный поток, предназначенный для воспроизведения, в модуль 215 управления буфером.

Блок 233 функции считывания видеоданных модуля 215 управления буфером (фиг.5) принимает stream_id потока видеоданных, состоящий из элементарных потоков, предназначенных для воспроизведения, из модуля 212 управления проигрывателем, и сохраняет его в регистре 242 stream_id. Кроме того, блок 234 функции считывания звука принимает stream_id и private_stream_id потока звука элементарных потоков, предназначенных для воспроизведения, из модуля 212 управления проигрывателем, и сохраняет их в регистре 252 stream_id и в регистре 253 private_stream_id соответственно.

Поскольку файл потока клипа "00002.PS", мультиплексированный с элементарным потоком, предназначенным для воспроизведения, содержит поток субтитров, stream_id и private_stream_id потока субтитров элементарных потоков, предназначенных для воспроизведения, передают из модуля 212 управления проигрывателем на блок 235 функции считывания субтитров. Блок 235 функции считывания субтитров сохраняет stream_id и private_stream_id в регистре 263 stream_id и в регистре 264 private_stream_id соответственно.

Модуль 212 управления проигрывателем устанавливает флаг функции считывания субтитров, который имеет значение, соответствующее файлу потока клипа, мультиплексированному с элементарным потоком, предназначенным для воспроизведения, на блоке 261 сохранения флага функции считывания субтитров, для инициирования модуля 215 управления буфером (фиг.5).

В этом случае, поскольку файл потока клипа "00002.PS", мультиплексированный с элементарными потоками, предназначенными для воспроизведения, содержит поток субтитров, флаг функции считывания субтитров, значение которого установлено в 1, установлен на блоке 261 сохранения флага функции считывания субтитров, для

активирования блока 235 функции считывания субтитров.

Модуль 212 управления проигрывателем передает значение 90000 как IN_time и значение 27090000 как OUT_time второго PlayItem#1, предназначенного для воспроизведения (фиг.29) в модуль 214 управления декодированием.

Кроме того, модуль 212 управления проигрывателем инициализирует команду режима отображения потока субтитров для модуля 219 обработки графических изображений. Другими словами, модуль 212 управления проигрывателем управляет модулем 219 обработки графических изображений для отображения потока субтитров в режиме отображения, принятом по умолчанию.

Когда configurable_flag (фиг.14) потока субтитров, предназначенного для воспроизведения, установлен в 1, что обозначает, что режим отображения разрешено изменять, команда режима отображения потока субтитров, которую модуль 212 управления проигрывателем передает в модуль 219 обработки графических изображений, может поддерживаться с помощью команды текущего режима отображения.

Таким же образом, что и первый PlayItem#0, воспроизводится второй PlayItem#1. В то время как воспроизводится второй PlayItem#1, модуль 214 управления декодированием проверяет время, которое рассчитывает блок 214A подсчета времени. Когда время, которое рассчитал блок 214A подсчета времени становится равным 27090000 (фиг.29), то есть, OUT_time второго PlayItem#1, переданного из модуля 212 управления проигрывателем на этапе S152 (фиг.40), модуль 214 управления декодированием выполняет то же управление отмены декодирования, что и на этапе S151, для завершения воспроизведения PlayItem#1. Как описано выше, момент времени, который подсчитывает блок 214A подсчета времени, может не точно соответствовать OUT_time PlayItem#0. В таком случае, в момент времени, в который время OUT_time PlayItem#0 становится близким к значению времени, который подсчитал блок 214A подсчета времени, выполняется обработка отмены декодирования и, таким образом, заканчивается воспроизведение PlayItem#0. Этот процесс будет описан ниже со ссылкой на фиг.51 и фиг.52.

[Отображение кода времени]

Далее, как описано выше, на этапе S123, показанном на фиг.34, представлен код времени. Отображение кода времени последовательно обновляется.

Далее, со ссылкой на блок-схему последовательности операций, показанную на фиг.41, будет описана обработка отображения кода времени.

Когда блок 214A подсчета времени, встроенный в модуль 214 управления декодированием (фиг.2A и фиг.2B), подсчитывает одну секунду, поток обработки переходит на этап S171. На этапе S171 модуль 214 управления декодированием передает сообщение, которое обозначает, что прошла одна секунда, и текущее время, которое подсчитал блок 214A подсчета времени для модуля 212 управления проигрывателем. После этого поток обработки переходит на этап S172. На этапе S172 модуль 212 управления проигрывателем принимает сообщение и значение текущего времени из модуля 214 управления декодированием и преобразует текущее время во временной код. После этого поток переходит на этап S173.

На этапе S173 модуль 212 управления проигрывателем управляет модулем 219 обработки графических изображений для отображения кода времени, полученного на этапе S172. После этого поток обработки возвращается на этап S171.

Таким образом, код времени обновляется через интервалы одна секунда. Интервалы обновления кода времени не ограничиваются одной секундой.

[Смена потоков]

Файл потока клипа "00001.PS", воспроизводимый первым PlayItem#0, который составляет первый PlayList#0, описанный со ссылкой на фиг.29, и файл "00002.PS" потока клипа, воспроизводимый вторым PlayItem#1, мультиплексированы с двумя потоками субтитров, как описано со ссылкой на фиг.30A и фиг.30B.

Когда множество элементарных потоков, имеющих одинаковый атрибут,

мультиплексируют с файлом потока клипа, элементарные потоки, предназначенные для воспроизведения, могут быть изменены с одного элементарного потока на другой элементарный поток.

Далее, со ссылкой на блок-схему последовательности операций, показанную на фиг.42, будет описана обработка смены потока.

Когда команда смены потока описана как программа сценария, например, в файле "SCRIPT.DAT" (фиг.6), и модуль 211 управления сценарием выполняет программу сценария, или пользователь работает с пультом дистанционного управления для смены потоков, команда смены потоков поступает в модуль 212 управления проигрывателем.

Когда модуль 211 управления сценарием выполняет программу сценария, которая описывает запрос на смену потока, модуль 211 управления сценарием передает сообщение запроса на изменение потока в модуль 212 управления проигрывателем. Когда пользователь вводит команду смены потока, используя пульт дистанционного управления, входной интерфейс 115 принимает сигнал команды на смену потока из пульта дистанционного управления и передает сообщение запроса на смену потока в модуль 212 управления проигрывателем.

Когда сообщение запроса на смену потока субтитров, которое обеспечивает смену потоков субтитров модулем 212 управления проигрывателем, поступает в модуль 212 управления проигрывателем, он проверяет количество потоков субтитров для элементарных потоков, предназначенных для воспроизведения, которые были распознаны на этапе S125, показанном на фиг.34.

Когда количество потоков субтитров, которые проверил модуль 212 управления проигрывателем, равно 1 или меньше, модуль 212 управления проигрывателем игнорирует сообщение запроса на смену потока субтитров. Таким образом, модуль 212 управления проигрывателем не выполняет обработку с этапа S192 до этапа S194.

В отличие от этого, когда количество потоков субтитров равно двум или больше, поток обработки переходит на этапы S192-S194. На этих этапах модуль 212 управления проигрывателем сменяет поток субтитров, который воспроизводится, на другой поток субтитров.

Другими словами, на этапе S192 модуль 212 управления проигрывателем идентифицирует воспроизводимый поток субтитров в файле информации клипа. В частности, если предположить, что поток субтитров, stream_id которого равен 0xBD и private_stream_id равен 0x80 и который мультиплексирован с файлом потока клипа "00002.PS", воспроизводится в соответствии со вторым PlayItem#1, который составляет первый PlayList#0, описанный со ссылкой на фиг.29, модуль 212 управления проигрывателем идентифицирует поток субтитров, который воспроизводится, как stream#2, который представляет собой третий поток субтитров в файле информации клипа "00002.CLP", показанном на фиг.30A и фиг.30B, из двух потоков субтитров, мультиплексированных с файлом потока клипа "00002.PS" на этапе S192.

После этого поток обработки переходит на этап S193. На этапе S193 модуль 212 управления проигрывателем идентифицирует следующий поток субтитров файла информации клипа, идентифицированного на этапе S192, как поток субтитров, предназначенный для воспроизведения следующим. На фиг.30A и фиг.30B следующий поток субтитров третьего потока субтитров stream#2 представляет собой четвертый поток субтитров stream#3 в файле информации клипа "00002.CLP". Таким образом, на этапе S193 модуль 212 управления проигрывателем распознает четвертый поток субтитров stream#3 как поток субтитров, предназначенный для воспроизведения следующим.

Когда поток субтитров, который воспроизводится, идентифицирован как stream#3, который представляет собой четвертый поток субтитров в файле информации клипа "00002.CLP", показанном на фиг.30A и фиг.30B, из двух потоков субтитров, мультиплексированных с файлом потока клипа "00002.PS", модуль 212 управления проигрывателем распознает, например, третий поток субтитров stream#2 как поток субтитров, предназначенный для воспроизведения следующим.

После этого поток переходит на этап S194. На этапе S194 модуль 212 управления проигрывателем передает stream_id и private_stream_id субтитров, распознанные на этапе S193, как поток субтитров, предназначенный для воспроизведения следующим, на блок 235 функции считывания субтитров модуля 215 управления буфером (фиг.5) так, что блок 235 функции считывания субтитров использует stream_id и private_stream_id для считывания следующего модуля доступа субтитров из буфера 215A.

Блок 235 функции считывания субтитров модуля 215 управления буфером (фиг.5) вновь устанавливает stream_id и private_stream_id, переданные из модуля 212 управления проигрывателем на этапе S194, в регистре 263 stream_id и в регистре 264 private_stream_id соответственно. Блок 235 функции считывания субтитров считывает следующий модуль доступа к субтитрам, идентифицированный по stream_id и private_stream_id, вновь установленным в регистре 263 stream_id и в регистре 264 private_stream_id соответственно.

Аналогично описанному выше, поток субтитров, который воспроизводят, изменяют на другой поток субтитров, который будет воспроизводиться следующим.

[Обработка модуля 215 управления буфером]

Далее, со ссылкой на фиг.43 - фиг.47 будет описана обработка, выполняемая модулем 215 управления буфером (фиг.5), обработка записи данных в буфер 215A и обработка считывания данных из него.

Как описано со ссылкой на фиг.5, модуль 215 управления буфером имеет пять указателей, которые используются для считывания из буфера 215A и записи данных в него.

Другими словами, как показано на фиг.43 и фиг.44, модуль 215 управления буфером имеет указатель начала данных, сохраненный на блоке 231 сохранения указателя начала данных, указатель записи данных, сохраненный на блоке 232 сохранения указателя записи данных, блок считывания видеоданных, сохраненный на блоке 241 сохранения указателя считывания видеоданных, указатель считывания звука, сохраненный на блоке 251 сохранения указателя считывания звука, и указатель считывания субтитров, сохраненный на блоке 262 сохранения указателя считывания субтитров.

На фиг.43 и фиг.44 не представлены регистр 242 stream_id и регистр 243 au_information() блока 233 функции считывания видеоданных, показанный на фиг.5, регистр 252 stream_id и регистр 253 private_stream_id блока 234 функции считывания звука и блок 261 сохранения флага функции считывания субтитров, регистр 263 stream_id и регистр 264 private_stream_id блока 235 функции считывания субтитров.

Указатель начала данных, сохраненный на блоке 231 сохранения указателя начала данных, представляет положение самых старых данных (которые должны быть считаны и которые еще не были считаны), сохраненных в буфере 215A. Указатель записи данных, сохраненный на блоке 232 сохранения указателя записи данных, представляет положение записи данных в буфере 215A. Это положение представляет собой положение, в котором записывают самые новые данные.

Указатель считывания видеоданных, сохраненный на блоке 241 сохранения указателя считывания видеоданных, представляет положение потока видеоданных, который считывают из буфера 215A. Указатель считывания звука, сохраненный на блоке 251 сохранения указателя считывания звука, представляет положение потока звука, считываемого из буфера 215A. Указатель считывания субтитров, сохраненный на блоке 262 сохранения указателя считывания субтитров, представляет положение потока субтитров, считываемого из буфера 215A.

Как описано со ссылкой на фиг.5, указатель начала данных, указатель записи данных, указатель считывания видеоданных, указатель считывания звука и указатель считывания субтитров перемещаются в направлении по часовой стрелке внутри буфера 215A.

Как показано на фиг.44, в соответствии с данным вариантом выполнения указатель начала данных обычно обновляют таким образом, что он представляет то же положение, что и положение самых старых данных указателя считывания видеоданных, указателя считывания звука и указателя считывания субтитров. На фиг.44 указатель считывания

звука представляет положение самых старых данных в указателе считывания видеоданных, указателе считывания звука или в указателе считывания субтитров. Указатель начала данных соответствует указателю считывания звука.

В модуле 215 управления буфером, который имеет указатель начала данных, указатель записи данных, указатель считывания видеоданных, указатель считывания звука и указатель считывания субтитров, когда новые данные считывают с диска 101 и записывают в буфер 215A, указатель записи данных обновляют в направлении по часовой стрелке так, что указатель записи данных представляет положение непосредственно после вновь записанных данных.

Когда видеопоток, поток звука или поток субтитров считывают из буфера 215A, указатель считывания видеоданных, указатель считывания звука или указатель считывания субтитров обновляют в направлении по часовой стрелке, на количество данных, которое было считано. Количество данных, которое было считано, представляет собой сумму видеоданных, данных звука или данных субтитров, которые были фактически считаны, и блок данных другого потока, который вклинился в считываемые данные, обходят при их считывании.

Когда указатель считывания видеоданных, указатель считывания звука или указатель считывания субтитров обновляют, указатель начала данных обновляют так, что он представляет положение самых старших данных, представленных указателем считывания видеоданных, указателем считывания звука или указателем считывания субтитров.

Модуль 215 управления буфером управляет операцией записи данных в буфере 215A так, что указатель записи данных не выходит перед указателем начала данных.

До тех пор пока указатель записи данных не выйдет перед указателем начала данных, модуль 215 управления буфером записывает данные, считываемые с диска 101, в положение в буфере 215A, представленном указателем записи данных, и обновляет указатель записи данных. С другой стороны, если указатель записи данных выходит перед указателем начала данных, модуль 215 управления буфером обеспечивает прекращение считывания данных модулем 213 подачи данных содержания с диска 101 и прекращает запись данных в буфере 215A. В результате предотвращается переполнение буфера 215A.

Как описано выше, данные, считываемые с диска 101, записывают в буфер 215A в соответствии с взаимозависимостью положений двух указателей, указателя начала данных и указателя записи данных.

С другой стороны, модуль 215 управления буфером управляет операцией считывания данных в буфере 215A таким образом, что указатель считывания видеоданных, указатель считывания звука и указатель считывания субтитров, а также указатель начала данных не выходят перед указателем записи данных.

Другими словами, до тех пор, пока указатель считывания видеоданных, указатель считывания звука или указатель считывания субтитров не выходят перед указателем записи данных, модуль 215 управления буфером считывает данные из положения в буфере 215A, представленном указателем считывания видеоданных, указателем считывания звука или указателем считывания субтитров в соответствии с запросом, полученным из модуля 216 управления видеодекодером, модуля 217 управления декодером звука или модуля 218 управления декодером субтитров, и обновляет указатель считывания видеоданных, указатель считывания звука или указатель считывания субтитров, и, в случае необходимости, указатель начала данных. С другой стороны, если указатель считывания видеоданных, указатель считывания звука или указатель считывания субтитров выходит перед указателем записи данных, модуль 215 управления буфером обеспечивает прекращение передачи запроса модулем 216 управления видеодекодером, модулем 217 управления декодером звука или модулем 218 управления декодером субтитров, до тех пор, пока в буфере 215A не накопится достаточное количество данных. В результате предотвращается переполнение буфера 215A.

Таким образом, в буфере 215A сохраняются данные, предназначенные для подачи в модуль 216 управления видеодекодером, модуль 217 управления декодером звука и в

модуль 218 управления декодером субтитров в области (заштрихованной на фиг.43 и фиг.44) в направлении по часовой стрелке, от положения, представленного указателем начала данных, в направлении, представленном указателем записи данных. Кроме того, указатель считывания видеоданных, указатель считывания звука и указатель считывания субтитров представлены в этой области.

В описанном выше случае указатель начала данных обновляется таким образом, что он представляет положение самых старых данных, представленных указателем считывания видеоданных, указателем считывания звука или указателем считывания субтитров. В качестве альтернативы, указатель начала данных может быть обновлен таким образом, что он будет представлять положения данных, которые следуют раньше на заданное время (например, на одну секунду), чем положение самых старых данных.

Ожидается, что указатель считывания видеоданных и указатель считывания звука в указателе считывания видеоданных, указателе считывания звука и указателе считывания субтитров представляют положение самых старых данных.

Таким образом, когда указатель начала данных обновляют так, что он представляет положение данных, которые следуют раньше, например, на одну секунду, чем положение самых старых данных, которое представляет указатель считывания видеоданных или указатель считывания звука, как показано на фиг.43, данные, следующие раньше на одну секунду, чем самые старые данные, которое представляет указатель считывания видеоданных или указатель считывания звука, могут быть сохранены в буфере 215A. На фиг.43 указатель считывания звука представляет положение самых старых данных, тогда как указатель начала данных представляет положение данных, которые следуют раньше на одну секунду, чем самые старые данные.

Когда указатель начала данных обновляют так, что он представляет положение данных, которые следуют раньше на одну секунду, чем положение самых старых данных, можно улучшить отклик устройства воспроизведения с диска.

Другими словами, как показано на фиг.44, когда указатель начала данных обновляют так, что он представляет положение самых старых данных, которые представляет указатель считывания звука, если будет подана команда на специальное воспроизведение, например команда воспроизведения в обратном направлении, данные, которые были считаны из буфера 215A, потребуется снова повторно считать с диска 101. Таким образом, после подачи команды на специальное воспроизведение до тех пор, пока не будет выполнена операция специального воспроизведения, потребуется определенное время.

В отличие от этого, как показано на фиг.43, когда указатель начала данных обновляют так, что он представляет положение данных, которые следуют раньше на одну секунду, чем положение самых старых данных, которые представляет указатель считывания звука, если будет подана команда на специальное воспроизведение, например команда на воспроизведение в обратном порядке, и данные, необходимые для начала операции специального воспроизведения, следуют раньше на одну секунду, чем самые старые данные и сохраненные в буфере 215A, операция специального воспроизведения может быть быстро начата, без повторного считывания данных с диска 101.

Когда указатель начала данных обновляется так, что он представляет положение данных, которые следуют раньше на одну секунду, чем положение самых старых данных, которые представляет указатель считывания звука, данные, необходимые для начала операции специального воспроизведения, не могут быть сохранены в буфере 215A. В этом случае, данные, необходимые для начала операции специального воспроизведения, повторно считывают с диска 101.

Далее будут подробно описаны ниже операции считывания для потока видеоданных, потока звука и потока субтитров из буфера 215A.

Как описано на этапе S127, показанном на фиг.34, когда начинается операция воспроизведения файла потока клипа, модуль 215 управления буфером инициализирует указатель начала данных, указатель записи данных, указатель считывания видеоданных,

указатель считывания звука и указатель считывания субтитров так, что они представляют одно и то же положение в буфере 215A.

Когда программный поток (программный поток MPEG2-System), сохраненный в файле потока клипа, считывают с диска 101 и передают в модуль 215 управления буфером, в нем сохранен программный поток в положении, которое представляет указатель записи данных буфера 215A. Кроме того, указатель записи данных обновляется в направлении по часовой стрелке.

Кроме того, блок 233 функции считывания видеоданных модуля 215 управления буфером (фиг.5) анализирует программный поток, сохраненный в буфере 215A, выделяет модуль доступа видеоизображения из программного потока, сохраненного в буфере 215A в соответствии с запросом, принятым из модуля 216 управления видеодекодером, и передает выделенный модуль доступа к видеоизображению в модуль 216 управления видеодекодером.

Аналогично, блок 234 функции считывания звука анализирует программный поток, сохраненный в буфере 215A, выделяет модуль доступа к звуку из программного потока, сохраненного в буфере 215A в соответствии с запросом, полученным модуля 217 управления декодером звука, и передает модуль доступа к звуку в модуль 217 управления декодером звука. Блок 235 функции считывания субтитров анализирует программный поток, сохраненный в буфере 215A, выделяет модуль доступа субтитров из программного потока, сохраненного в буфере 215A в соответствии с запросом, принятым из модуля 218 управления декодером субтитров, и передает модуль доступа субтитров в модуль 218 управления декодером субтитров.

[Считывание потока видеоданных]

Далее, со ссылкой на блок-схему последовательности операций, показанную на фиг.45, будет подробно описана обработка считывания потока видеоданных для буфера 215A блоком 233 функции считывания видеоданных (фиг.5).

На этапе S211 блок 233 функции считывания видеоданных выполняет поиск программного потока, сохраненного в буфере 215A для PES_packet() private_stream_2. Другими словами, как описано со ссылкой на фиг.23, в PES_packet() private_stream_2, stream_id составляет 10111111B (=0xBF). Блок 233 функции считывания видеоданных выполняет поиск в PES_packet(), stream_id которого составляет 10111111B.

Если предположить, что элементарный поток, мультиплексированный с программным потоком, сохраненным в файле потока клипа "00001.PS", представляет собой элементарный поток, предназначенный для воспроизведения, когда программный поток считывают с диска 101 и сохраняют в буфере 215A, на этапе S122, показанном на фиг.34, сектор 305 определяется как положение начала воспроизведения с информацией о точке возможного начала декодирования, описанной в EP_map() (фиг.31) файла потока клипа "00001.PS". На этапе S128, показанном на фиг.34, обозначен сектор 305, который представляет собой точку начала воспроизведения. Блок 233 функции считывания видеоданных обеспечивает считывание с помощью операционной системы 201, программный поток из файла потока клипа "00001.PS".

Информация о точке возможного начала декодирования, описанная в EP_map() потока видеоданных, представляет положение PES_packet() private_stream_2, после которого немедленно следует реальная точка возможного начала декодирования.

Таким образом, немедленно после того, как программный поток, сохраненный в файле потока клипа "00001.PS", будет считан с диска 101 и сохранен в буфере 215A, PES_packet() private_stream_2 будет сохранен в положении, представленном указателем начала данных и указателем считывания видеоданных в буфере 215A.

Когда блок 233 функции считывания видеоданных находит PES_packet() для private_stream_2 на этапе S211, поток переходит на этап S212. На этапе S212 блок 233 функции считывания видеоданных выделяет video_stream_id из private_stream2_PES_payload() (фиг.26), который представляет собой PES_packet_data_byte для PES_packet() private_stream_2. На этапе S127, показанном на

фиг.34, блок 233 функции считывания видеоданных определяет, соответствует ли video_stream_id stream_id для воспроизводимого потока видеоданных, который сохранен в регистре 242 stream_id (фиг.5).

Когда результат, определенный на этапе S212, обозначает, что video_stream_id, описанный в private_stream2_PES_payload(), не соответствует stream_id, сохраненному в регистре 242 stream_id, а именно PES_packet() private_stream_2, найденный на этапе S211, не находится в точке возможного начала декодирования потока видеоданных, предназначенного для воспроизведения, поток обработки возвращается на этап S211. На этапе S211 блок 233 функции считывания видеоданных выполняет поиск программного потока, сохраненного в буфере 215A для PES_packet() другого private_stream_2, и повторяет ту же обработку.

В отличие от этого, когда результат, определенный на этапе S212, обозначает, что video_stream_id, описанный в private_stream2_PES_payload(), соответствует stream_id, сохраненному в регистре 242 stream_id, а именно PES_packet() для private_stream_2, найденного на этапе S211, находится в точке возможного начала декодирования потока видеоданных, предназначенного для воспроизведения, поток обработки переходит на этап S213. На этапе S213 блок 233 функции считывания видеоданных считывает au_information(), описанный в private_stream2_PES_payload() PES_packet() для private_stream_2 из буфера 215A и сохраняет au_information() в регистре 243 au_information() (фиг.5). После этого поток обработки переходит на этап S214.

На этапе S214 блок 233 функции считывания видеоданных обновляет указатель считывания видеоданных, сохраненный на блоке 231 сохранения указателя начала данных для размера PES_packet() в private_stream_2, найденного на этапе S211 (PES_packet() в private_stream_2, в котором video_stream_id (фиг.26) соответствует stream_id, сохраненному в регистре 242 stream_id (фиг.5))

Другими словами, в файле потока клипа после PES_packet() для private_stream_2 непосредственно следует видеопоток (PES_packet()), stream_id которого соответствует video_stream_id. Таким образом, на этапе S214 блок 233 функции считывания видеоданных обновляет указатель считывания видеоданных так, что он представляет положение реальной точки возможного начала декодирования потока видеоданных.

После этого поток переходит с этапа S214 на этап S215. Блок 233 функции считывания видеоданных определяет, передал ли модуль 216 управления видеодекодером запрос на считывание данных. Когда результат, определенный на этапе S215, обозначает, что блок 233 функции считывания видеоданных не передал запрос на считывание данных, поток обработки возвращается на этап S215. На этапе S215 модуль 216 управления видеодекодером повторяет ту же обработку.

В отличие от этого, когда результат, определенный на этапе S215, обозначает, что модуль 216 управления видеодекодером передал запрос на считывание данных, поток обработки переходит на этап S216. На этапе S216 блок 233 функции считывания видеоданных анализирует программный поток из положения, представленного указателем считывания видеоданных в буфере 215A, считывает данные в байтах, описанные в AU_length в au_information(), сохраненном в регистре 243 au_information(), а именно один модуль доступа к видеоданным из буфера 215A, передает эти данные в модуль 216 управления видеодекодером и обновляет указатель считывания видеоданных на размер одного модуля доступа видеоданных, который был считан из буфера 215A.

Другими словами, как описано со ссылкой на фиг.27, au_information() описывает number_of_access_unit, который представляет количество модулей доступа к видеоданным (изображениям), полученное из PES_packet() private_stream_2, содержащих au_information() в PES_packet() следующего private_stream_2.

Кроме того, как описано со ссылкой на фиг.27, au_information() описывает pic_struct_copy, au_ref_flag и AU_length как информацию о каждом из модулей доступа к видеоданным, представленным number_of_access_unit.

Как описано на фиг.27, поскольку каждый из AU_length, описанных в au_information()

в соответствии с number_of_access_unit, представляет размер каждого из модулей доступа к видеоданным, представленным number_of_access_unit из PES_packet() private_stream_2, содержащим au_information, в PES_packet() следующего private_stream_2, блок 233 функции считывания видеоданных может выделить модули доступа с AU_length без необходимости анализа потока видеоданных.

Другими словами, когда выделяют модули доступа MPEG2-Video или MPEG4-AVC, необходимо знать структуру потока видеоданных и затем ее проанализировать. Однако программный поток, сохраненный в файле потока клипа, записанном на диске 101, содержит PES_packet() private_stream_2, который описывает AU_length, который представляет размер модуля видеодоступа и после которого непосредственно следует, по меньшей мере, одна точка возможного начала декодирования потока видеоданных. Таким образом, блок 233 функции считывания видеоданных может считывать модули доступа видеоданных (видеопоток как модули видеодоступа) из буфера 215A и передавать модули видеодоступа в модуль 216 управления видеодекодером в соответствии с AU_length, описанным в PES_packet() private_stream_2, без необходимости анализа потока видеоданных.

На этапе S216, когда блок 233 функции считывания видео данных передает модули доступа к видео данным в модуль 216 управления видеодекодером, блок 233 функции считывания видеоданных также передает pic_struct_copy, au_ref_flag и AU_length, описанные в au_information(), а также временную метку (PTS, DTS), добавленную к каждому из модулей видеодоступа, как информацию о модулях доступа к видеоданным в модуль 216 управления видеодекодером.

После того как блок 233 функции считывания видеоданных считает один модуль доступа к видеоданным из буфера 215A и передаст их в модуль 216 управления видеодекодером на этапе S216, поток обработки переходит на этап S217. На этапе S217 блок 233 функции считывания видеоданных определяет, обработал ли он модули доступа, представленные number_of_access_unit для au_information() (фиг.27), сохраненном в регистре 243 au_information().

Когда результат, определенный на этапе S217, обозначает, что блок 233 функции считывания видеоданных еще не обработал модули доступа, представленные number_of_access_unit, а именно блок 233 функции считывания видеоданных еще не считал модули доступа, представленные number_of_access_unit из буфера 215A, и не передал их в модуль 216 управления видеодекодером, поток возвращается на этап S215. На этапе S215 блок 233 функции считывания видеоданных повторяет ту же обработку.

В отличие от этого, когда результат, определенный на этапе S217, обозначает, что блок 233 функции считывания видеоданных уже обработал модули доступа, представленные number_of_access_unit, а именно блок 233 функции считывания видеоданных уже считал модули доступа, представленные number_of_access_unit из буфера 215A, и передал их в модуль 216 управления видеодекодером, поток обработки возвращается на этап S211. На этапе S211 блок 233 функции считывания видеоданных выполняет поиск PES_packet() следующего private_stream_2 и повторяет ту же обработку.

[Считывание потока звука]

Далее, со ссылкой на блок-схему последовательности операций, показанную на фиг.46, будет описана обработка считывания потока звука для буфера 215A блок 234 функции считывания звука (фиг.5).

На этапе S230 блок 234 функции считывания звука определяет, представляет ли stream_id поток звука, предназначенный для воспроизведения, который был сохранен в регистре 252 stream_id (фиг.5) на этапе S127, показанном на фиг.34, PES_packet() private_stream_1.

Когда результат, определенный на этапе S230 обозначает, что stream_id, сохраненный в регистре 252 stream_id, не представляет PES_packet() private_stream_1, а именно, как описано со ссылкой на фиг.23, stream_id, сохраненный в регистре 252 stream_id, представляет собой 110xxxxxB, установленный в потоке звука, который был кодирован в

соответствии со стандарт MPEG, поток обработки переходит на этап S231. На этапе S231 блок 234 функции считывания звука выполняет поиск в программном потоке, сохраненном в буфере 215A синхронного кода, который представляет начало фрейма звука, определенного в MPEG Audio. Поскольку положение синхронного кода находится в начале фрейма звука, блок 234 функции считывания звука обновляет указатель считывания звука таким образом, что он представляет положение начала фрейма звука. После этого поток обработки переходит с этапа S231 на этап S232. На этапе S232 блок 234 функции считывания звука выполняет поиск в программном потоке, сохраненном в буфере 215A, PES_packet(), который соответствует stream_id, сохраненному в регистре 252, stream_id, который соответствует положению, представленному указателем считывания звука, и получает PES_packet(). После этого поток обработки переходит на этап S233.

На этапе S233 блок 234 функции считывания звука обновляет указатель считывания звука, сохраненный на блоке 251 сохранения указателя считывания звука, так, что указатель считывания звука представляет начало PES_packet_data_byte для PES_packet() (фиг.19A и фиг.19B - фиг.21A и фиг.21B), который был найден на этапе S232. После этого поток обработки переходит на этап S237.

На этапе S237 блок 234 функции считывания звука определяет, передал ли модуль 217 управления декодером звука запрос на получение данных. Когда результат, определенный на этапе S237, обозначает, что модуль 217 управления декодером звука не передал запрос на получение данных, поток обработки возвращается на этап S237. На этапе S237 блок 234 функции считывания звука повторяет ту же обработку.

В отличие от этого, когда результат, определенный в этапе S237, обозначает, что модуль 217 управления декодером передал запрос на получение данных, поток обработки переходит на этап S238. На этапе S238 блок 234 функции считывания звука анализирует программный поток от положения, представленного указателем считывания звука в буфере 215A, считывает один модуль доступа к звуку, имеющий заданную фиксированную длину из буфера 215A, и передает модуль доступа к звуку вместе с временной меткой (PTS, DTS), добавленной к модулю доступа к звуку, в модуль 217 управления декодером звука.

Блок 234 функции считывания звука обновляет указатель считывания звука до размера одного модуля доступа к звуку, считываемого из буфера 215A. После этого поток обработки возвращается на этап S237. На этапе S237 блок 234 функции считывания звука повторяет ту же обработку.

В отличие от этого, когда результат, определенный на этапе S230, обозначает, что stream_id, сохраненный в регистре 252 stream_id, представляет PES_packet() для private_stream_1, а именно stream_id, сохраненный в регистре 252 streamed представляет собой 10111101B (=0xBD) и представляет PES_packet() для private_stream_1, как описано со ссылкой на фиг.23, поток обработки переходит на этап S234. На этапе S234 блок 234 функции считывания звука выполняет поиск в программном потоке, сохраненном в буфере 215A PES_packet() в private_stream_1, и получает PES_packet(). Другими словами, блок 234 функции считывания звука выполняет поиск PES_packet(), stream_id которого составляет 101111101B и получает PES_packet().

Когда блок 234 функции считывания звука находит PES_packet() в private_stream_1 на этапе S234, поток обработки переходит на этап S235. На этапе S235 блок 234 функции считывания звука выделяет private_stream_id из private_stream1_PES_payload() (фиг.24), который представляет собой PES_packet_data_byte для PES_packet() private_stream_1, и определяет, соответствует ли этот private_stream_id private_stream_id потока звука, предназначенного для воспроизведения, который был сохранен в регистре 253 private_stream_id (фиг.5) на этапе S127, показанном на фиг.34.

Когда результат, определенный на этапе S235, обозначает, что private_stream_id, описанный в private_stream1_PES_payload(), не соответствует private_stream_id, сохраненному в регистре 253 private_stream_id, а именно PES_packet() private_stream_1, найденного на этапе S234, не является потоком звука, предназначенным для воспроизведения, поток обработки возвращается на этап S234. На

этапе S234 блок 234 функции считывания звука выполняет поиск в программном потоке, сохраненном в буфере 215A PES_packet() другого private_stream_1. После этого блок 234 функции считывания звука повторяет ту же обработку.

В отличие от этого, когда результат, определенный на этапе S235, обозначает, что private_stream_id, описанный в program_stream_PES_payload(), соответствует private_stream_id, сохраненному в регистре 253 private_stream_id, а именно PES_packet() private_stream_1, найденный на этапе S234, представляет собой поток звука, предназначенный для воспроизведения, поток обработки переходит на этап S236. На этапе S236 блок 234 функции считывания звука считывает AU_locator, описанный в private_stream1_PES_payload() (фиг.24) для PES_packet() private_stream_1 из буфера 215A, суммирует положение, непосредственно следующее после AU_locator, и значение, которое представляет AU_locator, и получает начальное положение модуля доступа к звуку.

Другими словами, как описано на фиг.24, AU_locator представляет начальное положение модуля доступа к звуку или модуля доступа субтитров, сохраненное в private_payload() в private_stream1_PES_payload() на основе положения, непосредственно следующего после AU_locator. Таким образом, путем суммирования значения, которое представляет AU_locator, и положения, непосредственно следующего после AU_locator, можно получить начальное положение модуля доступа к звуку.

На этапе S236 блок 234 функции считывания звука обновляет указатель считывания звука, сохраненный на блоке 251 сохранения указателя считывания звука, в результате указатель считывания звука представляет начальное положение полученного модуля доступа к звуку. После этого поток обработки переходит на этап S237.

На этапе S237 блок 234 функции считывания звука определяет, передал ли модуль 217 управления декодером звука запрос на получение данных. Когда результат, определенный на этапе S237, обозначает, что модуль 217 управления декодером звука не передал запрос на получение данных, поток обработки возвращается на этап S237. На этапе S237 блок 234 функции считывания звука повторяет ту же обработку.

В отличие от этого, когда результат, определенный на этапе S237, обозначает, что модуль 217 управления декодером звука передал запрос на получение данных, поток обработки переходит на этап S238. На этапе S238 блок 234 функции считывания звука анализирует программный поток из положения, представленного указателем считывания звука в буфере 215A, считывает один модуль доступа к звуку, имеющий заданную длину из буфера 215A, и передает модуль доступа к звуку вместе с временной меткой, добавленной к модулю доступа к звуку, в модуль 217 управления декодером звука.

Блок 234 функции считывания звука обновляет указатель считывания звука на размер одного модуля доступа к звуку, считываемого из буфера 215A. После этого поток обработки возвращается на этап S237. На этапе S237 блок 234 функции считывания звука повторяет ту же обработку.

[Считывание потока субтитров]

Далее, со ссылкой на блок-схему последовательности операций, показанную на фиг.47, будет описана обработка считывания потока субтитров из буфера 215A блоком 235 функции считывания субтитров (фиг.5).

На этапе S251 блок 235 функции считывания субтитров определяет флаг функции считывания субтитров, который был сохранен в модуле 216 управления видеодекодером на этапе S127, показанном на фиг.34. Когда результат, определенный на этапе S251, обозначает, что флаг функции считывания субтитров установлен равным 0, а именно файл потока клипа, мультиплексированный с элементарным потоком для воспроизведения, не содержит поток субтитров, и 0 был установлен на блоке 261 сохранения флага функции считывания субтитров на этапе S127, показанном на фиг.34, блок 235 функции считывания субтитров не выполняет какую-либо обработку.

В отличие от этого, когда результат, определенный на этапе S251, обозначает, что флаг функции считывания субтитров установлен в 1, а именно файл потока клипа, мультиплексированный с элементарным потоком, предназначенным для воспроизведения,

содержит поток субтитров, и 1 была установлена на блоке 261 сохранения флага функции считывания субтитров на этапе S127, показанном на фиг.34, поток обработки переходит на этап S252. На этапе S252 блок 235 функции считывания субтитров выполняет поиск в программном потоке, сохраненном в буфере 215A, PES_packet(), который соответствует stream_id потока субтитров, предназначенного для воспроизведения, который был сохранен в регистре 263 stream_id (фиг.5).

Как описано на этапе S127, показанном на фиг.34, streamed потока субтитров, предназначенный для воспроизведения, сохранен в регистре 263 stream_id (фиг.5). С другой стороны, как описано со ссылкой на фиг.23, stream_id потока субтитров равен 10111101B (=0xBD), который представляет PES_packet() для private_stream_1.

Таким образом, на этапе S252, блок 235 функции считывания субтитров выполняет поиск в программном потоке, сохраненном в буфере 215A PES_packet() для private_stream_1.

Когда блок 235 функции считывания субтитров выполняет поиск PES_packet() в private_stream_1 и получает его, поток обработки переходит на этап S253. На этапе S253 блок 235 функции считывания субтитров выделяет private_stream_id из private_stream_PES_payload() (фиг.24), который представляет собой PES_packet_data_byte для PES_packet() private_stream_1 и определяет, соответствует ли private_stream_id private_stream_id потока субтитров, предназначенного для воспроизведения, который был сохранен в регистре 264, private_stream_id (фиг.5) на этапе S127, показанном на фиг.34.

Когда результат, определенный на этапе S253, обозначает, что private_stream_id, описанный в private_stream_PES_payload(), не соответствует private_stream_id, сохраненному в регистре 264 private_stream_id, а именно PES_packet() private_stream_1, который был найден на этапе S252, не является потоком субтитров, предназначенным для воспроизведения, поток обработки возвращается на этап S252. На этапе S252 блок 235 функции считывания субтитров выполняет поиск в программном потоке, сохраненном в буфере 215A PES_packet() другого private_stream_1. После этого блок 235 функции считывания субтитров повторно выполняет ту же обработку.

В отличие от этого, когда результат, определенный на этапе S253, обозначает, что private_stream_id, описанный в private_stream1_PES_payload(), соответствует private_stream_id, сохраненному в регистре 264, private_stream_id, а именно, PES_packet() private_stream_1, который был найден на этапе S252, представляет собой поток субтитров, предназначенный для воспроизведения, поток обработки переходит на этап S254. На этапе S254 блок 235 функции считывания субтитров считывает AU_locator, описанный в private_stream1_PES_payload() (фиг.24) PES_packet() private_stream_1 из буфера 215A, суммирует положение, непосредственно следующее после AU_locator, и значение, которое представляет AU_locator, и получает начальное положение модуля доступа субтитров.

Как описано на фиг.24, AU_locator представляет начальное положение модуля доступа субтитров (или модуля доступа к звуку), сохраненного в private_payload() private_stream1_PES_payload() на основе положения, немедленно следующего после AU_locator. Таким образом, путем суммирования значения, которое представляет AU_locator, и положения, немедленно следующего после AU_locator, можно получить начальное положение модуля доступа субтитров.

Кроме того, на этапе S254 блок 235 функции считывания субтитров обновляет указатель считывания субтитров, сохраненный на блоке 262 сохранения указателя считывания субтитров, в результате чего указатель считывания субтитров представляет начальное положение полученного модуля доступа субтитров. После этого поток обработки переходит на этап S255.

На этапе S255 блок 235 функции считывания субтитров определяет, выработал ли модуль 218 управления декодером субтитров запрос на получение данных. Когда результат, определенный на этапе S255, обозначает, что блок 235 функции считывания

субтитров не выработал запрос на получение данных, поток обработки возвращается на этап S255. На этапе S255 блок 235 функции считывания субтитров повторно выполняет эту же обработку. В отличие от этого, когда результат, определенный на этапе S255, обозначает, что модуль 218 управления декодером субтитров выработал запрос на получение данных, поток обработки переходит на этап S256. На этапе S256 блок 235 функции считывания субтитров анализирует программный поток из положения, представленного указателем считывания субтитров в буфере 215A, считывает один модуль доступа субтитров с размером, описанным в начале модуля доступа субтитров, из буфера 215A и передает модуль доступа субтитров вместе с временной меткой, добавленной к модулю доступа субтитров в модуль 218 управления декодером субтитров. Как описано на фиг.2A и фиг.2B, размер модуля доступа субтитров описан в его начале. Блок 235 функции считывания субтитров считывает данные, относящиеся к размеру, из положения, представленного указателем считывания субтитров, из буфера 215A и передает модуль доступа субтитров вместе с временной меткой, добавленной к модулю доступа субтитров в модуль 218 управления декодером субтитров.

Блок 235 функции считывания субтитров обновляет указатель считывания субтитров на величину размера одного модуля доступа субтитров, считываемого из буфера 215A. После этого поток обработки возвращается на этап S255. На этапе S255 блок 235 функции считывания субтитров повторно выполняет ту же обработку.

[Обработка повторной синхронизации]

Далее будет описано управление синхронизацией видеоданных и звуковых данных, выполняемое модулем 214 управления декодированием, показанным на фиг.2A и фиг.2B.

Как описано на этапе S130, показанном на фиг.34, модуль 214 управления декодированием обеспечивает начало декодирования своих данным модулем 216 управления видеодекодером, модулем 217 управления декодером звука и модулем 218 управления декодером субтитров. В случае необходимости модуль 214 управления декодированием обеспечивает начало декодирования своих данных этими модулями в разные моменты времени для их синхронизации. Например, когда видеодекодер 116 и декодер 117 звука выполняют свою обработку декодирования, в зависимости от своих состояний, полученных в ходе работы, они могут выводить видеоданные и данные звука в разные моменты времени.

Таким образом, модуль 214 управления декодированием выполняет обработку повторной синхронизации, которая компенсирует разность выходного времени видеоданных и данных звука и обеспечивает синхронный вывод видеодекодером 116 и декодером 117 звука видеоданных и данных звука.

Далее, со ссылкой на блок-схему последовательности операций, показанную на фиг.48, будет описана обработка повторной синхронизации.

Во время обработки повторной синхронизации, на этапе S271, модуль 214 управления декодированием определяет, является ли большой разность между временной меткой модуля доступа к видеоданным, которую выводят из модуля 216 управления видеодекодером, и временной меткой модуля доступа к звуку, которую выводят из модуля 217 управления декодером звука.

Другими словами, как описано на этапе S129, показанном на фиг.34, всякий раз, когда модуль 216 управления видеодекодером получает модуль доступа к видеоданным из модуля 215 управления буфером, модуль 216 управления видеодекодером передает временную метку модуля доступа к видеоданным в модуль 214 управления декодированием. Аналогично, всякий раз, когда модуль 217 управления декодером звука получает модуль доступа к звуку из модуля 215 управления буфером, модуль 217 управления декодером звука передает временную метку модуля доступа к звуку в модуль 214 управления декодированием.

На этапе S271 модуль 214 управления декодированием сравнивает временные метки, полученные из модуля 216 управления видеодекодером, и модуля 217 управления декодером звука в заданный период времени, который рассматривается, как одинаковые

моменты времени, и определяет, является ли разность между временными метками большой.

Когда результат, определенный на этапе S271, обозначает, что разность между временной меткой модуля доступа к видеоданным, полученной из модуля 216 управления видеодекодером, и временной меткой модуля доступа к звуку, полученной из модуля 217 управления декодером звука, невелика, а именно, когда разность между временной меткой модуля доступа к видеоданным и временной меткой модуля доступа к звуку находится в пределах заданного диапазона, в пределах которого модули доступа можно рассматривать как синхронизированные, например, для двух видеокадров (приблизительно 66 миллисекунд), поток обработки возвращается на этап S271. На этапе S271 модуль 214 управления декодированием определяет разность временных меток.

В отличие от этого, когда результат, определенный на этапе S271, обозначает, что разность между временной меткой модуля доступа к видеоданным, полученной модуля 216 управления видеодекодером и временной меткой модуля доступа к звуку, полученной из модуля 217 управления декодером звука, велика, а именно эта разность выходит за пределы заданного диапазона так, что при этом модули доступа нельзя рассматривать как синхронизированные, поток обработки переходит на этап S272. На этапе S272 модуль 214 управления декодированием сравнивает временную метку модуля доступа к видеоданным, полученную из модуля 216 управления видеодекодером, и временную метку модуля доступа к звуку, полученную из модуля 217 управления декодером звука, для определения, которые из выходных видеоданных и выходных данных звука отстают от других.

Когда результат, определенный на этапе S272, обозначает, что вывод видеоданных запаздывает по сравнению с выводом данных звука, поток обработки переходит на этап S273. На этапе S273 модуль 214 управления декодированием обеспечивает остановку декодирования модулем 216 управления видеодекодером и отображения модуля доступа видеоданных, а именно пропускают обработку модуля доступа видеоданных, чтобы ускорить на один модуль обработку доступа к видеоданным. После этого поток обработки переходит на этап S274.

На этапе S274 модуль 216 управления видеодекодером принимает запрос на пропуск из модуля 214 управления декодированием и проверяет au_ref_flag (фиг.27), переданный вместе с модулем доступа к видеоданным из модуля 215 управления буфером.

Другими словами, au_information() (фиг.27), сохраненный в private_stream2_PES_payload() (фиг.26) PES_packet() для private_stream_2, содержит au_ref_flag как информацию о модуле доступа. Как описано на этапе S129, показанном на фиг.34, и на этапе S216, показанном на фиг.45, вместе с модулем доступа к видеоданным модуль 215 управления буфером передает его au_ref_flag этого в модуль 216 управления видеодекодером.

На этапе S274 модуль 216 управления видеодекодером проверяет au_ref_flag модуля доступа, переданный вместе с модулем доступа.

После этого поток обработки переходит с этапа S274 на этап S275. На этапе S275 модуль 216 управления видеодекодером определяет, что модуль доступа к видеоданным не является эталонным изображением, к которому обращаются при кодировании другого изображения, в соответствии с результатом проверки au_ref_flag модуля доступа к видеоданным, который был передан из модуля 215 управления буфером.

Как описано со ссылкой на фиг.27, au_ref_flag модуля видеодоступа обозначает, является ли модуль доступа эталонным изображением документа. Когда модуль доступа представляет собой эталонное изображение, au_ref_flag равен 1. В отличие от этого, когда модуль доступа не является эталонным изображением, au_ref_flag равен 0.

Когда результат, определенный на этапе S275, обозначает, что модуль видеодоступа, переданный из модуля 215 управления буфером, не является модулем видеодоступа, который не является эталонным изображением, а именно модуль видеодоступа, переданный из модуля 215 управления буфером, является эталонным изображением,

поток обработки переходит на этап S276. На этапе S276 модуль 216 управления видеодекодером обеспечивает нормальную обработку видеодекодером 116 модуля доступа к видеоданным. После того как модуль 216 управления видеодекодером получит следующий модуль доступа к видеоданным из модуля 215 управления буфером, поток обработки возвращается на этап S274.

В отличие от этого, когда результат, определенный на этапе S275, обозначает, что модуль видеодоступа, переданный из модуля 215 управления буфером, не является эталонным изображением, поток обработки переходит на этап S277. На этапе S277 модуль 216 управления видеодекодером обеспечивает пропуск видеодекодером обработки модуля 116 доступа к видеоданным. После того как модуль 215 управления буфером передаст следующий модуль доступа к видеоданным, поток обработки возвращается на этап S271.

Поскольку обработка одного модуля доступа к видеоданным была пропущена, их обработка продвинется вперед приблизительно на один модуль доступа к видеоданным. В результате будет ускорен вывод видеоданных, которые запаздывали по сравнению с выводом данных звука.

В отличие от этого, когда результат, определенный на этапе S272, обозначает, что вывод видеоданных не запаздывает по сравнению с выводом данных звука, а именно вывод данных звука запаздывает по сравнению с видеоданными, поток обработки переходит на этап S278. На этапе S278 модуль 214 управления декодированием выводит непрерывную выходную команду в модуль 216 управления видеодекодером для непрерывного вывода видеоданных, соответствующих модулю доступа к видеоданным, который декодируют для поддержания модуля 216 управления видеодекодером в режиме ожидания для обработки следующего модуля доступа к видеоданным. После этого поток обработки переходит на этап S279.

На этапе S279 модуль 216 управления видеодекодером получает непрерывный запрос на вывод из модуля 214 управления декодированием и постоянно выводит видеоданные модуля доступа к видеоданным, декодирование которого выполняет видеодекодер 116 в модуль 219 обработки графических изображений в соответствии с непрерывным запрос на вывод. После того как модуль 215 управления буфером передаст следующий модуль доступа к видеоданным, поток обработки переходит на этап S271.

Как описано выше, модуль 214 управления декодированием определяет, запаздывает ли вывод видеоданных по сравнению с выводом данных звука. Когда вывод видеоданных запаздывает относительно вывода данных звука, модуль 214 управления декодированием обеспечивает пропуск обработки модулем 216 управления видеодекодером одного модуля доступа. Модуль 216 управления видеодекодером определяет, является ли модуль доступа, который будет пропущен, эталонным изображением или не является эталонным изображением в соответствии с au_ref_flag модуля доступа. Когда модуль доступа не является эталонным изображением, модуль 214 управления декодированием обеспечивает пропуск обработки видеодекодером 116 этого модуля доступа. Таким образом, вывод видеоданных и выход данных звука можно легко синхронизировать.

Другими словами, когда модуль доступа, который должен быть пропущен, представляет собой эталонное изображение, видеоданные этого модуля доступа должны быть декодированы так, чтобы на эти видеоданные мог ссылаться другой модуль доступа, когда он будет декодирован. Таким образом, при управлении синхронизацией, при котором вывод видеоданных и вывод данных звука синхронизированы, если будет пропущена обработка модуля доступа эталонного изображения, другой модуль доступа, который ссылается на это эталонное изображение, нельзя будет декодировать. В результате, когда отображают видеоданные, синхронизированные с данными звука, появляются шумы.

Таким образом, предпочтительно, чтобы модуль доступа, который не является эталонным изображением, а именно является не эталонным изображением, был пропущен.

С другой стороны, при выполнении поиска в обычном элементарном потоке модуля доступа, который не является эталонным изображением, такой элементарный поток должен быть проанализирован. Элементарный поток, кодированный в соответствии с,

например, системой MPEG4-AVC, является очень сложным. Таким образом, когда выполняет анализ такого элементарного потока, он требует значительных затрат.

В отличие от этого программный поток, сохраненный в файле потока клипа, записанном на диск 101, мультиплексирован с PES_packet() для private_stream_2, который содержит private_stream2_PES_payload() (фиг.26), который представляет собой расширение PES_packet_data_byte, помимо PES_packet() (фиг.19A и фиг.19B - фиг.21A и фиг.21B), имеющий PES_packet_data_byte, который содержит модуль доступа к видеоданным. au_information() (фиг.27) для private_stream2_PES_payload() описывает au_ref_flag, который обозначает, является ли модуль доступа к видеоданным эталонным изображением или не является эталонным изображением. au_ref_flag поступает вместе с соответствующим модулем доступа к видеоданным из модуля 215 управления буфером в модуль 216 управления видеодекодером. Таким образом, модуль 216 управления видеодекодером может определять, является ли модуль видеодоступа эталонным изображением или не является эталонным изображением, путем проверки au_ref_flag модуля видеодоступа, практически без каких-либо дополнительных затрат.

[Обработка метки]

Далее, со ссылкой на блок-схему последовательности операций, показанную на фиг.49, будет описана обработка метки на основе Mark(), описанной в PlayListMark() (фиг.9).

Модуль 214 управления декодированием обычно проверяет текущее время, подсчитываемое встроенным блоком 214A подсчета времени. На этапе S301 модуль 214 управления декодированием определяет, соответствует ли текущее время mark_time_stamp какой-либо Mark(), описанной в PlayListMark() (фиг.9).

Как описано на этапе S124, показанном на фиг.34, когда модуль 212 управления проигрывателем воспроизводит первый PlayItem#0 первого PlayList#0, показанного на фиг.29, модуль 212 управления проигрывателем распознает, что четыре Mark(), которые представляют собой с первой по четвертую Mark() из семи Mark(), содержащихся в PlayListMark(), в верхней таблице, показанной на фиг.32, принадлежат первому PlayItem#0 PlayList#0, и передает значения {180090}, {5580090}, {10980090} и {16380090}, которые представляют собой mark_time_stamp четырех Mark(), вместе с информацией, которая обозначает, что атрибут моментов времени, содержащихся в mark_time_stamp, представляет "обработку метки" для модуля 214 управления декодированием.

На этапе S301 модуль 214 управления декодированием определяет, какое из четырех значений времени (mark_time_stamp), имеющих атрибут "обработка метки", который был передан из модуля 212 управления проигрывателем, соответствует текущему времени.

Когда результат, определенный на этапе S301, обозначает, что текущее время не соответствует ни одному из значений времени, имеющих атрибут "обработка метки", поток обработки возвращается на этап S301. На этапе S301 модуль 214 управления декодированием повторно выполняет ту же обработку.

[Определение соответствия при обработке метки]

При обработке метки, на этапе S301, модуль 214 управления декодированием определяет, соответствует ли текущее время одной из mark_time_stamp. Однако в данном варианте выполнения, поскольку блок 214A подсчета времени обозначает дискретные значения, в случае, если их соответствие определяется простым способом, может возникнуть проблема.

Такая проблема будет описана на простом примере со ссылкой на фиг.50. В верхней части фиг.50, I0, P1, P2 и P3 представляют модули доступа к видеоданным. Предполагается, что pic_struct каждого из этих модулей доступа к видеоданным равняется 3, а именно длительность отображения одному кадру (3003 при 90 кГц). В данном примере предполагается, что порядок их декодирования совпадает с порядком их отображения, а именно не выполняется изменение порядка. 10 представляет собой модуль доступа, зарегистрированный в EP_map(), как описано в разделе "Обработка при подготовке к воспроизведению". Модуль доступа 10 имеет временную метку PTS=180090. В

отличие от этого, модули доступа P1, P2 и P3 не имеют временной метки.

Когда выполняется обработка таких видеоданных, часы блока 214A подсчета времени обновляют, как показано на нижнем блоке на фиг.50, Когда выводят 10, подают PTS и pic_struct для 10. Поскольку 10 имеет PTS, его подставляют на блоке 214A подсчета времени. В результате PTS блока 214A подсчета времени становится равным 180090. Когда выводят P1, поскольку он не имеет PTS, подают только pic_struct для этого P1. Поскольку pic_struct 10 равен 3, время для одного кадра (3003 при 90 кГц) добавляют к блоку 214A подсчета времени. Таким образом, значение блока 214A подсчета времени становится равным 183093. Аналогично, когда выводят P2, поскольку pic_struct P1 равно 3,3003, добавляют к времени блока 214A подсчета времени. В результате значение блока 214A подсчета времени становится равным 186096. Когда выводят P3, аналогично 3003 добавляют к значению времени блока 214A подсчета времени. В результате значение блока 214A подсчета времени становится равным 189099.

Теперь рассмотрим обработку, которая выполнятся, когда mark_time_stamp одной из меток, зарегистрированных в PlayListMark() (фиг.9 и фиг.32), равно 186000. Как описано выше, значение, которое выводят часы (блок 214A подсчета времени), равно 180090, 183093, 186096, 189099, значение 186000, которое соответствует времени метки, не выводится. Таким образом, когда mark_time_stamp и значение времени сравнивают простым способов, а именно определяют, равно ли значение их разности 0 или нет, возникает проблема.

Таким образом, заданное правило применяют при определении соответствия времени. Другими словами, в данном примере, когда mark_time_stamp конкретного события содержится в длительности отображения конкретного изображения, определяют, что конкретное событие происходит в момент времени начала отображения соответствующего изображения. В предыдущем примере mark_time_stamp=186000, содержится в длительности отображения изображения P1. Таким образом, определяется, что это событие происходит в момент времени начала отображения P1, а именно 183093.

Далее будет описан алгоритм определения модуля 214 управления декодированием, который выполняет определение в соответствии с приведенным выше определением.

В этом примере время (блок 214A подсчета времени) обновляют только, когда обновляют видеоданные. Другими словами, обработка, выполняемая на этапе S301, показанном на фиг.49, выполняется только, когда обновляют время. Таким образом, в устройстве воспроизведения, состоящем из программных средств, поскольку количество этапов обработки может быть значительно уменьшено, такая структура является предпочтительной.

Когда обновляют время, вызывают обработку, показанную на фиг.49. На этапе S301 модуль 214 управления декодированием проверяет, произошло ли событие, которое определено как событие, которое должно соответствовать текущему времени. Другими словами, модуль 214 управления декодированием проверяет, произошло ли событие, содержащееся в длительности отображения изображения, которое отображают на основе текущего времени и длительности отображения отображаемого изображения. Когда отсутствует событие, определенное как событие, которое должно соответствовать текущему времени, поток обработки возвращается на этап S301. Когда отсутствует событие, определенное как событие, которое должно соответствовать текущему времени, поток обработки переходит на этап S302. Когда отсутствует событие, определенное как событие, которое должно соответствовать текущему времени, обработка может быть закончена.

В частности, хотя, например, отображается 10, поскольку время равно 180 090 и pic_struct для 10 равно 3, понятно, что длительность отображения I0 составит 3003. Таким образом, выполняют поиск mark_time_stamp, который удовлетворяет условию 180090 < mark_time_stamp < 180090 + 3003. В этот момент, поскольку время события 186 000, приведенное в качестве примера в данном случае, не удовлетворяет этому выражению, не будет получено определение соответствия времени.

Пока выполняется отображение I1, поскольку время 183093 и pic_struct для II равно 3, понятно, что длительность отображения 10 составляет 3003. Таким образом, выполняется поиск mark_time_stamp, который удовлетворяет условию 183093 ≤ mark_time_stamp < 183093 + 3003. Поскольку время 186000 события, рассматриваемого как пример, в этот момент удовлетворяет такому выражению, определяется, что эти значения времени соответствуют. После этого выполняется процесс обработки после этапа S302.

В приведенном выше описании в качестве примера было представлено одно определение соответствия времени. Вместо этого можно применять другое определение. Например, когда mark_time_stamp конкретного события равно или больше, чем "время начала отображения конкретного изображения - α", и меньше, чем "время начала отображения следующего отображаемого изображения - α", можно определить, что событие произошло в момент начала отображения соответствующего изображения. Вместо этого в соответствии с тем же критерием определения можно определить, что время, в которое происходит событие, представляет собой, например, "время начала отображения - α соответствующего изображения".

Когда вводят такое определение, не обязательно знать время потока видеоданных, когда время метки, а именно mark_time_stamp, установлено. Таким образом, когда выполняется операция авторской разработки мультимедийного продукта, поскольку обработка кодирования видеоизображения становится строго зависимой от создания базы данных, их можно выполнять отдельно друг от друга.

В отличие от этого, когда результат, определенный на этапе S301, обозначает, что текущее значение времени соответствует одному из этих четырех значений времени, имеющих атрибут "обработка метки", модуль 214 управления декодированием передает сообщение, которое обозначает, что текущее время становится временем, имеющим атрибут "обработка метки", вместе с соответствующим временем, имеющим атрибут "обработка метки", в модуль 212 управления проигрывателем. После этого поток обработки переходит на этап S302.

На этапе S302 модуль 212 управления проигрывателем принимает сообщение, которое обозначает, что текущее время становится временем, имеющим атрибут "обработки метки", вместе с соответствующим временем, которое имеет атрибут "обработка метки", из модуля 214 управления декодированием, и распознает Mark(), для которой mark_time_stamp соответствует текущему времени, как Mark(), которая должна быть обработана для обработки метки (ниже такая Mark() иногда называется целевой меткой).

Другими словами, модуль 212 управления проигрывателем распознает PlayItem() для воспроизводимого PlayList(). Путем обращения к PlayListMark() (фиг.9) файла "PLAYLIST.DAT" (фиг.7) с PlayList(), PlayItem() и временем (mark_time_stamp), которое имеет атрибут "обработка метки" (ниже это время иногда называется временем метки), которое соответствует текущему времени и передано из модуля 214 управления декодированием, модуль 212 управления проигрывателем распознает целевую метку.

В частности, если предположить, что воспроизводится первый PlayItem#0 для первого PlayList#0, показанного на фиг.29, модуль 212 управления проигрывателем распознает, что время метки представляет собой mark_time_stamp любой одной из четырех Mark(), которая представляет собой с первой по четвертую Mark() из семи меток Mark(), содержащихся в PlayListMark(), в верхней таблице, показанной на фиг.32.

Когда время метки, переданное из модуля 214 управления декодированием в модуль 212 управления проигрывателем, например, составляет 16380090, модуль 212 управления проигрывателем распознает в качестве целевой метки четвертую Mark(), mark_time_stamp которой соответствует 16380090, которое представляет собой время метки, из четырех Mark(), которые представляют собой с первой по четвертую Mark(), содержащиеся в PlayListMark() в верхней таблице, показанной на фиг.32.

Когда модуль 212 управления проигрывателем распознает целевую метку, поток обработки переходит с этапа S302 на этап S303. На этапе S303 модуль 212 управления проигрывателем определяет, описывает ли целевая метка entry_ES_stream_id и entry

JES_private_stream_id (фиг.9), которые идентифицируют элементарный поток.

Когда результат, определенный на этапе S303, обозначает, что целевая метка не описывает entry_ES_stream_id и entry_ES_private_stream_id (фиг.9), которые идентифицируют элементарный поток, а именно оба entry_ES_stream_id и entry_ES_private_stream_id равны 0x00, поток обработки переходит на этап S305, пропуская этап S304. На этапе S305 модуль 214 управления декодированием выполняет обработку целевой метки.

В отличие от этого, когда результат, определенный на этапе S303, обозначает, что целевая метка описывает entry_ES_stream_id и entry_ES_private_stream_id (фиг.9), которые идентифицируют элементарный поток, поток обработки переходит на этап S304. На этапе S304 модуль 212 управления проигрывателем определяет, содержит ли воспроизводимый элементарный поток элементарный поток, идентифицированный по entry_ES_stream_id и, в случае необходимости, entry_ES_private_stream_id.

Когда результат, определенный на этапе S304, обозначает, что элементарный поток, который воспроизводится, не содержит элементарный поток, идентифицированный по entry_ES_stream_id и entry_ES_private_stream_id целевой метки, поток обработки возвращается на этап S301. Другими словами, когда элементарный поток, идентифицированный по entry_ES_stream_id и entry_ES_private_stream_id целевой метки, не воспроизводится, целевая метка игнорируется.

В отличие от этого, когда результат, определенный на этапе S304, обозначает, что элементарный поток, который воспроизводится, содержит элементарный поток, идентифицированный по entry_ES_stream_id и entry_ES_private_stream_id целевой метки, а именно воспроизводится элементарный поток, идентифицированный по entry_ES_stream_id и entry_ES_private_stream_id целевой метки, определяют, что целевая метка является достоверной. После этого поток обработки переходит на этап S305. На этапе S305 модуль 212 управления проигрывателем выполняет обработку целевой метки.

Другими словами, на этапе S305, благодаря обращению к mark_type целевой метки (фиг.9), модуль 212 управления проигрывателем определяет целевую метку.

Когда результат, определенный на этапе S305, обозначает, что целевая метка представляет собой метку раздела или метку индекса, а именно mark_type целевой метки представляет собой "Раздел" или "Индекс", поток обработки переходит на этап S306. На этапе S306 модуль 212 управления проигрывателем обеспечивает обновление с помощью модуля 219 обработки графических изображений, номер раздела или номер индекса, используя номер целевой метки. После этого поток обработки возвращается на этап S301.

Когда результат, определенный на этапе S305, обозначает, что целевая метка представляет собой метку события, а именно mark_type целевой метки, представляет собой "Событие", поток обработки переходит на этап S307. На этапе S307 модуль 212 управления проигрывателем передает как сообщение о событии, которое обозначает, что событие произошло, так и mark_data целевой метки в модуль 211 управления сценарием. После этого поток обработки переходит на этап S308.

На этапе S308 модуль 211 управления сценарием принимает сообщение о событии и mark_data из модуля 212 управления проигрывателем и выполняет последовательность процессов, описанных в файле "SCRIPT.DAT" с аргументом mark_data в соответствии с сообщением события, в качестве запроса на прерывание. После этого поток обработки возвращается на этап S301.

Другими словами, модуль 211 управления сценарием выполняет обработку, соответствующую mark_data.

В частности, в PlayListMark() PlayList#1 в нижней таблице, показанной на фиг.32, mark_type каждой из второй Mark() (Mark#1) и третьей Mark() (Mark#2), представляют собой "Событие". Однако mark_data Mark#1 равно 1, тогда как mark_data Mark#2 равно 2.

Когда модуль 211 управления сценарием принимает сообщение о событии, соответствующее второй Mark(), и сообщение о событии, соответствующее третьей Mark(),

модуль 211 управления сценарием выполняет обработку в соответствии с принятыми сообщениями о событиях, используя один и тот же обработчик события (процедура обработки прерывания). Модуль 211 управления сценарием проверяет mark_data, переданный вместе с сообщением о событии, и выполняет обработку в соответствии с mark_data, используя обработчик события.

В частности, когда mark_data равно, например, 1, модуль 211 управления сценарием управляет модулем 219 обработки графических изображений для отображения пиктограммы первого типа. Когда mark_data представляет собой, например, 2, модуль 211 управления сценарием управляет модулем 219 обработки графических изображений для отображения пиктограммы второго типа.

Значение mark_data не ограничивается 1 и 2. Кроме того, обработка, соответствующая mark_data, не ограничивается отображением простых пиктограмм.

Другими словами, когда mark_data находится в диапазоне от 3 до 18, модуль 211 управления сценарием управляет модулем 219 обработки графических изображений для отображения пиктограммы первого типа с интенсивностью, соответствующей значению, в котором 2 вычтено из mark_data (цифровое значение в диапазоне от 1 до 16). С другой стороны, когда mark_data находится в диапазоне от 19 до 34, модуль 211 управления сценарием управляет модулем 219 обработки графических изображений для отображения пиктограммы второго типа с интенсивностью, соответствующей значению, в котором 18 вычтено из mark_data (цифровое значение в диапазоне от 1 до 16).

Когда контроллер, которым управляет пользователь, подключен к входному интерфейсу 115 (фиг.1), и контроллер имеет механизм вибрации, который представляет собой двигатель постоянного тока (ПТ, DC) с эксцентричным грузом, установленным на валу двигателя, и который вибрирует, когда включают двигатель, если значение mark_data находится в диапазоне от 35 до 42, двигатель вибратора может быть включен на период времени работы, соответствующий значению, в котором 34 вычтено из значения mark_data (цифровое значение в диапазоне от 1 до 8).

mark_data представляет собой цифровое значение. Использование и алгоритм mark_data могут быть описаны с помощью программы сценария, которую выполняет модуль 211 управления сценарием. Таким образом, mark_data можно использовать в соответствии с заданным правилом или оригинальным правилом, обозначенным производителем диска 101 или провайдером содержания, который предоставил данные, записанные на диске 101.

Когда текущее время соответствует времени, имеющему атрибут "обработка метки", целевая метка распознается по времени метки, которая представляет собой время, имеющее атрибут "обработка метки". Когда целевая метка не описывает entry_ES_stream_id и entry_ES_private_stream_id, которые идентифицируют элементарный поток, выполняется обработка в соответствии с mark_type целевой метки. Даже если целевая метка описывает entry_ES_stream_id и entry_ES_private_stream_id, которые идентифицируют элементарный поток, в то время как воспроизводится элементарный поток, идентифицированный по entry_ES_stream_id и entry_ES_private_stream_id, выполняется обработка в соответствии с mark_type целевой метки.

В то время как воспроизводится второй PlayList#1, показанный на фиг.29, выполняется следующая обработка метки.

Другими словами, как показано в нижней таблице, представленной на фиг.32, PlayListMark() в соответствии со вторым PlayList#1, описывает первую Mark() (Mark#0), вторую Mark() (Mark#1) и третью Mark() (Mark#2), для которых значения mark_time_stamp равны 90000, 27090000 и 27540000 соответственно.

Кроме того, поскольку entry_ES_stream_id второй Mark() и третьей Mark() PlayListMark() в нижней таблице, показанной на фиг.32, описывают 0xE0 и 0xE1, вторая Mark() и третья Mark() скоррелированы с элементарными потоками, идентифицированными stream_id, которые составляют 0xE0 и 0xE1 соответственно.

Как описано со ссылкой на фиг.29, второй PlayList#1 описывает только один

PlayItem() (PlayItem#0). В соответствии с PlayItem#0 воспроизводится файл потока клипа "00003.PS". Как описано в файле информации клипа "00003.CLP", показанном на фиг.30A и фиг.30B, который соответствует файлу потока клипа "00003.PS", файл поток клипа "00003.PS" мультиплексирован с тремя элементарными потоками, которые представляют собой видеопоток stream#0, идентифицированный по stream_id, который равен 0xE0, видеопоток stream#1, идентифицированный по stream_id, который равен 0xE1, и поток звука stream#2, идентифицированный по private_stream_id, который равен 0x00.

Таким образом, вторая Mark() в PlayListMark() в нижней таблице, показанной на фиг.32, скоррелирована с файлом потока видеоданных stream#0, stream_id которого равен 0xE0, который мультиплексирован с файлом потока клипа "00003.PS". Третья Mark() скоррелирована с видеопотоком stream#1, в котором stream_id равен 0xE1, который мультиплексирован с файлом потока клипа "00003.PS".

Когда PlayItem#0 второго PlayList#1, показанный на фиг.29, воспроизводят, как описано на этапе S124, показанном на фиг.34, модуль 212 управления проигрывателем распознает, что три Mark(), содержащиеся в PlayListMark() в нижней таблице, показанной на фиг.32, принадлежат PlayItem#0 PlayList#1, и передает значения {90000}, {27090000} и {27540000}, которые представляют собой mark_time_stamp для трех Mark(), вместе с информацией, которая обозначает, что эти значения времени имеют атрибут "обработка метки" в модуль 214 управления декодированием.

При обработке метки, в то время как воспроизводят PlayItem#0 PlayList#1, модуль 214 управления декодированием обычно определяет, какое из значений времени {90000}, {27090000} и {27540000} соответствуют текущему времени, подсчитанному блоком 214A подсчета времени (на этапе S301). Когда текущее время соответствует времени, имеющему атрибут "обработка метки", модуль 214 управления декодированием передает время метки, которое представляет собой время, имеющее атрибут "обработка метки", вместе с сообщением, которое обозначает, что текущее время становится временем, имеющим атрибут "обработка метки", в модуль 212 управления проигрывателем.

Когда текущее время соответствует 27090000 среди значений времени {90000}, {27090000} и {27540000}, имеющих атрибут "обработка метки", модуль 214 управления декодированием передает время метки, имеющее атрибут "обработка метки", 27090000, вместе с сообщением, которое обозначает, что текущее время становится временем, имеющим атрибут "обработка метки" в модуль 212 управления проигрывателем.

Модуль 212 управления проигрывателем распознает, что воспроизводится PlayItem#0 из PlayList#1. Модуль 212 управления проигрывателем сравнивает значения 90000, 27090000 и 27540000, которые представляют собой mark_time_stamp трех Mark(), принадлежащих PlayItem#0 Mark(), описанной в PlayListMark() в нижней таблице, показанной на фиг.32 со значением 27090000, которое представляет собой время метки, переданное из модуля 214 управления декодированием, и распознает, что Mark(), mark_time_stamp которой соответствует 27090000, которое представляет собой время метки, а именно второй Mark() (Mark#1), описанной в PlayListMark() в нижней таблице, показанной на фиг.32, как целевую метку (на этапе S302).

Во второй Mark(), которая представляет собой целевую метку, описанную в PlayListMark(), в нижней таблице, показанной на фиг.32, entry_ES_stream_id равен 0xE0. Как описано выше, entry_ES_stream_id, который равен 0xE0, представляет собой видеопоток stream#0 (фиг.30A и фиг.30B), stream_id которого равен 0xE0, мультиплексированный с файлом потока клипа "00003.PS". Модуль 212 управления проигрывателем определяет, содержит ли элементарный поток, который воспроизводится, поток видеоданных stream#0 (на этапах S303 и S304).

Когда элементарный поток, который воспроизводится, не содержит видеопоток stream#0, модуль 212 управления проигрывателем игнорирует целевую метку (на этапе S304).

В отличие от этого, когда элементарный поток, который воспроизводят, содержит видеопоток stream#0, модуль 212 управления проигрывателем обрабатывает целевую метку так, чтобы она была достоверной, и выполняет обработку в соответствии с целевой

меткой (на этапах S305-S308).

В этом случае mark_type второй Mark(), которая представляет собой целевую метку, описанную в PlayListMark(), в нижней таблице, показанной на фиг.32, представляет собой "Событие". Таким образом, вторая Mark() представляет собой метку события. Модуль 212 управления проигрывателем передает сообщение о событии, которое обозначает, что событие произошло, и mark_data целевой метки в модуль 211 управления сценарием (на этапах S305 и S307). Модуль 211 управления сценарием выполняет последовательность процессов, описанных в "SCRIPT.DAT", с аргументом mark_data, в соответствии с сообщением о событии, принятым из модуля 212 управления проигрывателем, как запрос на прерывание (на этапе S308).

Как описано выше, при обработке метки модуль управления проигрывателем определяет, является ли текущее время временем воспроизведения файла потока клипа, воспроизводимого в соответствии с mark_time_stamp, которая представляет одно значение времени воспроизведения на временной оси PlayList(),mark_type, который представляет тип Mark(), и PlayList() (фиг.7), который содержит PlayListMark() (фиг.9), который не имеет Mark() или более чем одну Mark(), которая содержит mark_data как аргумент того, что метка события соответствует mark_time_stamp. Когда текущее время соответствует mark_time_stamp, модуль 212 управления проигрывателем распознает Mark(), которая имеет mark_time_stamp, равную времени метки, которое представляет собой текущее время, в качестве целевой метки. Когда mark_type целевой метки представляет тип, в котором происходит событие, а именно целевая метка представляет собой метку события, передают mark_type целевой метки и сообщение о событии. Модуль 212 управления проигрывателем выполняет обработку в соответствии с mark_data. Таким образом, обработка в соответствии с mark_data может быть выполнена в соответствии со временем воспроизведения файла потока клипа.

[Определение соответствия при обработке out_time]

Как описано выше, когда время, которое подсчитывает блок 214A подсчета времени, становится равным OUT_time для Play Item, передаваемого модулем 212 управления проигрывателем, модуль 214 управления декодированием отменяет обработку декодирования и заканчивает воспроизведение PlayItem. В данном варианте выполнения завершение PlayItem#0 описано на этапе S151 блок-схемы последовательности операций, показанной на фиг.40.

В этом случае, когда время и OUT_time просто сравнивают при определении соответствия, существует вероятность возникновения проблемы. Таким образом, когда сравнивают время и OUT_time, используется определение, соответствующее описанному выше определению соответствия.

Другими словами, как показано на фиг.51, в случае, когда OUT_time PlayItem соответствующего playListEnd, меньше, чем PET FoCFP (кадр или взаимодополняющая пара полей потока видеоданных, которые воспроизводятся в данный момент времени), отображаемый в конце playList, когда возникает событие PlayListEnd в момент времени начала отображения (PTS) FoCFP, длительность отображения которого содержит OUT_time, соответствующий времени PlayListEnd, а именно $PTS_{FOCFP\ [3]} \leq OUT_time < PTS_{FOCFP\ [3]}$, происходит событие PlayListEnd в момент времени начала отображения $PTS_{FOCFP\ [3]}$ для FoCFP [3]. В данном примере $PET_{FOCFP\ [k]}$ представляет собой "время, длительность отображения которого, основанную на pic_struct, добавляют к $PTS_{FOCFP[k]}$".

Таким образом, поскольку определение соответствия времени выполняют только, когда выводят видеопоток, облегчается нагрузка, связанная с обработкой. Кроме того, как описано выше, подготовка потока видеоданных становится строго независимой от подготовки базы данных.

Кроме того, модуль 214 управления декодированием информирует модуль 212 управления проигрывателем о завершении воспроизведения PlayItem. Когда модуль 212 управления проигрывателем определяет, что PlayItem представляет собой последний

PlayItem из PlayList, модуль 212 управления проигрывателем обеспечивает генерирование модулем 211 управления сценарием события playListEnd.

Когда модуль 211 управления сценарием принимает событие playListEnd, модуль 211 управления сценарием получает информацию, что воспроизведение PlayList, в отношении которого поступила команда, завершено, и продолжает выполнять запрограммированную операцию. Другими словами, модуль 211 управления сценарием, например, воспроизводит другой playList, отображает меню или завершает работу.

В случае, показанном на фиг.52, когда OUT_time равно времени окончания отображения последнего изображения в PlayItem, такой случай может оказаться не пригодным для обработки в описанном выше определении соответствия. На фиг.52, например, когда отображается FoCFP [2] и установлен режим паузы, если вызывают playStop(), FoCFP [3] отображают в режиме паузы. После этого, если снова вызывают playStop(), отображаемое изображение не изменяется, но происходит playListEnd.

Другими словами, из-за начального времени отображения последнего изображения + длительность на основе pic_struct = OUT_time, не удовлетворяется взаимозависимость начального времени отображения последнего изображения + длительность на основе pic_struct<OUT_time.

В этом случае, после того как модуль 216 управления видеодекодером выводит последнее изображение и затем проходит длительность отображения изображения, модуль 216 управления видеодекодером передает информацию, которая представляет окончание отображения, в модуль 214 управления декодированием. Таким образом, часы устанавливаются с опережением на "время начала отображения последнего изображения + длительность на основе pic_struct". Таким образом, может удовлетворяться условие соответствия.

[Декодирование субтитров]

Всякий раз, когда модуль 218 управления декодером субтитров получает один модуль доступа субтитров, сохраненный в буфере 215A, и временную метку, добавленную к нему, из блока 235 функции считывания субтитров модуля 215 управления буфером (фиг.5), модуль 218 управления декодером субтитров обеспечивает декодирование внутренним программным обеспечением декодера субтитров модуля доступа к субтитрам. Кроме того, модуль 218 управления декодером субтитров передает временную метку и длительность в модуль 214 управления декодированием,

Когда модуль 214 управления декодированием изменяет время часов (блок 214A подсчета времени), используя информацию, переданную из модуля 216 управления видеодекодером, модуль 214 управления декодированием проверяет PTS модуля доступа к субтитрам, переданный из модуля 216 управления видеодекодером. Другими словами, когда модуль 214 управления декодированием определил, что PTS модуля доступа к субтитрам соответствуют времени, на основе критерия определения соответствия, модуль 214 управления декодированием обеспечивает вывод субтитра модулем 219 обработки графических изображений и вывод субтитра модулем 218 управления декодером субтитров.

Когда модуль 214 управления декодированием обеспечивает вывод субтитра модулем 218 управления декодером субтитра, модуль 218 управления декодером субтитра передает декодированные данные изображения субтитра в модуль 219 обработки графических изображений. В модуле 219 обработки графических изображений сохраняются вводимые данные субтитра, и они комбинируются затем с видеоданными, которые будут введены позже.

Модуль 214 управления декодированием также проверяет длительность отображения субтитра. Другими словами, когда будет определено, что значение "времени начала отображения субтитра + длительность отображения" соответствует текущему времени, на основе критерия определения соответствия, модуль 214 управления декодированием обеспечивает стирание субтитров модулем 219 обработки графических изображений. В результате модуль 219 обработки графических изображений стирает данные субтитров,

которые были сохранены и введены, и прекращает комбинирование данных субтитров с видеоданными, которые будут введены позже.

[Необходимость интервалов метки]

В описанном выше критерии определения соответствия время в заданном диапазоне округляют до одного времени. Другими словами, время t, которое удовлетворяет взаимозависимости времени начала отображения конкретных видеоданных ≤ t < времени окончания отображения, округляют до времени начала отображения видеоданных.

Таким образом, значения времени двух соседних событий могут быть округлены до одного времени, в зависимости от взаимоотношения положений этих событий. Например, в примере, показанном на фиг.50, если mark_time_stamp события, которому непосредственно предшествует событие, время которого составляет 186000, равно 184000, будет определено, что эти два события возникают в момент времени начала отображения P1.

Для исключения такой ситуации необходимо обеспечить, чтобы для одного видеоизображения назначалось только одно событие. Таким образом, когда интервалы соседних событий представляют собой три поля или больше (больше, чем максимальное время отображения, обозначенное pic_struct), указанное выше условие гарантируется.

На фиг.53 показан пример предыдущего условия. Другими словами, на фиг.53, случай A обозначает, что частота кадров равна 5005/240000 (прогрессивное, с частотой 23,976 Гц), и минимальный интервал события составляет 7507, в то время как случай B обозначает частоту кадров 4004/240000 (с перемежением, с частотой 59,94 Гц), и минимальный интервал события на частоте 90 кГц составляет 6006.

В системе кодирования видеоизображения, такой как AVC и MPEG2 Video, сигнал для одного кадра отображают в течение времени трех полей для эффективного кодирования 2-3 разворачивающегося изображения. Таким образом, максимальная длительность сигнала для одного кадра составляет три поля. Другими словами, когда интервал соседних событий разделен только временем для трех полей или больше, исключается определение возникновения этих двух соседних событий в момент времени начала отображения одних видеоданных.

Кроме того, интервал соседних событий может быть определен при использовании более чем трех полей. Например, интервал соседних событий может быть определен с двумя кадрами или больше.

Вместо этого, указанное выше условие может быть гарантировано путем проверки видеоданных для всех событий и определения, что они не перекрываются друг другом.

[Обработка управления выходным атрибутом]

Далее, со ссылкой на блок-схему последовательности операций, показанную на фиг.54, будет подробно описана обработка управления выходным атрибутом, выполняемая на этапе S126, показанном на фиг.34 и т.д.

Как описано на этапе S126, показанном на фиг.34, модуль 212 управления проигрывателем проверяет, по меньшей мере, один элементарный поток, предназначенный для воспроизведения, а именно number_of_DynamicInfo (фиг.12), который представляет количество DynamicInfo() (фиг.15), которые описывают выходной атрибут, по меньшей мере, для одного элементарного потока, который был определен для воспроизведения на этапе S125, показанном на фиг.34.

Когда number_of_DynamicInfo каждого из, по меньшей мере, одного элементарного потока, предназначенного для воспроизведения, равно 0, модуль 212 управления проигрывателем не выполняет никакую обработку.

В отличие от этого, когда number_of_DynamicInfo элементарного потока, предназначенного для воспроизведения, не равно 0, модуль 212 управления проигрывателем выполняет обработку управления выходным атрибутом в соответствии с блок-схемой последовательности операций, показанной на фиг.54.

Таким образом, когда три файла информации клипа "00001.CLP", "00002.CLP" и "00003.CLP", записанные на диске 101, представляют собой такие файлы, как показаны на фиг.30A и фиг.30B, и воспроизводится первый PlayItem#0 первого PlayList#0, который

воспроизводит файл потока клипа "00001.PS", соответствующий файлу информации клипа "00001.CLP", поскольку number_of_DynamicInfo всех четырех элементарных потоков, мультиплексированных с файлом потока клипа "00001PS", которые представляют собой stream#0 - stream#3, равны 0, для файла информации клипа "0001.CLP" (фиг.30A и фиг.30B), модуль 212 управления проигрывателем не выполняет обработку управления выходным атрибутом.

Аналогично, когда воспроизводится второй PlayItem#1 первого PlayList#0, который воспроизводит файл потока клипа "00002.PS", соответствующий файлу информации клипа "00002.CLP", поскольку number_of_DynamicInfo четырех элементарных потоков, мультиплексированных с файлом потока клипа "00002.PS", которые представляют собой stream#0 - stream#3, равен 0, для файла информации клипа "00002.CLP" (фиг.30A и фиг.30B), модуль 212 управления проигрывателем не выполняет обработку управления выходным атрибутом.

В отличие от этого, когда воспроизводят PlayItem#0 второго PlayList#1, который воспроизводит файл потока клипа "00003.PS", соответствующий файлу информации клипа "00003.CLP", поскольку number_of_DynamicInfo потока видеоданных stream#0, который представляет собой первый элементарный поток, и потока звука stream#2, который представляют собой третий элементарный потоком, равны 2 и 3, соответственно, эти три элементарных потока, stream#0 - stream#3, мультиплексированы с файлом потока клипа "00003.PS" для файла информации клипа "00003.CLP" (фиг.30A и фиг.30B), модуль 212 управления проигрывателем выполняет обработку управления выходным атрибутом.

Другими словами, при обработке управления выходным атрибутом, на этапе S320, модуль 212 управления проигрывателем передает pts_change_point, описанный в файле информации клипа Clip() (фиг.12), соответствующем файлу потока клипа, который должен воспроизводиться вместе с информацией, которая представляет время, имеющее атрибут "процесса DynamicInfo()", для модуля 214 управления декодированием. Модуль 214 управления декодированием принимает pts_change_point, который представляет собой момент времени, имеющий атрибут "процесса DynamicInfo()", из модуля 212 управления проигрывателем. После этого поток обработки переходит на этап S321.

На этапе S321 модуль 214 управления декодированием определяет, соответствует ли текущее время, подсчитанное блоком 214A подсчета времени, значению pts_change_point, которое представляет собой время, имеющее атрибут "процесса DynamicInfo()". Когда результат, определенный на этапе S321, обозначает, что текущее время не соответствует pts_change_point, поток обработки возвращается на этап S321.

В отличие от этого, когда результат, определенный на этапе S321, обозначает, что текущее время соответствует любому одному из моментов времени, имеющих атрибут "процесса DynamicInfo()"; модуль 214 управления декодированием передает сообщение, которое обозначает, что текущее время стало временем, имеющим атрибут "процесса DynamicInfo()", и время, которое имеет атрибут "процесса DynamicInfo()" (ниже иногда называется временем DynamicInfo), в модуль 212 управления проигрывателем. После этого поток обработки переходит на этап S322.

На этапе S332 модуль 212 управления проигрывателем получает сообщение, которое обозначает, что текущее время стало временем, имеющим атрибут "процесса DynamicInfo()", и время DynamicInfo из модуля 214 управления декодированием, и распознает DynamicInfo() в паре с pts_change_point (фиг.12), которое соответствует времени DynamicInfo, в качестве целевого значения DynamicInfo(). После этого поток переходит на этап S323.

На этапе S323 модуль 212 управления проигрывателем передает выходной атрибут, описанный в DynamicInfo() (фиг.15), который представляет собой целевое значение DynamicInfo(), в модуль 219 обработки графических изображений или в модуль 221 вывода звука. После этого поток обработки переходит на этап S324.

На этапе S324 модуль 219 обработки графических изображений или модуль 221 вывода звука начинает управление выводом видеоданных или данных звука в соответствии с

выходным атрибутом, который был передан из модуля 212 управления проигрывателем, на этапе S323. После этого поток обработки возвращается на этап S321.

Таким образом, видеоданные выводят в соответствии с, например, соотношением сторон, описанным как режим вывода. В качестве альтернативы данные звука выводят в соответствии с, например, стерео режимом или режимом вывода двойного звука (двуязычного), описанным как режим вывода,

Далее со ссылкой на фиг.55 будет подробно описана обработка управления выходным атрибутом.

А именно, на фиг.55 показана пара pts_change_point и DynamicInfo() (фиг.12), описанных в файле информации клипа "00003.CLP", показанном на фиг.30A и фиг.30B.

Как описано выше, в файле информации клипа "00003.CLP", показанном на фиг.30A и фиг.30B, number_of_DynamicInfo для потока видеоданных stream#0 и для потока звука stream#2, которые представляют собой первый элементарный поток и третий элементарный поток из трех элементарных потоков, stream#0 - stream#2, мультиплексированных с файлом потока клипа "00003.PS", равны 2 и 3 соответственно. Таким образом, в файле информации клипа "00003.CLP" два набора pts_change_point's и DynamicInfo() описаны для первого потока видеоданных stream#0 файла потока клипа "00003.PS", и три набора pts_change_point's и DynamicInfo() описаны для третьего потока звука stream#2 файла потока клипа "00003.PS".

В верхней таблице, показанной на фиг.55, описаны два набора pts_change_point и DynamicInfo() первого потока видеоданных stream#0 файла потока клипа "00003.PS". В нижней таблице, показанной на фиг.55, описаны три набора pts_change_point и DynamicInfo() третьего потока звука stream#2 файла потока клипа "00003.PS".

В верхней таблице, показанной на фиг.55, в дополнение к двум наборам pts_change_point и DynamicInfo() первого потока видеоданных stream#0 описаны stream_id (=0xE0), private_stream_id (=0x00) и number_of_DynamicInfo (=2) первого потока видеоданных stream#0 файла информации клипа "00003.CLP", показанного на фиг.30A и фиг.30B. Аналогично, в нижней таблице, показанной на фиг.55, в дополнение к трем наборам pts_change_point и DynamicInfo() третьего потока звука stream#2 описаны streamed (=0xBD), private_stream_id (=0x00) и number_of_DynamicInfo (=3) потока звука stream#2 файла информации клипа "00003.CLP", показанных на фиг.30A и фиг.30B.

В верхней таблице, показанной на фиг.55, pts_change_point первого набора из двух наборов pts_change_point и DynamicInfo() потока видеоданных stream#0 составляет 90 000, и display_aspect_ratio (фиг.15) для его DynamicInfo() составляет "4:3". pts_change_point второго набора составляет 54090000 и display_aspect_ratio для его DynamicInfo() равно "16:9".

В нижней таблице, показанной на фиг.55, pts_change_point первого набора из трех наборов pts_change_point и DynamicInfo() потока звука stream#2 равен 90000, и channel_assignment (фиг.15) для его DynamicInfo() составляет "Двойной". pts_change_point второго набора 27 090 000, и channel_assignment для его DynamicInfo() равен "Stereo". pts_change_point третьего набора равен 32490000, и channel_assignment для его DynamicInfo() равен "Двойной".

Теперь предположим, что на этапе S125, показанном на фиг.34, первый видеопоток stream#0, идентифицированный по stream_id, который представляет собой 0xE0, и третий поток звука stream#2, идентифицированный по stream_id, который представляет собой 0xBD, и private_stream_id, который составляет 0x00, были определены как потоки, предназначенные для воспроизведения файла потока клипа "00003.PS".

В этом случае модуль 212 управления проигрывателем проверяет эти два набора pts_change_point и DynamicInfo() в верхней таблице, показанной на фиг.55, для потока видеоданных stream#0, идентифицированного по stream_id, который составляет 0xE0, и три набора pts_change_point и DynamicInfo() в нижней таблице, показанной на фиг.55 для потока звука stream#2, идентифицированного по stream_id, который составляет 0xBD и private_stream_id, который составляет 0x00, и распознает исходное значение.

Другими словами, pts_change_point первого набора из двух наборов pts_change_point и DynamicInfo() в верхней таблице, показанной на фиг.55 для потока видеоданных stream#0, идентифицированного по stream_id, который составляет 0xE0, равняется 90000. Время 90000 соответствует времени 90000, описанному в presentation_start_time, которое представляет время начала файла потока клипа "00003.PS" в файле информации клипа "00003.CLP", показанном на фиг.30A и фиг.30B соответственно для файла потока клипа "00003.PS", с которым был мультиплексирован видеопоток stream#0.

Аналогично, pts_change_point первого набора из трех наборов pts_change_point и DynamicInfo() в нижней таблице, показанной на фиг.55 для потока звука stream#2, идентифицированного по stream_id, который представляет собой 0xBD и private_stream_id, который представляет собой 0x00, 90000. Время 90000 соответствует времени 90 000, описанному в presentation_start_time, который представляет собой начальное время файла потока клипа "00003.PS" в файле информации клипа "00003.CLP", показанном на фиг.30A и фиг.30B, соответствующих файлу потока клипа "00003.PS", с которым был мультиплексирован поток звука stream#2.

Модуль 212 управления проигрывателем распознает pts_change_point, который соответствует времени 90000, описанному в presentation_start_time, который представляет начальное время файла потока клипа "00003.PS" в качестве исходного значения. Таким образом, модуль 212 управления проигрывателем распознает pts_change_point первого набора из двух наборов pts_change_point и DynamicInfo() в верхней таблице, показанной на фиг.55 и pts_change_point в первом наборе из трех наборов pts_change_point и DynamicInfo() в нижней таблице, показанной на фиг.55, в качестве исходных значений.

Модуль 212 управления проигрывателем обозначает выходной атрибут элементарного потока в соответствии с DynamicInfo() в паре с pts_change_point, распознанном как исходное значение на этапе S126, показанном на фиг.34, перед воспроизведением файла потока клипа "00003.PS".

Для потока видеоданных stream#0, идентифицированного по stream_id, который составляет 0xE0, в верхней таблице, показанной на фиг.55, display_aspect_ratio для DynamicInfo() в паре с pts_change_point, который составляет 90000 в качестве исходного значения, составляет "4:3". В этом случае модуль 212 управления проигрывателем управляет модулем 219 обработки графических изображений, используя информацию, которая обозначает, что display_aspect_ratio составляет "4:3", а именно информацию о выходном атрибуте, который обозначает, что видеопоток stream#0 представляет собой видеоданные, с соотношением сторон 4:3.

Для потока звука stream#2, идентифицированного по stream_id, который представляет собой 0xBD и private_stream_id, который представляет собой 0x00, в нижней таблице, показанной на фиг.55, channel_assignment для DynamicInfo() в паре с pts_change_point, который представляет собой 90000, в качестве исходного значения, представляет собой "Двойной". В данном случае модуль 212 управления проигрывателем передает информацию, которая обозначает, что channel_assignment представляет собой "Двойной", а именно информацию о выходном атрибуте, который обозначает, что поток звука stream#2 представляет собой данные двойного звука в модуль 221 вывода звука.

На этапе S126, показанном на фиг.34, модуль 212 управления проигрывателем выполняет обработку управления выходным атрибутом для pts_change_point в качестве исходных значений.

После этого модуль 212 управления проигрывателем передает значения 90000 и 54090000, которые представляют собой два pts_change_point, в видеопоток stream#0 в верхней таблице, показанной на фиг.55, и {27090000}, и {32490000}, и {54090000} для 90000, 27090000 и 32490000, которые представляют собой значения времени трех pts_change_point, за исключением 90000, которое представляет собой исходное значение для потока звука stream#2 в нижней таблице, показанной на фиг.55, вместе с информацией, которая обозначает, что эти три момента времени имеют атрибут "процесса

DynamicInfo()" для модуля 214 управления декодированием (на этапе S3 20).

Модуль 214 управления декодированием принимает значения времени {27090000}, {32490000} и {54090000}, имеющие атрибут "процесса DynamicInfo()", из модуля 212 управления проигрывателем. После начала воспроизведения потока видеоданных stream#0 и потока звука stream#2 модуль управления декодированием начинает проверку текущего времени, отсчитываемого блоком 214A подсчета времени.

Когда текущее время соответствует одному из моментов времени {27090000}, {32490000} и {54090000}, которые имеют атрибут "процесс DynamicInfo()", модуль 214 управления декодированием передает время DynamicInfo, которое представляет собой время, которое имеет атрибут "процесс DynamicInfo()" и которое соответствует текущему времени, в модуль 212 управления проигрывателем (на этапе S321).

Когда текущее время становится равным, например, 27090000, модуль 214 управления декодированием передает 27090000, которое соответствует текущему времени, и представляет собой одно из значений времени, имеющих атрибут "процесс DynamicInfo()", в качестве времени DynamicInfo, в модуль 212 управления проигрывателем.

Модуль 212 управления проигрывателем принимает значение 27090000, которое представляет собой время DynamicInfo, из модуля 214 управления декодированием, проверяет pts_change_point, которое соответствует 27090000, в качестве времени DynamicInfo из двух pts_change_point для потока видеоданных stream#0, в верхней таблице, показанной на фиг.55, и три pts_change_point для потока звука stream#2 в нижней таблице, показанной на фиг.55, и распознает DynamicInfo() в паре с pts_change_point, который соответствует 27090000, а именно второй DynamicInfo() для потока звука stream#2, в нижней таблице, показанной на фиг.55, в качестве целевого значения DynamicInfo() (на этапе S322).

В случае, когда целевое значение DynamicInfo() представляет собой DynamicInfo() потока видеоданных, модуль 212 управления проигрывателем передает выходной атрибут, описанный в целевом DynamicInfo(), в модуль 219 обработки графических изображений (на этапе S323). Когда целевое значение DynamicInfo() представляет собой DynamicInfo() потока звука, модуль 212 управления проигрывателем передает выходной атрибут, описанный в целевом DynamicInfo(), в модуль 221 вывода звука (на этапе S323).

Когда модуль 219 обработки графических изображений принимает выходной атрибут из модуля 212 управления проигрывателем, модуль 219 обработки графических изображений начинает управление выводом видеоданных в соответствии с выходным атрибутом (на этапе S324).

Другими словами, модуль 219 обработки графических изображений преобразует соотношение размера видеоданных, которые выводятся в модуль 220 вывода видеоизображения, например, в соответствии с соотношением размеров видеоданных (display_aspect_ratio (фиг.15)), представленных выходным атрибутом, полученным, например, из модуля 212 управления проигрывателем, и соотношением размеров устройства вывода видеоизображения, подключенного к выходному выводу 120 видеоизображения, показанному на фиг.1.

В частности, когда соотношение размеров устройства вывода видеоизображения представляет собой, например, 16:9 и соотношение размеров видеоданных, представленных атрибутом вывода, представляет собой 4:3, модуль 219 обработки графических изображений выполняет обработку сжатия видеоданных, которые выводят в модуль 220 вывода видеоданных, в горизонтальном направлении, размещает данные, которые закрашивают левый и правый края видеоданных в черный цвет, среди видеоданных, и выводит полученные в результате видеоданные. Когда соотношение размеров видеоданных устройства вывода видеоизображения составляет, например, 4:3 и соотношение размером видеоданных, представленных выходным атрибутом, составляет 16:9, модуль 219 обработки графических изображений выполняет обработку сжатия видеоданных, которые выводят в модуль 220 вывода видеоизображения в вертикальном направлении, размещает эти данные, которые закрашивают верхний и нижний края

видеоданных в черный цвет, среди видеоданных, и выводит полученные в результате видеоданные. Когда соотношение размеров устройства вывода видеоизображения и соотношение размера видеоданных в качестве выходного атрибута совпадают, например равны 4:3 или 16:9, модуль 219 обработки графических изображений выводит эти видеоданные в модуль 220 вывода видеоизображения, без выполнения обработки сжатия видеоданных.

Используя два набора pts_change_point и DynamicInfo() для потока видеоданных stream#0, идентифицированного по stream_id, который представляет собой 0xE0, в верхней таблице, показанной на фиг.55, видеоданные, имеющие соотношение размеров 4:3, получают из потока видеоданных #0 после момента времени 90000, который представляет собой время начала воспроизведения потока видеоданных stream#0, перед временем 54090000. После времени 54090000 видеоданные, имеющие соотношение размеров 16:9, получают из потока видеоданных stream#0.

Таким образом, если предположить, что соотношение размеров устройства вывода видеоизображения, подключенного к выходному выводу 120 видеоизображения, показанного на фиг.1, составляет, например, 4:3, модуль 219 обработки графических изображений передает видеоданные, имеющие соотношение размеров 4:3, полученное из потока видеоданных stream#0, в устройство вывода видеоизображения, соотношение размеров которого составляет 4:3, после момента времени 90000, перед моментом времени 54090000. Устройство вывода видеоизображения отображает полученные видеоданные.

После времени 54090000 модуль 219 обработки графических изображений выполняет обработку сжатия для видеоданных, имеющих соотношение размеров 16:9 в вертикальном направлении, и преобразует эти видеоданные, имеющие соотношение размеров 16:9, в видеосигнал, имеющий соотношение размеров 4:3, используя данные, которые обеспечивают закрашивание верхнего и нижнего краев видеоданных в черный цвет. Преобразованный видеосигнал передают в устройство вывода видеоизображения. Устройство вывода видеоизображения отображает преобразованные видеоданные.

Когда модуль 221 вывода звука принимает атрибут вывода из модуля 212 управления проигрывателем, модуль 221 вывода звука начинает управление выводом данных звука в соответствии с выходным атрибутом (на этапе S324).

Другими словами, модуль 221 вывода звука обрабатывает данные звука, принятые из модуля 217 управления декодером звука в соответствии с назначением канала для данных звука (channel_assignment (фиг.15)), представленных выходным атрибутом, полученным из модуля 212 управления проигрывателем, и в соответствии с режимом вывода звука из модуля 212 управления проигрывателем через входной интерфейс 115 (фиг.1), с которым пользователь выполняет операции, используя пульт дистанционного управления, и выводит обработанные данные звука на выходной вывод 121 звука (фиг.1).

В частности, когда назначение канала для данных звука, представленных выходным атрибутом, представляет собой двойной (двуязычный) режим, в котором левый канал представляет собой данные "основного звука", и правый канал представляет собой данные "вспомогательного звука", модуль 221 вывода звука обрабатывает данные звука, переданные из модуля 217 управления декодером звука, в соответствии с режимом вывода звука, переданным из модуля 212 управления проигрывателем, и выводит обработанные данные звука на выходной вывод 121 звука.

Другими словами, если "основной звук" был обозначен как режим вывода звука, модуль 221 вывода звука копирует левый канал данных звука, полученных из модуля 217 управления декодером звука, как правый канал данных звука, и выводит левый и правый каналы данных звука (данных "основного звука") на выходной вывод 121 звука. Если "вспомогательный звук" был обозначен как режим вывода звука, модуль 221 вывода звука копирует правый канал данных звука, полученных из модуля 217 управления декодером звука как левый канал, и выводит левый и правый каналы (данные "вспомогательного звука") на выходной вывод 121 звука. Если оба "основной и вспомогательный звуки" были

обозначены как режим вывода звука, модуль 221 вывода звука непосредственно выводит данные звука, полученные из модуля 217 управления декодером звука на выходной вывод 121 звука.

Если назначение канала данных звука, представленное выходным атрибутом, составляет, например, режим стерео, модуль 221 вывода звука непосредственно выводит данные звука, принятые из модуля 217 управления декодером звука, на выходной вывод 121 звука, независимо от того, какой режим вывода звука был обозначен.

Используя три набора pts_change_point и DynamicInfo() для потока звука stream#2, идентифицированных по stream_id, которые представляют собой 0xBD и private_stream_id, который представляет собой 0x00, в нижней таблице, показанной на фиг.55, данные двойного звука получают из потока звука stream#2 через время 90000 после времени начала воспроизведения, перед временем 27090000. Кроме того, данные стерео звука получают из потока звука stream#2, через время 27090000, перед временем 32490000. Кроме того, данные двойного звука получают из потока звука stream#2 после момента времени 32490000.

Таким образом, когда "основной звук" был обозначен как режим вывода звука, модуль 221 вывода звука копирует данные звука левого канала для данных двойного звука, которые были получены из потока звука stream#2 через время 90000, после времени 27090000, как правый канал данных звука. Левый канал и правый канал данных звука выводят на выходной вывод 121 звука.

Данные стерео звука, полученные от потока звука stream#2 после времени 27090000, перед временем 32490000, выводят на выходной вывод 121 звука.

Левый канал данных двойного звука, полученных из потока звука stream#2, через время 32490000, копируют как правый канал данных звука. Левый канал и правый канал данных звука выводят на выходной вывод 121 звука.

Как описано выше, при обработке управления выходным атрибутом определяют, соответствует ли время воспроизведения элементарного потока, который воспроизводит pts_change_point, файлу информации клипа Clip() (фиг.12), который содержит n наборов pts_change_point, которые представляют время воспроизведения каждого элементарного потока, мультиплексированного с файлом потока клипа, и DynamicInfo(), которые представляют выходной атрибут каждого элементарного потока (где n представляет собой 0 или любое целое число больше него). Когда время воспроизведения элементарного потока, который воспроизводят, соответствует pts_change_point, распознают DynamicInfo() в паре с pts_change_point. Выходом воспроизводимого элементарного потока управляют в соответствии с выходным атрибутом, описанным в DynamicInfo(). Таким образом, выводом элементарного потока можно управлять в соответствии со временем воспроизведения элементарного потока и выходным атрибутом.

[Обработка управления отображением субтитров]

Далее, со ссылкой на блок-схему последовательности операций, показанную на Фиг.56, будет описана обработка управления отображением субтитров, которая управляет отображением данных субтитров, соответствующих потоку субтитров.

Когда начинается воспроизведение PlayList(), модуль 212 управления проигрывателем инициализирует режим отображения данных субтитров для модуля 219 обработки графических изображений, на этапе S341. Другими словами, модуль 212 управления проигрывателем управляет модулем 219 обработки графических изображений для изменения режима отображения данных субтитров на режим отображения, принятый по умолчанию. Инициализация режима визуального отображения, выполняемого на этапе S341, соответствует инициализации режима отображения, выполняемого на этапе S127, показанного на фиг.34.

После этапа S341 поток обработки переходит на этап S342. На этапе S342 модуль 212 управления проигрывателем определяет, ввел ли пользователь новую команду режима отображения данных субтитров во входной интерфейс 115 через пульт дистанционного управления.

Когда результат, определенный на этапе S342, обозначает, что новая команда режима отображения была введена, поток обработки переходит на этап S343. На этапе S343 модуль 212 управления проигрывателем определяет, воспроизводится ли поток субтитров.

Когда результат, определенный на этапе S343, обозначает, что поток субтитров не воспроизводится, поток обработки возвращается на этап S342.

В отличие от этого, когда результат, определенный на этапе S343, обозначает, что поток субтитров воспроизводится, поток обработки переходит на этап S345. На этапе S345 модуль 212 управления проигрывателем определяет, является ли новая команда режима отображения командой режима отображения, принятого по умолчанию. Когда результат, определенный на этапе S343, обозначает, что новая команда режима отображения представляет собой команду принятого по умолчанию режима отображения, поток обработки возвращается на этап S341. На этапе S341, как описано выше, модуль 212 управления проигрывателем управляет модулем 219 обработки графических изображений для изменения режима отображения данных субтитров на режим отображения, принятый по умолчанию.

В отличие от этого, когда результат, определенный на этапе S345, обозначает, что новая команда режима отображения не является командой режима отображения, принятого по умолчанию, а именно новая команда режима отображения представляет собой команду режима отображения, который не был принят по умолчанию, например команду увеличения данных субтитров, команду уменьшения данных субтитров или команду повышения яркости, поток обработки переходит на этап S346. На этапе S346 модуль 212 управления проигрывателем получает StaticInfo() потока субтитров, который воспроизводят из StaticInfo() (фиг.14) файла информации клипа Clip() (фиг.12), который соответствует файлу потока клипа, с которым мультиплексирован воспроизводимый поток субтитров. После этого поток обработки переходит на этап S347.

На этапе S347 модуль 212 управления проигрывателем определяет configurable_flag для StaticInfo(), полученный на этапе S346.

Когда результат, определенный на этапе S347, обозначает, что configurable_flag равен 0, что обозначает, что не разрешено изменение режима отображения данных субтитров, поток обработки переходит на этап S348. На этапе S348 модуль 212 управления проигрывателем управляет модулем 219 обработки графических изображений для наложения на выходные видеоданные сообщения, которое обозначает, что режим отображения данных субтитров не может быть изменен. После этого поток обработки возвращается на этап S342. На этапе S342 отображается сообщение об ошибке.

В отличие от этого, когда результат, определенный на этапе S347, обозначает, что configurable_flag равен 1, что обозначает, что разрешено изменение режима отображения данных субтитров, поток обработки переходит на этап S349. На этапе S349 модуль 212 управления проигрывателем передает новую команду режима отображения, которая была введена через пульт дистанционного управления пользователем через входной интерфейс 115, в модуль 219 обработки графических изображений. После этого поток обработки переходит на этап S350.

На этапе S350 модуль 219 обработки графических изображений начинает выполнять процесс увеличения, процесс уменьшения или процесс изменения яркости данных субтитров, переданных из модуля 218 управления выводом субтитров, в соответствии с командой режима отображения, которая была передана из модуля 212 управления проигрывателем на этапе S349. После этого поток обработки переходит на этап S342. Таким образом, данные субтитров отображаются с размером отображения, в положении отображения, или с цветами отображения, в соответствии с командой режима отображения, которая была введена пользователем через пульт дистанционного управления.

В отличие от этого, когда результат, определенный на этапе S342, обозначает, что новая команда режима отображения не была введена, поток обработки переходит на этап S351. На этапе S351 модуль 212 управления проигрывателем определяет, был ли изменен

PlayItem(), как описано со ссылкой на фиг.40. Когда результат, определенный на этапе S342, обозначает, что PlayItem() не был изменен, поток обработки возвращается на этап S342.

В отличие от этого, когда результат, определенный на этапе S351, обозначает, что PlayItem() был изменен, поток обработки возвращается на этап S341. На этапе S341, как описано выше, модуль 212 управления проигрывателем управляет модулем 219 обработки графических изображений для изменения режима отображения данных субтитров на режим отображения, принятый по умолчанию. Другими словами, когда PlayItem() был изменен, режим отображения данных субтитров восстанавливают в режим отображения, принятый по умолчанию.

Как описано выше, только когда configurable_flag потока субтитров равен 1, что обозначает, что режим отображения разрешено изменять, режим отображения данных субтитров для потока субтитров может быть изменен, в соответствии с командой режима отображения, введенной пользователем через пульт дистанционного управления.

Таким образом, например, в файле информации клипа "00001.CLP", показанном на фиг.30A и фиг.30B, поскольку configurable_flag потока субтитров stream#2, который представляет собой третий элементарный поток из четырех элементарных потоков, мультиплексированных с файлом потока клипа "00001.PS", равен 0, что обозначает, что режим отображения не разрешено изменять, в то время как отображают поток субтитров stream#2, даже если пользователь работает с пультом дистанционного управления, пытаясь изменить режим отображения субтитров, режим отображения не будет изменен.

В отличие от этого, поскольку configurable_flag потока субтитров stream#3, который представляет собой четвертый элементарный поток из четырех элементарных потоков, мультиплексированных с файлом потока клипа "00001.PS", равен 1, что обозначает, что режим отображения разрешено изменять, в то время как отображают поток субтитров stream#3, когда пользователь работает с пультом дистанционного управления, для изменения режима отображения субтитров, размер отображения субтитров изменяется.

Теперь предполагается, что файл потока клипа "00001.PS" воспроизводится в соответствии с первым PlayItem#1 первого PlayList#1, показанным на фиг.29. Кроме того, в файле информации клипа "00001.CLP", показанном на фиг.30A и фиг.30B, предполагается, что третий и четвертый элементарные потоки четырех элементарных потоков, мультиплексированных с файлом потока клипа "00001.PS", представляют собой потоки субтитров и что воспроизводят третий поток субтитров stream#2 третьего и четвертого потоков субтитров stream#2 и stream#3.

Когда пользователь работает с пультом дистанционного управления для ввода команды режима отображения субтитров (на этапе S342), команда режима отображения поступает из входного интерфейса 115 (фиг.1) в модуль 212 управления проигрывателем. Когда модуль 212 управления проигрывателем принимает команду режима отображения, модуль 212 управления проигрывателем выполняет поиск StaticInfo() в файле информации клипа (фиг.12), который соответствует воспроизводимому потоку субтитров (на этапе S346).

Другими словами, поток субтитров, который воспроизводят, представляет собой третий поток субтитров stream#2, мультиплексированный с файлом потока клипа "00001.PS". Модуль 212 управления проигрывателем выполняет поиск StaticInfo() в файле информации клипа, соответствующий "00001.CLP" третьего потока субтитров stream#2.

Кроме того, модуль 212 управления проигрывателем определяет configurable_flag, который равен 0, описанный в StaticInfo() третьего потока субтитров stream#2, показанного на фиг.30A и фиг.30B (на этапе S3 47). Таким образом, модуль 212 управления проигрывателем распознает, что режим отображения третьего потока субтитров stream#2 не разрешено изменять.

В этом случае модуль 212 управления проигрывателем определяет, что поток субтитров, который воспроизводят, не соответствует режимам увеличения и уменьшения, и управляет модулем 219 обработки графических изображений, так что он генерирует соответствующее сообщение об ошибке (на этапе S348), накладывает на видеоданные это сообщение об

ошибке и выводит видеоданные с наложенным изображением.

В то время как воспроизводится четвертый поток субтитров stream#3 третьего и четвертого потоков субтитров stream#2 и stream#3 из четырех элементарных потоков, мультиплексированных с файлом потоком клипа "00001.PS", когда модуль 212 управления проигрывателем принимает команду режима отображения, которая была введена пользователем через пульт дистанционного управления, модуль 212 управления проигрывателем выполняет поиск StaticInfo() в соответствующем файле информации клипа "00001.CLP" четвертого потока субтитров stream#3.

Модуль 212 управления проигрывателем определяет configurable_flag, который равен 1, описанный в StaticInfo() четвертого потока субтитров stream#3, показанного на фиг.30A и фиг.30B (на этапе S347). Таким образом, модуль 212 управления проигрывателем распознает, что режим отображения четвертого потока субтитров stream#3 было разрешено изменять.

В этом случае модуль 212 управления проигрывателем определяет, что воспроизводимый поток субтитров соответствует режиму увеличения или режиму уменьшения и передает команду режима отображения, которая была введена пользователем через пульт дистанционного управления, в модуль 219 обработки графических изображений (на этапе S349).

Таким образом, модуль 219 обработки графических изображений, например, увеличивает или уменьшает данные субтитров, полученные из модуля 218 управления декодером субтитров, в соответствии с командой режима отображения, полученной из модуля 212 управления проигрывателем, накладывает полученные данные субтитров на видеоданные, переданные из модуля 212 управления видеодекодером, и выводит видеоданные с наложенным изображением.

Когда модуль 212 управления проигрывателем начинает воспроизводить первый PlayItem() PlayList(), модуль 212 управления проигрывателем инициализируют режим отображения данных субтитров модуля 219 обработки графических изображений (на этапе S341). Другими словами, модуль 212 управления проигрывателем управляет модулем 219 обработки графических изображений для изменения режима отображения данных субтитров на режим отображения, принятый по умолчанию.

Когда PlayItem() изменяются, модуль 212 управления проигрывателем инициализирует режим отображения данных субтитров модуля 219 обработки графических изображений (на этапах S341 и S351).

Когда PlayItem() изменяются, модуль 212 управления проигрывателем проверяет configurable_flag на наличие нового потока субтитров, который должен воспроизводиться в соответствии с новым воспроизводимым PlayItem(). Когда configurable_flag равен 0, модуль 212 управления проигрывателем инициализирует режим отображения данных субтитров модуля 219 обработки графических изображений. Когда configurable_flag равен 1, модуль 212 управления проигрывателем обеспечивает поддержание модулем 219 обработки графических изображений режима отображения, в котором PlayItem() не были изменены.

При обработке управления отображением субтитров, показанной на фиг.56, когда новая команда режима отображения вводится пользователем через пульт дистанционного управления, новую команду режима отображения передают в модуль 219 обработки графических изображений (на этапе S349). Команда режима отображения может быть сохранена, например, в энергонезависимом запоминающем устройстве, из которого состоит запоминающее устройство 113 (фиг.1). Команда режима отображения, сохраненная в энергонезависимом запоминающем устройстве, может быть передана в модуль 219 обработки графических изображений.

В случае, когда команда режима отображения, которую установил пользователь, сохранена в энергонезависимом запоминающем устройстве, в качестве исходной установки устройства воспроизведения диска, показанного на фиг.1, когда он или она вводит новую команду режима отображения с использованием пульта дистанционного управления,

команда режима отображения, сохраненная в энергонезависимом запоминающем устройстве, может быть заменена новой командой режима отображения, и новая команда режима отображения, сохраненная в энергонезависимом запоминающем устройстве, может быть передана в модуль 219 обработки графических изображений. В этом случае, поскольку в энергонезависимом запоминающем устройстве сохранена команда режима отображения, которая была установлена после завершения списка воспроизведения, когда воспроизводится следующий PlayList(), данные субтитров отображаются с командой режима отображения, без необходимости ввода команды отображения через пульт дистанционного управления.

В этом случае предполагается, что команда режима отображения, сохраненная в энергонезависимом запоминающем устройстве, включает в себя, например, степень увеличения или степень уменьшения, в которой поток субтитров был увеличен или уменьшен.

Как описано выше, при обработке управления отображением субтитров определяется, разрешено ли изменение режима отображения данных субтитров из режима отображения, принятого по умолчанию в configurable_flag, который обозначает, разрешено ли изменять режим отображения из режима отображения, принятого по умолчанию, содержащегося в StaticInfo(), для данных субтитров, которые не изменяются, при воспроизведении элементарных потоков, содержащихся в файле информации клипа Clip() (фиг.12). Когда режим отображения, принятый по умолчанию данных субтитров, которые воспроизводят, разрешено изменять, выполняют процесс отображения, например процесс увеличения, процесс уменьшения или процесс изменения цвета данных субтитров. Таким образом, можно управлять режимом отображения данных субтитров.

[Обработка управления захватом]

Далее, со ссылкой на блок-схему последовательности операций, показанную на фиг.57, будет описана обработка управления захватом, которая управляет захватом видеоданных, соответствующих видеопотоку. На фиг.57 также показана блок-схема последовательности операций, которая описывает обработку фонового изображения/скринсейвера, которая выполняет вторичное использование видеоданных, которые были зафиксированы при обработке управления захватом.

Когда команда захвата видеоданных поступает от пользователя через пульт дистанционного управления и входной интерфейс 115 (фиг.1) в модуле 212 управления проигрывателем начинается обработка управления захватом.

Другими словами, при обработке управления захватом, на этапе S371, модуль 212 управления проигрывателем определяет, воспроизводится ли видеопоток. Когда результат, определенный на этапе S371, обозначает, что видеопоток не воспроизводится, модуль 212 управления проигрывателем завершает обработку управления захватом.

В отличие от этого, когда результат, определенный на этапе S371, обозначает, что видеопоток воспроизводится, поток обработки переходит на этап S372. Модуль 212 управления проигрывателем получает capture_enable_flag_PlayList из PlayList() (фиг.7), который соответствует воспроизводимому видеопотоку, и capture_enable_flag_Clip из файла информации клипа Clip() (фиг.12), который соответствует воспроизводимому видеопотоку.

Как описано на фиг.7, capture_enable_flag_PlayList в PlayList() обозначает, разрешено ли вторичное использование видеоданных, соответствующих видеопотоку, воспроизводимому в соответствии с PlayList(). С другой стороны, как описано на фиг.12, capture_enable_flag_Clip файла информации клипа Clip() обозначает, разрешено ли вторичное использование видеоданных, соответствующих видеопотоку, сохраненному в файле потока клипа, который соответствует файлу информации клипа Clip().

После этапа S372 поток обработки переходит на этап S373. На этапе S373 модуль 212 управления проигрывателем определяет, разрешен ли захват изображения воспроизводимых видеоданных, когда поступает команда захвата через входной интерфейс 115 (фиг.1) в соответствии с capture_enable_flag_PlayList и

capture_enable_flag_Clip, которые были получены на этапе S373.

Когда результат, определенный на этапе S373, обозначает, что захват изображения воспроизводимых видеоданных при поступлении команды захвата через входной интерфейс 115 не разрешен, а именно, по меньшей мере, один из capture_enable_flag_PlayList и capture_enable_flag_Clip, полученных на этапе S373, равен О, что обозначает, что не разрешено вторичное использование видеоданных, поток обработки переходит на этап S374. На этапе S374 модуль 212 управления проигрывателем управляет модулем 219 обработки графических изображений для наложения на видеоданные сообщения об ошибке, которое обозначает, что захват видеоданных не разрешен, и завершает обработку управления захватом. В результате отображается сообщение об ошибке.

В отличие от этого, когда результат, определенный на этапе S373, обозначает, что разрешен захват изображения воспроизводимых видеоданных, когда поступает команда захвата через входной интерфейс 115, а именно оба capture_enable_flag_PlayList и capture_enable_flag_Clip, которые были получены на этапе S373, равны 1, что обозначает, что разрешено вторичное использование видеоданных, поток обработки переходит на этап S375. На этапе S375 модуль 212 управления проигрывателем передает команду захвата воспроизводимых видеоданных, когда команда захвата поступает через входной интерфейс 115, в модуль 219 обработки графических изображений. После этого поток обработки переходит на этап S376.

На этапе S376 модуль 219 обработки графических изображений выполняет захват изображения видеоданных из модуля 216 управления видеодекодером в соответствии с командой захвата, полученной из модуля 212 управления проигрывателем, сохраняет изображение в запоминающем устройстве 113 (фиг.1) и завершает обработку управления захватом. Когда capture_enable_flag состоит из множества битов и условия их использования ограничены, в этой точке выполняется соответствующая операция. Другими словами, когда размер захваченного изображения ограничен, выполняется захват изображения с уменьшенным размером. Когда используемая прикладная программа имеет ограничение, флаг, который представляет такое ограничение, также записывают.

Как описано выше, при обработке управления захватом, capture_enable_flag_PlayList и capture_enable_flag_Clip для PlayList() (фиг.7), а также файл информации клипа Clip() (фиг.12), соответствующие видеопотоку, который воспроизводят, когда пользователь вводит команду захвата, обрабатывают с использованием логической операции "И". Когда результат операции "И" равен 1, а именно, когда все capture_enable_flag_PlayList и capture_enable_flag_Clip равны 1, что обозначает, что разрешено вторичное использование видеоданных, определяется, что видеоданные можно использовать вторично. В результате, эти видеоданные захватывают.

Когда видеопоток воспроизводят в соответствии с первым PlayItem#0 первого PlayList#0, показанного на фиг.29, а именно видеопоток, мультиплексированный с файлом потока клипа "00001.PS", воспроизводят, когда пользователь вводит команду захвата, поскольку capture_enable_flag_PlayList первого PlayList#0 равен 1 и capture_enable_flag_Clip файла информации клипа "00001.CLP", показанного на фиг.30А и фиг.30В, соответствующие файлу потока клипа "00001.PS", воспроизводимого первым PlayItem#0, равны 1, определяется, что воспроизводимые видеоданные (видеоданные, соответствующие видеопотоку, мультиплексированному с файлом потока клипа "00001.PS"), возможно использовать вторично, и выполняется захват видеоданных.

В то время как видеопоток воспроизводят в соответствии со вторым PlayItem#1 первого PlayList#0, показанного на фиг.29, а именно воспроизводят видеопоток, мультиплексированный с файлом потока клипа "00002.PS", когда пользователь вводит команду захвата, поскольку capture_enable_flag_PlayList первого PlayList#0 равен 1 и capture_enable_flag_Clip файла информации клипа "00002.CLP", показанного на фиг.30А и фиг.30В, соответствующих файлу потока клипа "00002.PS", воспроизводимого в соответствии со вторым PlayItem#1, равен 0, определяется, что воспроизводимые

видеоданные (видеоданные, соответствующие видеопотоку, мультиплексированному с файлом потока клипа "00002.PS") не разрешено использовать вторично, и захват видеоданных не выполняется.

В то время как видеопоток воспроизводится в соответствии с PlayItem#0 второго PlayList#1, показанного на фиг.29, а именно воспроизводят видеопоток, мультиплексированный с файлом потока клипа "00003.PS", когда пользователь вводит команду захвата, поскольку capture_enable_flag_PlayList второго PlayList#1 равен 0 и capture_enable_flag_Clip файла информации клипа "00003.CLP", показанного на фиг.30A и фиг.30B, соответствующих файлу потока клипа "00003.PS", воспроизводимому в соответствии с PlayItem#0 второго PlayList#1, равен 1, определяется, что воспроизводимые видеоданные (видеоданные, соответствующие видеопотоку, мультиплексированному с файлом потока клипа "00003.PS") не разрешено использовать вторично. Таким образом, захват видеоданных не выполняется.

В этом случае, когда проверяют, что capture_enable_flag_PlayList второго PlayList#1 равен 0, можно определить, что не может быть разрешено вторичное использование видеоданных. Таким образом, можно исключить проверку capture_enable_flag_Clip файла информации клипа "00003.CLP", показанного на фиг.30A и фиг.30B, соответствующего файлу потока клипа "00003.PS", воспроизводимого в соответствии с PlayItem#0 второго PlayList#1.

Изображение, захваченное в процессе управления захватом и сохраненное в запоминающем устройстве 113, можно повторно использовать при обработке фонового изображения/скринсейвера.

Обработка фонового изображения/скринсейвера выполняется, например, когда работает модуль 212 управления проигрывателем, но элементарный поток при этом не воспроизводится, а именно диск 101 не был вставлен в привод 102 диска (фиг.1), или элементарный поток уже был воспроизведен.

При обработке фонового изображения/скринсейвера, на этапе S3 80, модуль 212 управления проигрывателем управляет модулем 219 обработки графических изображений для отображения изображения, сохраненного в запоминающем устройстве 113, при обработке управления захватом. Модуль 219 обработки графических изображений отображает изображение, сохраненное в запоминающем устройстве 113, при обработке управления захватом под управлением модуля 212 управления проигрывателем.

Когда модуль 219 обработки графических изображений отображает изображение, сохраненное в запоминающем устройстве 113 как неподвижное изображение, выполняется так называемое представление фонового изображения. Когда изображение отображается с увеличением, уменьшением и с перемещением через заданные интервалы, получается скринсейвер. Обработка фонового изображения/скринсейвера, который отображает изображение, сохраненное в запоминающем устройстве 113, при обработке управления захватом, может выполняться с использованием другой независимой прикладной программы, вместо модуля 212 управления проигрывателем.

Когда флаг, который представляет ограничение, добавлен к изображению, сохраненному в запоминающем устройстве 133, отображаемое изображение ограничено в соответствии с этим флагом.

Как описано выше, capture_enable_flag_PlayList и capture_enable_flag_Clip, которые обозначают, разрешено ли вторичное использование воспроизводимых видеоданных, получают в соответствии с, например, PlayList() или PlayItem(), которые больше, чем модуль доступа к видеоданным. В соответствии с capture_enable_flag_PlayList и capture_enable_flag_Clip определяют, разрешено ли вторичное использование воспроизводимых видеоданных. Когда определенный результат обозначает, что воспроизводимые видеоданные разрешено вторично использовать, выполняют захват воспроизводимых видеоданных и обработку фонового изображения/скринсейвера. Таким образом, можно управлять вторичным использованием видеоданных.

При обработке управления захватом, показанной на фиг.57, PlayList() (фиг.7)

содержит capture_enable_flag_PlayList и файл информации клипа Clip() (фиг.12), который соответствует файлу потока клипа, воспроизводимому в соответствии с PlayItem(), содержит capture_enable_flag_Clip. При использовании одновременно capture_enable_flag_PlayList и capture_enable_flag_Clip, определяют, разрешено ли вторичное использование видеоданных. В качестве альтернативы, когда PlayList() (фиг.7) содержит capture_enable_flag_PlayList или файл информации клипа Clip() (фиг.12), соответствующий файлу потока клипа, воспроизводимому в соответствии с PlayItem(), содержит capture_enable_flag_Clip, а именно либо capture_enable_flag_PlayList, либо capture_enable_flag_Clip, можно определить, разрешено ли вторичное использование видеоданных.

При обработке управления захватом, показанной на фиг.57, на этапе S376, модуль 219 обработки графических изображений выполняет захват видеоданных из модуля 216 управления видеодекодером в соответствии с командой захвата, полученной из модуля 212 управления проигрывателем, а именно только одно изображение. В качестве альтернативы модуль 219 обработки графических изображений может выполнять захват множества изображений. Другими словами, может быть захвачено множество изображений, которые модуль 216 управления видеодекодером выводит в виде временной последовательности. В этом случае можно заранее обозначить количество изображений, захватываемых в определенный момент времени. В качестве альтернативы биты capture_enable_flag_PlayList и capture_enable_flag_Clip могут быть расширены до информации, которая представляет количество изображений, которые могут быть захвачены в определенный момент времени.

В описанном выше случае информация разрешения, которая обозначает, разрешено ли вторичное использование видеоданных, которые представляют собой capture_enable_flag_PlayList и capture_enable_flag_Clip, описана в PlayList() и файле информации клипа Clip(). При использовании информации разрешения определяют, разрешено ли вторичное использование всех видеоданных, воспроизводимых в соответствии с PlayList(), и всех видеоданных, соответствующих видеопотоку, мультиплексированному с файлом потока клипа, который соответствует файлу информации клипа Clip(). Информация о разрешении на использование может описывать видеоданные любого входа. При использовании информации разрешения можно определить, разрешено ли вторичное использование видеоданных в любом модуле.

Другими словами, на фиг.58 показан синтаксис private_stream2_PES_payload(), который содержит информацию о разрешении на использование. На фиг.59 показан синтаксис au_information(), который содержит информацию о разрешении на использование.

private_stream2_PES_payload(), показанный на фиг.58, является тем же, что показан на фиг.26, за исключением того, что video_stream_id расположен непосредственно перед capture_enable_flag_ps2 в качестве информации разрешения использования. Аналогично, au_information(), показанный на фиг.59, является тем же, что и показан на фиг.27, за исключением того, что pic_struct_copy расположен непосредственно перед capture_enable_flag_AU в качестве информации разрешения использования.

capture_enable_flag_ps2, содержащийся в private_stream2_PES_payload(), показанном на фиг.58, обозначает, разрешено ли вторичное использование видеоданных потока видеоданных после PES_packet() private_stream_2, который содержит private_stream2_PES_payload() перед PES_packet() следующего private_stream_2. Таким образом, с помощью capture_enable_flag_ps2, содержащегося в private_stream2_PES_payload(), показанного на фиг.58, можно определить, разрешено ли вторичное использование видеоданных, следующих после определенной точки возможного начала декодирования до следующей точки возможного начала декодирования.

Кроме того, capture_enable_flag_AU, содержащийся в au_information(), показанном на фиг.59, обозначает, разрешено ли вторичное использование видеоданных модулем видеодоступа, соответствующего capture_enable_flag_AU. Таким образом, при использовании capture_enable_flag_AU, содержащегося в au_information(), показанном на

фиг.59, может быть определено, разрешено ли вторичное использование видеоданных каждого модуля доступа видеоданных, а именно каждое изображения.

По меньшей мере, два из capture_enable_flag_PlayList, которые используют информацию разрешения PlayList() (фиг.7), capture_enable_flag_Clip, который использует информацию разрешения файла информации клипа Clip() (фиг.12), capture_enable_flag_ps2 в качестве информации разрешения на использование private_stream2_PES_payload() (фиг.58) и capture_enable_flag_AU как информацию разрешения использования au_information() (фиг.59), можно использовать с избыточностью. В этом случае, по результату, для получения которого выполняют операцию логическое "И", по меньшей мере, двух типов информации разрешения использования, используемой как избыточная информация, можно определить, разрешено ли вторичное использование изображения видеоданных.

Как описано на этапе S211, показанном на фиг.45, блок 233 функции считывания видеоданных, в модуле 215 управления буфером (фиг.5) выполняет поиск PES_packet() в программном потоке, сохраненном в буфере 215A private_stream_2, который содержит private_stream2_PES_payload(), показанный на фиг.26 или фиг.58, которые содержат au_information(); показанную на фиг.59. Таким образом, когда используются private_stream2_PES_payload(), показанный на фиг.58, который содержит capture_enable_flag_ps2 и au_information(), показанные на фиг.59, которые содержат capture_enable_flag_AU, модуль 212 управления проигрывателем должен запросить на блоке 233 функции считывания видеоизображений capture_enable_flag_ps2 и capture_enable_flag_AU, для определения, разрешено ли вторичное использование видеоданных.

Далее, со ссылкой на фиг.60, будет описана структура аппаратных средств устройства записи диска.

Устройство записи диска, показанное на фиг.60, может быть применено, например, в проигрывателе диска, игровом устройстве, автомобильной навигационной системе и т.д.

В устройстве записи диска, показанном на фиг.60, диск 410, например, представляет собой оптический диск, такой как DVD, магнитооптический диск или магнитный диск. На диск 410 можно записывать данные содержания, такие как видеоданные, данные звука и данные субтитров. Кроме того, на диск 410 можно записывать данные содержания. Когда данные различных типов записаны на диске 410, его можно использовать как диск 101, показанный на фиг.1.

Входной вывод 400A видеоданных соединен с модулем ввода видеоданных, таким как устройство съемки изображения (не показано). Входной вывод 400A видеоданных передает видеоданные, переданные из модуля ввода видеоданных во входной интерфейс 401 видеоданных. Входной вывод 400 B звука соединен с модулем ввода звука, таким как микрофон и усилитель (не показан). Входной вывод 400 B звука передает входные данные звука в интерфейс 402 ввода звука.

Интерфейс 401 ввода видеоданных выполняет требуемую обработку для входных видеоданных и передает полученные в результате видеоданные в кодер 403 видеоданных через шину 411. Интерфейс 402 ввода звука выполняет требуемую обработку для входных данных звука и передает полученные в результате данные звука в кодер 404 звука через шину 411.

Кодер 403 видеоданных кодирует видеоданные, переданные из ЦПУ 405, и интерфейс 401 ввода видеоданных и обеспечивает запись приводом 409 диска полученных в результате данных, кодированных со сжатием (кодированных видеоданных, например, видеопоток MPEG2) на диск 410 через шину 411.

Кодер 404 звука кодирует данные звука, переданные из ЦПУ 405, и интерфейс 402 ввода звука, и обеспечивает запись приводом 409 диска полученных в результате кодированных со сжатием данных (кодированных данных звука, например, в виде потока звука MPEG2) на диск 410 через шину 411.

ЦПУ 405 и запоминающее устройство 406 составляют компьютерную систему. Другими

словами, ЦПУ 405 выполняет программу, сохраненную в запоминающем устройстве 4Q6, управляет всем устройством записи диска и выполняет различные типы обработки, которые будут описаны ниже. В запоминающем устройстве 406 сохранена программа, которую выполняет ЦПУ 405. Кроме того, в запоминающем устройстве 406 временно сохранены данные, с которыми должно работать ЦПУ 405. Запоминающее устройство 406 может состоять только из энергонезависимого запоминающего устройства или из комбинации энергозависимого и энергонезависимого запоминающего устройства. Когда устройство записи диска, показанное на фиг.60, содержит жесткий диск, на котором записана программа, которую выполняет ЦПУ 405, и эта программа записана (установлена) на жестком диске, запоминающее устройство 406 может состоять только из энергозависимого запоминающего устройства.

Программа, которую выполняет ЦПУ 405, может быть заранее записана в запоминающем устройстве 406 в качестве носителя записи, встроенного в устройство записи диска.

Вместо этого, программа может быть временно или постоянно записана в приводе 409 диска, гибкого диска, который не является приводом 409 диска, или на съемном носителе записи, таком как CD-ROM (компакт-диск - постоянное запоминающее устройство), MO (магнитооптический) диск, магнитном диске или карте памяти. Такой съемный носитель записи может быть предусмотрен как так называемое упакованное программное обеспечение.

Кроме того, программа может быть предварительно сохранена в запоминающем устройстве 406. Вместо этого программа может быть установлена со съемного носителя записи в устройстве записи диска. Вместо этого программа может быть передана по беспроводному каналу передачи данных с сайта загрузки в устройство записи диска через спутник, используемый для широковещательной спутниковой передачи цифровых данных. Вместо этого программа может быть передана в устройство записи диска с использованием кабеля, через который работает сеть, такая как ЛВС (локальная вычислительная сеть) или Интернет. Устройство записи диска может принимать программу через входной интерфейс 408 и может устанавливать ее во встроенном запоминающем устройстве 406.

Кроме того, программа может обрабатываться одним ЦПУ. Вместо этого программа может обрабатываться в распределенном режиме, с использованием множества ЦПУ.

Интерфейс 407 привода управляет приводом 409 диска под управлением ЦПУ 405. Таким образом, интерфейс 407 привода передает данные, переданные из ЦПУ 405, запоминающего устройства 406, кодера 403 видеоданных и кодера 404 звука в привод 409 привода диска через шину 411, и обеспечивает запись всем приводом 409 диска данных на диске 410. Вместо этого интерфейс 407 привода считывает данные с диска 410 и передает эти данные в ЦПУ 405 и в запоминающее устройство 406 через шину 411.

Входной интерфейс 408 принимает сигнал в соответствии с операцией пользователя, выполняемой с кнопками (клавишами) и пультом дистанционного управления (пультом дистанционного ввода команд), и передает этот сигнал в ЦПУ 405 через шину 411. Кроме того, входной интерфейс 408 выполняет функцию интерфейса передачи данных, например модем (включая модем ADSL) (асимметричная цифровая абонентская линия) или NIC (карта сетевого интерфейса).

Видеоданные и данные звука могут быть переданы по кабелю или по беспроводному каналу передачи данных из модуля ввода видеоданных и из модуля ввода звука соответственно.

Диск 410 может быть загружен в привод 409 диска и может быть выгружен из него. Привод 409 диска имеет встроенный интерфейс (не показан). Привод 409 диска соединен с интерфейсом 407 привода через интерфейс. Привод 409 диска выполняет привод загруженного диска 410 и выполняет, например, обработку записи данных на диске 410, в соответствии, например, с командой записи, полученной через интерфейс 407 привода.

В необходимых случаях данные (данные записи), записанные на диск 410, включают в

себя программу, которую может выполнять компьютер. В данном варианте выполнения в качестве носителя записи используется диск 410, который представляет собой носитель записи в форме диска. Вместо этого в качестве носителя записи может использоваться полупроводниковое запоминающее устройство или носитель записи в форме ленты.

К шине 411 подключены ЦПУ (центральное процессорное устройство) 405, запоминающее устройство 406, интерфейс 407 привода, входной интерфейс 408, кодер 403 видеоданных, кодер 404 звука, интерфейс 401 ввода видеоданных и интерфейс 402 ввода звука.

Далее, со ссылкой на фиг.61, будет описана функция, выполняемая устройством записи диска, с помощью которой воплощают способ кодирования данных в соответствии с настоящим изобретением. В функции, которую выполняет устройство записи диска, показанное на данном чертеже, кодер 404 звука выполняет кодирование со сжатием звукового сигнала, который был введен через входной вывод 400 B звука и интерфейс 402 ввода звука, и выводит полученный в результате сигнал в модуль 421 мультиплексирования.

Кодер 403 видеоданных выполняет кодирование со сжатием видеосигнала, который был введен через входной вывод 400A видеоданных и интерфейс 401 ввода видеоданных, и выводит полученный в результате сигнал, который должен быть мультиплексирован модулем 421.

Модуль 421 мультиплексирования выполняет пакетирование входного потока видеоданных MPEG2 и входного потока звука MPEG2 и мультиплексирует их на основе разделения времени, как описано со ссылкой на фиг.18A и фиг.18B - фиг.27. Модуль 421 мультиплексирования выбирает внутреннее кодирование изображения из потока и вводит PES_packet() private_stream_2, показанные на фиг.26, в изображение с внутренним кодированием с частотой, составляющей приблизительно два раза в секунду.

Модуль 421 мультиплексирования выводит мультиплексированный поток в модуль 424 перезаписи RAPI через ППО 422, а также в модуль 423 выделения информации RAPI. Модуль 423 выделения информации RAPI детектирует начальное положение PES_packet() private_stream_2 из потока видеоданных мультиплексированного потока, значение временной метки (PTS) изображения с внутренним кодированием, которому непосредственно предшествует PES_packet() private_stream_2, и конечные положения изображения с внутренним кодированием, второго, третьего и четвертого эталонных изображений, которым предшествует изображение с внутренним кодированием, и сохраняет их.

В этом случае RAPI представляет PES_packet() private_stream_2.

Модуль 423 выделения информации RAPI выводит конечные положения детектируемых внутренних изображений и второго, третьего и четвертого эталонных изображений, которым предшествует изображения с внутренним кодированием, в модуль 424 перезаписи RAPI. Модуль 424 перезаписи RAPI перезаписывает поля 1stRef_picture, 2ndRef_picture, 3rdRef_picture и 4thRef_picture, показанные на фиг.26, как информацию RAPI, записывает конечные положения верхнего, второго, третьего и четвертого эталонных изображений как цифровые значения в единицах секторов, и сохраняет их в выходном сервере 426.

После окончания обработки всего мультиплексированного потока контроллер 425 получает начальные положения всех RAPI, которые были выделены и сохранены модулем 423 выделения информации RAPI и которые были мультиплексированы в мультиплексированном потоке, и конечное положение изображения с внутренним кодированием, которому непосредственно предшествует каждый RAPI, а также конечные положения второго, третьего и четвертого эталонных изображений, которым предшествует изображение с внутренним кодированием.

Контроллер 425 создает EP_map(), описанную со ссылкой на фиг.16, с входной информацией.

Контроллер 425 создает EP_map() файла информации клипа с адресом каждого RAPI,

PTS изображения с внутренним кодированием, которому непосредственно предшествует каждый RAPI, и одним из конечных положений изображения с внутренним кодированием, и второго, третьего и четвертого изображений, которым предшествует изображение с внутренним кодированием, и сохраняет EP_map() в выходном сервере 426.

Далее, со ссылкой на блок-схему последовательности операций, показанную на фиг.62, будет описана обработка создания EP_map().

На этапе S381 кодер 403 видеоданных кодирует со сжатием видеосигнал, который был введен через вывод 400A ввода видеоданных и интерфейс 401 ввода видеоданных, и выводит полученный в результате сигнал в модуль 421 мультиплексирования. Кодер 404 звука выполняет кодирование со сжатием звукового сигнала, который был введен через вывод 400 B ввода звука и интерфейс 402 ввода звука, и выводит полученный в результате сигнал в модуль 421 мультиплексирования. В этом случае поток, который выведен из кодера 404 звука, представляет собой звуковой поток MPEG2. Аналогично, поток, который выводится из кодера 403 видеоизображения, представляет собой видеопоток MPEG2.

На этапе S382 модуль 421 мультиплексирования выполняет пакетирование входного потока видеоданных MPEG2 и потока звука MPEG2, и мультиплексирует их на основе разделения по времени (пакетирование), как описано со ссылкой на 18A и фиг.18B - фиг.27, выбирает изображение с внутренним кодированием из потока и вводит в него PES_packet() private_stream_2, показанного на фиг.26, с частотой приблизительно два раза в секунду. В этом примере PES_packet() private_stream_2 обозначает, что после него непосредственно следует изображение с внутренним кодированием видеоданных, которое может быть декодировано без необходимости обращения к другому изображению. В этой точке изображение с внутренним кодированием обычно имеет временную метку (PTS/DTS).

В этой точке данные не записывают в поля 1stRef_picture, 2ndRef_picture, 3rdRef_picture и 4thRef_picture, описанные со ссылкой на фиг.26. Кроме того, поток субтитров (не показан) может быть подан в модуль 421 мультиплексирования, в результате чего его мультиплексируют с видеопотоком и потоком звука.

На этапе S383 модуль 421 мультиплексирования выводит мультиплексированный поток в модуль 424 перезаписи RAPI, через ППО 422 и также в модуль 423 выделения информации RAPI. Модуль 423 выделения информации RAPI детектирует начальное положение PES_packet() для private_stream_2, значения временной метки (PTS) изображения с внутренним кодированием, которому непосредственно предшествует PES_packet() для private_stream_2, и конечные положения изображения с внутренним кодированием и второго, третьего и четвертого эталонных изображений, которым предшествует изображение с внутренним кодированием из потока видеоданных мультиплексированного потока, и сохраняет их.

Кроме того, модуль 423 выделения информации RAPI выводит конечные положения детектированного изображения с внутренним кодированием и второго, третьего и четвертого эталонных изображений, которым предшествует изображение с внутренним кодированием, в модуль 424 перезаписи RAPI. Модуль 424 перезаписи RAPI перезаписывает поля 1stRef_picture, 2ndRef_picture, 3rdRef_picture и 4thRef_picture, показанные на фиг.26, как информацию RAPI, записывает конечные положения верхнего, второго, третьего и четвертого эталонных изображений как значения в единицах секторов, и сохраняет их в выходном сервере 426.

На этапе S385 начальные положения RAPI, которые были выделены и сохранены в модуле 423 выделения информации RAPI и которые были мультиплексированы в мультиплексированный поток, и конечное положение изображения с внутренним кодированием, которому непосредственно предшествует каждый RAPI, и конечные положения второго, третьего и четвертого эталонных изображений, которым предшествует изображение с внутренним кодированием, подают в контроллер 425.

Контроллер 425 создает EP_map(), описанный со ссылкой на фиг.16 с входной

информацией. В данном примере предполагается, что EP_map() содержит информацию только потока видеоданных. EP_map() видеоданных представляет положения всех RAPI в потоке, а именно положения всех PES_packet() private_stream_2. Эту информацию создают с информацией, которую подают из модуля 423 выделения информации RAPI в контроллер 425.

Более конкретно, контроллер 425 создает EP_map() файла информации клипа с адресом каждого RAPI, PTS изображения с внутренним кодированием, которому непосредственно предшествует каждый RAPI, и одно из конечных положений изображения с внутренним кодированием и второго, третьего и четвертого изображений, которым предшествует изображение с внутренним кодированием, и сохраняет EP_map() в выходном сервере 426. Другими словами, контроллер 425 копирует значение, близкое к заданной величине подсчета секторов (количество секторов, которые могут быть считаны во время процесса кодирования конечных положений четырех эталонных изображений (1stRef_picture, 2ndRef_picture, 3rdRef_picture и 4thRef_picture) в N-th_Ref_picture_copy.

На этапе S386 контроллер 425 определяет index_minus1 на основе N-th_Ref_picture_copy и записывает его на диск 410. В данном примере потоковые данные и файл базы данных, сохраненные в выходном сервере 426, передают в привод 409 диска через интерфейс 407 привода и записывают на диске 410.

В описанной выше обработке EP_map() создают, как показано на фиг.31.

[Использование 1stRef_Picture, 2ndRef_Picture, 3rdRef_Picture и 4thRef_Picture]

Далее, со ссылкой на блок-схему последовательности операций, показанную на фиг.63 будет описана обработка воспроизведения с быстрой перемоткой вперед, используя EP_map(), показанный на фиг.31.

Предполагается, что пользователь (не показан) передает команду быстрой перемотки вперед в программу 210 воспроизведения видеосодержания. Модуль 212 управления проигрывателем выбирает одно из положений возможного начала воспроизведения, зарегистрированных в EP_map(), из файла информации клипа воспроизводимого потока (на этапе S3 91), и принимает решение в отношении данных для считывания в отношении размера N-th_Ref_Picture_copy из RPN_EP_start, описанных в EP_map (в этапе S393). Модуль 212 управления проигрывателем передает информацию в модуль 213 передачи данных содержания об этой информации и команды в модуль 214 управления декодированием, для выполнения воспроизведения с быстрой перемоткой вперед.

Используя функцию операционной системы 201, модуль 213 подачи данных содержания считывает файл потока клипа, который содержит программный поток, в котором был мультиплексирован элементарный поток, предназначенный для воспроизведения, и передает файл потока клипа в модуль 215 управления буфером (на этапе S393). Поскольку название файла и т.д. были обозначены, их не обозначают снова. В отличие от случая, когда воспроизведение началось, команду считывания передают с адресом начала считывания и размером данных, предназначенных для передачи.

Блок 233 функции считывания видеоданных демультиплексирует мультиплексированные данные, которые были введены в модуль 215 управления буфером (на этапе S394), и передает только видеопоток в модуль 216 управления видеодекодером. В этом случае, поскольку выполняют обработку воспроизведения с быстрой перемоткой вперед, модуль 217 управления декодером звука, модуль 218 управления декодером субтитров, блок 234 функции считывания звука и блок 233 функции считывания субтитров не работают.

Входные данные содержат одно эталонное изображение из четырех или меньше эталонных изображений. При обработке воспроизведения с быстрой перемоткой вперед только эталонные изображения декодируют и отображают на основе точки входа, выбранной при обработке выбора точки входа (на этапе S395), описанном со ссылкой на блок-схему последовательности операций, показанную на фиг.64. Однако index_N_minus1 был передан в модуль 214 управления декодированием. Таким образом, модуль 214 управления декодированием декодирует обозначенное количество эталонных

изображений, передает декодированные изображения на расположенный далее блок и обеспечивает отображение им эталонных изображений (на этапе S396).

После отображения эталонных изображений модуль 212 управления проигрывателем выбирает точку входа EP_map(), которая должна быть отображена следующей, повторяет указанную выше обработку и выводит изображения, предназначенные для воспроизведения, в режиме воспроизведения с быстрой перемоткой вперед (на этапе S397).

Когда выбирают точку входа в качестве места назначения перехода в EP_map(), для отображения, используют index_N_minus1. Далее будет описан способ использования index_N_minus1 в таком случае. Как описано выше, после считывается N_th_Ref_Picture_copy, index_N_minus1 представляет количество эталонных изображений, содержащихся в данных. В примере данных, показанном на фиг.31, поскольку index_N_minus1 каждой из первых и третьей точек входа, равен 0, количество внутренних изображений равно единице. Поскольку index_N_minus1 каждой из второй и четвертой точек входа равен 3, здесь содержатся четыре эталонных изображения.

Когда выводят два или больше эталонных изображений, проявляется тенденция улучшения субъективного качества изображений, воспроизводимых в режиме воспроизведения с быстрой перемоткой вперед. Однако для вывода большого количества эталонных изображений необходимо увеличить количество считываемых данных. Таким образом, частота обновления становится низкой. Другими словами, существует взаимосвязь на основе компромисса между ними. Таким образом, когда модуль 212 управления проигрывателем выбирает точку входа в EP_map(), которая должна отображаться следующей, модуль 212 управления проигрывателем оценивает значение index_N_minus1.

Другими словами, при высокой скорости воспроизведения с быстрой перемоткой вперед, хотя разреженные интервалы точек входа в EP_map() становятся большими, предпочтительно выбирают точку входа, index_N_minus1 которой является большим (а именно, высокое субъективное качество изображения). В отличие от этого, когда скорость воспроизведения с быстрой перемоткой вперед низкая, выбирают точку входа с малым значением index_N_minus1.

В описанном выше алгоритме определения index_N-minus1 выбирают точку входа, значение N-th_Ref_picture_copy которой близко к "30". Другими словами, в EP_map(), созданном в соответствии с таким алгоритмом, когда данные считывают в соответствии с N-th_Ref_picture_copy, количество считанных данных становится равным "приблизительно 30" секторов. Когда скорость считывания данных является доминирующей, важно, чтобы время считывания оставалось постоянным. Таким образом, эффективно использовать такой способ выбора точек входа.

Далее, со ссылкой на блок-схему последовательности операций, показанную на фиг.64, будет описана обработка выбора точек входа в режиме воспроизведения с быстрой перемоткой вперед.

На этапе S401 модуль 212 управления проигрывателем определяет, является ли режим воспроизведения с быстрой перемоткой вперед режимом с высокой скоростью или режимом с низкой скоростью. Когда результат определения обозначает, что режим воспроизведения с быстрой перемоткой вперед представляет собой режим с высокой скоростью, поток обработки переходит на этап S402. В отличие от этого, когда определенный результат обозначает, что режим воспроизведения с быстрой перемоткой вперед представляет собой режим с низкой скоростью, поток обработки переходит на этап S411.

<Описание выбора воспроизведения с быстрой перемоткой вперед с низкой скоростью>

На этапе S402, поскольку режим воспроизведения с быстрой перемоткой вперед представляет собой режим с низкой скоростью, модуль 212 управления проигрывателем выполняет последовательное приращение номера точки входа, которая должна быть выбрана (текущая точка входа (N) на 2 после последней точки входа (точка входа +=2).

На этапе S403 модуль 212 управления проигрывателем считывает index_N_minus1 из текущей точки входа N, входа (n-1), предшествующего на одну точку, и входа (N+1), следующего на одну точку позже.

На этапе S404 модуль 212 управления проигрывателем определяет, равно ли значение index_N_minus (N), а именно значение index_N_minus1 N-й точки входа, 0 или 1. Когда значение index_N_minus1 на этапе S404 равно 0 или 1, поток обработки переходит на этап S405. На этапе S405 модуль 212 управления проигрывателем выбирает N-ю точку входа и завершает обработку. В отличие от этого, когда значение index_N_minus1 на этапе S404 не равно ни 0, ни 1, поток обработки переходит на этап S406.

На этапе S406 модуль 212 управления проигрывателем определяет, равно ли значение index_N_minus (N+1), а именно значение index_N_minus1 для (N+1)-й точки входа, 0 или 1. На этапе S406, когда index_N_minus (N+1) равен 0 или 1, поток обработки переходит на этап S407. На этапе S407 модуль 212 управления проигрывателем выбирает (N+1)-ю точку входа и заканчивает обработку. В отличие от этого, когда index_N_minus (N+1) на этапе S406 не равен ни 0, ни 1, поток обработки переходит на этап S408.

На этапе S408 модуль 212 управления проигрывателем определяет, равно ли значение index_N_minus (n-1), а именно значение index_N_minus1 для (n-1)-й точки входа, 0 или 1. Когда значение index_N_minus1 на этапе S408 равно 0 или 1, поток обработки переходит на этап S409. На этапе S409 модуль 212 управления проигрывателем выбирает (n-1)-ю точку входа и завершает обработку. В отличие от этого, когда значение index_N_minus1 на этапе S408 не является ни 0, ни 1, поток обработки переходит на этап S410.

Поскольку очевидно, что значения index_N_minus1 всех точек N, (N+1) и (n-1) входа не равны ни 0, ни 1, на этапе S410, модуль 212 управления проигрывателем выбирает N-ю точку входа и завершает обработку.

<Описание выбора воспроизведения быстрой перемотки вперед, выполняемой с высокой скоростью>

Поскольку режим воспроизведения с быстрой перемоткой вперед выполняется в режиме высокой скорости, на этапе S411, модуль 212 управления проигрывателем выполняет последовательное приращение количества точек входа, которые будут выбраны (текущая точка входа) (N) на 5, начиная с последней точки входа (соответствующее количество точек входа +=5). На этапе S412 модуль 212 управления проигрывателем считывает index_N_minus1 из текущей точки входа N, из точки входа, следующей на одну точку раньше, 1 (N-1), и входа (N+1), следующего на одну точку позже.

На этапе S413 модуль 212 управления проигрывателем определяет, равно ли значение index_N_minus (N), а именно значение index_N_minus1 для N-й точки входа, 3 или 2. Когда значение index_N_minus1 для N-й точки входа на этапе S413 равно 3 или 2, поток обработки переходит на этап S414. На этапе S414 модуль 212 управления проигрывателем выбирает N-ю точку входа и завершает обработку. В отличие от этого, когда значение index_N_minus1 N-й точки входа на этапе S413 не равно ни 3, ни 2, поток обработки переходит на этап S415.

На этапе S415 модуль 212 управления проигрывателем определяет, равно ли значение index_N_minus (N+1), а именно значение index_N_minus1 для (N-1)-й точки входа 3 или 2. Когда значение index_N_minus1 для (N+1)-й точки входа на этапе S415 равно 3 или 2, поток обработки переходит на этап S416. На этапе S416 модуль 212 управления проигрывателем выбирает (N-1)-ю точку входа и завершает обработку. В отличие от этого, когда значение index_N_minus1 для (N-1)-й точки входа на этапе S415 не равно ни 3, ни 2, поток обработки переходит на этап S417.

На этапе S417 модуль 212 управления проигрывателем определяет, является ли значение index_N_minus (n-1), а именно значение index_N_minus1 для (n-1)-й точки входа, 3 или 2. Когда значение index_N_minus1 для (n-1)-й точки входа на этапе S417 равно 3 или 2, поток обработки переходит на этап S418. На этапе S418 модуль 212 управления проигрывателем выбирает (n-1)-ю точку входа и завершает обработку. В

отличие от этого, когда значение index_N_minus1 для (n-1)-й точки входа на этапе S417 не равен ни 3, ни 2, поток обработки переходит на этап S419.

Поскольку очевидно, что значения index_N_minus1 для всех точек входа N, (N+1) и (n-1) не равны ни 3, ни 2, на этапе S419, модуль 212 управления проигрывателем выбирает N-ю точку входа и завершает обработку.

Другими словами, когда скорость воспроизведения с быстрой перемоткой вперед слишком высока, хотя интервалы разреживания точек входа в EP_map() становятся большими, предпочтительно выбирают точку входа, index_N_minus1 которой является большим (а именно, высоко субъективное качество изображения). В отличие от этого, когда скорость воспроизведения при быстрой перемотке вперед мала, выбирают точку входа с малым index_N_minus1.

В предыдущей обработке воспроизведение с быстрой перемоткой вперед может быть выполнено с высокой скоростью без ухудшения субъективного качества изображения. Когда воспроизведение с быстрой перемоткой вперед выполняют с малой скоростью, поскольку воспроизведение с быстрой перемоткой вперед выполняют с большим количеством эталонных изображений, чем при высокой скорости, можно исключить снижение качества воспроизводимых изображения.

В приведенном выше примере предполагается, что информация о считываемых точках входа обычно является постоянной при обработке воспроизведения. В этом случае в устройстве воспроизведения диска, которое обладает высокими рабочими характеристиками, поскольку считывают множество точек входа, качество воспроизводимых изображений улучшается. Однако, если устройство воспроизведения диска, которое не имеет высоких характеристик обработки, считывает большое количество информации точек входа, может снизиться скорость обработки. Таким образом, устанавливают уровни приоритета для считываемой информации в точках входа. Устройство воспроизведения диска, которое имеет высокие характеристики обработки, может использовать информацию по всем точкам входа, тогда как устройство воспроизведения диска, которое имеет низкие характеристики обработки, может считывать только точки входа, имеющие высокие уровни приоритета.

На фиг.65 показана функциональная блок-схема, описывающая функции устройства записи диска, которое устанавливает уровни приоритета для точек входа. В устройстве записи диска, показанном на фиг.65, функции, аналогичные функциям устройства записи диска, показанного на фиг.61, будут обозначены аналогичными номерами ссылочных позиций, и их описание не будет приведено.

Кодер 443 субтитров считывает материал субтитров из сервера 442 материала субтитров, кодирует со сжатием этот материал, и записывает полученные в результате данные в сервер 444 данных субтитров. В отличие от видеоданных и данных звука, данные субтитров расположены с чередованием по временной оси. Таким образом, время начала отображения и длительность отображения материала субтитров, записанного в сервере 442 материала субтитров, и также данных субтитров, записанных в сервере 444 данных субтитров, обеспечиваются как информация 445 времени отображения субтитров.

Хотя модуль 441 мультиплексирования, в основном, имеет те же функции, что и модуль 421 мультиплексирования, показанный на фиг.61, последний также мультиплексирует данные субтитров и информацию о времени представления субтитров с видеоданными и данными звука. Другими словами, модуль 441 мультиплексирования считывает не только входной видеопоток MPEG2 и поток звука MPEG2, но также данные субтитров и информацию 445 времени представления субтитров, передаваемую из сервера 444 данных субтитров, и мультиплексирует их на основе с разделения времени, как описано со ссылкой на фиг.18A и фиг.18B - фиг.27.

Модуль 441 мультиплексирования выбирает изображение с внутренним кодированием из потока и вставляет PES_packet() для private_stream_2, показанного на фиг.23 в него, с частотой приблизительно два раза в секунду. В данном примере PES_packet() private_stream_2 обозначает, что после непосредственно следует изображение с

внутренним кодированием, состоящее из видеоданных, которые могут быть декодированы без необходимости другого эталонного изображения. В этой точке изображение с внутренним кодированием обычно имеет временную метку (PTS/DTS).

Все модули доступа данных субтитров обычно имеют временную метку.

В этой точке данные не были записаны в полях 1stRef_picture, 2ndRef_picture, 3rdRef_picture и 4thRef_picture, описанных со ссылкой на фиг.26. Кроме того, поток субтитров (не показан) может быть введен в модуль 441 мультиплексирования так, что его мультиплексируют с видеопотоком и потоком звука.

Модуль 441 мультиплексирования передает мультиплексированный поток в модуль 424 перезаписи RAPI через ППО 422, а также в модуль 423 выделения информации RAPI. Модуль 423 выделения информации RAPI выделяет информацию о потоке видеоданных и информацию о потоке субтитров из мультиплексированного потока, и сохраняет выделенные потоки. Другими словами, модуль 423 выделения информации RAPI детектирует начальное положение PES_packet() для private_stream_2, значения временной метки (PTS) изображения с внутренним кодированием, которому непосредственно предшествует PES_packet() private_stream_2, и конечные положения изображения с внутренним кодированием и второго, третьего и четвертого эталонных изображений, которым предшествует изображение с внутренним кодированием, из потока видеоданных и сохраняет их. Кроме того, модуль 423 выделения информации RAPI детектирует начальные положения и временные метки всех модулей доступа к субтитрам из потока субтитров.

Контроллер 446 создает EP_map(), как показано на фиг.66, с входной информацией. Предполагается, что EP_map() содержит информацию о потоке видеоданных и потоке субтитров. Основная информация о видеоданных в EP_map() представляет собой положения всех RAPI, а именно PES_packet() private_stream_2, и временные метки внутренних изображений, которым непосредственно предшествует PES_packet() для private_stream_2. Эта информация может быть создана с информацией, которую вводят из модуля 423 выделения информации RAPI в контроллер 446. Основная информация о субтитре в EP_map() представляет собой положения и временные метки модулей доступа субтитров. Эта информация также может быть создана с информацией, которую вводят из модуля 423 выделения информации RAPI в контроллер 446.

Контроллер 446 создает priority_flag, который не был определен в информации EP_map() с данными, вводимыми из модуля 423 выделения информации RAPI. Другими словами, контроллер 446 выполняет оценку временных меток всех точек входа потока видеоданных (точек входа RAPI) и модулей доступа субтитров и устанавливает priority_flag (который будет описан ниже) для каждой из них. Для установки priority_flag информация 447 об изменении сцены раздела была введена в контроллер 446.

[Описание EP_map]

Далее, со ссылкой на фиг.66, будет описан EP_map(), с помощью которого устанавливают уровни приоритетов для файлов. Как показано на фиг.66, после number_of_EP_entries следует priority_flag (2 бита) как информация о точке возможного начала декодирования элементарного потока, идентифицированного по stream_id и private_stream_id, которому непосредственно предшествуют number_of_EP_entries, reserved_for_future_use (14 бит), PTS_EP_start (32 бита) и RPN_EP_start (32 бита). priority_flag, reserved_for_future_use, PTS_EP_start (32 бита) и RPN_EP_start (32 бита) повторяются столько раз, как представлено number_of_EP_entries.

priority_flag имеет значение, показанное на фиг.67. Другими словами, для точки входа потока видеоданных, когда значение priority_flag равно 3, это обозначает, что точка входа соответствует началу раздела. Когда значение priority_flag равно 2, это означает, что эта точка входа соответствует важному изменению сцены через интервалы одна минута, другие, чем указанные выше точки входа. Когда значение priority_flag равно 1, это означает, что эта точка входа соответствует изменению сцены через интервалы три секунды, другие, чем указанные выше точки входа. Предполагается, что

значение priority_flag других точек входа равно 0.

Для точки входа потока субтитров, когда значение priority_flag равно 3, это означает, что эта точка входа соответствует началу раздела. Когда значение priority_flag равно 2, это означает, что точка входа соответствует важному другому изменению сцены, помимо указанной выше точки входа. Когда значение priority_flag равно 1, это означает, что эта точка входа соответствует изменению сцены, кроме указанных выше точек входа. Предполагается, что значение priority_flag других точек входа равно 0.

Когда клип представляет собой двухчасовой кинофильм и он содержит две точки оперативного доступа в секунду, общее количество точек входа составляет 14400 (=2 часа $\times$ 3600 секунд $\times$ 2 раза). Когда количество разделов составляет приблизительно несколько десятков, количество точек входа, для которых priority_flag=3, становится равным нескольким десяткам и равно количеству разделов. Поскольку количество важных изменений сцены (priority_flag=2) и количество других изменений сцены (priority_flag= 1) зависят от содержания, хотя и без обобщения, предполагается, что количество точек входа, для которых priority_flag=3 или 2, составляет приблизительно 200 и количество точек входа для которых priority_flag=3, 2 или 1, составляет около 2400, и общее количество точек входа составляет 14400. В этом случае также предполагается, что общее количество точек входа, для которых priority_flag=2 и 1, равно 1000. В этом случае предполагается что, когда считывают только точки входа, для которых priority_flag=3, 2 и 1, объем памяти становится равным приблизительно 1000/14 400, что составляет 1/14 в случае, когда считывают все точки входа. Кроме того, в этом случае, поскольку одна точка входа составляет 10 байт, емкость памяти может быть уменьшена для одного потока видеоданных на 10 байт $\times$ (14400-1000)=120134 килобайта.

Кроме того, считается, что двухчасовой кинофильм содержит 1000-2000 предложений субтитров. В отличие от этого имеется несколько десятков разделов. Таким образом, если предположить, что считывают только точки входа, для которых priority_flag=3, емкость памяти можно уменьшить на несколько десятков /1000 или на несколько десятков/ 2000. Поскольку количество потоков субтитров больше, чем количество потоков видеоданных, эффект уменьшения емкости памяти становится существенным.

В этом примере флаг представлен значениями 3, 2, 1 и 0. Вместо этого можно предположить, что он может быть представлен соответствующими битами и соответствующий бит может быть установлен равным 1. Другими словами, такое поле может состоять из трех битов. Когда старший значимый бит равен 1, это может означать, что эта точка входа представляет собой начало раздела. Когда следующий бит равен 1, это может означать, что эта точка входа представляет собой точку входа через интервалы одна минута. Когда наименьший значащий бит равен 1, это может означать, что эта точка входа представляет собой точку входа с интервалами в пять секунд. Когда все биты равны О, может быть определено, что данная точка входа не включена в эти три категории.

Для потока субтитров, когда значение priority_flag точки входа равно 1, это означает, что эта точка входа соответствует началу раздела. В этом случае предполагается, что значение priority_flag других точек входа равно 0.

Далее со ссылкой на блок-схему последовательности операций, показанную на фиг.68, будет описана обработка установки prioirty_flag.

На этапе S441 контроллер 446 определяет, является ли текущая точка входа видеоданных началом раздела, а именно, что оцениваемая точка входа соответствует времени раздела по информации 447 изменения сцены раздела. Когда текущая точка входа соответствует времени раздела, определяется, что разность равна 0. Когда текущая точка входа соответствует времени раздела, поток обработки переходит на этап S442. На этапе S442 контроллер 446 устанавливает priority_flag=3 для этой точки входа и сохраняет его в выходном сервере 426.

Когда результат, определенный на этапе S441, обозначает, что текущая точка входа не

является началом раздела видеоданных, поток обработки переходит на этап S443. На этапе S443 контроллер 446 определяет, является ли текущая точка входа положением важного изменения сцены видеоданных, а именно точкой входа, для которой будет выполнена следующая оценка, находится в положении через интервал одна минута от начала "важного изменения сцены" в информации 447 изменения сцены раздела. Когда результат, определенный на этапе S443, обозначает, что оцениваемая следующей точка входа находится в положении через интервал одна минута от начала "важного изменения сцены", поток обработки переходит на этап S444. На этапе S444 контроллер 446 устанавливает priority_flag=2 для текущей точки входа.

Когда результат, определенный на этапе S443, обозначает, что текущая точка входа не является важным изменением сцены, поток обработки переходит на этап S445. На этапе S445 контроллер 446 определяет, является ли текущая точка входа обычным изменением сцены видеоданных, а именно точка входа, оценка которой будет выполнена следующей, находится в положении через интервал три секунды от начала "изменения сцены" в информации 447 изменения сцены раздела. Когда результат, определенный на этапе S445, обозначает, что точка входа, которая будет следующей оценена, находится в положении через интервал три секунды от начала "изменения сцены", поток обработки переходит на этап S446. На этапе S446 контроллер 446 устанавливает priority_flag=1 для текущей точки входа.

Когда результат, определенный на этапе S445, обозначает, что текущая точка входа не является обычным изменением сцены, а именно текущая точка входа не соответствует какому-либо изменению сцены, поток обработки переходит на этап S447. На этапе S447 контроллер 446 устанавливает priority_flag=0 для текущей точки входа.

На этапе S448 контроллер 446 определяет, были ли обработаны все точки входа видеоданных. Когда определенный результат обозначает, что все точки входа не были обработаны, поток обработки возвращается на этап S441. На этапе S441 обработка повторяется. Другими словами, обработка повторяется от этапа S441 до этапа S448, пока priority_flag не будет установлен для всех точек входа видеоданных.

Когда результат, определенный на этапе S448, обозначает, что обработка была закончена для всех точек входа, поток обработки переходит на этап S449.

На этапе S449 контроллер 446 определяет, является ли текущая точка входа субтитров началом раздела, а именно точка входа, которую оценивают, соответствует времени раздела в информации 447 изменения сцены раздела. Когда результат, определенный на этапе S449, означает, что текущая точка входа соответствует времени раздела, поток обработки переходит на этап S450. На этапе S450 контроллер 446 устанавливает priority_flag=3 для текущей точки входа.

В отличие от этого, когда результат, определенный на этапе S449, означает, что текущая точка входа не находится в начале раздела субтитров, поток обработки переходит на этап S451. На этапе S451 контроллер 446 определяет, находится ли текущая точка входа в положении важного изменения сцены для субтитров, а именно точка входа, которую будут оценивать следующей, находится в положении через интервал одна минута от начала "важного изменения сцены" в информации 447 изменения сцены раздела. Когда результат, определенный на этапе S451, обозначает, что точка входа, которую будут оценивать следующей, находится в положении через интервал одна минута от начала "важного изменения сцены", поток обработки переходит на этап S452. На этапе S452 контроллер 446 устанавливает priority_flag=2 для текущей точки входа.

Когда результат, определенный на этапе S451, обозначает, что текущая точка входа не является важным изменением сцены, поток обработки переходит на этап S453. На этапе S453 контроллер 446 определяет, является ли текущая точка входа обычным изменением сцены субтитров, а именно точка входа, которая будет оценена следующей, находится в положении через интервал три секунды от начала "изменения сцены" в информации 447 изменения сцены раздела. Когда точка входа, которая будет оценена следующей, находится в положении через интервал три секунды от начала "изменения сцены", поток

обработки переходит на этап S454. На этапе S454 контроллер 446 устанавливает priority_flag=1 для текущей точки входа.

Когда результат, определенный на этапе S453, обозначает, что текущая точка входа не является обычным изменением сцены, а именно точка входа, которая будет оценена следующей, не является каким-либо изменением сцены, поток обработки переходит на этап S455. На этапе S455 контроллер 446 устанавливает priority_flag=0 для текущей точки входа.

На этапе S456 контроллер 446 определяет, были ли обработаны все точки входа субтитра. Когда результат, определенный на этапе S456, обозначает, что все точки входа не были обработаны, поток обработки возвращается на этап S449. На этапе S449 обработка повторяется. Другими словами, обработка повторяется от этапа S449 до этапа S456, до тех пор, пока priority_flag не будет установлен для всех точек входа субтитра. Когда результат, определенный на этапе S456, обозначает, что все точки входа субтитров были обработаны, контроллер 446 выводит данные EP_map() в соответствии с синтаксисом, показанным на фиг.66, в выходной сервер 426.

[Работа стороны воспроизведения: разрежение EP_map()]

Устройство воспроизведения диска выполняет разрежение EP_map() на основе priority_flag, который был установлен, как описано выше, и в пространстве накопителя запоминающего устройства (например, запоминающего устройства 113, показанного на фиг.1). Другими словами, в устройстве воспроизведения диска, которое имеет ограниченные функции из-за низкой стоимости, только точки входа, priority_flag которых имеет большое значение, сохраняют в запоминающем устройстве. Конечно, в устройстве, которое имеет память, которая позволяет полностью сохранять EP_map(), нет необходимости выполнять такую операцию.

На этапе S106 обработки в соответствии с блок-схемой последовательности операций, показанной на фиг.33, для видеоданных точки входа, для которых priority_flag=1 или выше, сохраняют в запоминающем устройстве. Аналогично, для субтитров точки входа, для которых priority_flag=1 или выше, сохраняют в запоминающем устройстве. В этом случае, когда считывают EP_map() для потока видеоданных, модуль 212 управления проигрывателем считывает точки входа, priority_flag которых равен 3, 2 или 1, из запоминающего устройства и не считывает точки входа, priority_flag которых равен 0, из запоминающего устройства на основе значения stream_id и private_stream_id. Для потока субтитров модуль 212 управления проигрывателем считывает точки входа, priority_flag которых равен 3 и 21, из запоминающего устройства и не считывает точки входа, priority_flag которых равен 1 или 0, из запоминающего устройства на основе значения stream_id и private_stream_id.

Когда выполнятся описанная выше обработка, емкость накопителя запоминающего устройства, необходимая для EP_map() для одного потока видеоданных, становится меньше на величину приблизительно от 1/6 до 1/10, по сравнению со случаем, когда эта обработка не выполняется. Кроме того, объем памяти, необходимый для EP_map(), для содержания одного потока субтитров, может быть уменьшен приблизительно на величину нескольких сотых. В результате в устройстве воспроизведения диска с малой стоимостью точки входа могут быть сохранены с учетом емкости запоминающего устройства. В результате обработка воспроизведения может быть эффективно выполнена.

В указанном выше примере, priority_flag=3 установлен для события в начале раздела. Вместо этого priority_flag=3 может быть установлен для любого значения, такого как важное изменение сцены, а также для начала раздела.

В указанном выше варианте выполнения последовательность обработки выполняется с помощью программных средств. Вместо этого она может быть выполнена с использованием специализированных аппаратных средств.

В указанном выше варианте выполнения в качестве видеодекодера 116 (фиг.1) и кодера 403 видеоданных (фиг.60) используется аппаратный декодер. Вместо этого в качестве видеодекодера 116 (фиг.1) и кодера 403 видеоданных (фиг.60) может использоваться

программный декодер. Это относится к декодеру 117 звука (фиг.1) и кодеру 404 звука (фиг.60).

В приведенном выше варианте выполнения в качестве декодера субтитров используется программный декодер. Вместо этого в качестве декодера субтитров может использоваться аппаратный декодер.

ОПИСАНИЕ НОМЕРОВ ССЫЛОЧНЫХ ПОЗИЦИЙ

101 ДИСК
102 ПРИВОД ДИСКА
111 ШИНА
112 ЦПУ
113 ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО
114 ИНТЕРФЕЙС ПРИВОДА
115 ВХОДНОЙ ИНТЕРФЕЙС
116 ВИДЕОДЕКОДЕР
116A МЕХАНИЗМ ДЕКОДИРОВАНИЯ ВИДЕОДАННЫХ
116B DPB (БУФЕР ДЕКОДИРОВАННОГО ИЗОБРАЖЕНИЯ)
ОТ 116B-0 ДО 116B-n ОТ DPB-0 ДО DPB-n
117 ДЕКОДЕР ЗВУКА
118 ИНТЕРФЕЙС ВЫВОДА ВИДЕОИЗОБРАЖЕНИЯ
119 ИНТЕРФЕЙС ВЫВОДА ЗВУКА
120 ВЫХОДНОЙ ВЫВОД ВИДЕОИЗОБРАЖЕНИЯ
121 ВЫХОДНОЙ ВЫВОД ЗВУКА
201 ОПЕРАЦИОННАЯ СИСТЕМА
210 ПРОГРАММА ВОСПРОИЗВЕДЕНИЯ ВИДЕОСОДЕРЖАНИЯ
211 МОДУЛЬ УПРАВЛЕНИЯ СЦЕНАРИЕМ
212 МОДУЛЬ УПРАВЛЕНИЯ ПРОИГРЫВАТЕЛЕМ
213 МОДУЛЬ ПОДАЧИ ДАННЫХ СОДЕРЖАНИЯ
214 МОДУЛЬ УПРАВЛЕНИЯ ДЕКОДИРОВАНИЕМ
214A БЛОК ПОДСЧЕТА ВРЕМЕНИ
215 МОДУЛЬ УПРАВЛЕНИЯ БУФЕРОМ
215A БУФЕР
216 МОДУЛЬ УПРАВЛЕНИЯ ВИДЕОДЕКОДЕРОМ
217 МОДУЛЬ УПРАВЛЕНИЯ ДЕКОДЕРОМ ЗВУКА
218 МОДУЛЬ УПРАВЛЕНИЯ ДЕКОДЕРОМ СУБТИТРОВ
219 МОДУЛЬ ОБРАБОТКИ ГРАФИЧЕСКИХ ИЗОБРАЖЕНИЙ
220 МОДУЛЬ ВЫВОДА ВИДЕОИЗОБАЖЕНИЯ 220A ПППО
221 МОДУЛЬ ВЫВОДА ЗВУКА
221 ПППО
231 БЛОК СОХРАНЕНИЯ УКАЗАТЕЛЯ НАЧАЛА ДАННЫХ
232 БЛОК СОХРАНЕНИЯ УКАЗАТЕЛЯ ЗАПИСИ ДАННЫХ
233 БЛОК ФУНКЦИИ СЧИТЫВАНИЯ ВИДЕОДАННЫХ
234 БЛОК ФУНКЦИИ СЧИТЫВАНИЯ ЗВУКА
235 БЛОК ФУНКЦИИ СЧИТЫВАНИЯ СУБТИТРОВ
241 БЛОК СОХРАНЕНИЯ УКАЗАТЕЛЯ СЧИТЫВАНИЯ ВИДЕОДАННЫХ
242 РЕГИСТР stream_id
243 РЕГИСТРИР au_information()
251 БЛОК СОХРАНЕНИЯ УКАЗАТЕЛЯ СЧИТЫВАНИЯ ЗВУКА
252 РЕГИСТР stream_id
253 РЕГИСТР private_stream_id
261 БЛОК СОХРАНЕНИЯ ФЛАГА ФУНКЦИИ СЧИТЫВАНИЯ СУБТИТРОВ
262 БЛОК СОХРАНЕНИЯ УКАЗАТЕЛЯ СЧИТЫВАНИЯ СУБТИТРОВ
263 РЕГИСТР stream_id
264 РЕГИСТР private_stream_id

301 БУФЕР ВИДЕОДАННЫХ
302 БУФЕР ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ
400A ВЫВОД ВВОДА ВИДЕОИЗОБРАЖЕНИЯ
S101 ПРОВЕРКА ДИСКА
S102 ОБРАБОТКИ ОШИБКИ
S103 СЧИТАТЬ SCRIPT.DAT И PLAYLIST.DAT.
S104 ПОДАТЬ ФАЙЛЫ
S105 АНАЛИЗИРОВАТЬ PLAYLIST.DAT.
S106 СЧИТАТЬ ФАЙЛ ИНФОРМАЦИИ КЛИПА
S107 ПРОВЕРИТЬ РЕЗУЛЬТАТ ФАЙЛА?
S108 ИНТЕРПРЕТИРОВАТЬ И ВЫПОЛНИТЬ SCRIPT.DAT.
S121 РАСПОЗНАТЬ IN_time.
S122 ПРОВЕРИТЬ ПОЛОЖЕНИЕ НАЧАЛА ВОСПРОИЗВЕДЕНИЯ
S123 ОТОБРАЗИТЬ КОД ВРЕМЕНИ
S124 АНАЛИЗИРОВАТЬ PlayList Mark()
S125 ОПРЕДЕЛИТЬ ПОТОК ДЛЯ ВОСПРОИЗВЕДЕНИЯ
S126 УПРАВЛЯТЬ ВЫХОДНЫМ АТРИБУТОМ
S127 ПОДГОТОВИТЬСЯ К НАЧАЛУ ВОСПРОИЗВЕДЕНИЯ
S128 НАЧАТЬ СЧИТЫВАНИЕ ДАННЫХ
S129 НАЧАТЬ УПРАВЛЕНИЕ ДЕКОДЕРОМ
S130 НАЧАТЬ ДЕКОДИРОВАНИЕ
S131 НАЧАТЬ ОБРАБОТКУ ГРАФИЧЕСКИХ ИЗОБРАЖЕНИЙ
S132 НАЧАТЬ ОБРАБОТКУ ВЫВОДА
S141 СОДЕРЖИТ ЛИ МОДУЛЬ ДОСТУПА ВРЕМЕННУЮ МЕТКУ?
S142 ЗАМЕНИТЬ ВРЕМЕННУЮ МЕТКУ (PST).
S143 СОХРАНИТЬ ТЕКУЩЕЕ ЗНАЧЕНИЕ pic_struct
S144 ДОБАВИТЬ ИНТЕРВАЛ В СООТВЕТСТВИИ С ПРЕДЫДУЩИМ pic_struct
S151 ДЕКОДИРОВАТЬ УПРАВЛЕНИЕ ОТМЕНОЙ
S152 НАЧАТЬ ВОСПРОИЗВЕДЕНИЕ ЭЛЕМЕНТОВ ВОСПРОИЗВЕДЕНИЯ
S171 ВЫВЕСТИ СООБЩЕНИЕ
S172 ПРЕОБРАЗОВАТЬ ВРЕМЕННОЙ КОД
S173 ВЫВЕСТИ ВРЕМЕННОЙ КОД
S191 ПРОВЕРИТЬ КОЛИЧЕСТВО ПОТОКОВ
S192 ИДЕНТИФИЦИРОВАТЬ ВОСПРОИЗВОДИМЫЙ ПОТОК
S193 РАСПОЗНАТЬ ПОТОК, КОТОРЫЙ БУДЕТ ВОСПРОИЗВОДИТЬСЯ СЛЕДУЮЩИМ
S194 ОБЕСПЕЧИТЬ СЧИТЫВАНИЕ СЛЕДУЮЩЕГО ПОТОКА
S211 ВЫПОЛНИТЬ ПОИСК ПАКЕТА private_stream_2.
S212 ОБОЗНАЧЕННЫЙ steam_id?
S213 СЧИТАТЬ au_Information() ВО ВНУТРЕННИЙ РЕГИСТР
S214 ВСТАВИТЬ НАЧАЛО СЛЕДУЮЩЕГО ВИДЕОПАКЕТА В УКАЗАТЕЛЬ СЧИТЫВАНИЯ ВИДЕОДАННЫХ
S215 ЗАПРОШЕНЫ ЛИ ДАННЫЕ?
S216 ПРОАНАЛИЗИРОВАТЬ ПРОГРАММНЫЙ ПОТОК И ВЫВЕСТИ ВИДЕОДАННЫЕ ПО AU_length.
S217 ОБРАБОТАНЫ ЛИ МОДУЛИ ДОСТУПА ДЛЯ number_of_access_unit?
S230 private_stream_1?
S231 ВЫПОЛНИТЬ ПОИСК КОДА СИНХРОНИЗАЦИИ
S232 ВЫПОЛНИТЬ ПОИСК ПАКЕТА ЗВУКОВЫХ ДАННЫХ MPEG
S233 ОБНОВИТЬ УКАЗАТЕЛЬ СЧИТЫВАНИЯ ЗВУКА
S234 ВЫПОЛНИТЬ ПОИСК ПАКЕТА private_stream_1
S235 ОБОЗНАЧЕННЫЙ private_stream_id?
S236 УКАЗАТЕЛЬ 234 СЧИТЫВАНИЯ ЗВУКА ← ПОЛОЖЕНИЕ НЕПОСРЕДСТВЕННО ПОСЛЕ AU_locator+AU_locator

S237 ЗАПРОШЕНЫ ЛИ ДАННЫЕ?
S238 АНАЛИЗИРОВАТЬ ПРОГРАММНЫЕ ПОТОКИ И ВЫВЕСТИ ДОСТУП, ИМЕЮЩИЙ ФИКСИРОВАННУЮ ДЛИНУ
S251 ФЛАГ ФУНКЦИИ СЧИТЫВАНИЯ СУБТИТРА?
S252 ВЫПОЛНИТЬ ПОИСК ПАКЕТА private_stream_1
S253 ОБОЗНАЧЕННЫЙ private_stream_id?
S254 УКАЗАТЕЛЬ СЧИТЫВАНИЯ СУБТИТРА ← ПОЛОЖЕНИЕ НЕПОСРЕДСТВЕННО ПОСЛЕ AU_locator+AU_locator
S255 ЗАПРОШЕНЫ ЛИ ДАННЫЕ?
S256 ПРОАНАЛИЗИРОВАТЬ ПРОГРАММНЫЕ ПОТОКИ И ВЫВЕСТИ МОДУЛЬ ДОСТУПА СУБТИТРОВ НА ДЛИНУ, ОПИСАННУЮ В ЕГО НАЧАЛЕ
S271 БОЛЬШАЯ РАЗНОСТЬ ВРЕМЕННЫХ МЕТОК?
S272 ВИДЕОДАННЫЕ ЗАПАЗДЫВАЮТ ОТНОСИТЕЛЬНО ЗВУКА?
S273 ОБЕСПЕЧИТЬ ПРОПУСК МОДУЛЯ ДОСТУПА К ВИДЕОДАННЫМ
S274 ПРОВЕРИТЬ au_ref_flag ДЛЯ МОДУЛЯ ДОСТУПА
S275 МОДУЛЬ ДОСТУПА НЕ ЯВЛЯЕТСЯ ЭТАЛОННЫМ ИЗОБРАЖЕНИЕМ?
S276 ОБРАБОТАТЬ МОДУЛЬ ДОСТУПА К ВИДЕОДАННЫМ
S277 ПРОПУСТИТЬ ОБРАБОТКУ ДЛЯ МОДУЛЯ ДОСТУПА К ВИДЕОДАННЫМ
S278 ОБЕСПЕЧИТЬ НЕПРЕРЫВНЫЙ ВЫВОД ВИДЕОИЗОБРАЖЕНИЯ
S279 НЕПРЕРЫВНЫЙ ВЫВОД ВИДЕОИЗОБРАЖЕНИЯ
S301 ТЕКУЩЕЕ ВРЕМЯ СООТВЕТСТВУЕТ mark_time_stamp?
S302 РАСПОЗНАТЬ ЦЕЛЕВУЮ метку
S303 ОПИСЫВАЕТ ЛИ ЦЕЛЕВАЯ метка entry_ES_stream_id/entry_ES_private_stream_id?
S304 СОДЕРЖИТ ЛИ ВОСПРОИЗВОДИМЫЙ ПОТОК ПОТОК ИЗ entry_ES_stream_id/entry_ES_private_stream_id?
S305 mark_type ЦЕЛЕВОЙ метки?
S306 ОБРАБОТКА ОБНОВЛЕНИЯ ДЛЯ РАЗДЕЛА/ИНДЕКСА
5307 ИНФОРМИРОВАТЬ МОДУЛЬ 211 УПРАВЛЕНИЯ СЦЕНАРИЕМ О СООБЩЕНИИ О СОБЫТИИ И mark_data
S308 ОБРАБОТАТЬ mark_data
S320 ПЕРЕДАТЬ pts_change_point
S321 ТЕКУЩЕЕ ВРЕМЯ СООТВЕТСТВУЕТ pts_change_point?
S322 РАСПОЗНАТЬ DynamicInfo() ДЛЯ pts_change_point, КОТОРАЯ СООТВЕТСТВУЕТ ТЕКУЩЕМУ ВРЕМЕНИ
S323 ПЕРЕДАТЬ ВЫХОДНОЙ АТРИБУТ, ОПИСАННЫЙ В DynamicInfo().
S324 НАЧАТЬ УПРАВЛЕНИЕ ВЫВОДОМ ВИДЕО/АУДИО В СООТВЕТСТВИИ С ВЫХОДНЫМ АТРИБУТОМ
S341 ОБЕСПЕЧИТЬ ИНИЦИАЛИЗАЦИЮ РЕЖИМА ОТОБРАЖЕНИЯ СУБТИТРОВ
S342 БЫЛА ЛИ ВВЕДЕНА НОВАЯ КОМАНДА РЕЖИМА ОТОБРАЖЕНИЯ ДЛЯ СУБТИТРА?
S343 ВОСПРОИЗВОДИТСЯ ЛИ ПОТОК СУБТИТРОВ?
S345 КОМАНДА РЕЖИМА ОТОБРАЖЕНИЯ, ПРИНЯТОГО ПО УМОЛЧАНИЮ?
S346 ПОЛУЧИТЬ StaticInfo() ДЛЯ ВОСПРОИЗВОДИМОГО ПОТОКА СУБТИТРОВ
S347 ОПРЕДЕЛИТЬ configurable_flag ДЛЯ StaticInfo()?
S348 ОТОБРАЗИТЬ СООБЩЕНИЕ ОБ ОШИБКЕ
S349 ПЕРЕДАТЬ КОМАНДУ РЕЖИМА ОТОБРАЖЕНИЯ
S350 НАЧАТЬ ОБРАБОТКУ ОТОБРАЖЕНИЯ СУБТИТРОВ В СООТВЕТСТВИИ С КОМАНДОЙ РЕЖИМА ОТОБРАЖЕНИЯ
S351 ИЗМЕНИЛИСЬ ЛИ PlayItem?
S371 ВОСПРОИЗВОДЯТСЯ ЛИ ВИДЕОДАННЫЕ?
S372 ПОЛУЧИТЬ capture_enable_f ИЗ PlayList() И Clip() ВОСПРОИЗВОДИМЫХ ВИДЕОДАННЫХ
S373 РАЗРЕШЕН ЛИ ЗАХВАТ ВИДЕОДАННЫХ В СООТВЕТСТВИИ С

capture_enable_flag?
S374 ОТОБРАЗИТЬ СООБЩЕНИЕ ОБ ОШИБКЕ
S375 ПЕРЕДАТЬ КОМАНДУ НА ЗАХВАТ
S376 ЗАХВАТИТЬ И СОХРАНИТЬ ВИДЕОДАННЫЕ
S380 ОТОБРАЗИТЬ ИЗОБРАЖЕНИЕ
S381 КОДИРОВАТЬ
S382 МУЛЬТИПЛЕКСИРОВАТЬ И ВЫБРАТЬ ВНУТРЕНЕЕ ИЗОБРАЖЕНИЕ
S383 ДЕТЕКТИРОВАТЬ ПОЛОЖЕНИЯ ИЗОБРАЖЕНИЯ С ВНУТРЕННИМ КОДИРОВАНИЕМ И СЛЕДУЮЩЕГО ИЗОБРАЖЕНИЯ.
S384 ПЕРЕЗАПИСАТЬ ИНФОРМАЦИЮ RAPI
S385 КОПИРОВАТЬ ПОЛОЖЕНИЕ ИЗОБРАЖЕНИЯ, БЛИЗКОГО К 30 СЕКТОРАМ В N-th_Ref_picture_copy В СООТВЕТСТВИИ С ИНФОРМАЦИЕЙ RAPI.
S386 ОПРЕДЕЛИТЬ И ЗАПИСАТЬ index_N_minus1 В СООТВЕТСТВИИ С N-th_Ref_picture_copy.
S391 ВЫБРАТЬ БЛОК, ПРИГОДНЫЙ ДЛЯ ВОСПРОИЗВЕДЕНИЯ
S392 ОПРЕДЕЛИТЬ ДАННЫЕ ДЛЯ СЧИТЫВАНИЯ В СООТВЕТСТВИИ С RPN_EP_Start И N-th_Ref_picture_copy.
S393 СЧИТАТЬ ПОТОК КЛИПА, СОДЕРЖАЩИЙ ПРОГРАММНЫЙ ПОТОК, В КОТОРОМ БЫЛ МУЛЬТИПЛЕКСИРОВАН ЭЛЕМЕНТАРНЫЙ ПОТОК, ПРЕДНАЗНАЧЕННЫЙ ДЛЯ ВОСПРОИЗВЕДЕНИЯ?
S394 ДЕМУЛЬТИПЛЕКСИРОВАТЬ ВИДЕОПОТОК
S395 ВЫПОЛНИТЬ ОБРАБОТКУ ВЫБОРА ТОЧКИ ВХОДА
S396 ДЕКОДИРОВАТЬ И ОТОБРАЗИТЬ ЭТАЛОННОЕ ИЗОБРАЖЕНИЕ
S397 БЫЛА ЛИ ПОДАНА КОМАНДА ОСТАНОВКИ ВОСПРОИЗВЕДЕНИЯ ПРИ БЫСТРОЙ ПЕРЕМОТКЕ ВПЕРЕД?
S401 ВЫСОКАЯ СКОРОСТЬ?
S402 КОЛИЧЕСТВО ТОЧЕК ВХОДА (N)+=2
S403 СЧИТАТЬ index_N_minus1 ДЛЯ КАЖДОЙ ИЗ ТОЧЕК ВХОДА N, N+1 И N-1.
S404 index_N_minus1 (N)==0 ИЛИ 1?
S405 ИСПОЛЬЗОВАТЬ N
S406 index_N_minus1 (N+1)==0 ИЛИ 1?
S407 ИСПОЛЬЗОВАТЬ N+1
S408 index_N_minus1 (n-1)=0 ИЛИ 1?
S409 ИСПОЛЬЗОВАТЬ N-1
S410 ИСПОЛЬЗОВАТЬ N
S411 НОМЕР (N) ТОЧКИ ВХОДА +=5
S412 СЧИТАТЬ index_N_minus1 ДЛЯ КАЖДОЙ ИЗ ТОЧЕК ВХОДА N, N+1 И N-1
S413 index_N_minus1 (N)==3 ИЛИ 2?
S414 ИСПОЛЬЗОВАТЬ N
S415 index_N_minus1 (N+1)=3 ИЛИ 2?
S416 ИСПОЛЬЗОВАТЬ N+1
S417 index_N_minus1 (n-1)==3 ИЛИ 2?
S418 ИСПОЛЬЗОВАТЬ N-1
S419 ИСПОЛЬЗОВАТЬ N
S441 НАЧАЛО РАЗДЕЛА ВИДЕОДАННЫХ?
S442 priority_flag=3
S443 ВАЖНОЕ ИЗМЕНЕНИЕ СЦЕНЫ ВИДЕОДАННЫХ?
S444 priority_flag=2
S445 ОБЫЧНОЕ ИЗМЕНЕНИЕ СЦЕНЫ ВИДЕОИЗОБРАЖЕНИЯ?
S446 priority_flag=1
S447 priority_flag=0
S448 ВСЕ ЗАКОНЧЕНО?
S449 НАЧАЛО РАЗДЕЛА СУБТИТРА?

S450 priority_flag=3
S451 ВАЖНОЕ ИЗМЕНЕНИЕ СЦЕНЫ СУБТИТРА?
S452 priority_flag=2
S453 ОБЫЧНОЕ ИЗМЕНЕНИЕ СЦЕНЫ СУБТИТРА?
S454 priority_flag=1
S455 priority_flag=0
S456 ВСЕ ЗАКОНЧЕНО?

Формула изобретения

1. Устройство записи данных, содержащее: блок детектирования, который выбирает информацию об изображении с внутренним кодированием, содержащемся в видеоданных, и которая становится точкой оперативного доступа и детектирует информацию положения множества изображений, перед которыми расположено выбранное изображение с внутренним кодированием; блок обозначения, который обозначает информацию положения изображения, расположенного на заданном расстоянии от изображения с внутренним кодированием в информации положения множества изображений, детектированных блоком детектирования; блок записи, который записывает информацию положения множества изображений, детектированных блоком детектирования, и информацию об изображении, обозначенном блоком обозначения, на носителе записи данных, вместе с видеоданными, и блок выделения, который выделяет информацию положения для изображения, соответствующую заданному количеству секторов носителя записи данных, в информации положения множества изображений, детектированных блоком детектирования, в котором блок обозначения обозначает информацию положения изображения, расположенного на заданном расстоянии от изображения с внутренним кодированием в информации положения множества изображений, детектированных блоком детектирования, в соответствии с информацией положения изображения, соответствующей заданному количеству секторов в информации положения множества изображений, выделенных блоком выделения, и в котором блок записи записывает информацию положения изображения, соответствующую заданному количеству секторов носителя записи данных, в информации положения множества изображений, выделенных блоком выделения.

2. Устройство записи данных по п.1, в котором заданное количество секторов установлено в соответствии с емкостью, которую устройство, воспроизводящее видеоданные с носителя записи данных, использует как буфер видеоданных.

3. Способ записи данных, содержащий следующие этапы: выбор информации об изображении с внутренним кодированием, которая содержится в видеоданных и становится точкой оперативного доступа, и детектирования информации положения множества изображений, перед которыми расположено выбранное изображение с внутренним кодированием; выделения информации положения для изображения, соответствующей заданному количеству секторов носителя записи данных, в информации положения множества изображений, обозначения информации положения изображения, расположенной на заданном расстоянии от изображения с внутренним кодированием в информации положения множества изображений, детектированных блоком детектирования, в соответствии с информацией положения изображения, соответствующей заданному количеству секторов в информации положения множества изображений, выделенных блоком выделения, и записи информации положения множества изображений, соответствующей заданному количеству секторов носителя записи данных, в информации положения множества изображений, выделенной на этапе выделения, и информации об изображении, обозначенной на этапе обозначения, на носителе записи данных, вместе с видеоданными.

4. Машиночитаемый носитель записи программы, на котором записана программа, причем программа обеспечивает выполнение компьютером следующих этапов: выбор информации об изображении с внутренним кодированием, которая содержится в видеоданных и становится точкой оперативного доступа, и детектирования информации

положения множества изображений, перед которыми расположено выбранное изображение с внутренним кодированием; выделения информации положения для изображения, соответствующей заданному количеству секторов носителя записи данных, в информации положения множества изображений, обозначения информации положения изображения, расположенной на заданном расстоянии от изображения с внутренним кодированием в информации положения множества изображений, детектированных блоком детектирования, в соответствии с информацией положения изображения, соответствующей заданному количеству секторов в информации положения множества изображений, выделенных блоком выделения, и записи информации положения множества изображений, соответствующей заданному количеству секторов носителя записи данных, в информации положения множества изображений, выделенной на этапе выделения, и информации об изображении, обозначенной на этапе обозначения, на носителе записи данных, вместе с видеоданными.

5. Устройство обработки данных, содержащее: блок считывания, который считывает информацию положения множества изображений с внутренним кодированием и множества изображений, перед которыми расположено множество изображений с внутренним кодированием, и информацию обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, перед которыми расположено множество изображений с внутренним кодированием, с носителя записи данных, вместе с видеоданными, причем множество изображений с внутренним кодированием содержится в видеоданных и становится точками оперативного доступа, блок выбора, который выбирает изображения с внутренним кодированием и множество изображений, которым предшествуют изображения с внутренним кодированием, в соответствии с информацией обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием во множестве изображений с внутренним кодированием, которые становятся точками оперативного доступа видеоданных, считанных блоком считывания и записанных на носитель записи данных, и множество изображений, которым предшествуют изображения с внутренним кодированием, блок воспроизведения, который выполняет воспроизведение с быстрой перемоткой вперед видеоданных со всеми изображениями от изображений с внутренним кодированием до изображений, расположенных на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, которым предшествуют изображения с внутренним кодированием, и которые были выбраны блоком выбора, и в котором изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, представляют собой изображения, которые расположены от изображений с внутренним кодированием приблизительно на заданное количество секторов носителя записи данных.

6. Устройство обработки данных по п.5, в котором заданное количество секторов носителя записи данных устанавливают в соответствии с емкостью блока памяти, которая используется в качестве буфера для видеоданных, когда видеоданные воспроизводят с носителя записи данных.

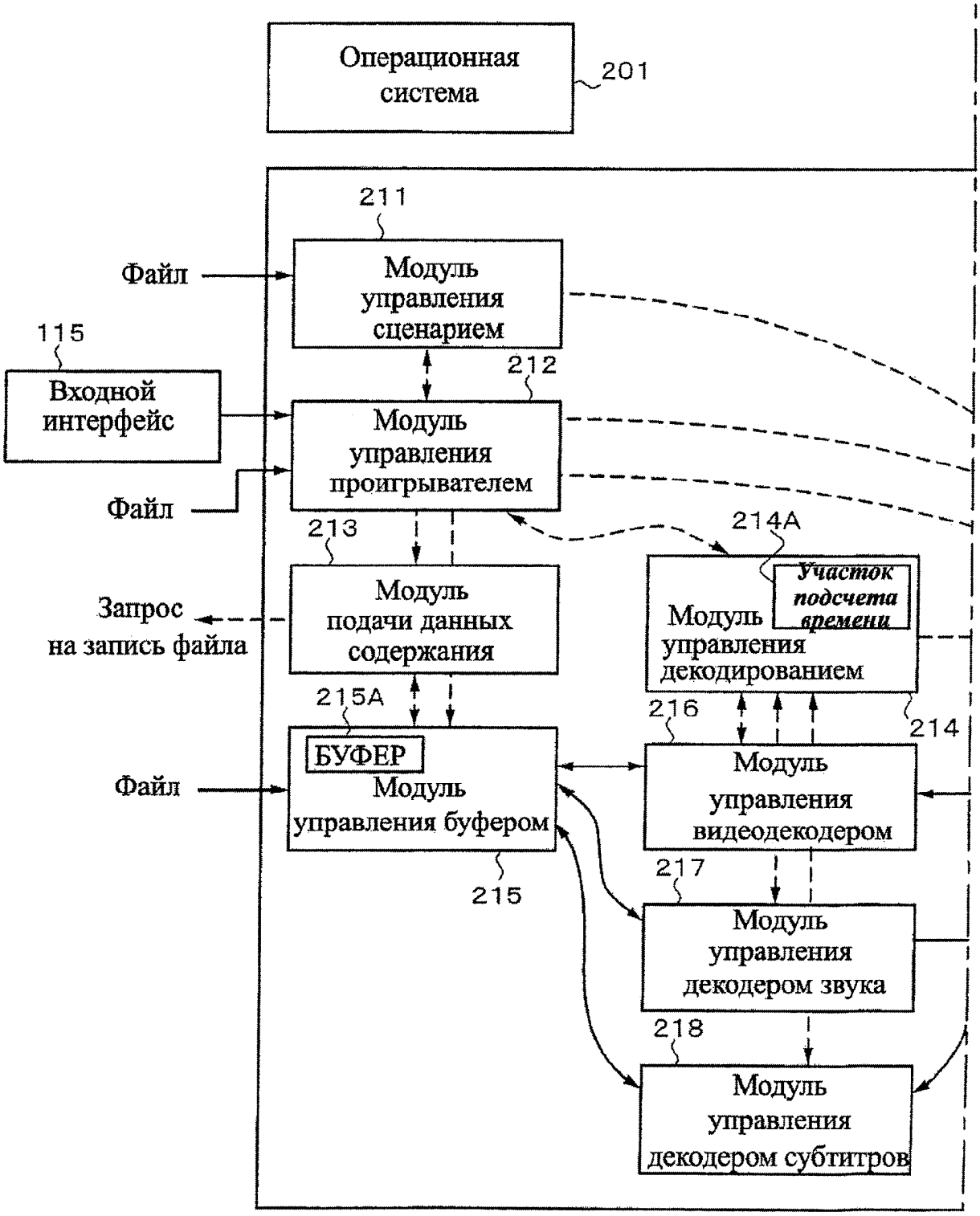
7. Способ обработки данных, содержащий следующие этапы: считывания информации положения множества изображений с внутренним кодированием и множества изображений, перед которыми расположено множество изображений с внутренним кодированием, и информации обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, перед которыми расположено множество изображений с внутренним кодированием, с носителя записи данных, вместе с видеоданными, причем множество изображений с внутренним кодированием, содержится в видеоданных и становится точками оперативного доступа; выбора изображений с внутренним кодированием и множества изображений, которым предшествуют изображения с внутренним кодированием, в соответствии с информацией обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений с внутренним

кодированием, которые становятся точками оперативного доступа видеоданных, считанных на этапе считывания и записанных на носителе записи данных, и множество изображений, которым предшествуют изображения с внутренним кодированием; и воспроизведения с быстрой перемоткой вперед видеоданных со всеми изображениями от изображений с внутренним кодированием до изображений, расположенных на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, которым предшествуют изображения с внутренним кодированием, и которые были выбраны на этапе выбора, при этом изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, представляют собой изображения, которые расположены от изображений с внутренним кодированием приблизительно на заданное количество секторов носителя записи данных.

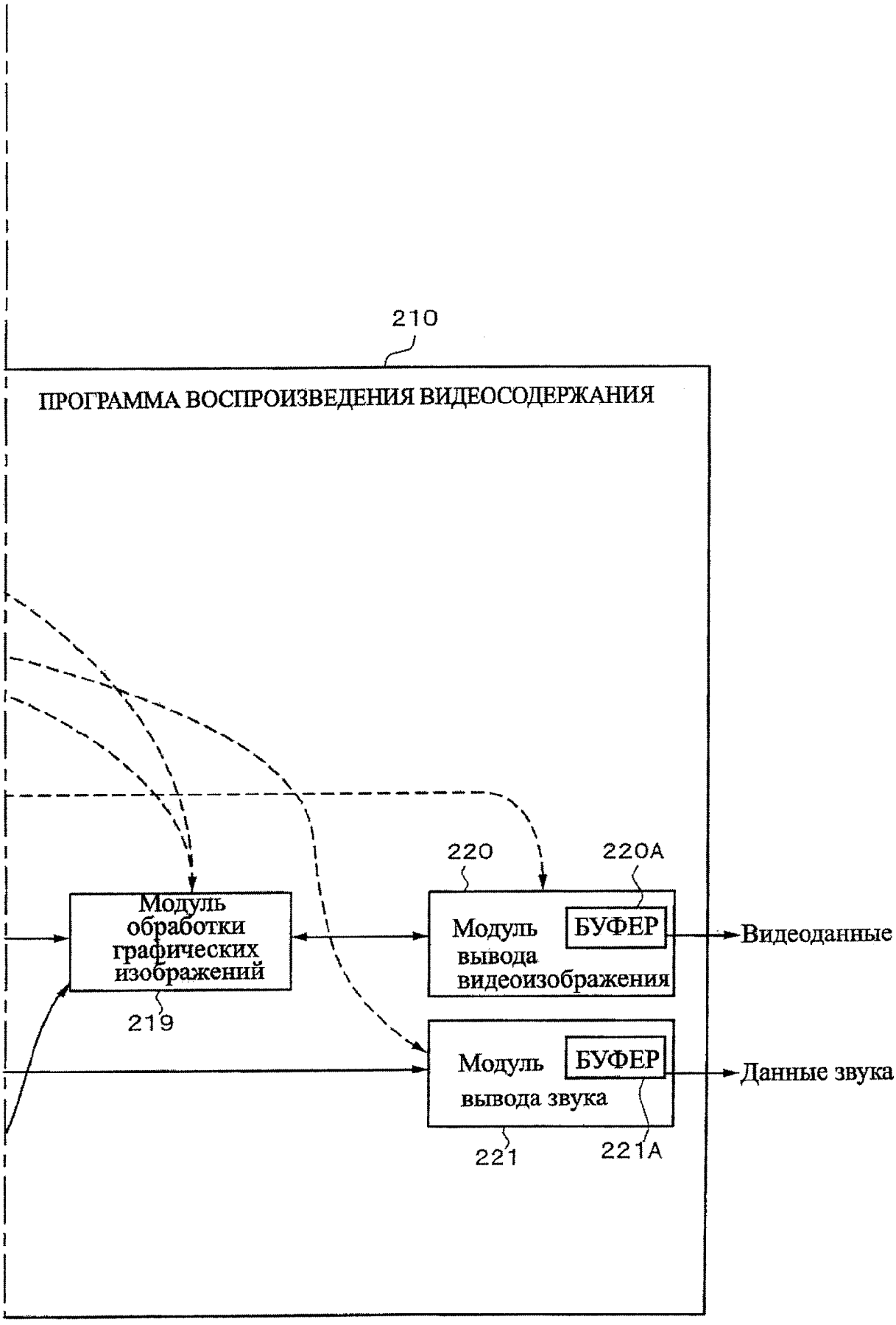
8. Машиночитаемый носитель записи программы, на котором записана программа, обеспечивающая выполнение компьютером следующих этапов: считывания информации положения множества изображений с внутренним кодированием и множества изображений, перед которыми расположено множество изображений с внутренним кодированием, и информации обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, перед которыми расположено множество изображений с внутренним кодированием, с носителя записи данных, вместе с видеоданными, причем множество изображений с внутренним кодированием содержатся в видеоданных и становится точками оперативного доступа; выбора изображений с внутренним кодированием и множества изображений, которым предшествуют изображения с внутренним кодированием, в соответствии с информацией обозначения, которая обозначает изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений с внутренним кодированием, которые становятся точками оперативного доступа видеоданных, считанных на этапе считывания и записанных на носителе записи данных, и множества изображений, перед которыми расположены изображения с внутренним кодированием; и воспроизведения с быстрой перемоткой вперед видеоданных со всеми изображениями, от изображений с внутренним кодированием, до изображений, расположенных на заданном расстоянии от изображений с внутренним кодированием, во множестве изображений, которым предшествуют изображения с внутренним кодированием, и которые были выбраны на этапе выбора, при этом изображения, расположенные на заданном расстоянии от изображений с внутренним кодированием, представляют собой изображения, которые расположены от изображений с внутренним кодированием приблизительно на заданное количество секторов носителя записи данных.

Фиг.2

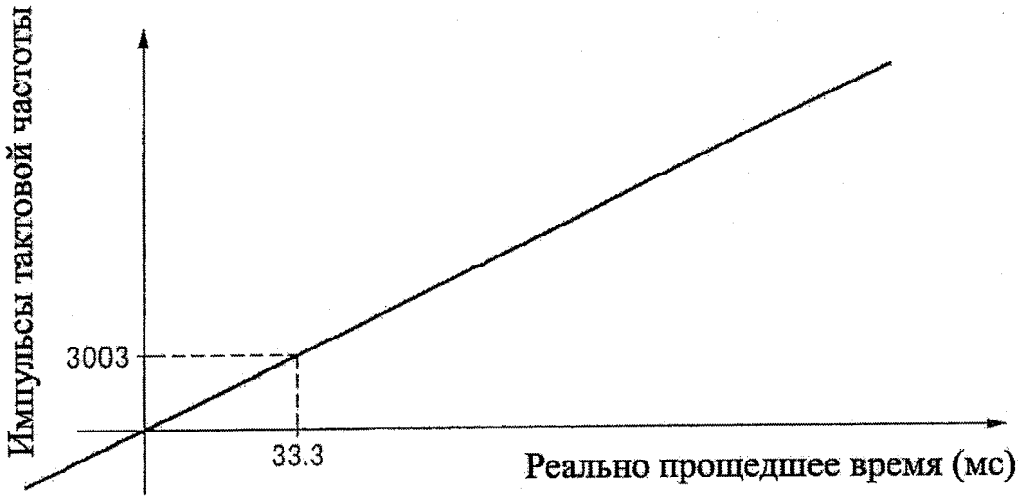
Фиг.2A | Фиг.2B



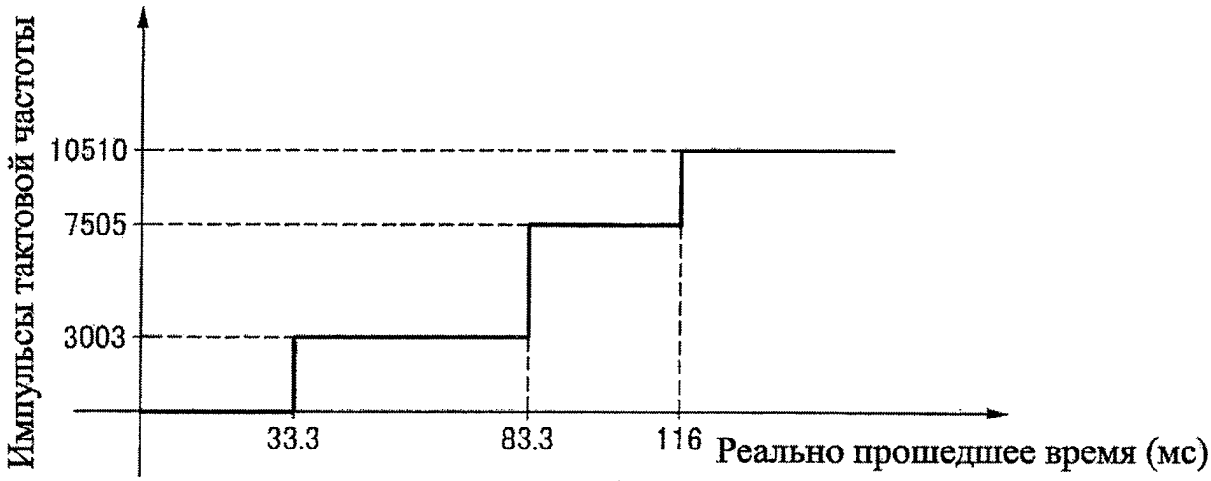
Фиг. 2a


210
ПРОГРАММА ВОСПРОИЗВЕДЕНИЯ ВИДЕОСОДЕРЖАНИЯ
220
220A
Модуль обработки графических изображений
219
Модуль вывода видеоизображения
БУФЕР
Видеоданные
Модуль вывода звука
БУФЕР
Данные звука
221
221A

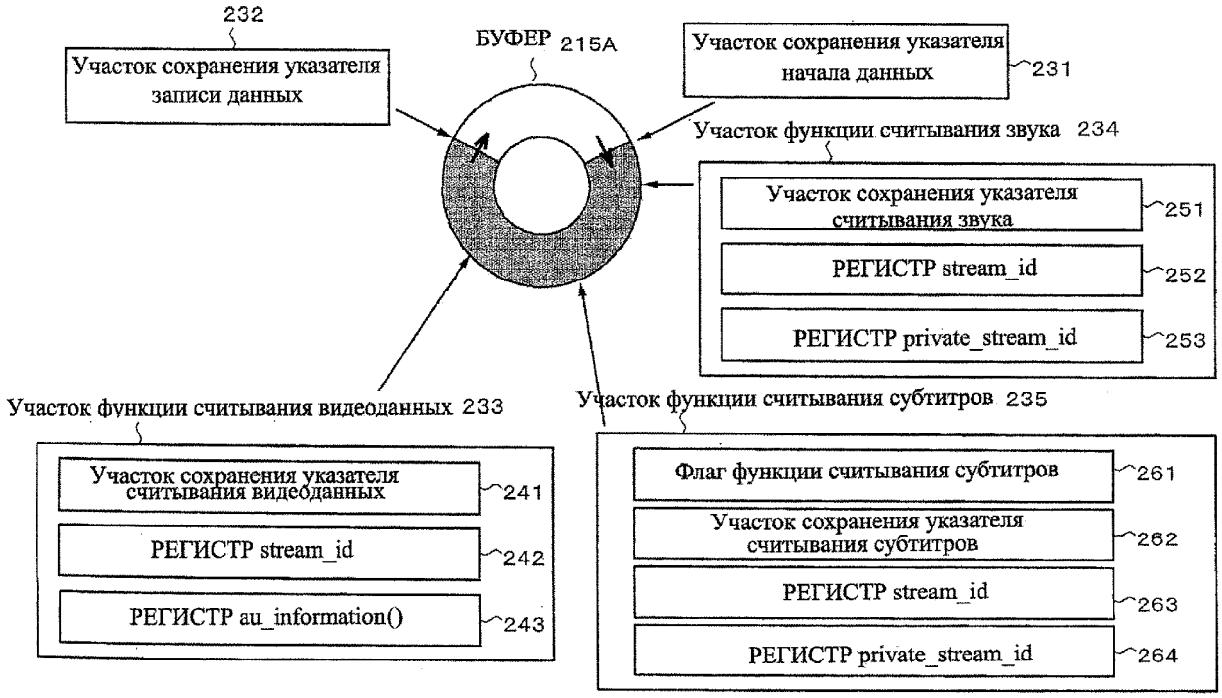
Фиг. 2b



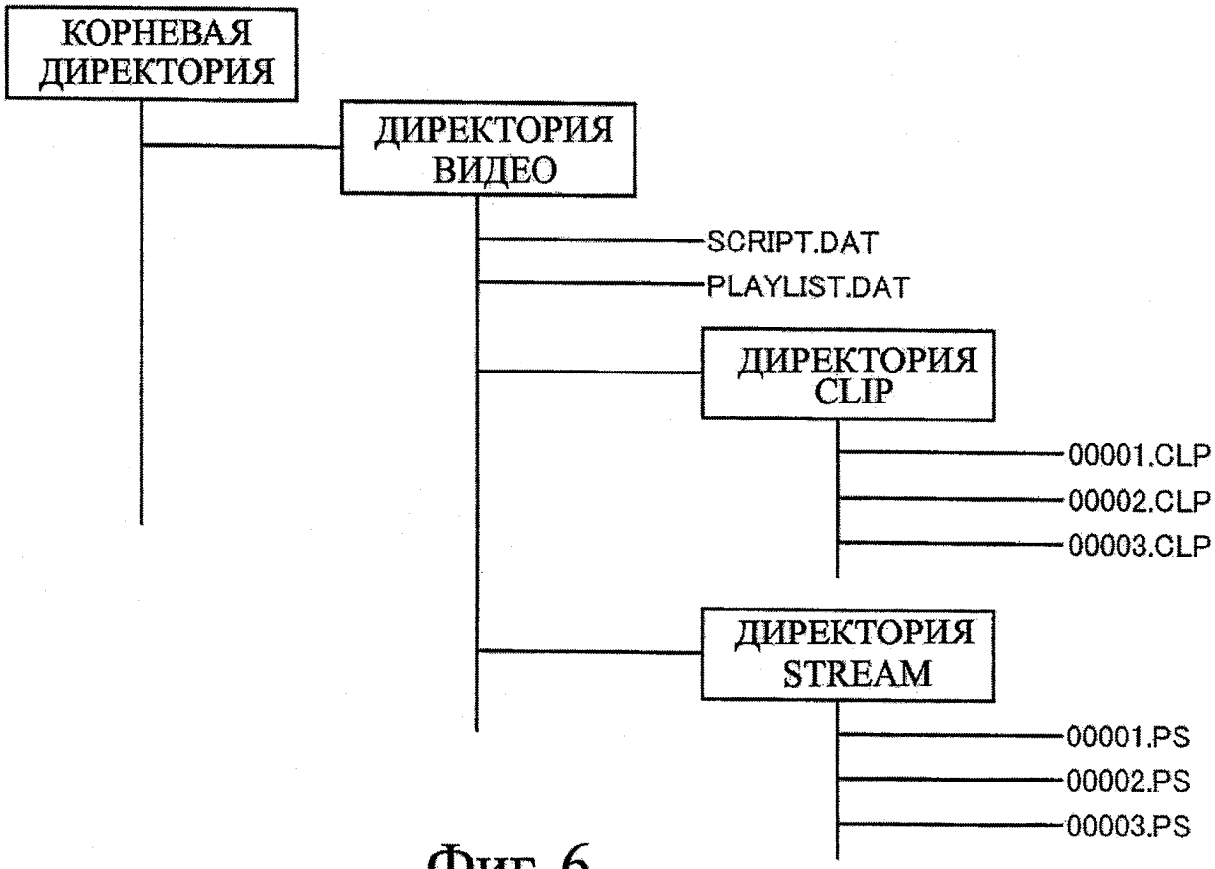
Фиг. 3



Фиг. 4



Фиг. 5



Фиг. 6

PlayList.PAT FILE		
СИНТАКСИС	КОЛИЧЕСТВО БИТОВ	МНЕМОНИЧЕСКОЕ ЗНАЧЕНИЕ
"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
//ИНФОРМАЦИЯ АТРИБУТА		
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()		
}		
}		
}		

Фиг. 7

PlayItem()		
Синтаксис	КОЛИЧЕСТВО БИТОВ	МНЕМОНИЧЕСКОЕ ЗНАЧЕНИЕ
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	
}		

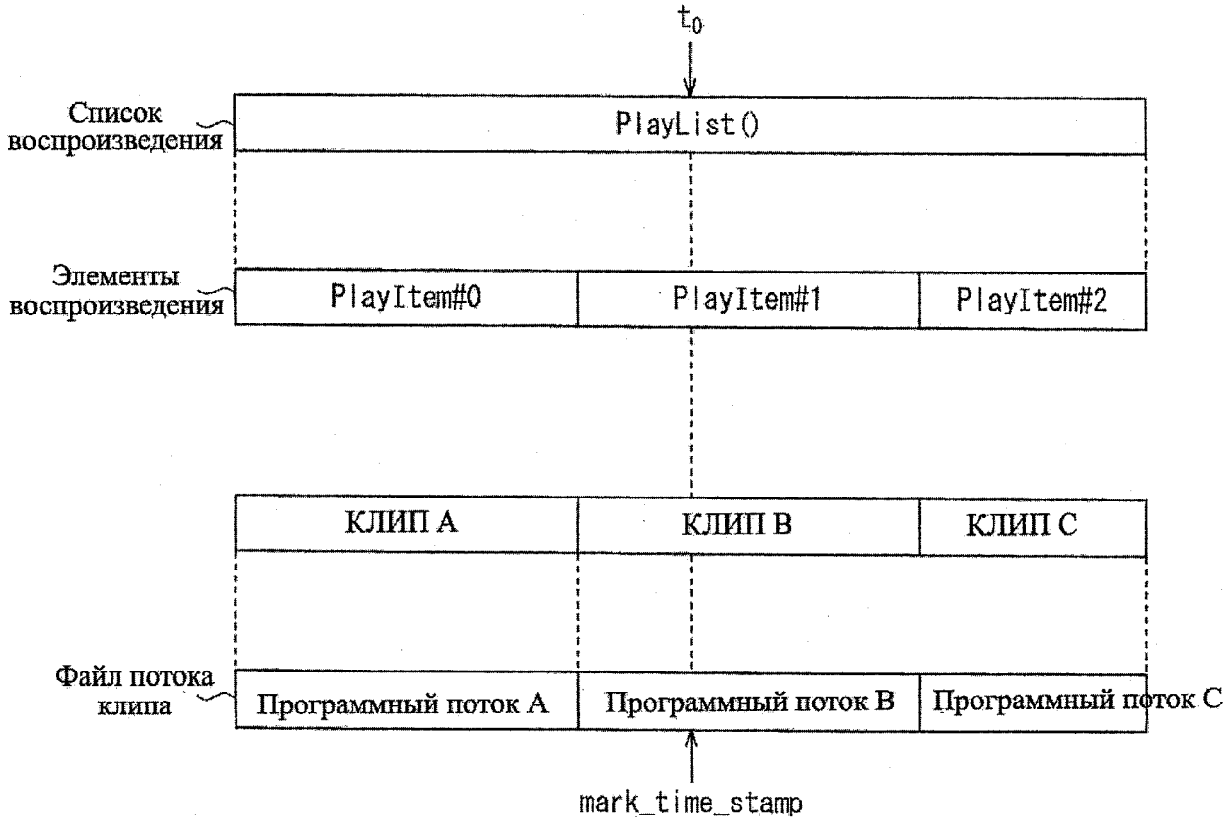
Фиг. 8

PlayListMark()		
СИНТАКСИС	КОЛИЧЕСТВО БИТОВ	МНЕМОНИЧЕСКОЕ ЗНАЧЕНИЕ
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
}		
}		
}		

Фиг. 9

mark_type	КОДИРОВАНИЕ ПОТОКА
0	ЗАРЕЗЕРВИРОВАНО
1	МЕТКА «Раздел»
2	МЕТКА «Индекс»
3	МЕТКА «Событие»
4−255	ЗАРЕЗЕРВИРОВАНО

Фиг. 10



Фиг. 11

clip() "CLIP INFORMATION FILE" (.CLP)		
СИНТАКСИС	КОЛИЧЕСТВО БИТОВ	МНЕМОНИЧЕСКОЕ ЗНАЧЕНИЕ
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()		
}		

Фиг. 12

Тип элементарного потока	stream_id	private_stream_id
Видеоданные	0xE0—0xEF	(NONE)
Звук ATRAC	0xBD	0x00—0x0F
Звук LPCM	0xBD	0x10—0x1F
Субтитр	0xBD	0x80—0x9F

Фиг. 13

StaticInfo()		
СИНТАКСИС	КОЛИЧЕСТВО БИТОВ	МНЕМОНИЧЕСКОЕ ЗНАЧЕНИЕ
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
}		
}		

Фиг. 14

БИТОВ

DynamicInfo()		
СИНТАКСИС	КОЛИЧЕСТВО БИТОВ	МНЕМОНИЧЕСКОЕ ЗНАЧЕНИЕ
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
}		
}		

Фиг. 15

EP_map()		
СИНТАКСИС	КОЛИЧЕСТВО БИТОВ	МНЕМОНИЧЕСКОЕ ЗНАЧЕНИЕ
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++){		
if(isVideoStream){		
index_N_minus1	2	uimsbf
N-th_Ref_picture_copy	14	uimsbf
}else		
reserved_for_future_use	16	bslbf
}		
PTS_EP_start	32	uimsbf
RPN_EP_start	32	uimsbf
}		
}		
}		

Фиг. 16

	index_N_minus1
1stRef_Picture	0
2ndRef_Picture	1
3rdRef_Picture	2
4thRef_Picture	3

Фиг. 17

ТАБЛИЦА 2-31, ПРОГРАММНЫЙ ПОТОК		
СИНТАКСИС	КОЛИЧЕСТВО БИТОВ	МНЕМОНИЧЕСКОЕ ЗНАЧЕНИЕ

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

ТАБЛИЦА 2-32, ПАКЕТ ПРОГРАММНОГО ПОТОКА		
СИНТАКСИС	КОЛИЧЕСТВО БИТОВ	МНЕМОНИЧЕСКОЕ ЗНАЧЕНИЕ

pack() {
pack_header()
while(nextbits()==packet_start_code_prefix) {
PES_packet()
}
}		

Фиг. 18a

ТАБЛИЦА 2-33, ЗАГОЛОВОК ПАКЕТА ПРОГРАММНОГО ПОТОКА		
СИНТАКСИС	КОЛИЧЕСТВО БИТОВ	МНЕМОНИЧЕСКОЕ ЗНАЧЕНИЕ
pack_header() {		
pack_start_code	32	bslbf
'01'	2	bslbf
system_clock_reference_base[32..30]	3	bslbf
marker_bit	1	bslbf
system_clock_reference_base[29..15]	15	bslbf
marker_bit	1	bslbf
system_clock_reference_base[14..0]	15	bslbf
marker_bit	1	bslbf
system_clock_reference_extension	9	uimsbf
marker_bit	1	bslbf
program_mux_rate	22	uimsbf
marker_bit	1	bslbf
marker_bit	1	bslbf
reserved	5	bslbf
pack_stuffing_length	3	uimsbf
for(i=0;i<pack_stuffing_length;i++) {		
stuffing_byte	8	bslbf
}		
if(nextbits()==system_header_start_code) {		
system_header()		
}		
}		

Фиг. 18b

Фиг.19

Фиг.19A

Фиг.19B

ТАБЛИЦА 2-17, ПАКЕТ PES

СИНТАКСИС	Количество битов	Мнемоническое значение
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		

Фиг. 19a

&& 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	uimsb
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
}		

Фиг. 19b

Фиг.20

Фиг.20A
Фиг.20B
Фиг.20C

ТАБЛИЦА 2-17, ПАКЕТ PES (ПРОДОЛЖЕНИЕ)

СИНТАКСИС	Количество битов	Мнемоническое значение
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
}		

Фиг. 20a

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
}		

Фиг. 20b

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
}		

Фиг. 20c

Фиг.21

Фиг.21A

Фиг.21B

ТАБЛИЦА 2-17, ПАКЕТ PES (КОНЕЦ)

СИНТАКСИС	Количество битов	Мнемоническое значение
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
}		

Фиг. 21a

```
                    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
                }
            }
        }
```

Фиг. 21b

Фиг.22

Фиг.22A
Фиг.22B

ТАБЛИЦА 2-18, НАЗНАЧЕНИЕ stream_id

stream_id	Примечание	КОДИРОВАНИЕ ПОТОКА
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 or ISO/IEC 11172-3 or ISO/IEC 13818-7 or ISO/IEC 14496-3 Номер потока звука X XXXX
1110 xxxx		ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 11172-2 or ISO/IEC 14496-2 Номер потока видеоданных XXXX
1111 0000	3	ECM_stream
1111 0001	3	EMM_stream
1111 0010	5	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 приложение A или ISO/IEC 13818-6_DSMCC_stream
1111 0011	2	ISO/IEC_13522_stream
1111 0100	6	ITU-T Rec. H.222.1 type A
1111 0101	6	ITU-T Rec. H.222.1 type B
1111 0110	6	ITU-T Rec. H.222.1 type C
1111 0111	6	ITU-T Rec. H.222.1 type D
1111 1000	6	ITU-T Rec. H.222.1 type 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		зарезервированный поток данных
1111 1111	4	program_stream_directory

Фиг. 22a

ПРИМЕЧАНИЕ: X означает, что значение "0" или "1" разрешены одновременно, и приводит к одному и тому же типу потока. номер потока задается с помощью X.

ПРИМЕЧАНИЕ 1: В пакете PES для program_stream_map используется уникальный синтаксис, определенный в 2.5.4.1.

ПРИМЕЧАНИЕ 2: Пакеты PES private_stream_1 и ISO/IEC.13552.stream следуют тому же синтаксису пакета, что и ITU-T REC.H262, ISO/IEC 13818-2 VIDEO и ISO/IEC 13818-3 AUDIO STREAMS

ПРИМЕЧАНИЕ 3: Пакеты PES для private_stream_2, ECM_STREAM, и EMM_stream аналогичны private_system_1, за исключением того, что syntax казан после PES_packet_length field.

ПРИМЕЧАНИЕ 4: В пакете PES для program _stream_directory используется уникальный синтаксис, определенный в 2.5.5.

ПРИМЕЧАНИЕ 5: В DSM-CC.stream используется уникальный синтаксис, определенный в ISO/IEC 13818-6

ПРИМЕЧАНИЕ 6: В этом stream_id используется комбинация со stream_type OX09 в таблице 2-29

ПРИМЕЧАНИЕ 7: Этот system_id используется только в пакете PES для программного потока или потока данных ISO/IEC 11172-1 system stream (см. 2.4.3.7).

Фиг. 22b

stream_id	Кодирование потока (тип потока)
1011 1101	private_stream_1
1011 1110	padding_stream
1011 1111	private_stream_2
110x xxxx	Номер потока звука X XXXX
1110 xxxx	Номер потока видеоданных XXXX

Фиг. 23

СИНТАКСИС	Количество битов	Мнемоническое значение
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()		
}		

Фиг. 24

private_stream_id	Кодирование потока (тип потока)
0000 xxxx 0001 xxxx 100x xxxx	Номер потока звука ATRAC XXXX Номер потока LPCM XXXX Номер потока субтитров X XXXX

Фиг. 25

СИНТАКСИС	Количество битов	Мнемоническое значение
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
}		
}		

Фиг. 26

СИНТАКСИС	Количество битов	Мнемоническое значение
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++){		
зарезервировано	4	bslbf
pic_struct_copy	4	uimsbf
au_ref_flag	1	uimsbf
reserved	2	bslbf
AU_length	21	uimsbf
}		uimsbf
}		

Фиг. 27

ОПИСАНИЕ D-1, pic_struct			
Значение	Отображение изображения	Ограничение	NumClockTS
0	Кадр	field_pic_flag = 0	1
1	Верхнее поле	field_pic_flag = 1 bottom_field_flag = 0	1
2	Нижнее поле	field_pic_flag = 1 bottom_field_flag = 1	1
3	В порядке: верхнее поле и нижнее поле	field_pic_flag = 0	2
4	В порядке: нижнее поле и верхнее поле	field_pic_flag = 0	2
5	В порядке: верхнее поле и нижнее поле, и верхнее поле (повторно отображается)	field_pic_flag = 0	3
6	В порядке: нижнее поле, верхнее поле и нижнее поле (повторно отображается)	field_pic_flag = 0	3
7	Кадр повторяют два раза	field_pic_flag = 0 fixed_frame_rate_flag = 1	2
8	Кадр повторяют три раза	field_pic_flag = 0 fixed_frame_rate_flag = 1	3
9..15	Зарезервировано		

Фиг. 28

// "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	

Фиг. 29

Фиг.30

Фиг.30A
Фиг.30B

// 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'
	lfe_existence='NO'	lfe_existence='NO'	cc_flag='No'
	sampling_frequency='48KHz'	sampling_frequency='48KHz'	
number_of_DynamicInfo	0	0	0

Фиг. 30a

// 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'
			lfe_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	

Фиг. 30b

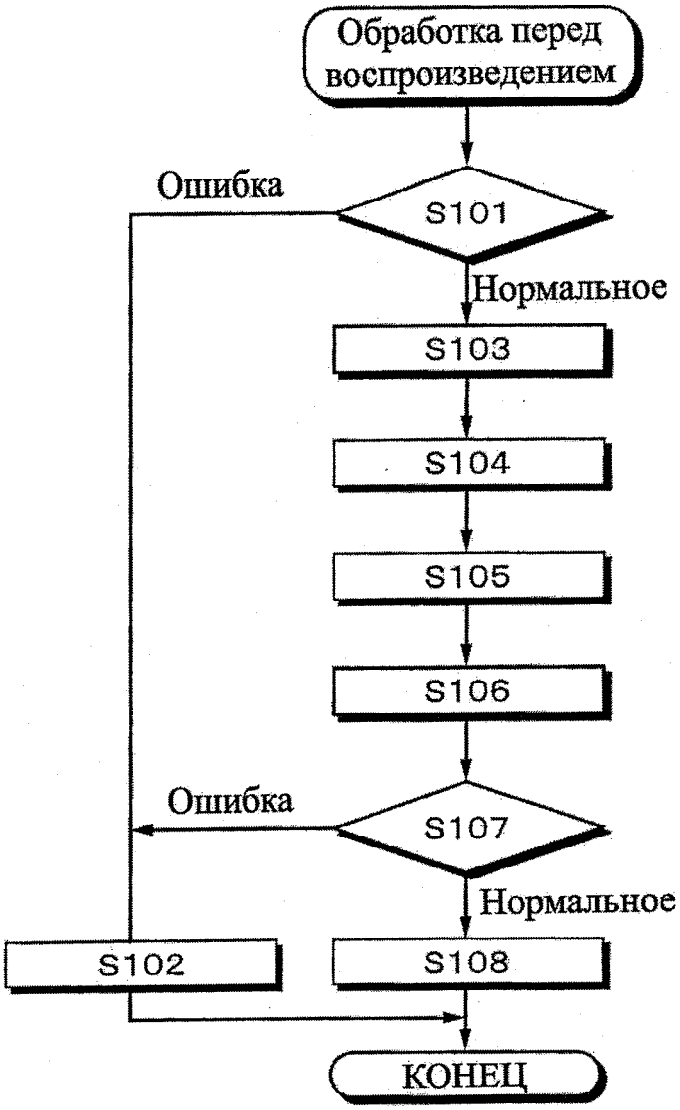
EP_map( ) for"00001.CLP"								
number_of_stream_id_entries	1							
stream_id	0xE0 (видеопоток)							
private_stream_id	0x00							
	PTS_EP_start	RPN_EP_start	1stRef_Picture	2ndRef_Picture	3rdRef_Picture	4thRef_Picture	index_N_minus1	N-th_Ref_picture_copy
	90000	0	28	37	48	58	0	28
	135045	244	10	18	25	31	3	31
	180090	305	25	44	50	54	0	25
	225135	427	8	15	22	29	3	29
	270180	701	26	32	41	48	1	32

Фиг. 31

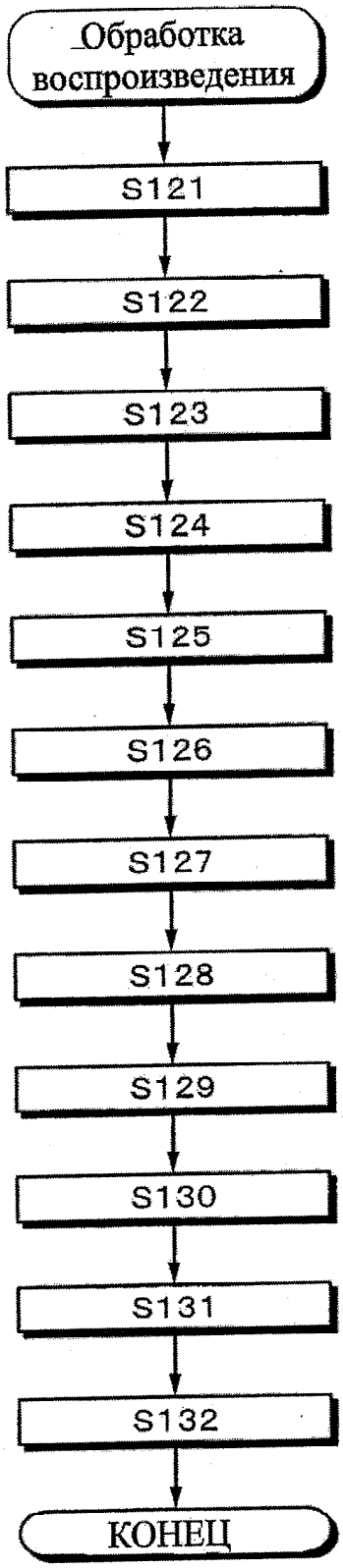
PlayListMARK() OF PlayList #0 number_of_PlayList_marks==7								
Mark()	mark_type	ref_to_ PlayItem_id	mark_time_ stamp	entry_ES_ stream_id	entry_ES_ private_ stream_id	mark_data		
#0	Раздел	0	180,090	0	0	1	Начало	Начало
#1	Индекс	0	5,580,090	0	0	1	1 минута	1 минута
#2	Индекс	0	10,980,090	0	0	2	2 минуты	2 минуты
#3	Событие	0	16,380,090	0	0	0	3 минуты	3 минуты
#4	Раздел	1	90,000	0	0	2	Начало	5 минут
#5	Индекс	1	5,490,000	0	0	1	1 минута	6 минут
#6	Индекс	1	10,890,000	0	0	2	2 минуты	7 минут

PlayListMARK() OF PlayList #1 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	Раздел	0	90,000	0	0	0	Начало
#1	Событие	0	27,090,000	0xE0	0	1	5 минут
#2	Событие	0	27,540,000	0xE1	0	2	5 минут, 5 секунд (305секунд)

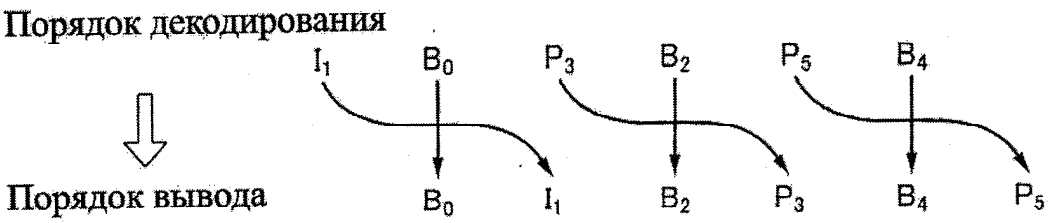
Фиг. 32


Обработка перед воспроизведением
S101
Ошибка
Нормальное
S103
S104
S105
S106
S107
Ошибка
Нормальное
S102
S108
КОНЕЦ

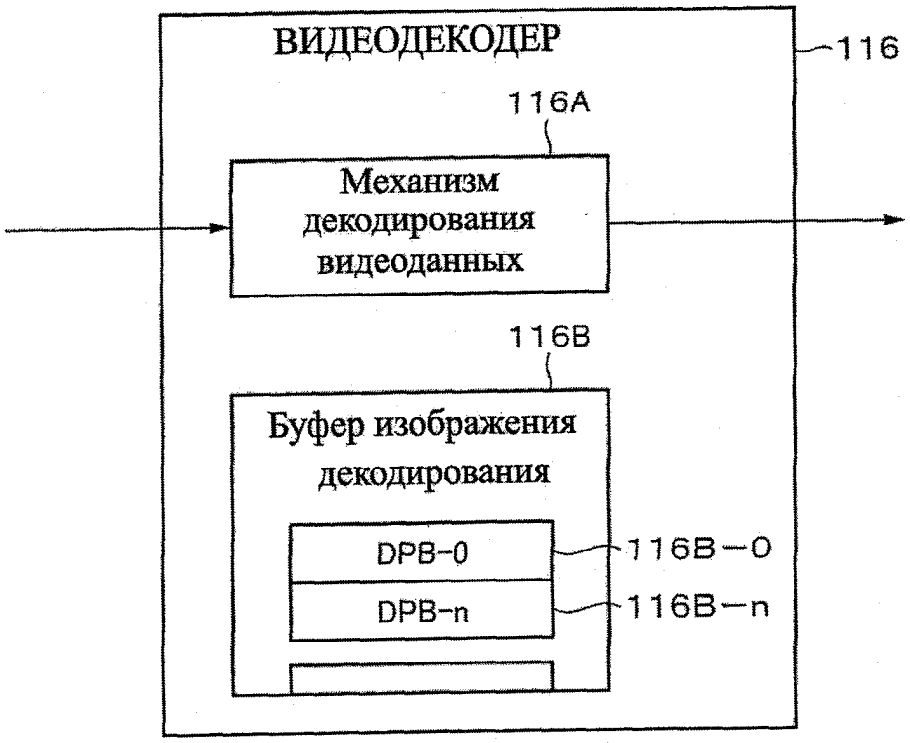
Фиг. 33


Обработка воспроизведения
S121
S122
S123
S124
S125
S126
S127
S128
S129
S130
S131
S132
КОНЕЦ

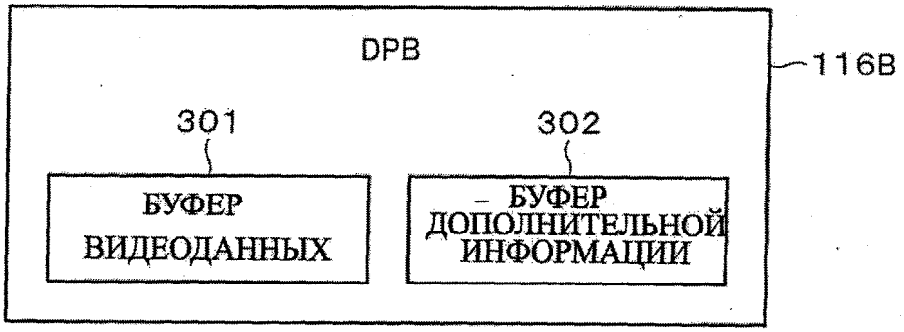
Фиг. 34



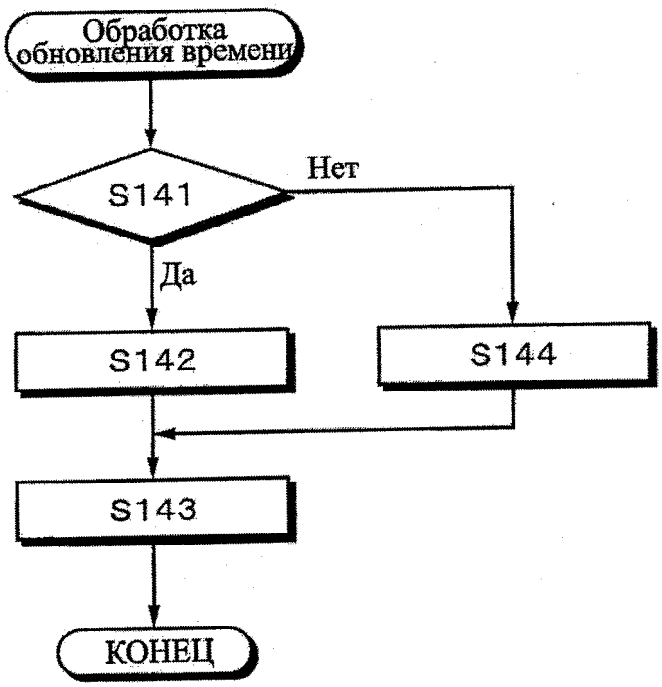
Фиг. 35



Фиг. 36



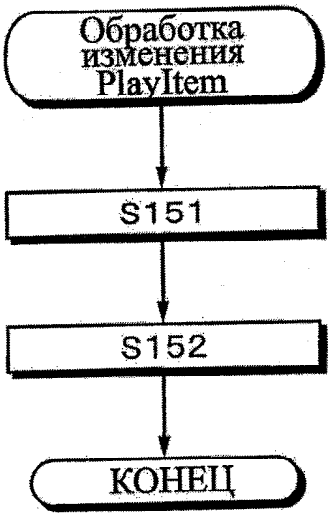
Фиг. 37



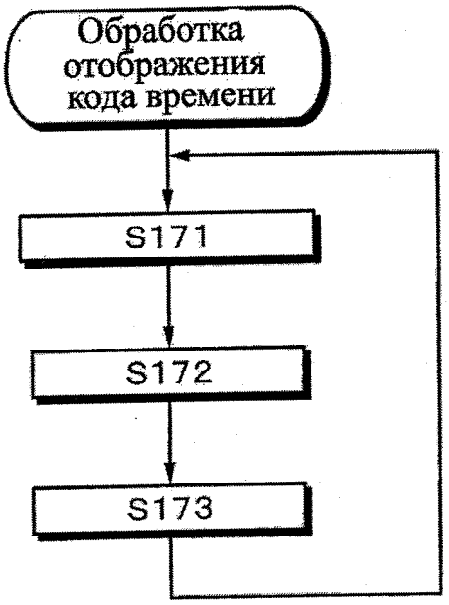
Фиг. 38

PTS	Значение сохраненного pic_struct	ОПЕРАЦИЯ
Присутствие	—	Заменить PTS
Отсутствие	0,1,2,3,4	Время + 2 поля (3003)
	5,6	Время + 3 поля (4505)

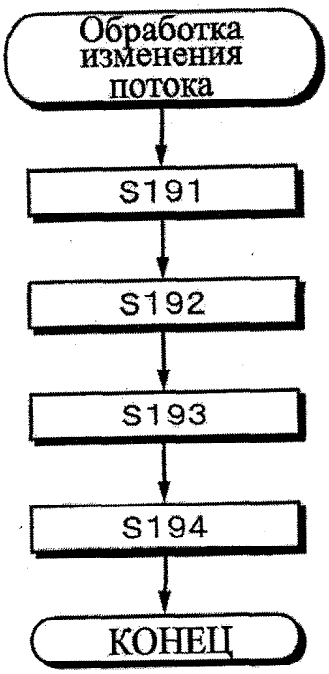
Фиг. 39



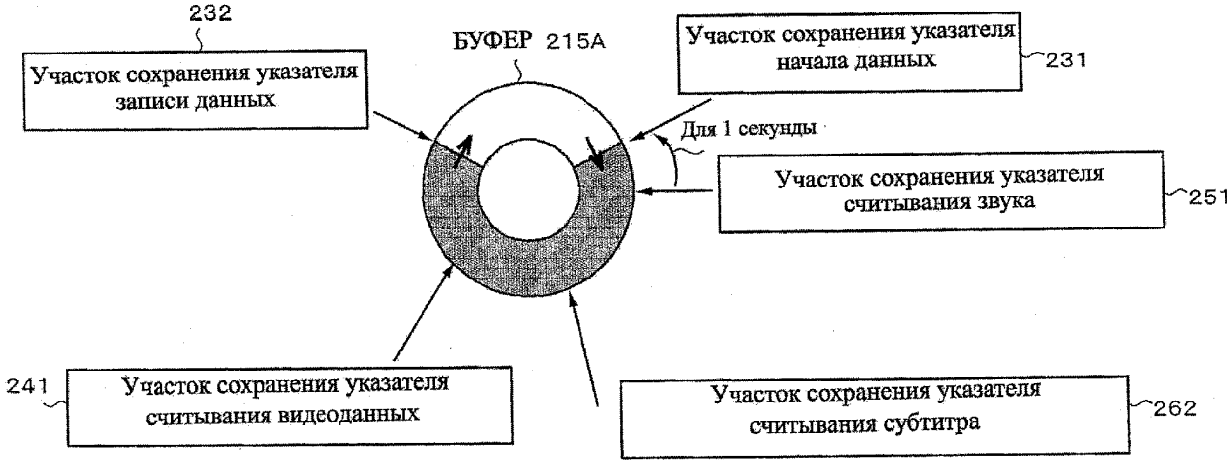
Фиг. 40



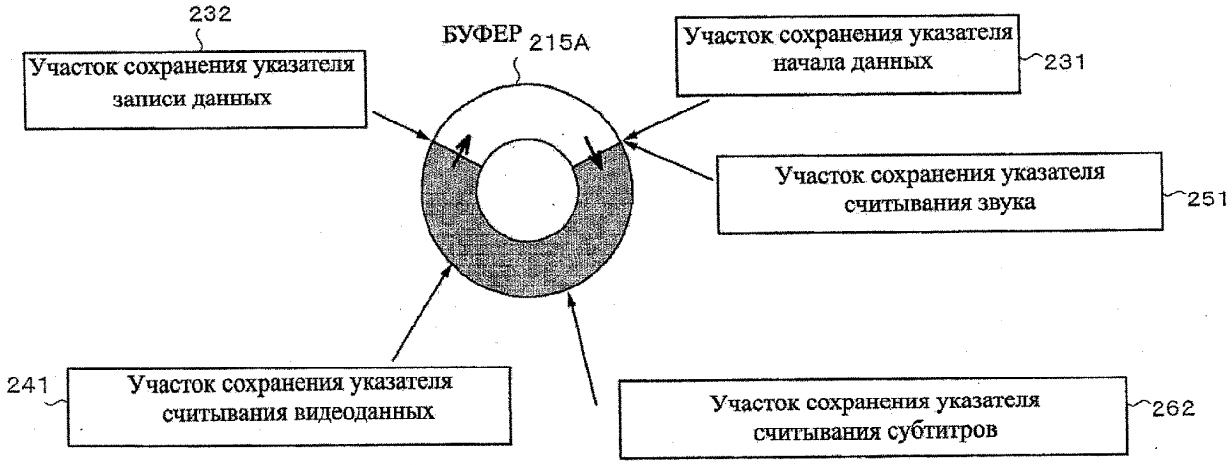
Фиг. 41



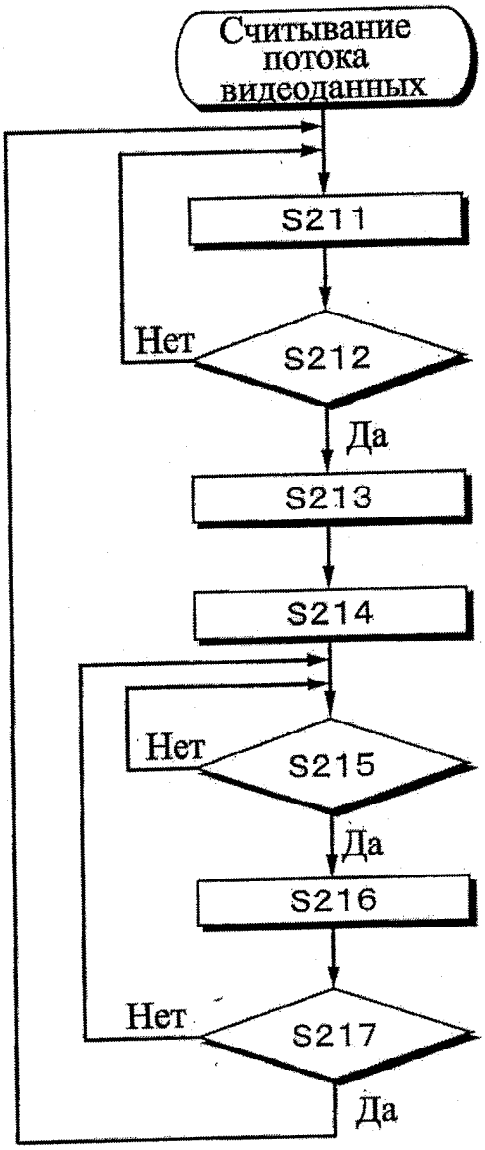
Фиг. 42



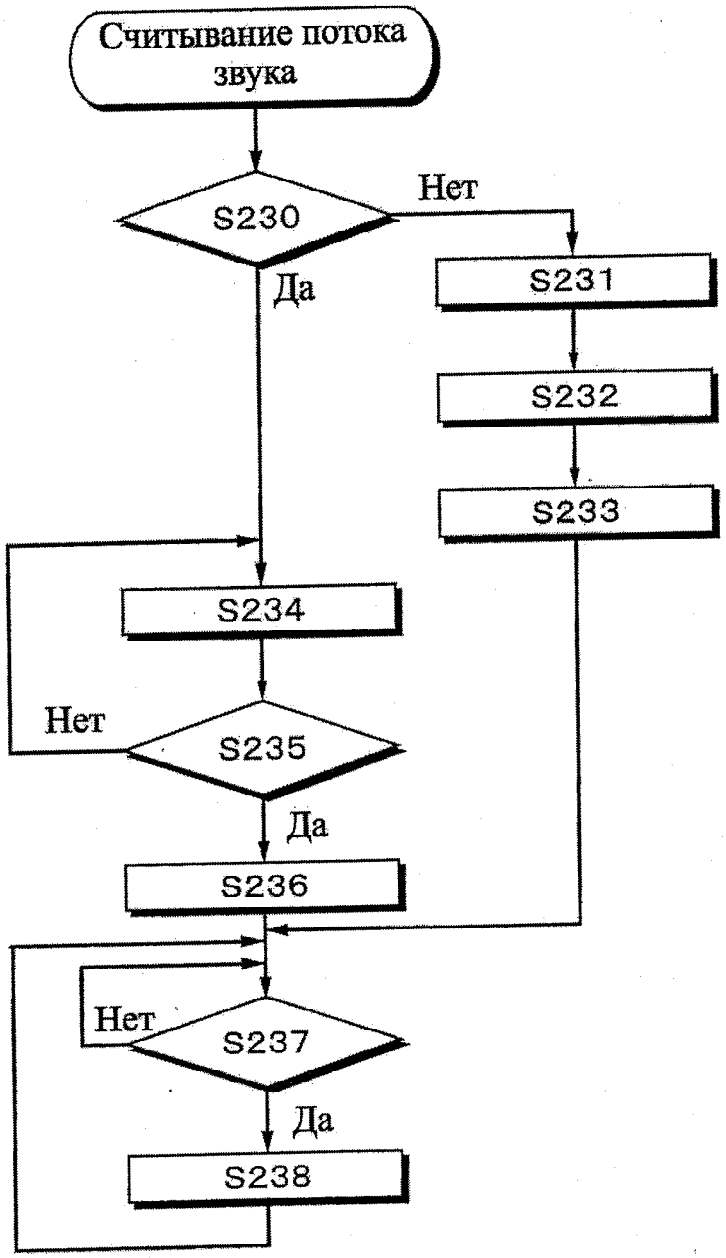
Фиг. 43


232
Участок сохранения указателя записи данных
БУФЕР 215A
Участок сохранения указателя начала данных
231
Участок сохранения указателя считывания звука
251
241
Участок сохранения указателя считывания видеоданных
Участок сохранения указателя считывания субтитров
262

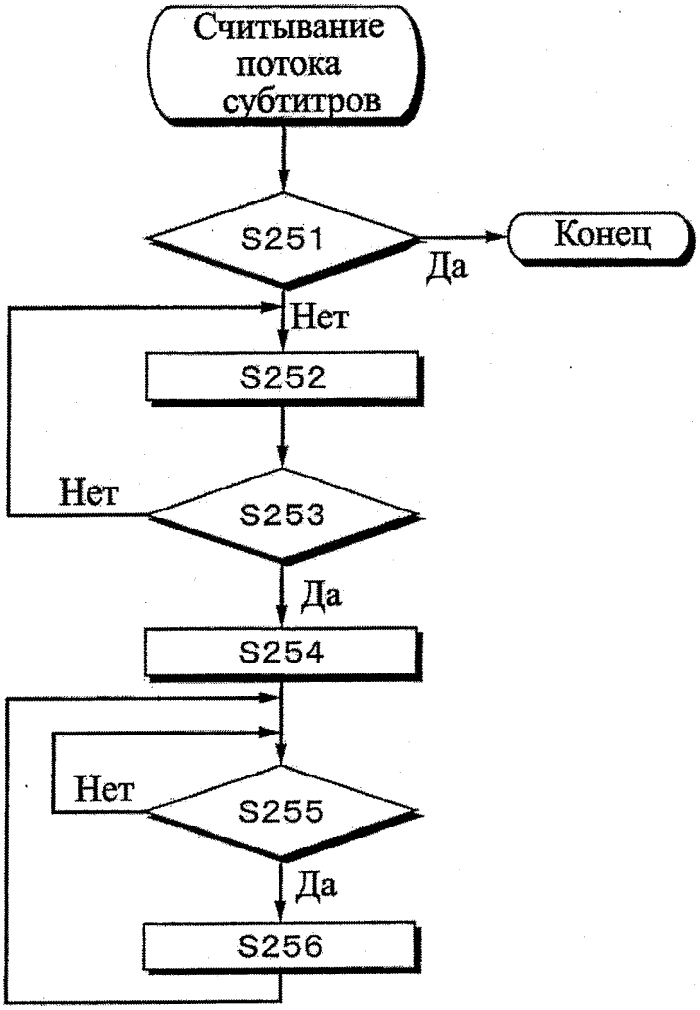
Фиг. 44


Считывание потока видеоданных
S211
Нет
S212
Да
S213
S214
Нет
S215
Да
S216
Нет
S217
Да

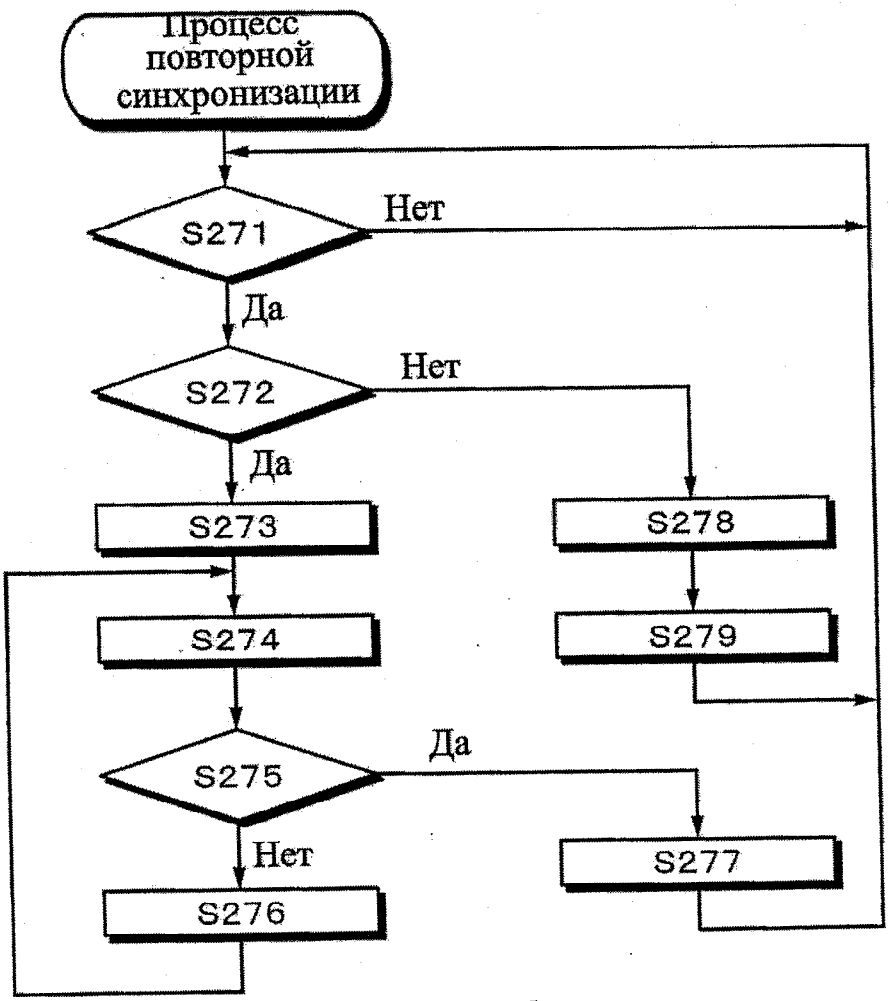
Фиг. 45


Считывание потока звука
S230
Нет
Да
S231
S232
S233
S234
Нет
S235
Да
S236
Нет
S237
Да
S238

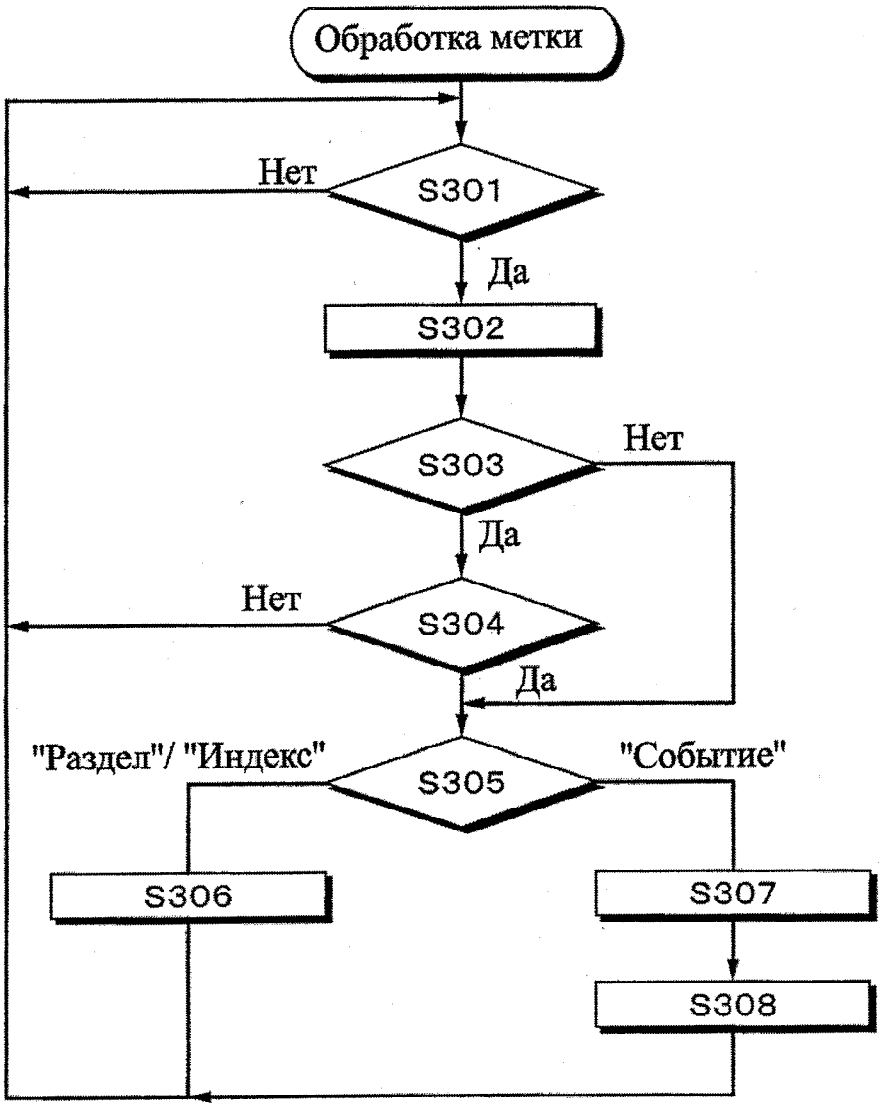
Фиг. 46


Считывание потока субтитров
S251
Да
Конец
Нет
S252
Нет
S253
Да
S254
Нет
S255
Да
S256

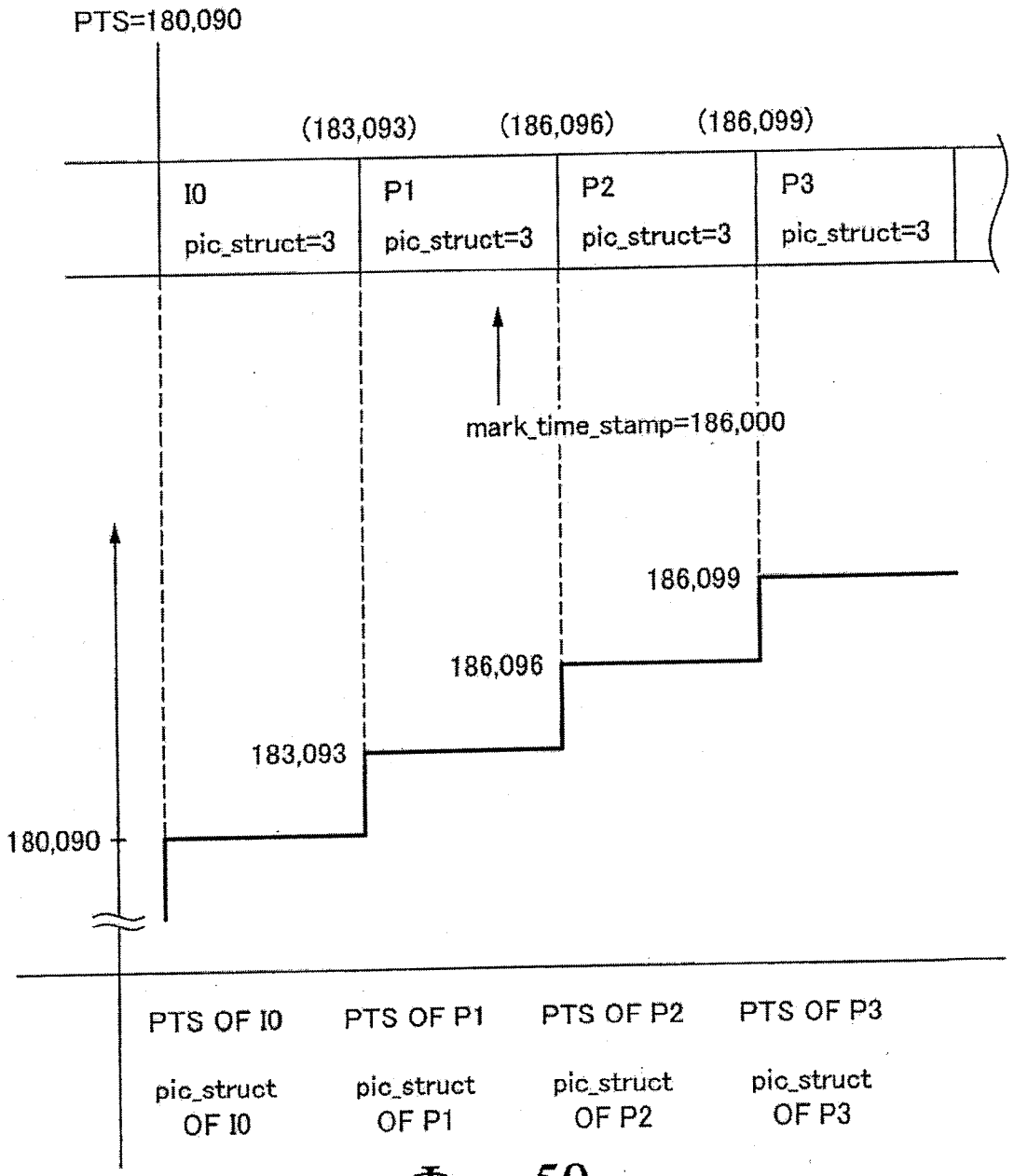
Фиг. 47


Процесс повторной синхронизации
S271
Нет
Да
S272
Нет
Да
S273
S278
S274
S279
S275
Да
Нет
S277
S276

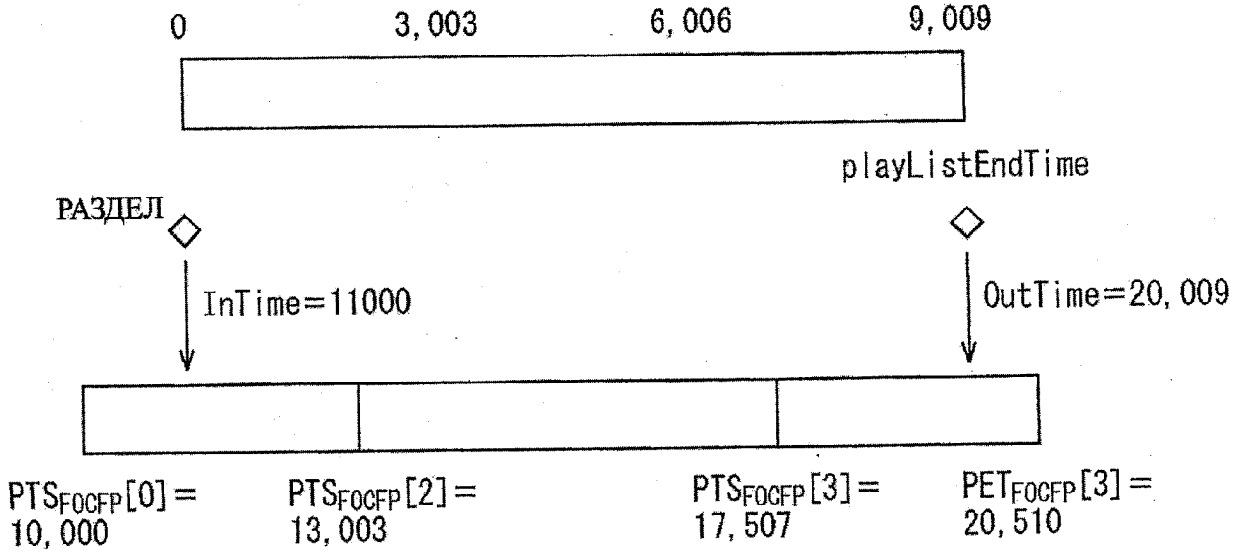
Фиг. 48


Обработка метки
S301
Нет
Да
S302
S303
Нет
Да
S304
Нет
Да
"Раздел"/ "Индекс"
S305
"Событие"
S306
S307
S308

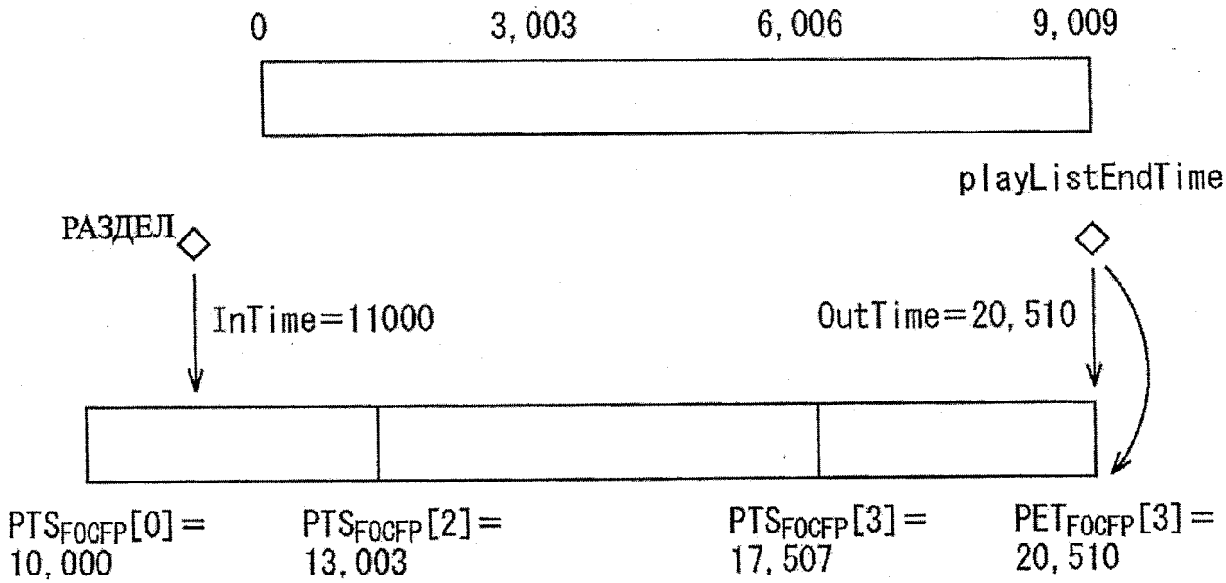
Фиг. 49


PTS=180,090
(183,093)
(186,096)
(186,099)
I0
pic_struct=3
P1
pic_struct=3
P2
pic_struct=3
P3
pic_struct=3
mark_time_stamp=186,000
186,099
186,096
183,093
180,090
PTS OF I0
PTS OF P1
PTS OF P2
PTS OF P3
pic_struct OF I0
pic_struct OF P1
pic_struct OF P2
pic_struct OF P3

Фиг. 50


0
3,003
6,006
9,009
playListEndTime
РАЗДЕЛ
InTime=11000
OutTime=20,009
PTS$_{FOCFP}$[0]= 10,000
PTS$_{FOCFP}$[2]= 13,003
PTS$_{FOCFP}$[3]= 17,507
PET$_{FOCFP}$[3]= 20,510

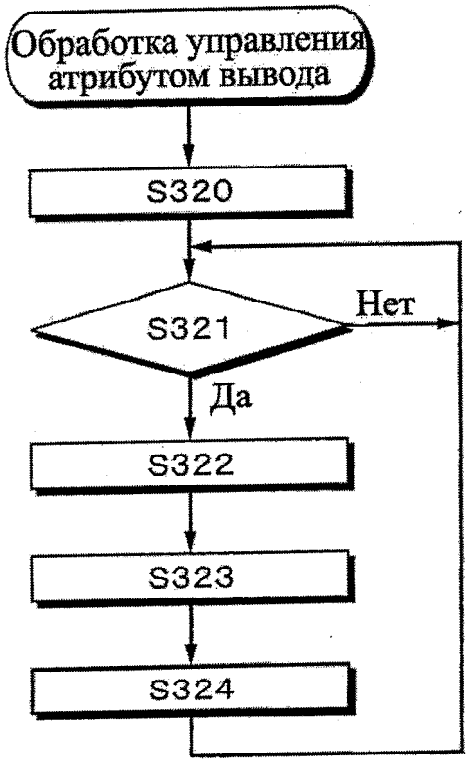
Фиг. 51



Фиг. 52

ЧАСТОТА КАДРОВ	СЛУЧАЙ A (23.976 p)	СЛУЧАЙ B (59.94 i)
МИНИМАЛЬНЫЙ ИНТЕРВАЛ СОБЫТИЯ	5005/240000	4004/240000
ПРИ ЧАСТОТЕ 90 кГЦ	7507	6006

Фиг. 53



Фиг. 54

"00003.CLP"

Stream#0

stream_id	0xE0
private_stream_id	0x00
number_of_DynamicInfo	2

(ВИДЕОПОТОК)

	pts_change_point	DynamicInfo()
#0	90,000	display_aspect_ratio=='4:3'
#1	54,090,000	display_aspect_ratio=='16:9'

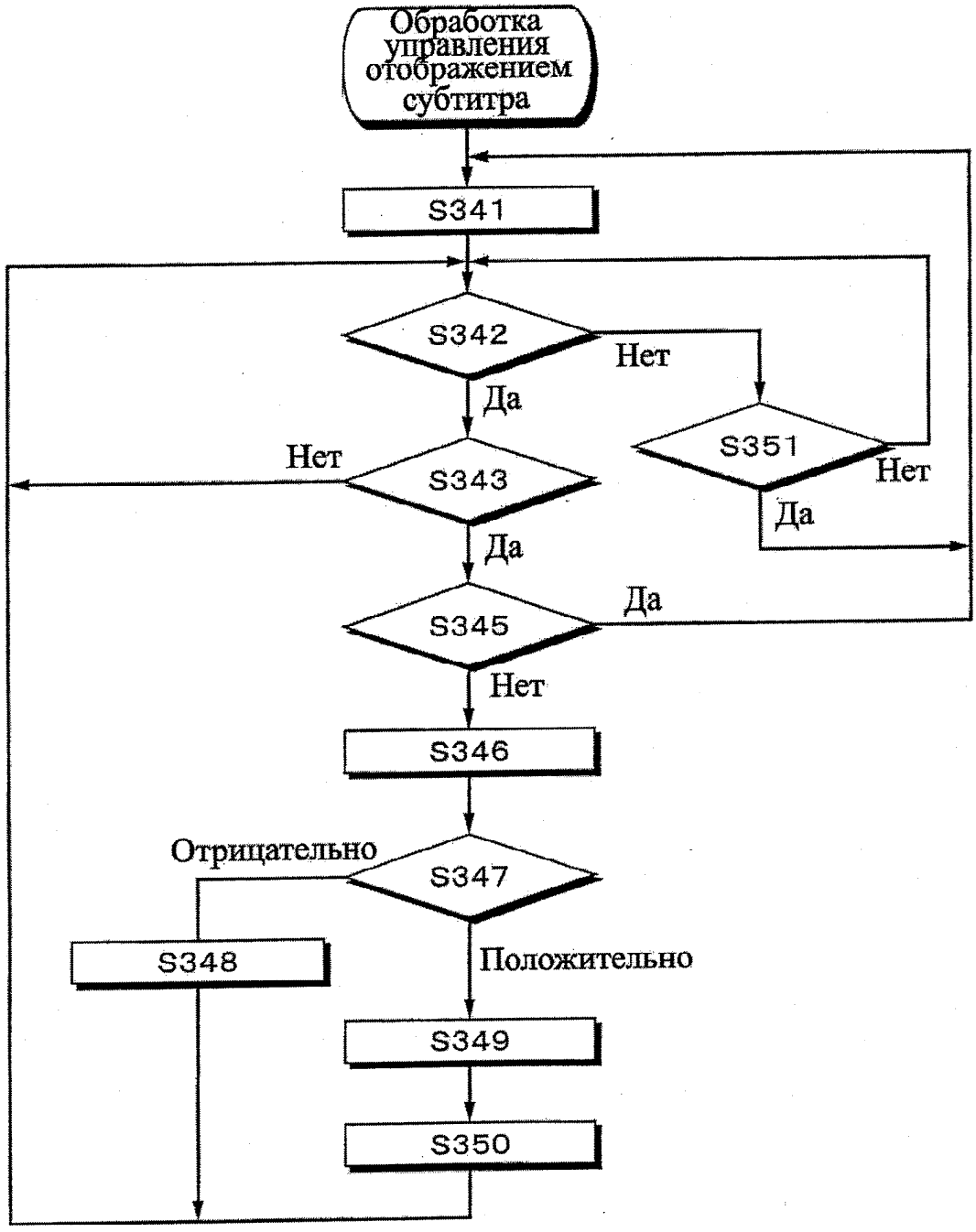
Stream#2

stream_id	0xBD
private_stream_id	0x00
number_of_DynamicInfo	3

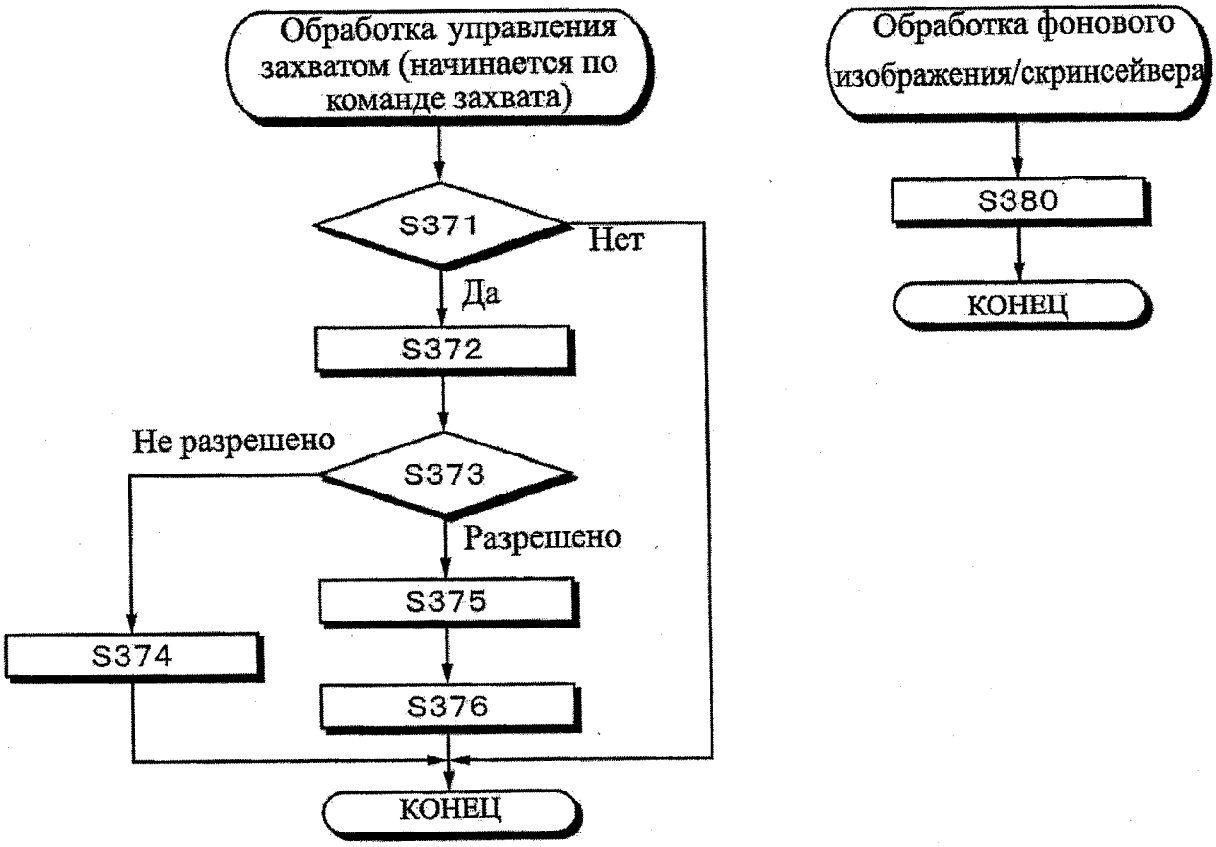
(private_stream_1)
(ATRAC)

	pts_change_point	DynamicInfo()
#0	90,000	channel_assignment=='Dual'
#1	27,090,000	channel_assignment=='Stereo'
#2	32,490,000	channel_assignment=='Dual'

Фиг. 55


Обработка управления отображением субтитра
S341
S342
Нет
Да
S351
Нет
Да
Нет
S343
Да
Да
S345
Нет
S346
Отрицательно
S347
S348
Положительно
S349
S350

Фиг. 56



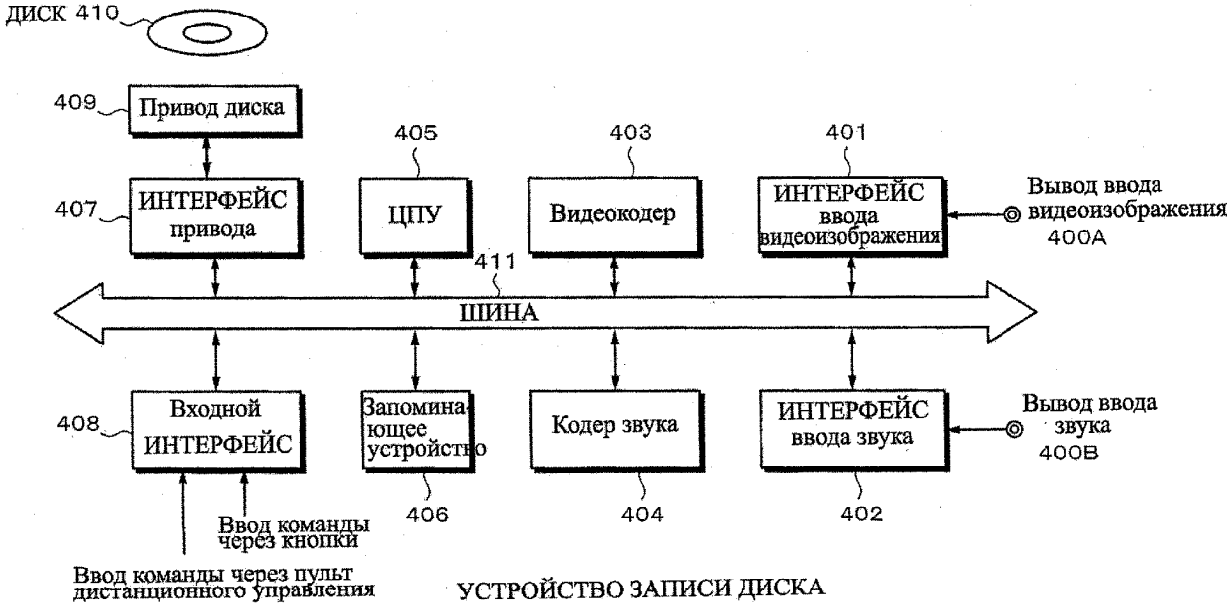
Фиг. 57

СИНТАКСИС	Количество битов	Мнемонические значения
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
}		
}		

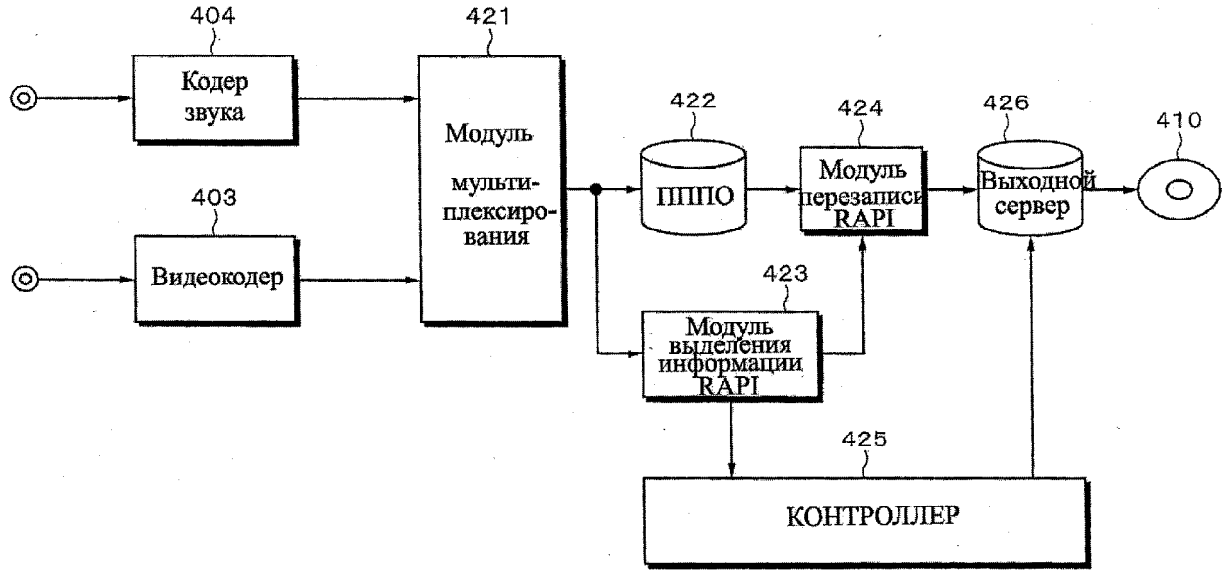
Фиг. 58

СИНТАКСИС	Количество битов	Мнемонические значения
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++){		
ЗАРЕЗЕРВИРОВАНО	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
}		

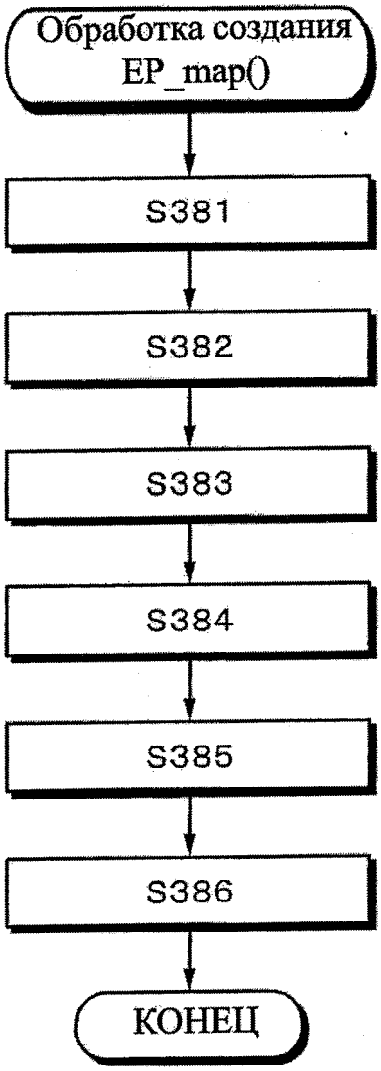
Фиг. 59



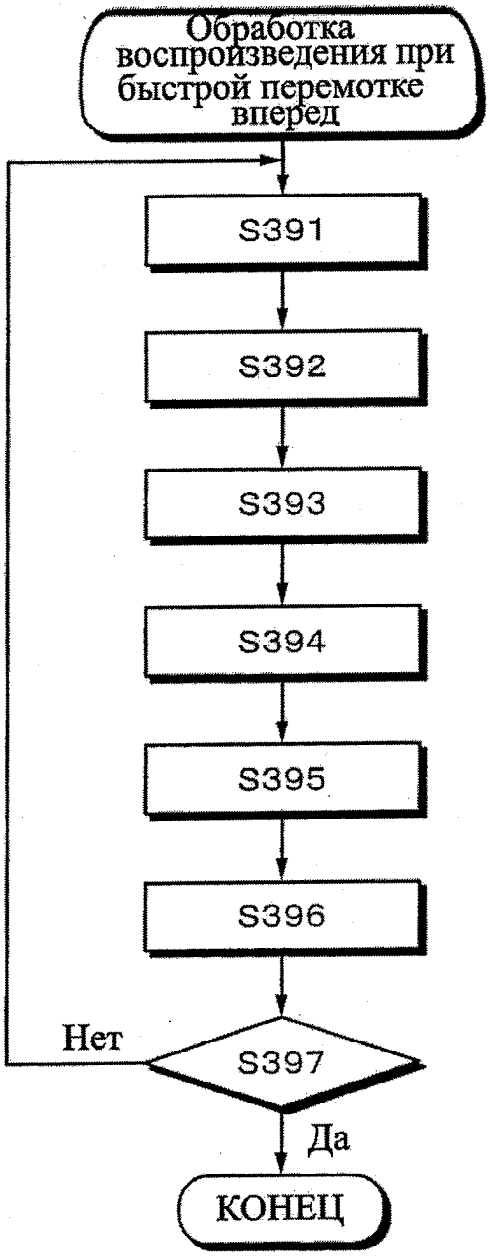
Фиг. 60



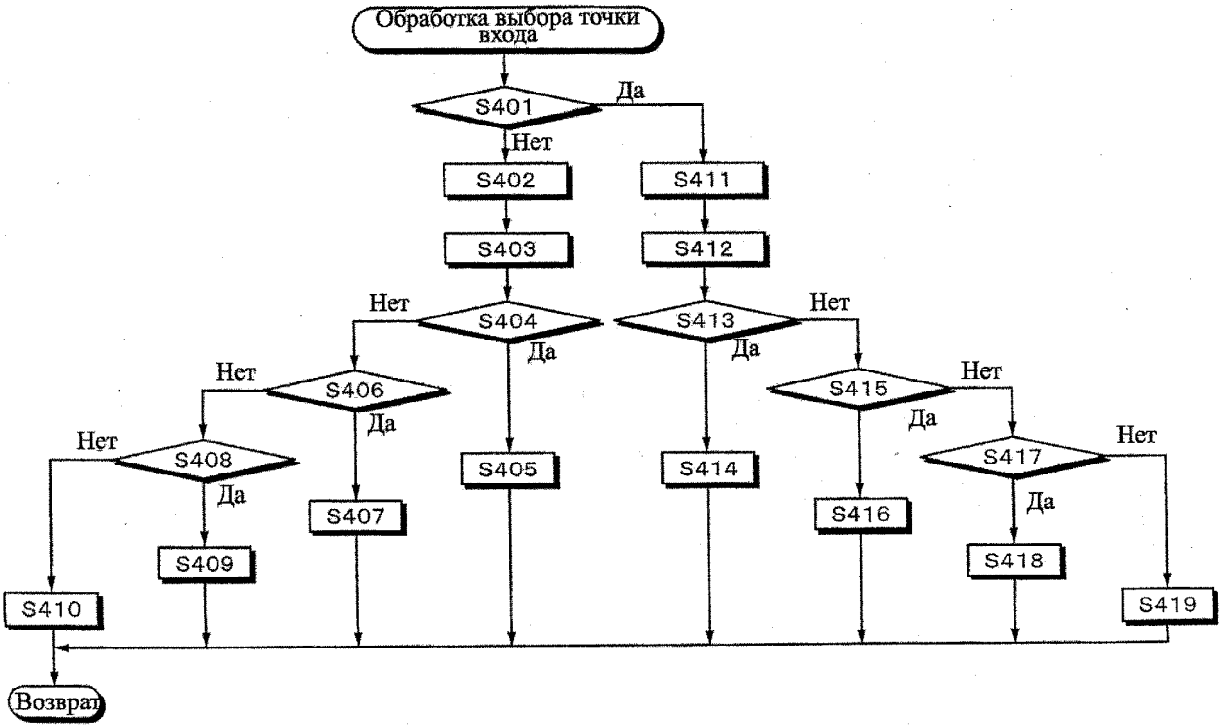
Фиг. 61


Обработка создания EP_map()
S381
S382
S383
S384
S385
S386
КОНЕЦ

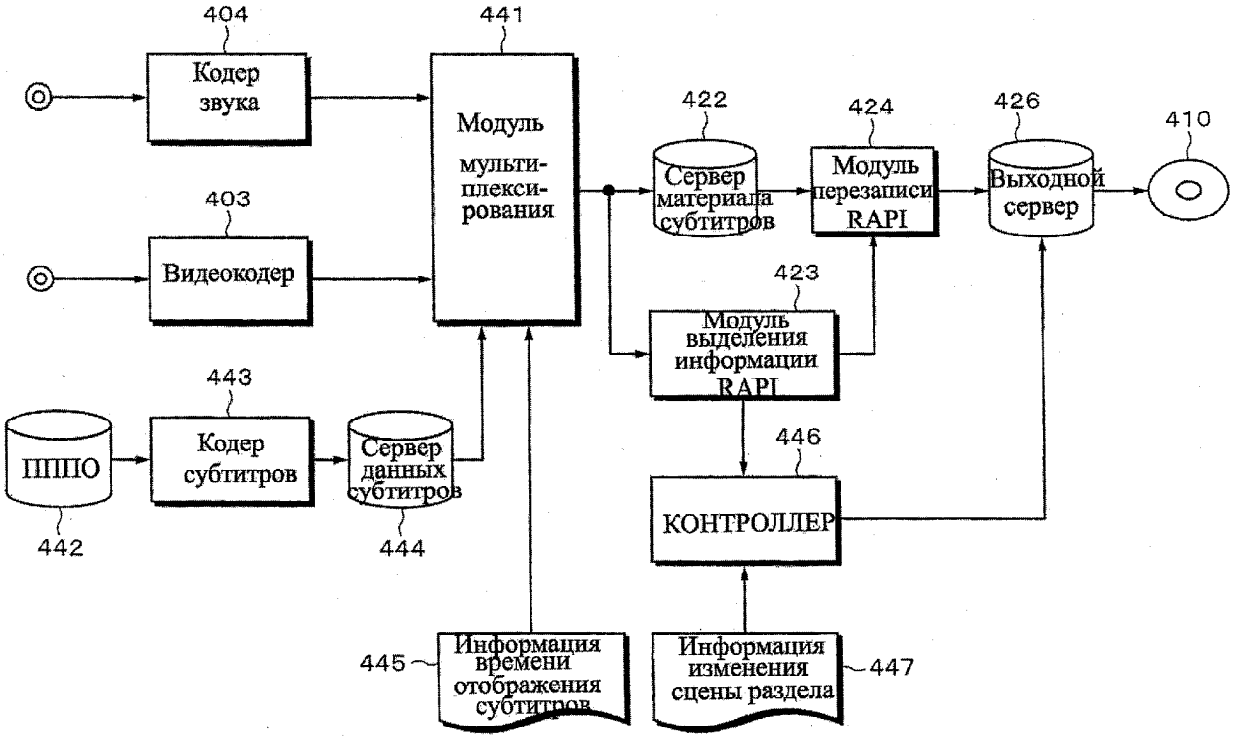
Фиг. 62


Обработка воспроизведения при быстрой перемотке вперед
S391
S392
S393
S394
S395
S396
S397
Нет
Да
КОНЕЦ

Фиг. 63


Обработка выбора точки входа
S401
Да
Нет
S402
S403
S404
Нет
Да
S406
Нет
Да
S408
Нет
Да
S407
S409
S410
S405
S411
S412
S413
Нет
Да
S414
S415
Нет
Да
S416
S417
Нет
Да
S418
S419
Возврат

Фиг. 64


404
Кодер звука
403
Видеокодер
443
ППО
Кодер субтитров
Сервер данных субтитров
442
444
441
Модуль мульти-плекси-рования
422
Сервер материала субтитров
424
Модуль перезаписи RAPI
426
Выходной сервер
410
423
Модуль выделения информации RAPI
446
КОНТРОЛЛЕР
445
Информация времени отображения субтитров
Информация изменения сцены раздела
447

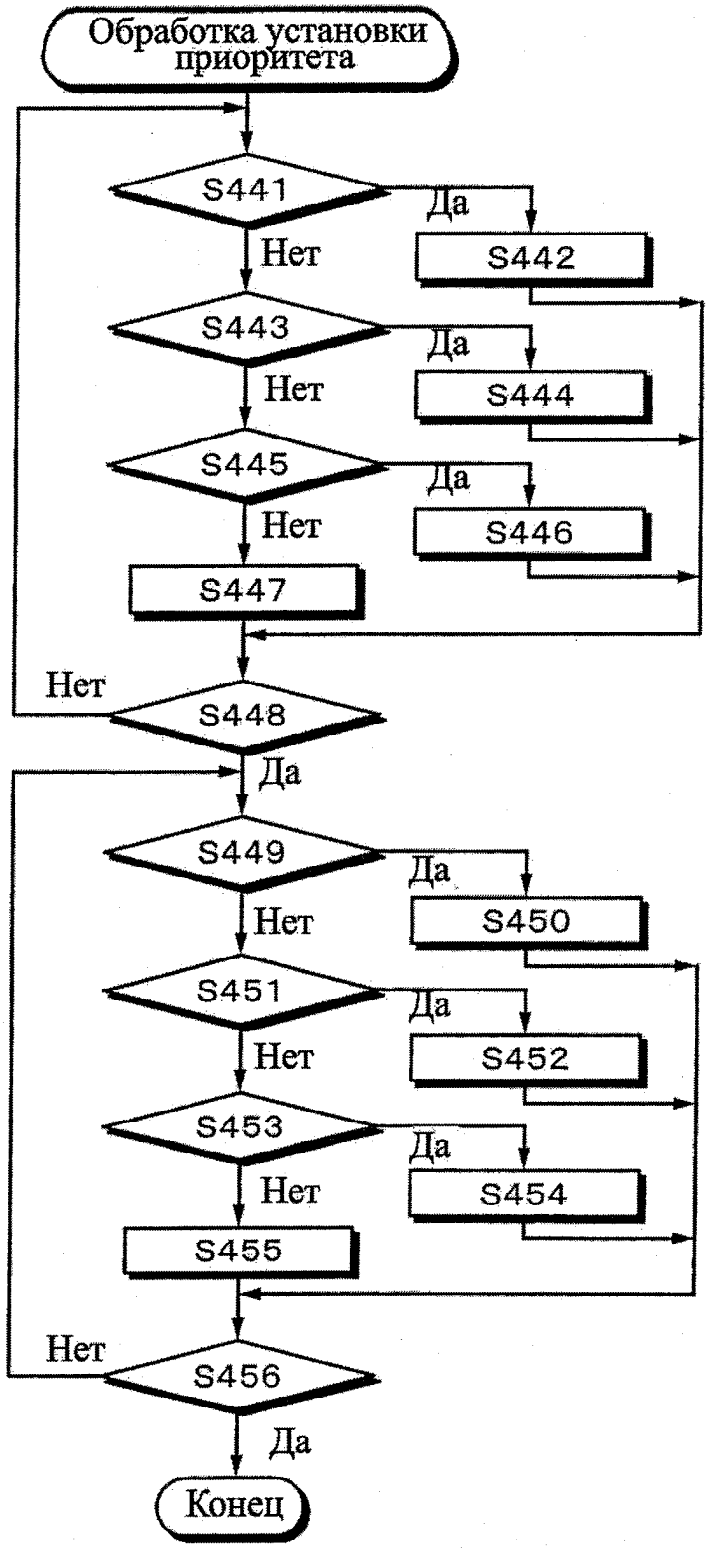
Фиг. 65

EP_map()		
СИНТАКСИС	Количество битов	Мнемоническое значение
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++){		
priority_flag	2	uimsbf
reserved_for_future_use	14	bslbf
PTS_EP_start	32	uimsbf
RPN_EP_start	32	uimsbf
}		
}		
}		

Фиг. 66

priority_flag	ВИДЕОПОТОК	ПОТОК СУБТИТРОВ
3	НАЧАЛО РАЗДЕЛА	НАЧАЛО РАЗДЕЛА
2	ВАЖНОЕ ИЗМЕНЕНИЕ СЦЕНЫ	ВАЖНОЕ ИЗМЕНЕНИЕ СЦЕНЫ
1	ОБЫЧНОЕ ИЗМЕНЕНИЕ СЦЕНЫ	ОБЫЧНОЕ ИЗМЕНЕНИЕ СЦЕНЫ
0	ВСЕ ДРУГИЕ ТОЧКИ ВХОДА	ВСЕ ДРУГИЕ ТОЧКИ ВХОДА

Фиг. 67


Обработка установки приоритета
S441
Да
S442
Нет
S443
Да
S444
Нет
S445
Да
S446
Нет
S447
Нет
S448
Да
S449
Да
S450
Нет
S451
Да
S452
Нет
S453
Да
S454
Нет
S455
Нет
S456
Да
Конец

Фиг. 68