Международный союз электросвязи

МСЭ-R
Сектор радиосвязи МСЭ

Рекомендация МСЭ-R BS.2088-1
(10/2019)

Развернутый формат файлов для международного обмена материалами звуковых программ, содержащих метаданные

Серия BS
Радиовещательная служба (звуковая)

Международный союз электросвязи

Предисловие

Роль Сектора радиосвязи заключается в обеспечении рационального, справедливого, эффективного и экономичного использования радиочастотного спектра всеми службами радиосвязи, включая спутниковые службы, и проведении в неограниченном частотном диапазоне исследований, на основании которых принимаются Рекомендации.

Всемирные и региональные конференции радиосвязи и ассамблеи радиосвязи при поддержке исследовательских комиссий выполняют регламентарную и политическую функции Сектора радиосвязи.

Политика в области прав интеллектуальной собственности (ПИС)

Политика МСЭ-R в области ПИС излагается в общей патентной политике МСЭ-T/МСЭ-R/ИСО/МЭК, упоминаемой в Резолюции МСЭ-R 1. Формы, которые владельцам патентов следует использовать для представления патентных заявлений и деклараций о лицензировании, представлены по адресу: https://www.itu.int/ITU-R/go/patents/en, где также содержатся Руководящие принципы по выполнению общей патентной политики МСЭ-T/МСЭ-R/ИСО/МЭК и база данных патентной информации МСЭ-R.

Серии Рекомендаций МСЭ-R

(Представлены также в онлайновой форме по адресу: https://www.itu.int/publ/R-REC/ru.)

Серия	Название
BO	Спутниковое радиовещание
BR	Запись для производства, архивирования и воспроизведения; пленки для телевидения
BS	**Радиовещательная служба (звуковая)**
BT	Радиовещательная служба (телевизионная)
F	Фиксированная служба
M	Подвижные службы, служба радиоопределения, любительская служба и относящиеся к ним спутниковые службы
P	Распространение радиоволн
RA	Радиоастрономия
RS	Системы дистанционного зондирования
S	Фиксированная спутниковая служба
SA	Космические применения и метеорология
SF	Совместное использование частот и координация между системами фиксированной спутниковой службы и фиксированной службы
SM	Управление использованием спектра
SNG	Спутниковый сбор новостей
TF	Передача сигналов времени и эталонных частот
V	Словарь и связанные с ним вопросы

***Примечание**. – Настоящая Рекомендация МСЭ-R утверждена на английском языке в соответствии с процедурой, изложенной в Резолюции МСЭ-R 1.*

Электронная публикация
Женева, 2019 г.

РЕКОМЕНДАЦИЯ МСЭ-R BS.2088-1*

Развернутый формат файлов для международного обмена материалами звуковых программ с метаданными

(2015-2019)

Сфера применения

В настоящей Рекомендации содержится спецификация формата звуковых файлов 64-битового формата WAVE для радиовещательных применений (BW64) с новыми фрагментами <ds64>, <axml>, <bxml>, <sxml> и <chna>, пригодного для представления больших многоканальных файлов с метаданными, в том числе на основе модели определения аудиофайла (ADM), описанной в Рекомендации МСЭ-R BS.2076.

Ключевые слова

Файл, формат файла, метаданные, WAVE, BW64, обмен, звуковая программа, WAV, BWF, RIFF, RF64, файл WAVE, звук с эффектом погружения, модель определения аудиофайла (ADM), последовательная ADM (S-ADM)

Ассамблея радиосвязи МСЭ,

учитывая,

a) что носители данных на базе информационных технологий, в том числе дисковые и ленточные, распространены во всех сферах производства звуковых программ для радиовещания, в том числе в нелинейном редактировании, перегоне в эфире и архивировании;

b) что эта технология обеспечивает значительные преимущества в части эксплуатационной гибкости, организации производственного процесса и автоматизации станций и потому представляется перспективной для модернизации и проектирования студий;

c) что внедрение единого формата файлов для обмена сигналами значительно упростило бы обеспечение взаимодействия отдельного оборудования и удаленных студий, а также способствовало бы желательной интеграции процессов монтажа, эфирного перегона и архивирования;

d) что в файл должен включаться минимальный набор информации, относящейся к радиовещанию, для документирования метаданных об аудиосигнале;

e) что в целях обеспечения совместимости приложений разной сложности должен быть согласован минимальный набор функций, являющийся общим для всех приложений, способных обрабатывать рекомендуемый формат файлов;

f) что в Рекомендации МСЭ-R BS.646 определен формат цифрового аудиосигнала, используемый в производстве звуковых программ для радио- и телевещания;

g) что совместимость с используемыми в настоящее время на рынке форматами файлов позволит сократить трудозатраты отрасли на реализацию этого формата в оборудовании;

h) что стандартный формат представления метаданных, в частности информации о применении кодирования, упростил бы использование информации после обмена программами;

i) что обработка аудиосигнала, в частности нелинейное кодирование и декодирование в процессах со снижением битовой скорости, влияет на его качество;

j) что для усовершенствованных аудиосистем требуется указание метаданных об аудиосигнале в файле;

---

* В феврале 2020 года и сентябре 2023 года 6-я Исследовательская комиссия по радиосвязи внесла поправки редакционного характера в настоящую Рекомендацию в соответствии с Резолюцией МСЭ-R 1.

k) что в усовершенствованных аудиосистемах используются разнообразные многоканальные конфигурации, в том числе звуковые форматы на основе канала, объекта и сцены (например, описанные в Рекомендации МСЭ-R BS.2051);

l) что относящиеся к звуку метаданные, используемые в усовершенствованных звуковых системах, определены в Рекомендации МСЭ-R BS.2076, их общие определения – в Рекомендации МСЭ-R BS.2094, а последовательное представление метаданных (S-ADM) – в Рекомендации МСЭ-R BS.2125;

m) что Рекомендация МСЭ-R BS.1352 имеет ограничения в части размера файла и возможности включения дополнительных метаданных;

n) что размер многоканальных аудиофайлов может превышать 4 ГБ,

рекомендует

1 задавать для целей обмена звуковыми программами частоту дискретизации (часть 1), разрядность (части 4 и 5) и предыскажение (часть 6) аудиосигнала согласно указаниям соответствующих частей Рекомендации МСЭ-R BS.646;

2 использовать формат файлов, описанный в Приложении 1, для обмена звуковыми программами в следующих контекстах:

- в средах с организацией работы на основе WAVE-файлов, где требуется модернизировать вещательные приложения в расчете на контент с эффектом погружения, обеспечив при этом прямую совместимость;
- в рабочих процессах на файловой основе, где предполагается совместно использовать традиционный контент в WAVE-файлах и контент с эффектом погружения;
- в рабочих процессах на файловой основе, где предпочтительно иметь данные и метаданные единым пакетом в общей оболочке.

ПРИМЕЧАНИЕ. – В Приложении 4 указаны изменения, внесенные в спецификации в Приложении 1 предыдущей версии настоящей Рекомендации; эти сведения приводятся только в информационных целях.

Приложение 1
(нормативное)

Спецификация формата файлов BW64

1 Введение

В основе формата BW64 лежит формат аудиофайлов WAVE (описание см. в Приложении 2), который является разновидностью формата RIFF (Resource Interchange File Format – формат файлов для обмена ресурсами). WAVE-файлы содержат звуковые данные. Основная структурная единица формата RIFF – так называемый фрагмент – содержит группу тесно связанных между собой элементов информации: идентификатор фрагмента, целочисленное значение длины фрагмента в байтах и собственно информацию, которую несет фрагмент. RIFF-файл представляет собой совокупность фрагментов. Базовые элементы этого формата согласно документу EBU Tech 3306 используются в формате BW64.

Формат файлов BWF, который описан в Рекомендации МСЭ-R BS.1352, имеет ряд ограничений, в частности:

- размер файла не может превышать 4 ГБ;
- отсутствует поддержка расширенных многоканальных звуковых форматов ввиду ограниченности метаданных, относящихся к звуку;
- недостаточная поддержка технических метаданных.

Формат BW64, описываемый в настоящей Рекомендации, призван устранить эти ограничения и обеспечить максимально возможную совместимость с Рекомендацией МСЭ-R BS.1352 – в частности, за счет общности многих базовых элементов.

В настоящее время растет потребность в передаче метаданных, особенно на основе модели определения аудиофайла (ADM), описанной в Рекомендации МСЭ-R BS.2076. В настоящей Рекомендации определены фрагменты <axml>, <bxml> и <sxml>, предназначенные для хранения и передачи метаданных в виде XML-кода соответственно в формате UTF-8, сжатом формате и в последовательной форме.

Основное назначение фрагмента <chna>, описываемого в настоящей Рекомендации – установление соответствия между каждой дорожкой в файле формата BW64 и идентификаторами метаданных ADM, которые определены в Рекомендации МСЭ-R BS.2076.

Помимо основной задачи – связывания каждой дорожки в файле с соответствующими метаданными ADM, фрагмент <chna> обеспечивает ускоренный доступ к идентификаторам ADM без необходимости обращаться к XML-метаданным (если значения этих идентификаторов находятся в диапазоне, установленном для стандартных конфигураций ADM). Поскольку фрагмент <chna> может иметь фиксированный размер и располагается перед фрагментами <data>, <axml>, <bxml> и <sxml>, его содержимое легче читать, генерировать и оперативно изменять.

Определения типов данных, используемых в этом документе, даны в Приложении 3.

2 Описание формата BW64

2.1 Содержимое файла в формате BW64

Файл формата BW64 должен начинаться с обязательного заголовка "WAVE" и содержать по крайней мере следующие фрагменты:

```
<WAVE-form> ->
    BW64('WAVE'
        <ds64-ck>       // Фрагмент ds64 для 64-битовой адресации
        <fmt-ck>        // Формат аудиосигнала (ИКМ или нет)
        <chna-ck>       // Фрагмент chna для ссылки на ADM
        <axml-ck>       // Фрагмент axml для XML-метаданных
        <bxml-ck>       // Фрагмент bxml для сжатых XML-метаданных
        <sxml-ck>       // Фрагмент sxml для XML-метаданных, относящихся к
                        // подфрагменту или звуковым данным
        <wave-data>     // Звуковые данные
```

ПРИМЕЧАНИЕ 1. – В файле могут присутствовать и другие фрагменты, в том числе выходящие за рамки настоящей Рекомендации. Приложения могут (хотя и не обязательно) интерпретировать или использовать эти фрагменты, поэтому целостность данных, содержащихся в таких неизвестных фрагментах, гарантировать нельзя. Однако приложения, соответствующие требованиям Рекомендации, обеспечивают прозрачную передачу неизвестных фрагментов.

ПРИМЕЧАНИЕ 2. – Допустимо расположить фрагмент <axml>, <bxml> или <sxml> после фрагмента <data>, поскольку длина XML-метаданных будет, скорее всего, неизвестна, и в ряде случаев практичнее было бы иметь известное начальное смещение отсчетов аудиосигнала.

2.2 Фрагменты, унаследованные из стандарта RIFF/WAVE

Определения некоторых фрагментов унаследованы из стандарта RIFF/WAVE. Это следующие фрагменты:

- <RIFF>

- <fmt>
- <data>

Они описаны в пп. 2.6.1–2.6.3.

Формат RIFF/WAVE является подмножеством формата, описанного в Рекомендации МСЭ-R BS.1352. В Рекомендации МСЭ-R BS.1352 определены также следующие дополнительные фрагменты:

- <bext>
- <ubxt>

Эти фрагменты не включаются в формат BW64, так как в нем предусмотрены более гибкие средства представления радиовещательных метаданных.

2.3 Фрагменты и структуры, введенные в формате BW64

В формате BW64 введены следующие новые фрагменты:

- <BW64>
- <ds64>
- <JUNK>
- <axml>, <bxml> или <sxml>
- <chna>

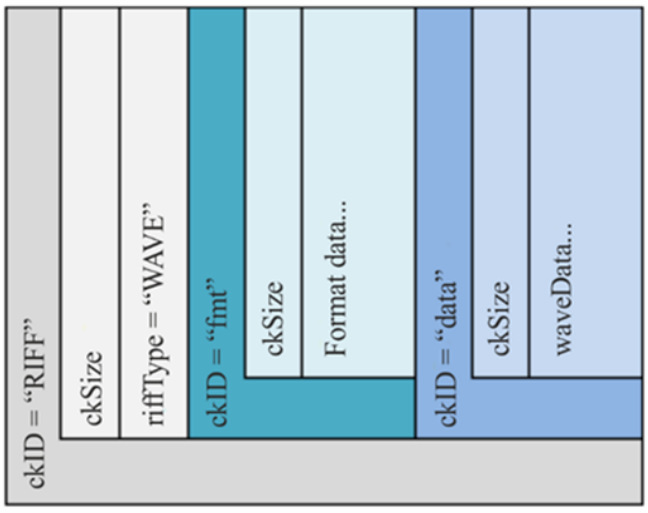
Они описаны в пп. 3–8.

2.4 Использование фрагмента <ds64> для работы с файлами, размер которых превышает 4 ГБ

Свойственное форматам RIFF/WAVE и BWF ограничение на размер файла в 4 ГБ обусловлено 32-битовой адресацией. Имея 32 разряда, можно адресовать не более 4 294 967 296 байт, или 4 ГБ. Для преодоления этого ограничения необходима 64-битовая адресация. Структура простейшего традиционного RIFF/WAVE-файла показана на рисунке 1, где поля ckSize – это 32-битовые числа, представляющие размеры соответствующих фрагментов.

РИСУНОК 1

Структура простейшего RIFF/WAVE-файла



BS.2088-1-01

Если просто изменить размер каждого поля в BWF-файле на 64-битовый, получившийся файл окажется несовместимым со стандартным форматом RIFF/WAVE – это важное, пусть и очевидное соображение.

Вместо этого решено было определить на базе RIFF новый 64-битовый формат под названием BW64, который во многом идентичен исходному RIFF/WAVE, но отличается от него следующим:

- Первые четыре байта файла содержат идентификатор "BW64" вместо "RIFF".
- Добавлен обязательный фрагмент <ds64> (с 64-битовыми данными), который должен идти первым после фрагмента "BW64".

Фрагмент "ds64" содержит два обязательных 64-битовых целочисленных значения, которые заменяют собой два 32-битовых поля формата RIFF/WAVE:

- bw64Size (заменяет поле размера фрагмента <RIFF>);
- dataSize (заменяет поле размера фрагмента <data>).

Оба 32-битовых поля формата RIFF/WAVE подчиняются следующему правилу:

Если содержащееся в поле 32-битовое значение не равно 0xFFFFFFFF, то используется это значение.

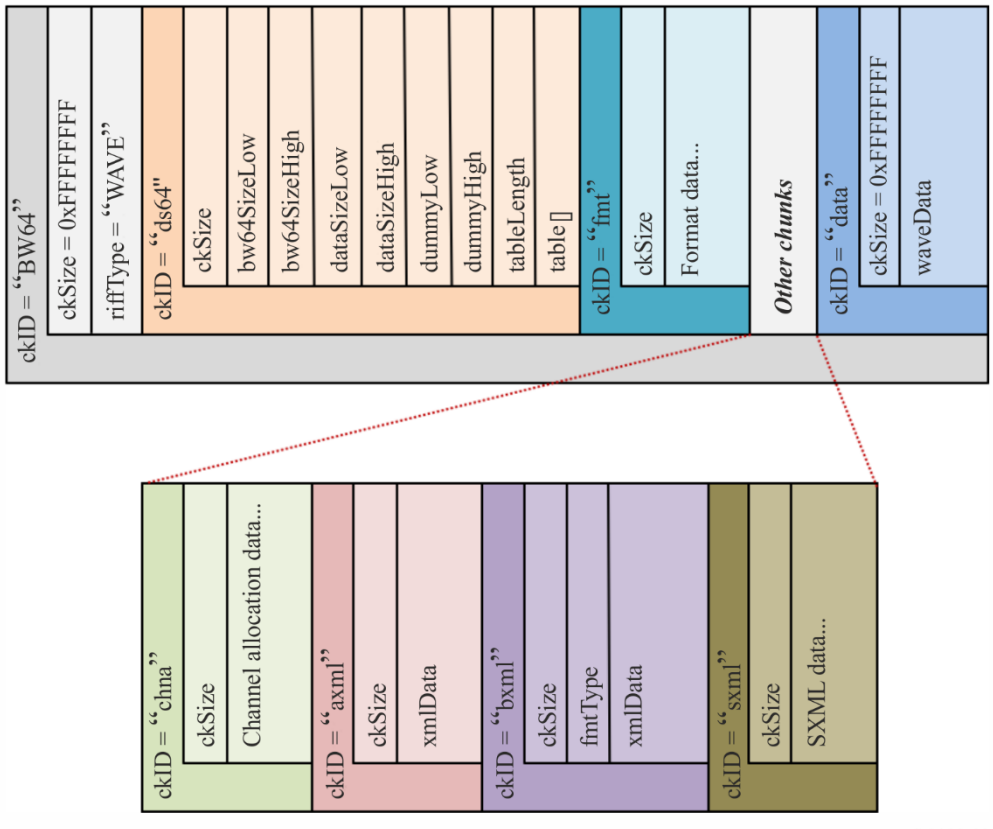
Если это 32-битовое значение равно 0xFFFFFFFF, вместо него используется 64-битовое значение из фрагмента "ds64"

- Может присутствовать один необязательный массив структур (см. Приложение 1) с размерами дополнительных 64-битовых фрагментов.

Полная структура формата файлов BW64 изображена на рисунке 2, где значения полей ckSize для фрагментов <BW64> и <data> установлены равными 0xFFFFFFFF, с тем чтобы можно было использовать 64-битовые значения размеров из фрагмента <ds64>.

РИСУНОК 2

Структура файла в формате BW64



ПРИМЕЧАНИЕ. – Размеры данных фрагментов могут изменяться. Начало каждого фрагмента выравнивается по словам относительно начала файла BW64 для сохранения совместимости с форматом файлов BWF, определенным в Рекомендации МСЭ-R BS.1352. Если размер фрагмента составляет нечетное число байтов, то после фрагмента добавляется заполняющий байт с нулевым значением. При этом значение ckSize не содержит заполняющего байта.

2.5 Обеспечение совместимости между форматами RIFF/WAVE и BW64

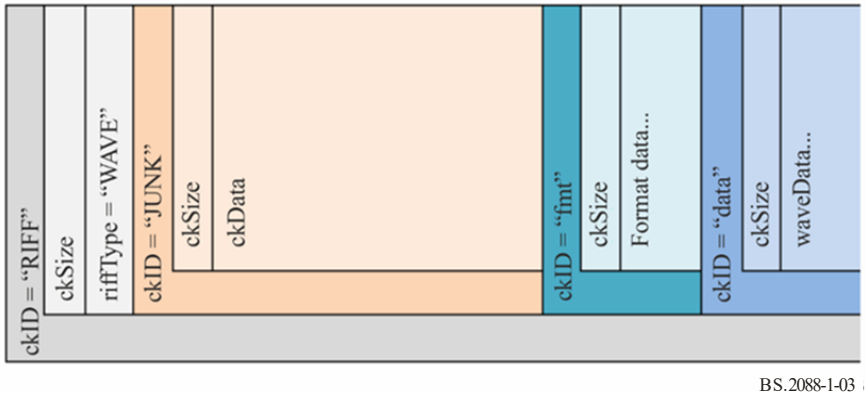
Несмотря на более высокие частоты дискретизации и многоканальный звук, некоторые используемые при производстве программ аудиофайлы по размеру неизбежно будут меньше 4 ГБ и поэтому должны оставаться в кратком формате RIFF/WAVE, который описан в Приложении 2. Проблема в том, что в приложении для записи звука невозможно заранее определить, превысит ли размер аудиофайла 4 ГБ по окончании записи (то есть необходимо ли использовать формат BW64).

Выход – предусмотреть в приложении возможность динамической смены формата с RIFF/WAVE на BW64 при переходе 4-гигабайтовой границы во время записи.

Для этого в RIFF/WAVE-файле дополнительно резервируют место, вставляя туда фрагмент <JUNK> того же размера, что и фрагмент <ds64>. Зарезервированное место не выполняет никакой функции в кратком формате WAVE, но становится фрагментом <ds64>, если возникает необходимость в переходе к формату BW64. На схеме, изображенной на рисунке 3, показан фрагмент-заполнитель <JUNK>, расположенный перед фрагментом <fmt>.

РИСУНОК 3

Структура файла с фрагментом JUNK



В начале записи в приложении, поддерживающем формат BW64, создается стандартный RIFF/WAVE-файл, в котором первым идет фрагмент "JUNK". Далее в ходе записи в приложении проверяются размеры RIFF-файла и данных. Если они превышают 4 ГБ, делается следующее.

- Идентификатор ckID фрагмента <JUNK> заменяется на <ds64>. (Тем самым ранее существовавший фрагмент <JUNK> преобразуется во фрагмент <ds64>).
- Во фрагмент <ds64> вставляются значения размера RIFF-файла и размера фрагмента "data".
- Значения размера RIFF-файла и размера фрагмента "data" в 32-битовых полях устанавливаются на 0xFFFFFFFF.
- В первых четырех байтах файла идентификатор "RIFF" заменяется на "BW64".
- Запись продолжается.

2.6 Фрагменты и структуры, унаследованные из формата RIFF/WAVE

Фрагменты, унаследованные из формата RIFF/WAVE, показаны ниже:

```
struct RiffChunk            //  объявление структуры RiffChunk
{
    CHAR   ckID[4];         //  'RIFF'
    DWORD  ckSize;          //  четырехбайтовое значение размера фрагмента в традиционном
                                файле формата RIFF/WAVE
    CHAR   riffType[4];     //  'WAVE'
};
```

```
struct FormatChunk          //  объявление структуры FormatChunk
{
    CHAR   ckID[4];         //  'fmt '
    DWORD  ckSize;          //  четырехбайтовое значение размера фрагмента 'fmt'
    WORD   formatTag;       //  WAVE_FORMAT_PCM = 0x0001 и т. д.
    WORD   channelCount;    //  1 - моно, 2 - стерео и т. д
    DWORD  sampleRate;      //  32000, 44100, 48000 и т. д.
    DWORD  bytesPerSecond;  //  важно только для форматов со сжатием
    WORD   blockAlignment;  //  размер контейнера (в байтах) для одного набора отсчетов
                               аудиосигнала
    WORD   bitsPerSample;   //  допустимые значения разрядности отсчета - 16, 20 или 24
    WORD   cbSize;          //  следует исключить, так как extraData не используется,
                            //  но если присутствует, то должно быть присвоено нулевое
                               значение
    CHAR   extraData[22];   //  дополнительные данные WAVE_FORMAT_EXTENSIBLE, в
                               необходимых случаях;
                            //  не должно использоваться, так как cbSize будет равняться
                               нулю или отсутствовать.
};

struct DataChunk            //  объявление структуры DataChunk
{
    CHAR   ckID[4];         //  'RIFF'
    DWORD  ckSize;          //  четырехбайтовое значение размера фрагмента 'data'
    CHAR   waveData[ ];     //  отсчеты аудиосигнала
};
```

Пустые квадратные скобки массива указывают на то, что массив может содержать переменное число элементов (в том числе ноль).

2.6.1 Элементы фрагмента <RIFF>

Фрагмент <RIFF> занимает верхний уровень в файле.

Поле	Описание
ckID	Четырехсимвольный массив {'R', 'I', 'F', 'F'}, служащий идентификатором фрагмента.
ckSize	Четырехбайтовое значение размера файла.
riffType	Четырехсимвольный массив {'W', 'A', 'V', 'E'}, который указывает на тип аудиофайла – WAVE.

2.6.2 Элементы фрагмента <fmt>

Фрагмент <fmt> содержит информацию о форматах отсчетов аудиосигнала, которые хранятся во фрагменте <data>.

Поле	Описание
ckID	Четырехсимвольный массив {'f', 'm', 't', ' '}, служащий идентификатором фрагмента.

ckSize	Четырехбайтовое значение размера фрагмента.
formatTag	Двухбайтовое значение, указывающее формат отсчетов аудиосигнала. Значение 0x0001 соответствует формату ИКМ, 0x0000 – неизвестным форматам.
channelCount	Двухбайтовое значение, указывающее количество звуковых дорожек в файле.
sampleRate	Четырехбайтовое значение, указывающее частоту дискретизации аудиосигнала в герцах.
bytesPerSecond	Средняя скорость в байтах в секунду, с которой должны передаваться данные звуковой волны. По этому значению может оцениваться размер буфера в программном обеспечении для воспроизведения звука.
blockAlignment	Размер блока данных звуковой волны в байтах. В программном обеспечении для воспроизведения звука за один прием должен обрабатываться объем данных, кратный **blockAlignment**, поэтому значение **blockAlignment** можно использовать для выравнивания буфера.
bitsPerSample	Разрядность отсчета аудиосигнала для каждого канала. Предполагается, что у всех каналов разрешение на отсчет одинаково. Если это поле не требуется, ему должно быть присвоено нулевое значение.
cbSize	Размер структуры extraData в байтах.
extraData	Дополнительные данные, служащие для хранения информации в формате WAVE_FORMAT_EXTENSIBLE. Не подлежат использованию в формате BW64.

Фрагмент FormatChunk специально предназначен для хранения звуковых данных в формате ИКМ.

Массив extraData во фрагменте FormatChunk используется, если элементу formatTag присвоено значение 0xFFFE (WAVE_FORMAT_EXTENSIBLE). Поскольку многоканальный звук следует описывать с помощью ADM-метаданных, следует избегать использования элемента formatTag. Вместе с тем в конкретных реализациях может быть обеспечена возможность чтения и адекватной обработки файлов с элементом formatTag.

Для обеспечения отсутствия противоречий между содержимым фрагмента FormatChunk и информацией, содержащейся во фрагментах <chna>, <axml>, <bxml> и <sxml>, рекомендуется присваивать элементу formatTag значение 0x0001 для звука в формате ИКМ и 0x0000 (неизвестный формат) для звука в форматах, отличных от ИКМ.

2.6.3 Элементы фрагмента <data>

Фрагмент <data> предназначен для хранения отсчетов аудиосигнала.

Поле	**Описание**
ckID	Четырехсимвольный массив {‘d’, ‘a’, ‘t’, ‘a’}, служащий идентификатором фрагмента.
ckSize	Четырехбайтовое значение размера фрагмента.
waveData	Здесь хранятся отсчеты аудиосигнала. Порядок следования байтов – начиная с младшего. Если дорожек несколько, то отсчеты всех дорожек чередуются. Например, в случае 16-битового двухдорожечного аудиофайла имеем:

Байт	Отсчет	Дорожка
0	0 – мл. байт	1
1	0 – ст. байт	1
2	0 – мл. байт	2
3	0 – ст. байт	2
4	1 – мл. байт	1
5	1 – ст. байт	1
6	1 – мл. байт	2
7	1 – ст. байт	2

3 Фрагмент верхнего уровня BW64

3.1 Определение

Фрагмент верхнего уровня <BW64> заменяет собой фрагмент <RIFF>, используемый в 32-битовых файлах. Если присутствует этот фрагмент, в файле также должен существовать фрагмент <ds64>, содержащий 64-битовые значения размеров. Фрагмент <BW64> описан ниже.

```
struct BW64Chunk          // объявление структуры BW64Chunk
{
    CHAR ckID[4];         // 'BW64'
    DWORD ckSize;         // 0xFFFFFFFF предписывает вместо этих данных использовать
                          // bw64SizeHigh и bw64SizeLow из фрагмента 'ds64'
    CHAR BW64Type[4];     // 'WAVE'
};
```

3.2 Элементы фрагмента <BW64>

Поле	Описание
ckID	Четырехсимвольный массив {'b', 'w', '6', '4'}, служащий идентификатором фрагмента.
ckSize	Четырехбайтовое значение, которое должно быть равно 0xFFFFFFFF для обозначения, что вместо него для определения размеров следует пользоваться фрагментом <ds64>.
BW64Type	Четырехсимвольный массив {'W', 'A', 'V', 'E'}, который указывает на тип аудиофайла – WAVE.

4 Фрагменты DS64 и JUNK

4.1 Определения

Фрагмент <ds64> содержит 64-битовые значения размера файла, фрагмента <data>, а также массив из 64-битовых значений размеров других фрагментов, которые могут быть определены в файле. Ниже показана структура фрагмента <ds64>, а за ней структура таблицы **ChunkSize64**, содержащей размеры возможных дополнительных фрагментов (помимо <data>). Пустые квадратные скобки массива указывают на то, что массив может содержать переменное число элементов (в том числе ноль).

```
struct DataSize64Chunk    //  объявление структуры DataSize64Chunk
{
    CHAR ckID[4];         //  'ds64' – идентификатор фрагмента FOURCC
```

```
    DWORD ckSize;          //  четырехбайтовое значение размера фрагмента <ds64>
    DWORD bw64SizeLow;     //  младшее слово четырехбайтового значения размера фрагмента
                               <BW64>
    DWORD bw64SizeHigh;    //  старшее слово четырехбайтового значения размера фрагмента
                               <BW64>
    DWORD dataSizeLow;     //  младшее слово четырехбайтового значения размера фрагмента
                               <data>
    DWORD dataSizeHigh;    //  старшее слово четырехбайтового значения размера фрагмента
                               <data>
    DWORD dummyLow;        //  холостое значение для перекрестной совместимости
    DWORD dummyHigh;       //  холостое значение для перекрестной совместимости
    DWORD tableLength;     //  количество действительных записей в массиве "table"
    ChunkSize64 table[ ]; //  массив размеров фрагментов, превышающих 4 ГБ
};

struct ChunkSize64         //  объявление структуры ChunkSize64
{
    CHAR ckID[4];          //  идентификатор фрагмента, требующего 64-битовой адресации;
                           //  например, если размер фрагмента <axml> превышает 4 ГБ,
                               указывается 'axml'
    DWORD ckSizeLow;       //  младшее слово четырехбайтового значения размера фрагмента
    DWORD ckSizeHigh;      //  старшее слово четырехбайтового значения размера фрагмента
};
```

Фрагмент <JUNK> – заполнитель для фрагмента <ds64>, используемый в том случае, если генерируется 32-битовый аудиофайл, который может впоследствии потребовать динамического преобразования в 64-битовый файл. По своему размеру фрагмент <JUNK> должен в точности совпадать с фрагментом <ds64>, который его заменит. Структура фрагмента показана ниже:

```
struct JunkChunk           //  объявление структуры JunkChunk
{
    CHAR   ckID[4];        //  'JUNK'
    DWORD  ckSize;         //  Четырехбайтовое значение размера фрагмента 'JUNK'. Должно
                               быть
                           //  не менее 28, если фрагмент служит заполнителем
                           //  для фрагмента 'ds64'.
    CHAR   ckData[];       //  холостые байты
};
```

4.2 Элементы фрагмента <ds64>

Поле	Описание
ckID	Четырехсимвольный массив {'d', 's', '6', '4'}, служащий идентификатором фрагмента.
ckSize	Четырехбайтовое значение размера фрагмента <ds64>.
bw64SizeLow	Младшее слово четырехбайтового значения размера фрагмента <BW64>. 64-битовое значение размера хранится в виде 0xHHHHLLLL, если <bw64SizeLow> = 0xLLLL и <bw64SizeHigh> = 0xHHHH. Это 32-битовое беззнаковое значение хранится в формате с порядком следования байтов начиная с младшего.

bw64SizeHigh	Старшее слово четырехбайтового значения размера фрагмента <BW64>. Это 32-битовое беззнаковое значение хранится в формате с порядком следования байтов начиная с младшего.
dataSizeLow	Младшее слово четырехбайтового значения размера фрагмента <data>. 64-битовое значение размера хранится в виде 0xHHHHLLLL, если <dataSizeLow> = 0xLLLL и <dataSizeHigh> = 0xHHHH. Это 32-битовое беззнаковое значение хранится в формате с порядком следования байтов начиная с младшего.
dataSizeHigh	Старшее слово четырехбайтового значения размера фрагмента <data>. Это 32-битовое беззнаковое значение хранится в формате с порядком следования байтов начиная с младшего.
dummyLow	Четырехбайтовое холостое значение, которое следует игнорировать при чтении и устанавливать равным нулю при записи. Присутствует для совместимости со спецификацией формата RF64 (см. документ EBU Tech 3306), где используется для хранения информации о размере фрагмента <fact>, отсутствующего в формате BW64.
dummyHigh	Четырехбайтовое холостое значение, которое следует игнорировать при чтении и устанавливать равным нулю при записи. Служит для той же цели, что и <dummyLow>.
tableLength	Количество действительных записей в массиве "ChunkSize64 table".
ChunkSize64 table	Массив размеров фрагментов, превышающих 4 ГБ.

Таблица **ChunkSize64** определяется следующим образом. Для хранения длин всех фрагментов, помимо <data>, в необязательной части фрагмента <ds64> используется массив структур **ChunkSize64**. В настоящее время есть только один тип фрагментов, кроме <data>, размер экземпляров которого может превышать 4 ГБ – это <axml> (в очень больших объектных аудиофайлах).

Поле	**Описание**
ckID	Четырехсимвольный массив для ссылки на поле <ckID> фрагмента, требующего 64-битовой адресации. Например, фрагменту <axml> соответствует массив {‘a’, ‘x’, ‘m’, ‘l’}.
ckSizeLow	Младшее слово четырехбайтового значения размера фрагмента, на <ckID> которого дается ссылка. Это 32-битовое беззнаковое значение хранится в формате с порядком следования байтов начиная с младшего.
ckSizeHigh	Старшее слово четырехбайтового значения размера фрагмента, на <ckID> которого дается ссылка. Это 32-битовое беззнаковое значение хранится в формате с порядком следования байтов начиная с младшего.

4.3 Элементы фрагмента <JUNK>

Поле	**Описание**
ckID	Четырехсимвольный массив {‘J’, ‘U’, ‘N’, ‘K’}, служащий идентификатором фрагмента.
ckSize	Четырехбайтовое значение размера фрагмента <JUNK>. Должно быть не менее 28, если этот фрагмент служит заполнителем для фрагмента <ds64>.
ckData	Холостые данные, которые следует игнорировать.

5 Фрагмент AXML

5.1 Определение

Фрагмент <axml> может содержать любые данные в формате XML 1.0 или более поздних версий. Этот широко распространенный формат обмена данными [1]. Следует иметь в виду, что фрагмент <axml> может содержать блоки XML-кода более чем из одной схемы. Он может следовать в любом порядке относительно других RIFF-фрагментов в том же файле.

Фрагмент <axml> состоит из заголовка и следующих за ним данных в формате XML. Общая длина этого фрагмента не фиксирована.

Пример того, как фрагмент <axml> в формате BW64 можно использовать для хранения радиовещательных метаданных (в том числе тех параметров, которые ранее указывались во фрагментах <bext> и <ubxt>), см. в п. 11.

```
            struct axml_chunk
 {
     CHAR    ckID[4];       // {'a','x','m','l'}
     DWORD   ckSize;        // размер фрагмента <axml> в байтах
     CHAR    xmlData[];    // текстовые данные в виде XML
 };
```

Поскольку XML-код может занимать более 4 ГБ, может возникнуть необходимость во фрагменте <ds64> с 64-битовым полем размера для фрагмента <axml>. Ниже приведен отрывок псевдокода, показывающий, как это можно осуществить с помощью массива "table" во фрагменте <ds64>.

```
DataSize64Chunk.tableLength = 1;   // количество действительных записей в массиве "table"
DataSize64Chunk.table[0] = {
    ChunkSize64.ckID = {`a`, `x`, `m`, `l`};   // идентификатор фрагмента <axml>
     ckSizeLow = xxxx     // младшее слово четырехбайтового значения размера фрагмента
     ckSizeHigh = xxxx    // старшее слово четырехбайтового значения размера фрагмента
}
```

5.2 Элементы фрагмента <axml>

ckID	Четырехсимвольный массив {‘a’, ‘x’, ‘m’, ‘l’}, служащий идентификатором фрагмента.
ckSize	Размеры раздела "xmlData" фрагмента в байтах. (Исключая 8 байтов, занимаемых ckID и ckSize.)
xmlData	Это поле содержит текстовую информацию в виде XML.

XML-данные имеют иерархическую структуру и хранятся в текстовых строках формата XML 1.0 или более поздних версий.

Если устройство-получатель не способно интерпретировать содержимое фрагмента <axml> в соответствии со спецификацией, указанной в XML-коде, весь фрагмент игнорируется.

6 Фрагмент BXML

6.1 Определение

Вместо фрагмента <axml> сжатые XML-данные могут содержаться во фрагменте <bxml>.

Фрагмент <bxml> состоит из заголовка, за которым следуют XML-данные, сжатые методом, который указан в поле **fmtType**. Общая длина фрагмента не фиксирована.

```
            struct bxml_chunk
{
    CHAR    ckID[4];       // {'b','x','m','l'}
    DWORD   ckSize;        // размер фрагмента <bxml> в байтах
    WORD    fmtType;       // тип сжатия, 0x0001="gzip" и т. д.
    CHAR    xmlData[];    // текстовые XML-данные, сжатые указанным методом
};
```

Поскольку сжатые XML-данные могут занимать более 4 Гбайт, может потребоваться использование фрагмента <ds64>, чтобы для фрагмента <bxml> было разрешено поле 64-битового размера. Ниже приведен псевдокод, иллюстрирующий, как этого можно добиться с помощью массива "table" во фрагменте <ds64>.

```
DataSize64Chunk.tableLength = 1;   // количество допустимых записей в массиве "table"
DataSize64Chunk.table[0] = {
    ChunkSize64.ckID = {`b`, `x`, `m`, `l`};   // идентификатор фрагмента <bxml>
     ckSizeLow = xxxx     // младшее слово четырехбайтового значения размера фрагмента
     ckSizeHigh = xxxx    // старшее слово четырехбайтового значения размера фрагмента
}
```

6.2 Элементы фрагмента <bxml>

ckID	Четырехсимвольный массив {‘b’, ‘x’, ‘m’, ‘l’}, служащий идентификатором фрагмента.
ckSize	Размеры раздела "xmlData" фрагмента в байтах. (Исключая 8 байтов, занимаемых ckID и ckSize.)
fmtType	2-байтовое значение, указывающее метод сжатия XML-текста. Значение 0x0001 означает, что используется метод сжатия gzip (IETF RFC 1952). Для несжатого XML-текста применяется значение 0x0000.
xmlData	Это поле содержит код XML, сжатый методом, указанным в поле fmtType.

7 Фрагмент SXML

7.1 Определение

Фрагмент <sxml> может содержать любые данные в виде сжатого или несжатого XML-текста, совместимые с форматом XML версии 1.0 или более поздней версии, которые связаны с сегментами звуковых данных. Эти фрагменты могут следовать в любом порядке вместе с другими фрагментами RIFF в одном и том же файле.

Фрагмент <sxml> состоит из заголовка, за которым следуют подфрагменты **(SubXMLChunk)** со сжатыми или несжатыми XML-данными, как указанно в поле **fmtType**. Каждому подфрагменту **SubXMLChunk** соответствует уникальное число отсчетов аудиосигнала, следующих за подфрагментами **SubXMLChunk**. Фрагмент <sxml> дополняется необязательной таблицей точек выравнивания, которая обеспечивает доступ к выбранному фрагменту **SubXMLChunk** на основе меток времени. Общая длина фрагмента <sxml> не фиксирована.

Фрагмент <sxml> может использоваться для передачи метаданных, изменяющихся во времени, например, последовательного представления ADM, определенного в Рекомендации МСЭ-R BS.2125.

```
            struct sxml_chunk
{
    CHAR    ckID[4];              // {'s','x','m','l'}
    DWORD   ckSize;               // размер фрагмента <sxml> в байтах
    WORD    fmtType;              // метод сжатия, 0x0001="gzip" и т. д.
    DWORD subXMLCkTbSizeLow; // младшее слово четырехбайтового значения размера
nSubXMLChunks +
                                  // SubXMLChunk table[]
    DWORD subXMLCkTbSizeHigh; // старшее слово четырехбайтового значения размера
nSubXMLChunks +
                                  // SubXMLChunk table[]
    DWORD   nSubXMLChunks;        // число подфрагментов с XML-данными
    SubXMLChunk table[];          // массив подфрагментов с XML-данными
    DWORD   nAlignmentPoints;     // число точек выравнивания
    AlignmentPoint table[];       // массив точек выравнивания
};

struct SubXMLChunk
{
    DWORD   subXMLChunkSize;      // размер фрагмента SubXMLChunk в байтах
    DWORD   nSamplesSubDataChunk;// число отсчетов аудиосигнала, связанных с фрагментом
                                     SubXMLChunk
    CHAR    xmlData[];           // сжатые или несжатые XML-данные
};

struct AlignmentPoint
{
    DWORD  subXMLChunkByteOffsetLow;  // младшие 4 байта смещения SubXMLChunk
    DWORD  subXMLChunkByteOffsetHigh; // старшие 4 байта смещения SubXMLChunk
    DWORD  nSamplesAlignPointLow;     // младшие 4 байта счетчика точек выравнивания
    DWORD  nSamplesAlignPointHigh;    // старшие 4 байта счетчика точек выравнивания
};
```

Поскольку сжатые или несжатые XML-данные могут занимать более 4 Гбайт, может потребоваться использование фрагмента <ds64>, чтобы для фрагмента <sxml> было разрешено поле 64-разрядного размера. Ниже приведен псевдокод, иллюстрирующий, как этого можно добиться с помощью массива "table" во фрагменте <ds64>.

```
DataSize64Chunk.tableLength = 1;   // количество действительных записей в массиве "table"
DataSize64Chunk.table[0] = {
    ChunkSize64.ckID = {`s`, `x`, `m`, `l`};   // идентификатор фрагмента <sxml>
     ckSizeLow = xxxx     // младшее слово четырехбайтового значения размера фрагмента
     ckSizeHigh = xxxx    // старшее слово четырехбайтового значения размера фрагмента
}
```

7.2 Элементы фрагмента <sxml>

Поле	Описание
ckID	Массив из четырех символов {‘s’, ‘x’, ‘m’, ‘l’}, используемый для идентификации фрагмента.
ckSize	Размер раздела данных фрагмента в байтах. Не включает 8 байтов, используемых полями ckID и ckSize.
fmtType	2-байтовое значение, указывающее метод сжатия XML-текста. Значение 0x0001 означает, что используется метод сжатия gzip (IETF RFC 1952). Для несжатых XML-данных применяется значение 0x0000.
subXMLCkTbSizeLow	Младшее слово четырехбайтового значения размера массива SubXMLChunk table[], включая 4 байта поля nSubXMLChunks. 64-битовое значение размера хранится в виде 0xHHHHLLLL, если <subXMLCkTbSizeLow> = 0xLLLL и <subXMLChTbSizeHigh> = 0xHHHH. Это 32-битовое беззнаковое значение хранится в формате с порядком следования байтов начиная с младшего.
subXMLCkTbSizeHigh	Старшее слово четырехбайтового значения размера массива SubXMLChunk table[], включая 4 байта поля nSubXMLChunks.
nSubXMLChunks	Количество действительных записей в массиве "SubXMLChunk table".
SubXMLChunk table	Массив подфрагментов с XML-данными.
nAlignmentPoints	Количество действительных записей в массиве "AlignmentPoint table".
AlignmentPoint table	Массив точек выравнивания.

Таблица **SubXMLChunk** определяется следующим образом.

Поле	Описание
subXMLChunkSize	Размеры раздела "xmlData" фрагмента в байтах, исключая 8 байтов, занимаемых subXMLChunkSize и nSamplesSubDataChunk.
nSamplesSubDataChunk	Число отсчетов аудиосигнала на канал, связанных с фрагментом SubXMLChunk.
xmlData	Это поле содержит XML-данные или данные XML, сжатые методом, указанным в поле fmtType.

Таблица **AlignmentPoint** определяется следующим образом.

Поле	Описание
subXMLChunkByteOffsetLow	Выраженное в байтах смещение первого байта поля SubXMLChunk с точкой выравнивания относительно начала фрагмента <sxml>, исключая 8 байтов, используемых полями ckID и ckSize. Это младшие 4 байта смещения первого байта поля SubXMLChunk. 64-битовое значение размера хранится в виде 0xHHHHLLLL, если <subXMLChunkByteOffsetLow> = 0xLLLL и <subXMLChunkByteOffsetHigh> = 0xHHHH. Это 32-битовое беззнаковое значение хранится в формате с порядком следования байтов начиная с младшего..

subXMLChunkByteOffsetHigh Это старшие 4 байта смещения первого байта поля SubXMLChunk с точкой выравнивания. Это 32-битовое беззнаковое значение хранится в формате с порядком следования байтов начиная с младшего.

nSamplesAlignPointLow Отметка времени точки выравнивания, выраженная в количестве отсчетов аудиосигнала на канал от начала фрагмента <data>. Это младшие 4 байта значения счетчика меток времени. 64-битовое значение размера хранится в виде 0xHHHHLLLL, если <nSamplesAlignPointLow> = 0xLLLL и <nSamplesAlignPointHigh> = 0xHHHH. Это 32-битовое беззнаковое значение хранится в формате с порядком следования байтов начиная с младшего.

nSamplesAlignPointHigh Это старшие 4 байта значения счетчика меток времени. Это 32-битовое беззнаковое значение хранится в формате с порядком следования байтов начиная с младшего.

8 Фрагмент CHNA

8.1 Определение

Фрагмент <chna> специально предназначен для использования с моделью определения аудиофайла (ADM), которая описана в Рекомендации МСЭ-R BS.2076. Фрагмент начинается с заголовка, за которым следуют поля количества используемых дорожек и уникальных идентификаторов дорожек. Далее идет массив идентификационных структур, каждая из которых содержит идентификаторы соответствующие идентификатору элементов ADM.

Размер фрагмента зависит от количества определяемых уникальных идентификаторов дорожек. Количество идентификационных структур должно быть больше или равно количеству используемых уникальных идентификаторов дорожек. За счет того, что фрагмент может содержать больше идентификационных структур, чем имеется уникальных идентификаторов, в нем облегчается изменение и добавление идентификаторов без изменения размера фрагмента. Например, может быть не ясно, сколько будет изначально сгенерировано уникальных идентификаторов, поэтому если установить количество идентификационных структур во фрагменте <chna> равным 64 (поскольку разработчик считает это более чем достаточным для выполнения поставленной задачи), а программа затем (как пример) сгенерирует 55 уникальных идентификаторов, то этими идентификаторами будут заполнены первые 55 идентификационных структур, а оставшиеся 9 структур будут заполнены нулями.

Идентификаторы элементов ADM могут служить ссылками на ADM-метаданные, хранящиеся во фрагментах <axml>, <bxml> и <sxml>, или на содержимое внешнего файла общих определений. Идентификаторы, у которых последние четыре шестнадцатеричных разряда имеют значение 0x0FFF и менее, считаются ссылками на общие определения согласно Рекомендации МСЭ-R BS.2094 "Общие определения для модели определения аудиофайла" (например, определения каналов "FrontLeft" и "FrontRight"). Любые идентификаторы со значениями 0x1000 считаются ссылками на нестандартные определения и поэтому будут содержаться во фрагментах <axml>, <bxml> и <sxml> аудиофайла.

Структура audioID содержит индекс дорожки во фрагменте <data> (содержащую отсчеты аудиосигнала); первой дорожке соответствует число 1. Она также содержит уникальный идентификатор дорожки из ADM-метаданных. Звуковые элементы дорожки могут меняться на протяжении файла; в данном случае каждому определению сопоставляется свой уникальный идентификатор. Таким образом, каждой дорожке может соответствовать несколько уникальных идентификаторов. Остальные два значения в этой структуре – это ссылки на идентификаторы элементов ADM audioTrackFormat и audioPackFormat. ADM позволяет пропустить элементы audioTrackFormat и audioStreamFormat, если тип формата звукоряда – линейная ИКМ. Тогда вместо элемента audioTrackFormat указывается элемент audioChannelFormat.

```
struct chna_chunk
```

```
{
    CHAR    ckID[4];        // {'c','h','n','a'}
    DWORD   ckSize;         // размер элемента <chna>
    WORD    numTracks;      // количество используемых дорожек
    WORD    numUIDs;        // количество используемых UID дорожек
    audioID ID[N];          // идентификаторы каждой дорожки (где N >= numUIDs)
};

struct audioID
{
    WORD    trackIndex;     // индекс дорожки в файле
    CHAR    UID[12];        // значение audioTrackUID
    CHAR   trackRef[14];   // ссылка на audioTrackFormatID или audioChannelFormatID
    CHAR    packRef[11];    // ссылка на audioPackFormatID
    CHAR    pad;            // байт, дополняющий общее количество байтов до четного
}
```

8.2 Элементы фрагмента <chna>

ckID Четырехсимвольный массив {‘c’, ‘h’, ‘n’, ‘a’}[1], служащий идентификатором фрагмента.

ckSize Размеры раздела "xmlData" фрагмента в байтах, исключая 8 байтов, занимаемых ckID и ckSize.

numTracks Количество используемых дорожек в файле. Даже если дорожка содержит более одного набора идентификаторов, они все равно относятся к одной дорожке.

numUIDs Количество используемых уникальных идентификаторов в файле. Так как одной дорожке может быть сопоставлено множество уникальных идентификаторов (охватывающих разные временны́е интервалы), их количество может превышать значение **numTracks**. Это значение должно совпадать с количеством идентификаторов, определенных в структуре **ID**.

ID Структура, содержащая набор ссылочных идентификаторов звуковых элементов дорожки. Этот массив содержит N идентификаторов, где N >= numUIDs. Если numUIDs < N, содержимое неиспользуемых идентификаторов дорожки заполняется нулями. При чтении фрагмента значение N можно определить по ckSize, так как ckSize = 4 + (N * 40), и N = (ckSize – 4) / 40.

trackIndex Индекс дорожки в файле, начинающийся с 1. Непосредственно соответствует порядку следования дорожек, чередующихся во фрагменте <data>.

UID Значение audioTrackUID дорожки – символьный массив в формате "ATU_xxxxxxxx", где x – шестнадцатеричная цифра.

---

[1] **Примечание**. – Определение DWORD ckID = "chna" не будет уникальным. В вычислительных системах разных архитектур порядок следования символов в строке различается, поэтому дано определение char ckID[4] = {‘c’, ‘h’, ‘n’, ‘a’}.

trackRef Ссылка на audioTrackFormatID дорожки – символьный массив в формате "AT_xxxxxxxx_xx", где x – шестнадцатеричная цифра. Формат AC_xxxxxxxx_00 (расширение "00" дополняет строку для соответствия формату строки audioTrackFormatID и не несет в себе никакого значения), где x – шестнадцатеричная цифра, также используется, когда оба элемента audioTrackFormat, и audioStreamFormat звукоряда с линейной ИКМ опущены и в XML-коде ADM дана прямая ссылка на audioChannelFormat.

packRef Ссылка на audioPackFormatID дорожки – символьный массив в формате "AP_xxxxxxxx", где x – шестнадцатеричная цифра. Когда элемент audioPackFormatID не требуется (то есть когда audioStreamFormat ссылается на audioPackFormat, а не на audioChannelFormat), это поле должно быть заполнено нулевыми символьными значениями.

pad байт, дополняющий общее количество байтов в структуре audioID до четного.

Когда структура **ID** не используется, полю **trackIndex** должно быть присвоено нулевое значение, а во все прочие поля должны быть записаны нулевые строки той же длины, что и обычная строка идентификатора. Например, в поле packRef должна быть записана строка из 11 нулевых символов (значение ASCII равно нулю), а в поле trackRef – из 14 нулевых символов.

8.3 Примеры для сведения

Здесь на нескольких простых примерах иллюстрируется логика использования фрагмента <chna>. В псевдокоде в каждом примере используется строковая запись идентификаторов (например, "AT_00010001_01"), хотя на практике вместо этого должны использоваться массивы символов, чтобы избежать вставки завершающего нулевого символа в конце (то есть в действительности указанный идентификатор должен выглядеть как массив {‘A’, ‘T’, ‘_’, ‘0’, ‘0’, ‘0’, ‘1’, ‘0’, ‘0’, ‘0’, ‘1’, ‘_’, ‘0’, ‘1’}).

8.3.1 Простой стереофайл

Большинство аудиофайлов до сих представляют собой двухканальные стереофайлы, в которых первая дорожка соответствует левому каналу, а вторая – правому. В ADM содержится определение левого канала с идентификатором "AT_00010001_01" и правого канала с идентификатором "AT_00010002_01". Имеется также определение стереофонического пакета с идентификатором "AP_00010002".

Соответствующий псевдокод показан ниже.

```
ckID = {‘c’,’h’,’n’,’a’};
ckSize = 84;
numTracks = 2;
numUIDs = 2;
ID[0]={ trackIndex=1; UID="ATU_00000001"; trackRef="AT_00010001_01"; packRef="AP_00010002"; pad='\0`; };
ID[1]={ trackIndex=2; UID="ATU_00000002"; trackRef="AT_00010002_01"; packRef="AP_00010002"; pad='\0`; };
```

Всего имеется две идентификационные структуры, поэтому неиспользуемых структур в этом примере нет.

Когда в ADM опущены оба элемента audioTrackFomat и audioStreamFormat и дана ссылка на audioChannelFormat, используется следующий код.

```
ckID = {'c','h','n','a'};
ckSize = 84;
numTracks = 2;
numUIDs = 2;
ID[0]={ trackIndex=1; UID="ATU_00000001"; trackRef="AC_00010001_00"; packRef="AP_00010002"; pad='\0`; };
ID[1]={ trackIndex=2; UID="ATU_00000002"; trackRef="AC_00010002_00"; packRef="AP_00010002"; pad='\0`; };
```

8.3.2 Простой пример объектного аудиофайла

Звуковые объекты могут занимать не все время звучания в аудиофайле. Для экономии места неперекрывающиеся объекты могут располагаться на одной дорожке. В этом случае одной дорожке будет соответствовать множество уникальных идентификаторов. В приведенном здесь примере количество идентификационных структур (32 в данном случае) превышает значение numUIDs для демонстрации того, как неиспользуемые структуры заполняются нулями.

```
ckID = {'c','h','n','a'};
ckSize = 1284;
numTracks = 2;
numUIDs = 4;
ID[0]={ trackIndex=1; UID="ATU_00000001"; trackRef="AT_00031001_01"; packRef="AP_00031001"; pad='\0`; };
ID[1]={ trackIndex=1; UID="ATU_00000002"; trackRef="AT_00031003_01"; packRef="AP_00031002"; pad='\0`; };
ID[2]={ trackIndex=1; UID="ATU_00000003"; trackRef="AT_00031004_01"; packRef="AP_00031003"; pad='\0`; };
ID[3]={ trackIndex=2; UID="ATU_00000004"; trackRef="AT_00031002_01"; packRef="AP_00031001"; pad='\0`; };
ID[4]={ trackIndex=0; UID=['\0']*12;      trackRef=['\0']*14;        packRef=['\0']*11;     pad='\0`; };
   :
ID[31]={ trackIndex=0; UID=['\0']*12;      trackRef=['\0']*14;        packRef=['\0']*11;     pad='\0`; };
```

С первой дорожкой связано три уникальных идентификатора, поэтому она будет содержать три различных объекта (с идентификаторами дорожки "AT_00031001_01", "AT_00031003_01" и "AT_00031004_01") на разных временны́х отметках в аудиофайле. Со второй дорожкой связан один уникальный идентификатор, поэтому она будет содержать один объект с тем же идентификатором пакета ("AP_00031001"), что и первый объект на дорожке 1. Это подсказывает, что первый объект имеет два канала, содержимое которых передается дорожками 1 и 2. Назначение каналов и дорожек в этом случае определяется по ADM-метаданным, содержащимся во фрагментах <axml>, <bxml> и <sxml>.

8.3.3 Пример смешанного контента

Файл формата BW64 может содержать контент нескольких типов, например на первых шести дорожках – основной микс 5.1, а на следующих двух – иноязычный стереофонический микс. В Рекомендации МСЭ-R BS.1738 описано несколько конфигураций. В приведенном здесь примере показано, как можно воспроизвести "сценарий производства 5" из этой Рекомендации с помощью фрагмента <chna>. В этом сценарии имеется восемь дорожек, из которых первые шесть содержат полный микс 5.1, а остальные две – международный стереофонический микс. Соответствующий фрагмент <chna> показан ниже.

```
ckID = {'c','h','n','a'};
ckSize = 324;
numTracks = 8;
numUIDs = 8;
ID[0]={ trackIndex=1; UID="ATU_00000001"; trackRef="AT_00010001_01"; packRef="AP_00010003"; pad='\0`; };
ID[1]={ trackIndex=2; UID="ATU_00000002"; trackRef="AT_00010002_01"; packRef="AP_00010003"; pad='\0`; };
ID[2]={ trackIndex=3; UID="ATU_00000003"; trackRef="AT_00010003_01"; packRef="AP_00010003"; pad='\0`; };
ID[3]={ trackIndex=4; UID="ATU_00000004"; trackRef="AT_00010004_01"; packRef="AP_00010003"; pad='\0`; };
ID[4]={ trackIndex=5; UID="ATU_00000005"; trackRef="AT_00010005_01"; packRef="AP_00010003"; pad='\0`; };
ID[5]={ trackIndex=6; UID="ATU_00000006"; trackRef="AT_00010006_01"; packRef="AP_00010003"; pad='\0`; };
ID[6]={ trackIndex=7; UID="ATU_00000007"; trackRef="AT_00010001_01"; packRef="AP_00010002"; pad='\0`; };
ID[7]={ trackIndex=8; UID="ATU_00000008"; trackRef="AT_00010002_01"; packRef="AP_00010002"; pad='\0`; };
```

ADM-метаданные во фрагментах <axml>, <bxml> и <sxml> будут содержать информацию о разделении этих двух миксов.

9 Правила для XML-фрагментов

XML-метаданные могут содержаться во фрагментах трех типов: <axml>, <bxml> и <sxml>. При том что основным назначением этих фрагментов является перенос XML-метаданных ADM (как указано в Рекомендации МСЭ-R BS.2076) или метаданных S-ADM (как указано в Рекомендации МСЭ-R BS.2125), они могут переносить и другие XML-метаданные, такие как радиовещательные метаданные, описанные в п. 11. Поскольку XML-метаданные могут содержаться в нескольких фрагментах, существует риск того, что метаданные одного фрагмента будут противоречить метаданным другого. Поэтому должны соблюдаться следующие правила:

1 Любой конкретный XML-фрагмент должен присутствовать только в одном экземпляре.

2 Если переносятся метаданные ADM:

   a) они должны присутствовать только во фрагменте <axml> или во фрагменте <bxml>, но не в обоих;

   b) должен присутствовать фрагмент <chna> с перекрестной ссылкой на метаданные ADM.

3 Если переносятся метаданные S-ADM, они должны присутствовать только во фрагменте <sxml>.

4 Если переносятся как метаданные ADM, так и метаданные S-ADM, они должны быть независимы друг от друга (то есть не должны ссылаться друг на друга).

5 Если переносятся другие метаданные (отличные от ADM и S-ADM):

   a) при необходимости они могут переноситься в одном фрагменте с метаданными ADM и S-ADM;

   b) в этих "других метаданных" не должно содержаться ничего, что уже описано в существующих метаданных ADM или S-ADM;

   c) если "другие метаданные" содержат ссылки на метаданные ADM или S-ADM, то указанные метаданные ADM или S-ADM должны присутствовать в файле.

10 Совместимость с Рекомендацией МСЭ-R BS.1352

Поскольку формат файлов BWF (Рекомендация МСЭ-R BS.1352) представляет собой краткую форму формата RIFF/WAVE (описанного в Приложении 2) с дополнительными фрагментами, в частности <bext>, возникает вопрос о совместимости между форматами BWF и BW64.

Фрагменты BWF Рек. МСЭ-R BS.1352-3	Фрагменты BW64 Рек. МСЭ-R BS.2088-0	Фрагменты BW64 Рек. МСЭ-R BS.2088-1	Способ обработки
<fmt>	<fmt>	<fmt>	Обычный
<data>	<data>	<data>	Обычный
<fact>	<fact>	<fact>	Обычный[, хотя этот фрагмент может быть опущен за вероятной избыточностью]
–	<ds64>	<ds64>	См. п. 2.4 и п. 4
–	<JUNK>	<JUNK>	См. п. 2.4 и п. 4
–	<chna>	<chna>	Распределение каналов см. в п. 8. Примечание. – Рекомендация МСЭ-R BS.2088-0 не поддерживает ссылок на audioChannelFormat.
–	<axml>	<axml>, <bxml> или <sxml>	См. пп. 5–7. Используется для радиовещательных метаданных, которые содержались бы во фрагменте <bext>.
<bext>	–	-	Если считывается фрагмент <bext>, его следует преобразовать в соответствующие фрагменты <axml>, <bxml> и <sxml>, которые будут содержать ADM-метаданные и другие радиовещательные метаданные в виде XML. Подробнее см. в п. 10.

11 Генерация радиовещательных метаданных в виде XML

Согласно Рекомендации МСЭ-R BS.1352, радиовещательные метаданные хранятся во фрагментах <bext> и <ubxt>. Длина и перечень полей в этих фрагментах фиксированы, что не позволяет хранить в них другие радиовещательные метаданные. Фрагменты <axml>, <bxml> и <sxml> в формате BW64 могут содержать любые метаданные в виде XML, поэтому в них можно хранить радиовещательные метаданные, в том числе те параметры, которые указаны во фрагментах <bext> и <ubxt>.

Для хранения параметров <bext>/<ubxt> во фрагментах <axml>, <bxml> и <sxml> должна использоваться следующая XML-структура, где эти параметры обозначены комментариями с префиксом "BEXT".

```
          <?xml version="1.0" encoding="UTF-8"?>
<ebuCoreMain                        xmlns="urn:ebu:metadata-schema:ebuCore_2015"
xmlns:dc="http://purl.org/dc/elements/1.1/"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <coreMetadata>
    <creator>
      <contactDetails>
        <name>
          <!--BEXT: bextOriginator -->
        </name>
      </contactDetails>
      <organisationDetails>
        <organisationName>
          <!--BEXT: bextOriginatorReference -->
        </organisationName>
      </organisationDetails>
    </creator>

    <description typeDefinition="bextDescription">
      <dc:description>
        <!--BEXT: bextDescription -->
      </dc:description>
    </description>
    <date>
```

```
      <!--BEXT: bextOriginationDate и bextOriginationTime -->
      <created startDate="2000-10-10" startTime="12:00:00"/>
    </date>

    <format>
      <audioFormatExtended>
        <!--BEXT: bextTimeReference -->
        <audioProgramme audioProgrammeID="..." start="00:00:00:00">
          <!--Другие метаданные звуковой программы (audioProgramme) -->
        </audioProgramme>
        <!--Другие ADM-метаданные согласно Рекомендации МСЭ-R BS.2076 -->
      </audioFormatExtended>
      <technicalAttributeString typeDefinition="CodingHistory">
        <!--BEXT: bextCodingHistory -->
      </technicalAttributeString>
    </format>

    <identifier                                              formatLabel="UMID"
formatLink="http://www.ebu.ch/metadata/cs/ebu_IdentifierTypeCodeCS.xml#1.1">
      <dc:identifier>
        <!--BEXT: bextUMID-->
      </dc:identifier>
    </identifier>
  </coreMetadata>
</ebuCoreMain>
```

В основе этого XML-кода лежат схемы метаданных EBUCore [2] и AESCore [3], которые совместимы с Рекомендацией МСЭ-R BS.2076.

При чтении BWF-файла, соответствующего Рекомендации МСЭ-R BS.1352, с намерением преобразовать его в файл формата BW64, следует преобразовать фрагменты <bext>/<ubxt> в описанный здесь XML-код, с тем чтобы включить его во фрагментах <axml>, <bxml> и <sxml>.

12 Расширение файла в формате BW64

Для файлов, соответствующих спецификации формата BW64, определено расширение ".wav". Это позволяет считывать с помощью ранее выпущенного программного обеспечения те фрагменты файла, которые предусмотрены в этом программном обеспечении (главным образом <fmt > и <data>), а тем чтобы по крайней мере иметь доступ к отсчетам аудиосигнала.

Файлам в формате BW64 не рекомендуется присваивать другие расширения, но можно ожидать, что будут случаи некорректного использования для них расширения ".bw64". Поэтому программное обеспечение с поддержкой формата BW64 должно позволять работать с файлами, имеющими такое альтернативное расширение.

13 Библиография

[1] Extensible Markup Language (XML) 1.0 W3C Recommendation 26-November-2008 http://www.w3.org/TR/2008/REC-xml-20081126

[2] EBU Tech 3293, "EBU Core Metadata Set v.1.6".

[3] AES 60-2011, "AES standard for audio metadata – Core audio metadata".

[4] IETF: RFC 1952, “GZIP file format specification version 4.3,” Internet Engineering Task Force, Reston, VA, May, 1996. http://tools.ietf.org/html/rfc1952

Приложение 2
(информационное)

Формат файлов RIFF WAVE (.WAV)

Информация, приведенная в этом Приложении, почерпнута из документов, составляющих спецификацию формата файлов RIFF. Она приводится здесь ввиду отсутствия надежного внешнего источника, на который можно было бы ссылаться, и предназначена только для справки.

1 Волновой формат аудиофайлов (WAVE)

Ниже дается определение формата WAVE. В программном обеспечении следует игнорировать любые неизвестные фрагменты, как и во всех разновидностях формата RIFF. Вместе с тем фрагмент <fmt-ck> всегда располагается перед фрагментом <wave-data>, и оба эти фрагмента являются обязательными в файле WAVE.

```
<WAVE-form> ->
    RIFF('WAVE'
        <fmt-ck>       // Фрагмент с описанием формата
        [<fact-ck>]    // Фрагмент fact (с дополнительными метаданными)
        [<other-ck>]   // Другие необязательные фрагменты
        <wave-data>)   // Звуковые данные
```

В следующих разделах описаны фрагменты, которые могут присутствовать в файлах формата WAVE.

1.1 Фрагмент с описанием формата WAVE

Фрагмент <fmt-ck> содержит описание формата фрагмента <wave-data> и определяется следующим образом.

```
<fmt-ck> ->fmt(<common-fields>
               <format-specific-fields>)
<common-fields> ->
    Struct {
        WORD  wFormatTag;           // Категория формата
        WORD  nChannels;            // Количество каналов
        DWORD nSamplesPerSec;       // Частота дискретизации
        DWORD nAvgBytesPerSec;      // Для оценки размера буфера
        WORD  nBlockAlign;          // Размер блока данных
    }
```

Раздел <common-fields> фрагмента содержит следующие поля:

Поле	Описание
wFormatTag	Число, обозначающее категорию формата WAVE, к которой принадлежит файл. От этого значения зависит содержимое раздела <format-specific-fields> фрагмента <fmt-ck>, а также способ интерпретации данных звуковой волны.

nchannels	Количество каналов, представленных данными звуковой волны, например 1 для монофонического файла и 2 для стереофонического.
nSamplesPerSec	Частота дискретизации (в отсчетах в секунду), на которой следует воспроизводить каждый канал.
nAvgBytesPerSec	Средняя скорость в байтах в секунду, с которой должны передаваться данные звуковой волны. По этому значению может оцениваться размер буфера в программном обеспечении для воспроизведения звука.
nBlockAlign	Выравнивание блока (в байтах) данных звуковой волны. В программном обеспечении для воспроизведения звука за один прием должен обрабатываться объем данных, кратный <nBlockAlign>, поэтому значение <nBlockAlign> можно использовать для выравнивания буфера.

Раздел <format-specific-fields> содержит ноль или более байтов параметров. Состав параметров зависит от категории формата WAVE – подробнее см. в следующих разделах. В программном обеспечении для воспроизведения следует допускать присутствие неизвестных параметров в конце раздела <format-specific-fields> и игнорировать эти параметры.

1.2 Категории формата WAVE

Категория формата файла WAVE задается значением поля <wFormatTag> фрагмента <fmt>. От категории формата зависит представление данных во фрагменте <wave-data>, а также содержимое раздела <format-specific-fields> фрагмента <fmt>.

На данный момент определены, в частности, следующие открытые, незапатентованные категории формата WAVE:

wFormatTag	Значение	Категория формата
WAVE_FORMAT_UNKNOWN	0x0000	Категория неизвестна
WAVE_FORMAT_PCM	0x0001	Формат ИКМ
WAVE_FORMAT_IEEE_FLOAT	0x0003	Формат IEEE с плавающей точкой
WAVE_FORMAT_EXTENSIBLE	0xFFFE	Расширяемый формат WAVE, определяемый субформатом

ПРИМЕЧАНИЕ. – В настоящее время с форматом BW64 используются только категории WAVE_FORMAT_PCM и WAVE_FORMAT_UNKNOWN. Формат файлов ИКМ WAVE подробно описывается ниже в п. 2. Общие сведения о других форматах WAVE приведены в п. 3. В будущем могут быть определены новые форматы WAVE.

Прежде для многоканальных файлов использовался бы формат WAVE_FORMAT_EXTENSIBLE, но в дальнейшем этого следует избегать.

1.3 Фрагмент Fact

Во фрагменте <fact-ck> хранится информация о содержимом файлов WAVE в формате, отличном от ИКМ. Состав этой информации зависит от конкретного формата. Соответственно в данной версии формата BW64 этот фрагмент не используется. Определяется он следующим образом.

```
<fact-ck> ->  fact( <dwSampleLength:DWORD> )
```

<dwSampleLength> – длина массива данных (количество отсчетов аудиосигнала). Поле <nSamplesPerSec> из заголовка формата WAVE в сочетании с полем <dwSampleLength> определяет длительность звучания в секундах.

Фрагмент Fact обязателен для всех новых форматов WAVE, отличных от ИКМ. Для стандартных файлов, формат которых обозначен как WAVE_FORMAT_PCM, этот фрагмент не требуется.

В дальнейшем фрагмент Fact будет включать в себя всю прочую информацию, заявленную в качестве обязательной в будущих разновидностях формата WAVE. Дополнительные поля будут следовать за полем <dwSampleLength>. В приложениях можно определять состав имеющихся полей по значению поля размера фрагмента.

1.4 Другие необязательные фрагменты

Формат WAVE допускает также присутствие ряда других фрагментов. Их подробное описание дается в спецификациях формата WAVE, а также может быть дано в будущих редакциях этих документов.

ПРИМЕЧАНИЕ. – В формате WAVE допускается присутствие других необязательных фрагментов, предназначенных для хранения той или иной информации. Эти фрагменты считаются частными и игнорируются приложениями, в которых не предусмотрена возможность их интерпретации.

2 Формат ИКМ

Если полю <wFormatTag> фрагмента <fmt-ck> присвоено значение WAVE_FORMAT_PCM, то данные звуковой волны представляют собой отсчеты аудиосигнала в формате ИКМ. В этом случае раздел <format-specific-fields> определяется следующим образом:

```
<PCM-format-specific> ->
    struct {
        WORD  nBitsPerSample;     // Размер отсчета
    }
```

Поле <nBitsPerSample> задает разрядность всех отсчетов в каждом канале. Если каналов несколько, разрядность отсчета одинакова во всех каналах.

Значение поля <nBlockAlign> должно равняться (с округлением результата до ближайшего большего целого)

$$nChannels \times BytesPerSample.$$

Значение поля BytesPerSample должно вычисляться путем округления значения nBitsPerSample в бо̸льшую сторону до целого байта. Если разрядность слова отсчета аудиосигнала меньше целого числа байтов, старшие биты отсчета записываются в старшие значащие биты слова данных, а неиспользуемые биты данных, соседствующие с его младшим значащим битом, должны заполняться нулями.

В случае ИКМ-данных значение поля <nAvgBytesPerSec> фрагмента <fmt> должно равняться

$$nSamplesPerSec \times nBblockAlign.$$

ПРИМЕЧАНИЕ 1. – Спецификация формата WAVE допускает, например, упаковку 20-битовых отсчетов из двух каналов в 5 байтов с записью младших битов обоих каналов в один и тот же байт. Настоящая Рекомендация предписывает выделять целое число байтов для хранения одного отсчета аудио сигнала, с тем чтобы снизить уровень неоднозначности интерпретации в различных реализациях и максимальную совместимость при обмене файлами.

2.1 Упаковка данных в файлах WAVE с ИКМ

В одноканальных файлах WAVE отсчеты хранятся последовательно. В стереофонических файлах WAVE канал 0 соответствует левому каналу, а канал 1 – правому. В многоканальных файлах WAVE отсчеты чередуются.

Ниже приведены схемы упаковки данных в 8-битовых монофонических и стереофонических файлах WAVE:

Упаковка данных в 8-битовом монофоническом ИКМ-файле

Отсчет 1	Отсчет 2	Отсчет 3	Отсчет 4
Канал 0	Канал 0	Канал 0	Канал 0

Упаковка данных в 8-битовом стереофоническом ИКМ-файле

Отсчет 1		Отсчет 2	
Канал 0 (левый)	Канал 1 (правый)	Канал 0 (левый)	Канал 1 (правый)

На следующих схемах показана упаковка данных в 16-битовых монофонических и стереофонических файлах WAVE:

Упаковка данных в 16-битовом монофоническом ИКМ-файле

Отсчет 1		Отсчет 2	
Канал 0 младший байт	Канал 0 старший байт	Канал 0 младший байт	Канал 0 старший байт

Упаковка данных в 16-битовом стереофоническом ИКМ-файле

Отсчет 1			
Канал 0 (левый)	Канал 0 (левый)	Канал 1 (правый)	Канал 1 (правый)
младший байт	старший байт	младший байт	старший байт

2.2 Формат данных отсчета аудиосигнала

Каждый отсчет содержится в целом числе i. Размер i есть наименьшее количество байтов, которым может быть представлен отсчет заданного размера. Порядок следования байтов – начиная с младшего. Биты, представляющие амплитуду отсчета, хранятся в старших битах i, а остальные биты заполняются нулями.

Например, если размер отсчета (указанный в поле <nBitsPerSample>) равен 12 битам, каждый отсчет хранится в двухбайтовом целочисленном значении. В этом случае четыре младших бита первого (младшего) байта заполняются нулями. Формат данных, максимальное значение и минимальное значение отсчета ИКМ-аудиосигнала в зависимости от его разрядности определяются следующим образом:

Размер отсчета	**Формат данных**	**Максимальное значение**	**Минимальное значение**
От одного до восьми бит	Беззнаковое целое	255 (0xFF)	0
Девять и более бит	Знаковое целое i	Наибольшее положительное значение i	Наибольшее отрицательное значение i

Например, для 8- и 16-битовых данных звуковой волны в формате ИКМ имеем:

Формат	Максимальное значение	Минимальное значение	Срединное значение
8-битовый ИКМ	255 (0xFF)	0	128 (0x80)
16-битовый ИКМ	32767 (0x7FFF)	–32768 (–0x8000)	0

2.3 Примеры файлов WAVE с ИКМ

Пример 8-битового монофонического файла WAVE с ИКМ, имеющего частоту дискретизации 11,025 кГц:

```
RIFF('WAVE' fmt(1, 1, 11025, 11025, 1, 8)
          data(<wave-data>) )
```

Пример 8-битового стереофонического файла WAVE с ИКМ, имеющего частоту дискретизации 22,05 кГц:

```
RIFF('WAVE' fmt(1, 2, 22050, 44100, 2, 8)
          data(<wave-data>) )
```

2.4 Хранение данных аудиосигнала

Фрагмент <**wave-data**> содержит данные звуковой волны. Он определяется следующим образом.

```
<wave-data> ->  { <data-ck> }
    <data-ck> -> data( <wave-data> )
```

2.5 Фрагмент Fact

Во фрагменте <fact-ck> хранится важная информация о содержимом файла WAVE. Он определяется следующим образом.

```
<fact-ck> ->   fact(<dwFileSize:DWORD>)   // Количество отсчетов
```

Для ИКМ-файлов этот фрагмент необязателен.

В дальнейшем фрагмент fact будет включать в себя всю прочую информацию, заявленную в качестве обязательной в будущих разновидностях формата WAVE. Дополнительные поля будут следовать за полем <dwFileSize>. В приложениях можно определять состав имеющихся полей по значению поля размера фрагмента.

2.6 Другие необязательные фрагменты

Формат WAVE допускает также присутствие ряда других фрагментов. Их подробное описание дается в спецификации формата WAVE, а также может быть дано в будущих ее редакциях.

ПРИМЕЧАНИЕ. – В формате WAVE допускается присутствие других необязательных фрагментов, предназначенных для хранения той или иной информации. Эти фрагменты считаются частными и игнорируются приложениями, в которых не предусмотрена возможность их интерпретации.

3 Расширение формата WAVE

С помощью описываемой ниже структуры для расширения формата WAVE, которая добавляется к фрагменту <fmt-ck>, определяются данные всех форматов, отличных от ИКМ. Эта общая структура используется для всех форматов, кроме ИКМ.

```
typedef struct waveformat_extended_tag {
    WORD  wFormatTag;      // тип формата
    WORD  nChannels;       // количество каналов (моно, стерео и т. д.)
    DWORD nSamplesPerSec;  // частота дискретизации
    DWORD nAvgBytesPerSec; // для оценки размера буфера
    WORD  nBlockAlign;     // размер блока данных
    WORD  wBitsPerSample;  // разрядность отсчета для монофонических данных
    WORD  cbSize;          // размер полей дополнительной информации в байтах
} WAVEFORMATEX;
```

Поле	Описание
wFormatTag	Определяет тип файла WAVE.
nChannels	Количество каналов: 1 – моно, 2 – стерео.
nSamplesPerSec	Частота дискретизации файла WAVE. Должна равняться 48 000, 44 100 и т. д. По этому значению и по количеству отсчетов, указанному во фрагменте fact с дополнительными метаданными, также определяется длительность звучания.
nAvgBytesPerSec	Средняя скорость передачи данных. По этому значению может оцениваться размер буфера в программном обеспечении для воспроизведения звука.
nBlockAlign	Размер блока данных фрагмента <data-ck> в байтах. В программном обеспечении для воспроизведения звука за один прием должен обрабатываться объем данных, кратный <nBlockAlign>, поэтому значение <nBlockAlign> можно использовать для выравнивания буфера.
wBitsPerSample	Разрядность отсчета аудиосигнала для каждого канала. Предполагается, что у всех каналов разрешение на отсчет одинаково. Если это поле не требуется, ему должно быть присвоено нулевое значение.
cbSize	Общий размер полей дополнительной информации в заголовке формата WAVE без учета размера структуры WAVEFORMATEX.

ПРИМЕЧАНИЕ. – В полях, следующих за полем <cbSize>, содержится конкретная информация, необходимая для той категории формата WAVE, которая указана в поле <wFormatTag>.

Приложение 3
(нормативное)

Определения элементарных типов данных

Ниже определены атомарные метки, представляющие собой ссылки на элементарные типы данных. Указан также соответствующий тип данных языка C.

Метка	Смысл	Тип данных C
<CHAR>	8-битовое знаковое целое	signed char
<BYTE>	8-битовое беззнаковое целое	unsigned char
<INT>	16-битовое знаковое целое в формате с порядком следования байтов начиная с младшего	signed int
<WORD>	16-битовое беззнаковое целое в формате с порядком следования байтов начиная с младшего	unsigned int
<LONG>	32-битовое знаковое целое в формате с порядком следования байтов начиная с младшего	signed long
<DWORD>	32-битовое беззнаковое целое в формате с порядком следования байтов начиная с младшего	unsigned long
<FLOAT>	32-битовое значение с плавающей точкой в формате IEEE	Float
<DOUBLE>	64-битовое значение с плавающей точкой в формате IEEE	Double
<STR>	Строка (последовательность символов)	
<ZSTR>	Строка с завершающим нулем	
<BSTR>	Строка с байтовым (8-битовым) префиксом длины	
<WSTR>	Строка с двухбайтовым (16-битовым) префиксом длины	
<BZSTR>	Строка с завершающим нулем и байтовым префиксом длины	

Приложение 4
(информационное)

Изменения в спецификациях Приложения 1

1 Рекомендация МСЭ-R BS.2088-1

В пересмотр 1 настоящей Рекомендации внесены следующие изменения в спецификациях Приложения 1:

– в п. 6 добавлен фрагмент BXML;

– в п. 7 добавлен фрагмент SXML;

– в п. 8 добавлена новая функция, позволяющая опускать элементы audioTrackFormat и audioStreamFormat.

_____