

---

ФЕДЕРАЛЬНОЕ АГЕНТСТВО  
ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

---



НАЦИОНАЛЬНЫЙ  
СТАНДАРТ  
РОССИЙСКОЙ  
ФЕДЕРАЦИИ

ГОСТ Р  
53556.10—  
2014

---

Звуковое вещание цифровое  
**КОДИРОВАНИЕ СИГНАЛОВ  
ЗВУКОВОГО ВЕЩАНИЯ С СОКРАЩЕНИЕМ  
ИЗБЫТОЧНОСТИ ДЛЯ ПЕРЕДАЧИ  
ПО ЦИФРОВЫМ КАНАЛАМ СВЯЗИ**

ЧАСТЬ III  
(MPEG-4 AUDIO)

**Передискретизация аудио**

(ISO/IEC 14496-3:2009, NEQ)

Издание официальное



Москва  
Стандартинформ  
2020

## Предисловие

1 РАЗРАБОТАН ТК 480 «Связь»

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 480 «Связь»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 18 апреля 2014 г. № 386-ст

4 Настоящий стандарт разработан с учетом основных нормативных положений международного стандарта ИСО/МЭК 14496-3:2009 «Информационные технологии. Кодирование аудиовизуальных объектов. Часть 3. Аудио» (ISO/IEC 14496-3:2009 «Information technology — Coding of audio-visual objects — Part 3: Audio», NEQ) [1]

5 ВВЕДЕН ВПЕРВЫЕ

6 ПЕРЕИЗДАНИЕ. Июль 2020 г.

*Правила применения настоящего стандарта установлены в статье 26 Федерального закона от 29 июня 2015 г. № 162-ФЗ «О стандартизации в Российской Федерации». Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок — в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет ([www.gost.ru](http://www.gost.ru))*

© Стандартинформ, оформление, 2014, 2020

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

**Содержание**

1 Область применения .....	1
2 Термины и определения .....	1
3 Условные обозначения .....	2
4 Основные типы .....	3
5 Полезные нагрузки для аудиообъекта .....	4
6 Семантика .....	9
7 Эталонная модель декодера <i>DST</i> .....	17
Приложение А (справочное) Описание кодера .....	24
Библиография .....	26

## НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

## Звуковое вещание цифровое

КОДИРОВАНИЕ СИГНАЛОВ ЗВУКОВОГО ВЕЩАНИЯ С СОКРАЩЕНИЕМ ИЗБЫТОЧНОСТИ  
ДЛЯ ПЕРЕДАЧИ ПО ЦИФРОВЫМ КАНАЛАМ СВЯЗИ

## ЧАСТЬ III (MPEG-4 AUDIO)

## Передискретизация аудио

Sound broadcasting digital. Coding of signals of sound broadcasting with reduction of redundancy for transfer on digital communication channels. Part III (MPEG-4 audio). Oversampled audio

Дата введения — 2015—01—01

## 1 Область применения

Стандарт описывает алгоритм кодирования без потерь MPEG-4 для передискретизированных аудиосигналов.

## 2 Термины и определения

В этом стандарте используются следующие термины и определения:

<i>Audio Channel</i>	Поток битов <i>DSD</i> , предназначенный для одного громкоговорителя.
<i>Audio Frame</i>	Фрейм (кадр), содержащий аудиоданные.
<i>Audio Channel Number</i>	Порядковый номер, присвоенный звуковому каналу. Номера звуковых каналов присваиваются непрерывно, начиная с единицы.
<i>Frame</i>	Блок данных, принадлежащий определенному временному коду. Время воспроизведения фрейма составляет 1/75 с.
<i>Reserved</i>	Все поля, маркированные <i>Reserved</i> (Зарезервировано), резервируются для будущей стандартизации. Все поля <i>Reserved</i> должны быть обнулены.
<i>Silence Pattern</i>	Сгенерированная в цифровой форме кодограмма <i>DSD</i> со следующими свойствами: у всех аудиобайтов одно и то же значение; каждый аудиобайт должен содержать 4 бита, равные нулю, и 4 бита, равные единице.
<i>Direct Stream Digital</i>	Однобитовое передискретизированное представление аудиосигнала.
<i>Direct Stream Transfer</i>	Метод кодирования без потерь, используемый для сигналов <i>DSD</i> в компакт-диске аудио высшего качества.
<i>Half Probability</i>	Половинная вероятность определяется для каждого звукового канала в аудиофрейме, кодируются ли первые биты <i>DSD</i> арифметически, используя значения <i>Ptable</i> , или используя вероятность, равную 1/2.
<i>Mapping</i>	Отображение определяется для каждого сегмента фильтр прогноза и таблицу вероятности.

<i>Prediction Filter</i>	Фильтр прогноза является трансверсальным фильтром, используемым, чтобы предсказать значение следующего бита <i>DSD</i> . Фильтр прогноза характеризуется порядком прогноза и коэффициентами.
<i>Probability Table</i>	Таблица вероятности содержит вероятность того, что для данного вывода фильтра прогноза значение бита <i>DSD</i> предсказывается ошибочно.
<i>Sampling Frequency</i>	Частота дискретизации сигнала <i>DSD</i> должна быть $64 * 44,1$ кГц, $128 * 44,1$ кГц или $256 * 44,1$ кГц.
<i>Segmentation</i>	Каждый звуковой канал в аудиофрейме может быть разделен на сегменты.

### 3 Условные обозначения

#### 3.1 Арифметические и битовые операции

$a \gg b$	Сдвиг <i>a</i> вправо на <i>b</i> битов. Новые биты <i>msb</i> устанавливаются в '0'.
$a \ll b$	Сдвиг <i>a</i> влево на <i>b</i> битов. Новые биты <i>lsb</i> устанавливаются в '0'.
$a b$	Поразрядное ИЛИ для <i>a</i> и <i>b</i> .
$a \& b$	Поразрядное И для <i>a</i> и <i>b</i> .
$\min(a, b)$	Наименьшее значение из <i>a</i> и <i>b</i> .
$\max(a, b)$	Наибольшее значение из <i>a</i> и <i>b</i> .
$\text{mod } b$	Значение <i>b</i> по модулю.
$\text{trunc}(a)$	Значение <i>a</i> , округленное в меньшую сторону.
$ a $	Абсолютное значение <i>a</i> .
$a == b$	Оценить, равны ли <i>a</i> и <i>b</i> .
$a != b$	Оценить, не равны ли <i>a</i> и <i>b</i> .
$a = b$	Переменная <i>a</i> устанавливается в значение <i>b</i> .
$a++$	$a = a + 1$
$a -= b$	$a = a - b$
$a += b$	$a = a + b$

#### 3.2 Упорядочивание разрядов

Графическое изображение всех многоразрядных величин является таким, что старший значащий бит (*msb*) расположен слева, а младший значащий бит (*lsb*) — справа. Рисунок 1 определяет позицию двоичного разряда в байте.

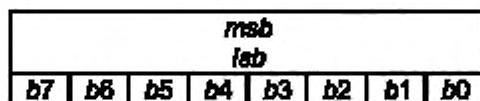


Рисунок 1 — Упорядочивание бита в байте

#### 3.3 Последовательность разрядов

Во всех местах, где используется последовательность битов, применяется нотация со старшим значащим разрядом на первом месте.

#### 3.4 Десятичная запись

Всем десятичным величинам предшествуют пробел или индикатор диапазона (..), когда включено в диапазон. Старшая значащая цифра находится слева, младшая значащая цифра — справа.

### 3.5 Порядок битов *DSD*

Первый выбранный бит *DSD* сохраняется в старшем значащем бите байта.

### 3.6 Полярность *DSD*

Бит *DSD*, равный единице, означает «плюс». Бит *DSD*, равный нулю, означает «минус».

### 3.7 Шестнадцатеричная нотация

Всем шестнадцатеричным значениям предшествует \$. Старший значащий полубайт располагается слева, младший значащий полубайт — справа.

### 3.8 Диапазон

*Constant\_1.. Constant\_2* обозначают диапазон от и включая *Constant\_1* до и включая *Constant\_2*, с инкрементами 1.

### 3.9 *Until*

*Until* используется в рисунках, чтобы указать, что для позиции байта структуры используются до, но не включая данное значение.

В позиции байта *B1* выражение «*until B2*» определяет байты *B2-B1*. В позиции байта *B1* выражение «*until esc*» определяет число байтов от *B1* до и включая последний байт текущего сектора. Позиция байта определяется относительно начала текущего или предыдущего сектора.

## 4 Основные типы

### 4.1 *BsMsb*

Последовательность битов, старший значащий бит сначала, должна интерпретироваться как строка битов.

### 4.2 *Char*

Закодированный однобайтовый символ. *NUL* (нулевой) символ (\$00) не разрешен для *Char*.

### 4.3 *SiMsb*

Последовательность битов должна интерпретироваться как целое число со знаком.

### 4.4 *UiMsb*

Последовательность битов должна интерпретироваться как целое число без знака.

### 4.5 *Uintn*

Закодированный двоичный файл *n* битов, численное значение без знака.

### 4.6 *Uint8*

Двоично закодированное 8-битовое численное значение без знака. Значение *Uint8* должно записываться в однобайтовом поле.

### 4.7 *Uint16*

Двоично закодированное 16-битовое численное значение без знака. Значение *Uint16*, представленное шестнадцатеричным представлением \$*ihxyz*, должно записываться в двухбайтовом поле как \$*ih* \$*xyz* (старший значащий байт сначала).

### 4.8 *Uint32*

Двоично закодированное 32-битовое численное значение без знака. Значение *Uint32*, представленное шестнадцатеричным представлением \$*stuvwxyz*, должно записываться в четырехбайтовом поле как \$*st* \$*uv* \$*xyz* \$*yz* (старший значащий байт сначала).

## 5 Полезные нагрузки для аудиообъекта

### 5.1 Конфигурация декодера (*DSTSpecificConfig*)

Таблица 1 — Синтаксис *Audio\_Frame ()*

Синтаксис	Количество битов	Мнемоника
<i>DSTSpecificConfig( channelConfiguration ) {</i>	1	<i>UiMsbf</i>
<i>DSDDST_Coded</i>	14	<i>UiMsbf</i>
<i>N_Channels</i>	1	<i>UiMsbf</i>
<i>reserved</i>		
}		

### 5.2 Полезная нагрузка потока битов

Таблица 2 — Синтаксис *Audio\_Frame ()*

Синтаксис	Количество битов	Мнемоника
<i>Audio_Frame() {</i>		
<i>if (DSDDST_Coded == 0)</i>		
<i>{</i>		
<i>DSD()</i>		<i>DSD</i>
<i>}</i>		
<i>else</i>		
<i>{</i>		
<i>DST()</i>		<i>DST</i>
<i>}</i>		
<i>}</i>		

Таблица 3 — Синтаксис *DSD*

Синтаксис	Количество битов	Мнемоника
<i>DSD() {</i>		
<i>For (Byte Nr=0; Byte Nr&lt;Frame Length; Byte Nr++)</i>		
<i>{</i>		
<i>For (Channel_Nr=1; Channel_Nr&lt;=N_Channels;</i>		
<i>Channel_Nr++)</i>		
<i>{</i>		
<i>DSD Byte[Channel_Nr][Byte_Nr]</i>	1	<i>Audio_Byte</i>
<i>}</i>		
<i>}</i>		

Таблица 4 — Синтаксис *DST*

Синтаксис	Количество битов	Мнемоника
<i>DST() {</i>	1	<i>BsMsbf</i>
<i>Processing_Mode</i>		
<i>if (Processing_Mode == 0)</i>		
<i>{</i>		
<i>DST_X_Bit</i>	1	<i>BsMsbf</i>
<i>Reserved</i>	6	<i>BsMsbf</i>
<i>DSD()</i>		<i>DSD</i>
<i>}</i>		
<i>else</i>		
<i>{</i>		
<i>Segmentation()</i>		<i>Segmentation</i>
<i>Mapping()</i>		<i>Mapping</i>
<i>Half_Probability()</i>		<i>Half_Probability</i>
<i>Filter_Coef_Sets()</i>		<i>Filter_Coef_Sets</i>
<i>Probability_Tables()</i>		<i>Probability_Tables</i>
<i>Arithmetic_Coded_Data()</i>		<i>Arithmetic_Coded_Data</i>
<i>}</i>		
<i>}</i>		

Таблица 5 — Синтаксис сегментации

Синтаксис	Количество битов	Мнемоника
<pre>Segmentation() {     Same_Segmentation     if(Same Segmentation == 0)     {         Filter_Segmentation()         Ptable Segmentation()     }     else     {         Filter And Ptable Segmentation()     } }</pre>	1	<i>BsMsbf</i> <i>Segment_Alloc</i> <i>Segment_Alloc</i> <i>Segment_Alloc</i>

Таблица 6 — Синтаксис сегментов

Синтаксис	Количество битов	Мнемоника
<pre>{     Channel Segmentation()[Channel Nr] } } else {     Channel Segmentation()[1] }</pre>	—	<i>Channel_Segmentation</i> <i>Channel_Segmentation</i>

Таблица 7 — Синтаксис Channel\_Segmentation

Синтаксис	Количество битов	Мнемоника
<pre>Scaled_Length[Nr_of_Segments] Segment_Length[Nr_of_Segments] = Resolution *     Scaled_Length[Nr_of_Segments] Start[Nr_of_Segments+1] = Start[Nr_of_Segments] +     Segment_Length[Nr_of_Segments] Nr_of_Segments++ End Of Channel Segm } Segment_Length[Nr_of_Segments] =     Frame Length - Start[Nr Of Segments]</pre>	1..13 1	<i>UiMsbf</i> <i>UiMsbf</i>

Таблица 8 — Синтаксис отображения

Синтаксис	Количество битов	Мнемоника
<pre>Mapping() {     Same_Mapping     if(Same Mapping == 0)     {         Filter_Mapping()         Ptable Mapping()     }     else     {         Filter And Ptable Mapping()     } }</pre>	1	<i>UiMsbf</i> <i>Maps</i> <i>Maps</i> <i>Maps</i>

Таблица 9 — Синтаксис отображений

Синтаксис	Количество битов	Мнемоника
<pre>Maps() {     Nr_Of_Elements = 0     Same_Maps_For_All_Channels     if(Same Maps For All Channels == 0)     {         for(Channel_Nr=1; Channel_Nr&lt;=N_Channels;             Channel_Nr++)         {             Channel Mapping(Channel_Nr)         }     }     else     {         Channel Mapping(1)     } }</pre>	1	<i>UiMsbf</i> <i>Channel_Mapping</i> <i>Channel_Mapping</i>

Таблица 10 — Синтаксис *Channel\_Mapping*

Синтаксис	Количество битов	Мнемоника
<pre>Channel_Mapping() {     for(Seg_Nr=1; Seg_Nr&lt;=Nr Of Segments[Channel_Nr]; Seg_Nr++)     {         Element[Channel_Nr][Seg_Nr]         if (Element[Channel_Nr][Seg_Nr] == Nr Of Elements)         {             Nr Of Elements++         }     } }</pre>	0..15	<i>UiMsbf</i>

Таблица 11 — Синтаксис *Half\_Probability*

Синтаксис	Количество битов	Мнемоника
<pre>Half_Probability() {     for (Channel_Nr=1; Channel_Nr&lt;=N_Channels; Channel_Nr++)     {         Half Prob(Channel_Nr)     } }</pre>	1	<i>BsMsbf</i>

Таблица 12 — Синтаксис *Arithmetic\_Coded\_Data*

Синтаксис	Количество битов	Мнемоника
<pre>Arithmetic Coded Data() {     j=0     do     {         A_Data[j]         j++     } until end of Audio Frame }</pre>	1	<i>BsMsbf</i>

Таблица 13 — Синтаксис Filter\_Coeff\_Sets

Синтаксис	Количество битов	Мнемоника
<i>Filter_Coeff_Sets()</i> { for ( <i>Filter Nr</i> =0; <i>Filter Nr</i> < <i>Nr Of Filters</i> ; <i>Filter Nr</i> ++) { <i>Coded_Pred_Order</i> <i>Pred_Order[Filter_Nr]</i> = <i>Coded_Pred_Order</i> +1 <i>Coded_Filter_Coeff_Set</i> if ( <i>Coded Filter Coef Set</i> ==0) { for ( <i>Coef Nr</i> =0; <i>Coef Nr</i> < <i>Pred Order[Filter Nr]</i> ; <i>Coef Nr</i> ++) { <i>Coef[Filter Nr][Coef Nr]</i> } } } } } } else { <i>CC_Method</i> for ( <i>Coef_Nr</i> =0; <i>Coef_Nr</i> < <i>CCPO</i> ; <i>Coef_Nr</i> ++) { <i>Coef[Filter_Nr][Coef_Nr]</i> } <i>CCM</i> for ( <i>Coef_Nr</i> = <i>CCPO</i> ; <i>Coef_Nr</i> < <i>Pred_Order[Filter_Nr]</i> ; <i>Coef_Nr</i> ++) { <i>Run_Length</i> =0 do { <i>RL_Bit</i> if ( <i>RL_Bit</i> ==0) { <i>Run_Length</i> ++ } } } while ( <i>RL_Bit</i> ==0) <i>LSBs</i> <i>Delta</i> =( <i>Run_Length</i> << <i>CC</i> <i>M</i> )+ <i>LSBs</i> if ( <i>Delta</i> !=0) { <i>Sign</i> if ( <i>Sign</i> ==1) { <i>Delta</i> = - <i>Delta</i> } } <i>Coef[Filter_Nr][Coef_Nr]</i> = <i>Delta</i> <i>Delta8</i> = 0 for ( <i>Tap_Nr</i> =0; <i>Tap_Nr</i> < <i>CCPO</i> ; <i>Tap_Nr</i> ++) { <i>Delta8</i> += 8* <i>CCPC[Tap_Nr]*Coef[Filter_Nr][Coef_Nr-Tap_Nr-1]</i> } if ( <i>Delta8</i> >=0) { <i>Coef[Filter_Nr][Coef_Nr]</i> -= <i>trunc((Delta8+4)/8)</i> } } else { <i>Coef[Filter_Nr][Coef_Nr]</i> += <i>trunc((-Delta8+3)/8)</i> } } } }	7 1 9 2 9 3 1 0...6 1	<i>UiMsbf</i> <i>BsMsbf</i> <i>SiMsbf</i> <i>BsMsbf</i> <i>SiMsbf</i> <i>UiMsbf</i> <i>BsMsbf</i> <i>UiMsbf</i> <i>BsMsbf</i>

Таблица 14 — Синтаксис Probability\_Tables

Синтаксис	Количество битов	Мнемоника
<pre>Probability_Tables() {     for (Ptable Nr=0; Ptable Nr&lt;Nr Of Ptables; Ptable Nr++)     {         Coded_Ptable_Len         Ptable_Len[Ptable_Nr] = Coded_Ptable_Len+1         if (Ptable_Len[Ptable_Nr] == 1)         {             P_one[Ptable_Nr][0] = 128         }     else     {         Coded_Ptable         If (Coded_Ptable==0)         {             for (Entry_Nr=0; Entry_Nr&lt;Ptable_Len[Ptable_Nr];                  Entry_Nr++)             {                 Coded_P_one                 P_one[Ptable_Nr][Entry_Nr] = Coded_P_one+1             }         }         else     {         PC_Method         for (Entry_Nr=0; Entry_Nr&lt;PCPO; Entry_Nr++)         {             Coded_P_one             P_one[Ptable_Nr][Entry_Nr] = Coded_P_one+1         }         PCM     for (Entry_Nr=PCPO; Entry_Nr&lt;Ptable_Len[Ptable_Nr]; Entry_Nr++)     {         Run_Length=0         do         {             RL_Bit             if (RL_Bit==0)             {                 Run_Length++             }         } while (RL_Bit==0)         LSBs         Delta = (Run_Length&lt;&lt;PCM)+LSBs         if (Delta != 0)         {             Sign             if (Sign==1)             {                 Delta = -Delta             }         }         P_one[Ptable_Nr][Entry_Nr] = Delta         for (Tap_Nr=0; Tap_Nr&lt;PCPO; Tap_Nr++)         {             P_one[Ptable_Nr][Entry_Nr] -=             PCPC[Tap_Nr]*P_one[Ptable_Nr][Entry_Nr-Tap_Nr-1]         }     } } }</pre>	6	UiMsbF
	1	BsMsbF
	7	UiMsbF
	2	BsMsbF
	7	UiMsbF
	3	UiMsbF
	1	BsMsbF
	0...4	UiMsbF
	1	BsMsbF

## 6 Семантика

### 6.1 Аудиопотоки

Аудиопоток содержит аудиосигнал *DSD* (простой *DSD* или *DST*-кодированный *DSD*). Аудиопоток является конкатенацией всех аудиофреймов в потоке байтов.

#### 6.1.1 Данные потока битов дискретизированного *DSD*

Для сигнала 2-канального аудио разрядная последовательность дискретизированного *DSD* определяется следующим образом:  $L_0, L_1, L_2, L_3, L_4, L_5, L_6, L_7, R_0, R_1, R_2, R_3, R_4, R_5, R_6, R_7, L_8, L_9, L_{10}, \dots$ , где  $L_0$  является первым битом выборки левого канала аудиофрейма.

Для каждого звукового канала восемь последовательных битов выборки группируются в один аудиобайт. Старший значащий бит аудиобайта является первым битом выборки этого байта.

#### 6.1.2 Структура аудиопотока

У кодированных аудиофреймов *DST* существует переменная длина. Эталонная модель декодера *DST* определяется в разделе 7.

#### 6.1.3 *Audio\_Frame*

*Audio\_Frame* (аудиофрейм) содержит кодированную *DST* или просто *DSD*-аудиоинформацию для одного фрейма. Максимальный размер *Audio\_Frame* равен размеру кодированного просто *DSD* *Audio\_Frame* плюс один байт. Синтаксис *Audio\_Frame* определяется в таблице 1.

*N\_Channels* является числом используемых звуковых каналов.

### 6.2 *DSD* содержит аудиоданные для одной *Audio\_Frame* простого *DSD*. Синтаксис *DSD* определяется в полезной нагрузке потока битов

*Channel\_Nr* является номером звукового канала.

*Frame\_Length* является длиной аудиофрейма в байтах на звуковой канал. *Frame\_Length* может быть вычислено исходя из частоты дискретизации по формуле

$$\text{Frame\_Length} = \frac{\text{Sampling Frequency}}{75 * 8} \text{ bytes per AudioChannel.}$$

Частота дискретизации может быть: 64\*44100 Гц, 128\*44100 Гц или 256\*44100 Гц. Соотношение между *Frame\_Length* и частотой дискретизации дается в таблице 15.

Таблица 15 — Соотношение *Frame\_Length* и частоты дискретизации

Частота дискретизации, Гц	<i>Frame_Length</i> , байт
64*44100	4704
128*44100	9408
256*44100	18816

*DSDDST\_Coded* сигнализирует, кодирован ли поток битов *DSD* или *DST*. Если *DSDDST\_Coded* = % 0 — это *DSD*-кодированный поток, а если *DSDDST\_Coded* = % 1 — *DST*-кодированный.

#### 6.2.1 *DSD\_Byte*

*DSD Byte* [*Channel\_Nr*] [*Byte\_Nr*] содержит сигнал *DSD*, как определено в 6.1.1.

##### 6.2.1.1 *DST*

*DST* содержит аудиоданные для кодированного *Audio\_Frame* одного *DST*. Синтаксис *DST* определяется в таблице 4.

###### 6.2.1.1.1 *Processing\_Mode*

Если бит *Processing\_Mode* устанавливается в единицу, *Audio\_Frame* содержит *DST\_X\_Bit* и сигнал *DSD* в форме кодирования без потерь. Если бит *Processing\_Mode* обнуляется, *Audio\_Frame* содержит *DST\_X\_Bit* и сигнал *DSD* без кодирования без потерь.

###### 6.2.1.1.2 *DST\_X\_Bit*

Если *Frame\_Format DST\_Coded*, то каждый *Audio\_Frame* содержит один бит *DST\_X\_Bit*. При кодировании *DST\_X\_Bit* должен быть обнулен. Декодер должен проигнорировать контент *DST\_X\_Bit*.

## 6.2.1.1.3 Зарезервировано

Это значение должно быть установлено в ноль.

## 6.2.1.1.4 DSD

См. 6.1.1.

## 6.2.1.1.5 Сегментация

Для каждого звукового канала аудиофрейм делится на один или более сегментов для фильтров и один или более сегментов для *Ptables*. Каждый сегмент может использовать различные фильтры прогноза/*Ptable*. Синтаксис сегментации определяется в таблице 5.

*Filter\_Segmentation*

Для каждого звукового канала аудиофрейм делится на один или более сегментов для фильтров прогноза. Каждый сегмент может использовать различные фильтры прогноза. Переменные *Nr\_Of\_Segments[]* и *Segment\_Length[]* из *Segment\_Alloc*, используемые для *Filter\_Segmentation*, упоминаются как *Filters.Nr\_Of\_Segments[Channel\_Nr]* и *Filters.Segment\_Length[Channel\_Nr][1..Filters.Nr\_Of\_Segments[Channel\_Nr]]*, где *Channel\_Nr* = 1.. *N\_Channels*.

*Ptable\_Segmentation*

Для каждого звукового канала аудиофрейм делится на один или более сегментов для *Ptables*. Каждый сегмент может использовать различные *Ptable*. Переменные *Nr\_Of\_Segments[]* и *Segment\_Length[]* из *Segment\_Alloc*, используемые для *Ptable\_Segmentation*, упоминаются как *Ptables.Nr\_Of\_Segments[Channel\_Nr]* и *Ptables.Segment\_Length[Channel\_Nr][1..Ptables.Nr\_Of\_Segments[Channel\_Nr]]*, где *Channel\_Nr* = 1.. *N\_Channels*.

*Filter\_And\_Ptable\_Segmentation*

Для каждого звукового канала аудиофрейм делится на один или более сегментов. Каждый сегмент может использовать различные комбинации *Prediction Filter* и *Ptable*. Для каждого звукового канала должны быть истиной следующие уравнения:

$$\begin{aligned} \text{Filters.Nr\_Of\_Segments[Channel\_Nr]} &= \text{Ptables.Nr\_Of\_Segments[Channel\_Nr]} \\ &= \text{Nr\_Of\_Segments[Channel\_Nr]} \end{aligned}$$

$$\begin{aligned} \text{Filters.Segment\_Length[Channel\_Nr]} &= \text{Ptables.Segment\_Length[Channel\_Nr]} \\ &= \text{Segment\_Length[Channel\_Nr]}. \end{aligned}$$

где *Channel\_Nr* = 1.. *N\_Channels*.

6.2.1.1.5.1 *Same\_Segmentation*

Если *Same\_Segmentation* равно единице, *Ptables* и фильтры прогноза используют одну и ту же сегментацию. Если *Same\_Segmentation* является нулем, разделение для аудиофрейма для фильтров прогноза независимо от разделения для *Ptables*.

6.2.1.1.5.2 *Segment\_Alloc*

*Segment\_Alloc* определяет сегментацию для фильтров прогноза и/или *Ptables*. Синтаксис *Segment\_Alloc* определяется в таблице 6.

Для каждого звукового канала переменные *Nr\_Of\_Segments* и *Segment\_Length[1..Nr\_Of\_Segments]* из *Channel\_Segmentation* упоминаются как *Nr\_Of\_Segments[Channel\_Nr]* или *Segment\_Length[Channel\_Nr][1..Nr\_Of\_Segments[Channel\_Nr]]*.

*Resolution\_Read* указывает, было ли считано разрешение из *Channel\_Segmentation*. *Resolution\_Read* устанавливается в истину в *Channel\_Segmentation* первого звукового канала больше чем с одним сегментом. Если фильтры прогноза и *Ptables* используют независимую сегментацию, они также используют независимое *Resolution\_Read*. *Channel\_Nr* является локальной индексной переменной. *N\_Channels* является числом используемых звуковых каналов.

6.2.1.1.5.2.1 *Same\_Segm\_For\_All\_Channels*

Если *Same\_Segm\_For\_All\_Channels* равно единице, сохраняется только сегментация для первого звукового канала и *Channel\_Segmentation()*[*Channel\_Nr*] = *Channel\_Segmentation()* для всех звуковых каналов. Если *Same\_Segm\_For\_All\_Channels* является нулем, аудиофрейм делится на сегменты, независимые для каждого звукового канала.

6.2.1.1.5.2.2 *Channel\_Segmentation*

*Channel\_Segmentation* определяет сегментацию фильтров прогноза и/или *Ptables*. Синтаксис *Channel\_Segmentation* определяется в таблице 7.

В синтаксисе *Channel\_Segmentation* используются переменные *Nr\_Of\_Segments*, *Start[1..Nr\_Of\_Segments]*, *Segment\_Length[1..Nr\_Of\_Segments]*.

*Nr\_Of\_Segments* является числом сегментов для текущего звукового канала. Максимальное количество сегментов является *MAXNRSEGS*. *MAXNRSEGS* должно быть 4 для *Filter\_Segmentation*, 8 для *Ptable\_Segmentation* и 4 для *Filter\_And\_Ptable\_Segmentation*.

*Resolution\_Read* указывает, была ли считана переменная *Resolution* в этом или предыдущем *Channel\_Segmentation*. *Resolution\_Read* устанавливается в истину в *Channel\_Segmentation* первого звукового канала с больше чем одним сегментом. Если *Prediction Filters* и *Ptables* используют независимую сегментацию, они также используют и независимое *Resolution Read*.

*Segment\_Length [Seg\_Nr]* содержит длину сегмента в байтах,  
где  $1 \leq Seg\_Nr \leq Nr\_Of\_Segments$ .

*Start[Seg\_Nr]* является стартовой позицией в байтах *Segment[Seg\_Nr]*.

#### 6.2.1.15.2.2.1 *End\_Of\_Channel\_Segm*

Если *End\_Of\_Channel\_Segm* является нулем, будет следовать одно или более значений для *Scaled\_Length*. Если *End\_Of\_Channel\_Segm* равно единице, структура *Channel\_Segmentation* заканчивается.

#### 6.2.1.15.2.2.2 *Resolution*

Каждое значение *Scaled\_Length* умножается на *Resolution* (разрешение), чтобы получить длину сегмента в байтах. Разрешение сохраняется только однажды, в начале первого звукового канала с больше чем одним сегментом. Если у всех звуковых каналов имеется только один сегмент, *Resolution* не кодируется.

Разрешение должно быть в диапазоне от 1 до *Frame\_Length - MINSEGLEN*. *MINSEGLEN* должно составлять 128 байтов для *Filter\_Segmentation*, 4 байта для *Ptable Segmentation* и 128 байтов для *Filter\_And\_Ptable\_Segmentation*.

#### 6.2.1.15.2.2.3 *Scaled\_Length*

Для каждого сегмента, кроме последнего, значение *Scaled\_Length* кодируется. Длина сегмента в байтах вычисляется по следующей формуле:

$$Segment\_Length [Seg_Nr] = Resolution * Scaled\_Length [Seg_Nr],$$

где  $1 \leq Seg\_Nr \leq Nr\_Of\_Segments$ .

Минимальная длина сегмента каждого *Segment* является *MINSEGLEN*.

Для *Ptable Segmentation* длина первого сегмента каждого звукового канала должна быть (*Pred\_Order* [*Filter[Channel\_Nr][1]*] + 7)/8 байтов.

Число битов, необходимых для кодирования *Scaled\_Length[Seg\_Nr]*, зависит от значения диапазона. Диапазон (*Range*) должен быть вычислен по формуле

$$Range = Trunc\left(\frac{Frame\_Length - Start[Seg_Nr] - MINSEGLEN}{Resolution}\right).$$

Если  $2^{n-1} \leq Range < 2^n$ , чтобы закодировать *Scaled\_Length[Seg\_Nr]*, должны использоваться *n* битов, см. таблицу 16. Минимальное значение *Range* равно 1. Длина последнего сегмента не кодируется. Длина последнего сегмента может быть вычислена исходя из длины фрейма и стартовой позиции последнего сегмента по формуле

$$Segment\_Length [Nr\_Of\_Segments] = Frame\_Length - Start[Nr\_Of\_Segments].$$

Таблица 16 — Диапазон и биты для кодирования *Scaled\_Length*

Диапазон	Используемые биты
1	1
2..3	2
4..7	3
8..15	4
16..31	5
32..63	6

Окончание таблицы 16

Диапазон	Используемые биты
64..127	7
128..255	8
256..511	9
512..1023	10
1024..2047	11
2048..4095	12
4096..8191	13

#### 6.2.1.1.6 Mapping

*Mapping* (отображение) определяет фильтры прогноза и *Ptables*, используемого с сегментами, определенными в 6.2.1.1.5. Синтаксис отображения определяется в таблице 8.

##### Filter\_Mapping

Для каждого звукового канала и каждого сегмента кодируется номер фильтра прогноза. Для *Filter\_Mapping* переменная *Element[]* из *Channel\_Mapping* содержит номера фильтра прогноза. Для *Filter\_Mapping* переменная *Element[]* упоминается как *Filter[Channel\_Nr][1.. Filters.Nr\_Of\_Segments[Channel\_Nr]]*. Для *Filter\_Mapping* переменная *Nr\_Of\_Elements* упоминается как *Nr\_Of\_Filters*.

##### Ptable\_Mapping

Для каждого звукового канала и каждого сегмента кодируется номер *Ptable*. Для *Ptable\_Mapping* переменная *Element[]* из *Channel\_Mapping* содержит номера *Ptable*. Для *Ptable\_Mapping* переменная *Element[]* упоминается как *Ptable[Channel\_Nr][1.. Ptables.Nr\_Of\_Segments[Channel\_Nr]]*. Для *Ptable\_Mapping* переменная *Nr\_Of\_Elements* упоминается как *Nr\_Of\_Ptables*.

##### Filter\_And\_Ptable\_Mapping

Для каждого звукового канала и каждого сегмента кодируется общий номер *Prediction\_Filter* и *Ptable*. Для *Filter\_and\_Ptable\_Mapping* переменная *Element[]* из *Channel\_Mapping* содержит общие номера *Prediction\_Filter* и *Ptable*. Для *Filter\_and\_Ptable\_Mapping* переменная *Element[]* упоминается как *Filter[Channel\_Nr][1.. Filters.Nr\_Of\_Segments[Channel\_Nr]]* или *Ptable[Channel\_Nr][1.. Ptables.Nr\_Of\_Segments[Channel\_Nr]]*. Для *Filter\_and\_Ptable\_Mapping* переменная *Nr\_Of\_Elements* упоминается как *Nr\_Of\_Filters*, а также как *Nr\_Of\_Ptables*.

##### 6.2.1.1.6.1 Same\_Mapping

Если *Same\_Mapping* равно единице, *Ptables* и фильтры прогноза используют одно и то же отображение. Если *Same\_Mapping* является нулем, для аудиофрейма отображение для фильтров прогноза независимо от отображения для *Ptables*.

##### 6.2.1.1.6.2 Maps

*Maps* (отображения) определяют отображение в фильтры прогноза и в сегменты, определенные в 6.2.1.1.5. Синтаксис отображений определяется в таблице 9.

В синтаксисе отображений используются переменные *Element[Channel\_Nr][1.. Nr\_Of\_Segments[Channel\_Nr]]* для каждого звукового канала и *Nr\_Of\_Elements*.

*Nr\_Of\_Elements* является общим номером фильтра прогноза и/или *Ptables*, используемого в отображениях. *Nr\_Of\_Elements* должен быть в диапазоне 1..2 \* *N\_Channels*.

*Channel\_Nr* является локальной индексной переменной, используемой в таблице 9 и таблице 10.

*N\_Channels* является числом используемых звуковых каналов.

##### 6.2.1.1.6.2.1 Same\_Maps\_For\_All\_Channels

Если *Same\_Maps\_For\_All\_Channels* равно единице, сохраняется только *Element[1]*, и каждый звуковой канал использует тот же самый *Element[Channel\_Nr][1] = Element[1]* массива. Если *Same\_Maps\_For\_All\_Channels* равно нулю, *Element[Channel\_Nr][1]* сохраняется независимо для каждого звукового канала. Если *Nr\_Of\_Segments[Channel\_Nr]* не имеет для всех звуковых каналов одинакового значения, то *Same\_Maps\_For\_All\_Channels* должно быть нулем.

##### 6.2.1.1.6.2.2 Channel\_Mapping

*Channel\_Mapping* содержит номера фильтра прогноза и/или *Ptable* на звуковой канал, используемые для каждого сегмента. Синтаксис *Channel\_Mapping* определяется в таблице 10.

*Nr\_Of\_Elements* является общей численностью фильтров прогноза и/или *Ptables* для всех каналов. *Nr\_Of\_Elements* инициализируется в отображениях, таблица 9.

*Channel\_Nr* является индексной переменной из отображений, таблица 9.

*Seg\_Nr* является локальной индексной переменной.

*Nr\_Of\_Segments [Channel\_Nr]* является общим количеством сегментов, используемых в текущем аудиофрейме для звукового канала *Channel\_Nr*.

#### 6.2.1.6.2.2.1 Element

*Element* является номером фильтра прогноза и/или *Ptable*, используемым в сегменте. Количество используемых для кодирования *Element* битов зависит от значения *Nr\_Of\_Elements*. В каждой итерации *Element* должен быть  $\leq Nr\_Of\_Elements$ . Поэтому *Element[1][1]* всегда является нулем и не сохраняется ( $\#bits = 0$ ). Для всех других звуковых каналов и сегментов  $Nr\_Of\_Elements > 0$  и число битов, необходимых для сохранения *Element*, равно  $n$ :  $2^{n-1} \leq Nr\_Of\_Elements < 2^n$ , таблица 17.

Таблица 17 — Биты, используемые для кодирования *Element*

<i>Nr_Of_Elements</i>	Номер используемого бита
0	0
1	1
2..3	2
4..7	3
8..15	4
16..31	5
32..63	6
64..127	7
128..255	8
256..511	9
512..1023	10
1024..2047	11
2048..4095	12
4096..8191	13
8192..16383	14
16384..32768	15

#### 6.2.1.1.7 Half\_Probability

Синтаксис *Half\_Probability* определяется в таблице 11.

*Channel\_Nr* является локальной индексной переменной.

*N\_Channels* является числом используемых звуковых каналов.

#### 6.2.1.1.7.1 Half\_Prob

*Half\_Prob* используется при кодировании для каждого звукового канала. Метод обычно используется для того, чтобы применить значение вероятности к арифметическому декодеру. Определение *Half\_Prob* дается в таблице 18.

Таблица 18 — Определение *Half\_Prob*

<i>Half_Prob[]</i>	Вероятность для использования во время первых битов <i>Pred_Order[]</i> аудиоканала
0	Использовать записи из <i>Ptable</i> .
1	Использовать $p = \frac{1}{2}$ (соответствует $P_{one} = 128$ )

Для оптимальной эффективности кодирования требуется, чтобы у следующего остаточного бита в  $E$  было значение, которое имеет наибольшую вероятность. Если применяется вероятность, которая отражает высокий шанс следующего бита  $E$  быть 1, в то время как следующий бит  $E$  является 0, то требуется более 1 бита в арифметическом коде, чтобы отправить этот бит.

Фильтр прогноза первоначально заполнен образцом инициализации. Во время первых выборок  $Pred\_Order$  в звуковом канале  $Channel\_Nr$  фильтр прогноза постепенно заполняется реальными данными  $DSD$ . Как следствие, распределение вероятности может отличаться от остальной части фрейма, и комбинация примененных  $E$  и  $P$  для этих битов приведет к большему количеству битов, чем требуется. Применяя вероятность 1/2 во время кодирования, каждый бит будет также стоить только одного бита в арифметическом коде.

Чтобы быть в состоянии отвергнуть плохую комбинацию  $E$  и  $P$  в начале фрейма,  $Half\_Prob$  доступен для каждого канала отдельно.

#### 6.2.1.1.8 Filter\_Coef\_Sets

Для каждого сегмента в каждом звуковом канале декодер  $DST$  использует фильтр прогноза. В случае если два или более фильтра прогноза одинаковы, соответствующие коэффициенты фильтра могут кодироваться только однажды. В синтаксисе  $Filter\_Coef\_Sets$  используются переменные  $Pred\_Order[Filter\_Nr]$  и  $Coef[Filter\_Nr][0..Pred\_Order[Filter\_Nr]-1]$ , где  $Filter\_Nr = 0..Nr\_Of\_Filters-1$ .

Все коэффициенты фильтра прогноза кодируются. Коэффициенты фильтра прогноза могут быть закодированы, используя простое линейное предсказание и кодирование Райса. Кодирование Райса является методом кодирования переменной длины (особый случай кодирования методом Хаффмана), который используется, чтобы сократить число битов, необходимых для определенного «сообщения», без потери информации. Синтаксис  $Filter\_Coef\_Sets$  определяется в таблице 13.

Младший значащий бит  $Coef[0][0]$  называют  $DST\_Y\_Bit$ .

$Nr\_Of\_Filters$  является значением, вычисленным в отображении.

$Pred\_Order[]$  является массивом, который содержит порядок прогноза для каждого фильтра прогноза, где  $Pred\_Order[Filter\_Nr] = Coded\_Pred\_Order + 1$ , для  $Filter\_Nr = 0..Nr\_Of\_Filters-1$ . Допустимый диапазон порядка прогноза равен:  $1 \leq Pred\_Order[Filter\_Nr] \leq 128$ .

$Coef[][]$  является двухмерным массивом, который содержит все коэффициенты всех фильтров прогноза. Каждая запись  $Coef[]$  должна быть в диапазоне от -256 до +255. Первый (левый) индекс является  $Filter\_Nr$  и простирается от 0 до  $Nr\_Of\_Filters-1$ . Второй (правый) индекс является номером коэффициента и простирается от 0 до  $Pred\_Order[Filter\_Nr]-1$ .

$CCPO$  является порядком прогноза кодирования коэффициента ( $CCPO$ ). Отношение между  $CC\_Method$  и  $CCPO$  определяется в таблице 19.

Таблица 19 — Отношение между  $CC\_Method$  и  $CCPO$

$CC\_Method$	$CCPO$
'00'	1
'01'	2
'10'	3
'11'	Не используется

Применяется ограничение  $CCPO < Pred\_Order[Filter\_Nr]$ .

$Run\_Length$  является вспомогательной переменной, предназначеннной для подсчета числа нулей в коде длины серии, который является частью кода Райса.

$Delta$  является вспомогательной переменной, чтобы вычислять кодированный по Райсу номер.

$Delta8$  является вспомогательной переменной, чтобы вычислять кодированный по Райсу номер.

$CCPC[]$  является массивом, который содержит коэффициенты прогноза кодирования коэффициентов ( $CCPC$ ), которые используются для линейного прогноза коэффициентов фильтра. Отношение между  $CC\_Method$  и  $CCPC[]$  определяется в таблице 20.

Таблица 20 — Отношение между  $CC\_Method$  и  $CCPC[]$

$CC\_Method$	$CCPC[0]$	$CCPC[1]$	$CCPC[2]$
'00'	-1	—	—
'01'	-2	1	—
'10'	-9/8	-5/8	6/8
'11'	Не используется	Не используется	Не используется

Линейное предсказание требует округления, как определено в синтаксисе таблицы 13.

#### 6.2.1.8.1 *DST\_Y\_Bit*

*DST Y Bit* является младшим значащим битом *Coeff[0][0]*. При кодировании *DST Y Bit* должен быть установлен в единицу. Декодер должен игнорировать контент *DST\_Y\_Bit*.

#### 6.2.1.8.2 *Coded\_Pred\_Order*

*Coded\_Pred\_Order* является 7-битовым целым числом без знака, которое содержит кодированный порядок прогноза текущего фильтра прогноза.

#### 6.2.1.8.3 *Coded\_Filter\_Coef\_Set*

*Coded\_Filter\_Coef\_Set* указывает, предсказываются ли коэффициенты фильтра прогноза и кодированы ли они по Райсу. *Coded\_Filter\_Coef\_Set* обнуляется, если коэффициенты фильтра прогноза сохраняются.

*Coded\_Filter\_Coef\_Set* устанавливается в единицу, если коэффициенты фильтра прогноза предсказываются и кодированы по Райсу.

Максимальное количество битов, разрешенных для единственного фильтра прогноза внутри *Filter\_Coef\_Sets*, равно  $7+1+Pred\_Order \cdot 9$ , где 7 — количество битов для *Coded\_Pred\_Order*, 1 — количество битов *Coded\_Filter\_Coef\_Set* и остальные коэффициенты *Pred\_Order* имеют по 9 битов каждый.

#### 6.2.1.8.4 *CC\_Method*

*CC\_Method* является 2-битовым кодом, который идентифицирует метод кодирования коэффициентов текущего фильтра прогноза.

#### 6.2.1.8.5 *CCM*

*CCM* является 3-битовым целым числом без знака, которое содержит параметр *M* кодирования коэффициентов, который используется для декодирования по Райсу коэффициентов текущего фильтра прогноза. Минимальное разрешенное значение для *CCM* равно нулю. Максимальное разрешенное значение для *CCM* равно 6.

#### 6.2.1.8.6 *RL\_Bit*

*RL\_Bit* используется, чтобы получить единственные биты кода длины серии, который состоит из нулей с завершающей единицей. Самым коротким кодом длины серии является '1'.

#### 6.2.1.8.7 *LSBs*

Младшие значащие биты *CCM* абсолютного значения предсказанного коэффициента считаются непосредственно из потока и сохраняются в *LSBs*.

#### 6.2.1.8.8 *Sign*

*Sign* (знак) является битом, который указывает, положителен ли предсказанный коэффициент (*Sign* = '0') или отрицателен (*Sign* = '1').

#### 6.2.1.8.9 *Probability\_Tables*

Для каждого сегмента в каждом звуковом канале декодер использует таблицу вероятности (*Ptable*). В случае если две или более таблицы вероятности равны, соответствующие записи таблицы вероятности могут быть доступными из потока только однажды. В синтаксисе *Probability\_Tables* используются переменные *Ptable\_Len* [*Ptable\_Nr*] и *P\_One* [*Ptable\_Nr*] [0...*Ptable\_Len* [*Ptable\_Nr*]-1], где *Ptable\_Nr* = 0..*Nr\_Of\_Ptables*-1.

В *Probability\_Tables* кодируются все записи таблицы вероятности. На таблицу вероятности кодируются длина таблицы (= число записей) и записи. Записи *Ptable* могут быть кодированы с использованием простого линейного прогноза и кодирования Райса. Синтаксис *Probability\_Tables* определяется в таблице 14.

*Nr\_Of\_Ptables* является значением, вычисленным в отображении. *Ptable\_Len* [] является массивом, который содержит длину таблицы вероятности для каждой *Ptable*, где *Ptable Len* [*Ptable Nr*] = *Coded\_Ptable\_Len* + 1, для *Ptable\_Nr* ∈ {0..*Nr\_Of\_Ptables*-1}.

Допустимый диапазон длины *Ptable*:  $1 \leq Ptable\_Len[Ptable\_Nr] \leq 64$ .

*P\_one* [] является двухмерным массивом, который содержит все записи всех таблиц вероятности. Первый (левый) индекс является *Ptable\_Nr* и находится в диапазоне от 0 до *Nr Of Ptables*-1. Второй (правый) индекс является числом записей и лежит в диапазоне от 0 до *Ptable\_Len* [*Ptable\_Nr*]-1. Каждая запись *P\_one* [] находится в диапазоне 1—128, соответствующа вероятности от 1/256 до 128/256 следующего ошибочного бита.

*PCPO* является порядком прогноза кодирования *Ptable* (*PCPO*). Отношение между *PC\_Method* и *PCPO* определяется в таблице 21.

Таблица 21 — Отношение между PC\_Method и PCPO

PC_Method	PCPO
'00'	1
'01'	2
'10'	3
'11'	Не используется

Применяется ограничение  $PCPO < Ptable\_Len [Ptable\_Nr]$ .

$Run\_Length$  является переменной справки, чтобы считать число нулей в коде длины серии, который является частью кода Райса.

$\Delta$  является переменной справки, чтобы вычислять декодированное число Райса.

$PCPC [ ]$  является массивом, который содержит коэффициенты прогноза кодирования  $Ptable$  ( $PCPC$ ), которые используются для линейного предсказания записей  $Ptable$ . Отношение между  $PC\_Method$  и  $PCPC [ ]$  определяется в таблице 22.

Таблица 22 — Отношение между PC\_Method и PCPC []

PC_Method	PCPC[0]	PCPC[1]	PCPC[2]
'00'	-1	—	—
'01'	-2	1	—
'10'	-3	3	-1
'11'	Не используется	Не используется	Не используется

#### 6.2.1.1.9.1 Coded\_Ptable\_Len

$Coded\_Ptable\_Len$  является 6-битовым целым числом без знака, которое содержит кодированную длину таблицы вероятности.

#### 6.2.1.1.9.2 Coded\_Ptable

$Coded\_Ptable$  указывает, прогнозируются ли записи  $Ptable$  и кодируются ли по Райсу.  $Coded\_Ptable$  обнуляется, если записи  $Ptable$  сохраняются.  $Coded\_Ptable$  устанавливается в единицу, если записи  $Ptable$  прогнозируются и кодируются по Райсу.

Максимальное количество битов, разрешенных для единственной  $Ptable$  внутри  $Probability\_Tables$  равно  $6+1+Ptable\_Len [ ]^7$ , где 6 — количество битов для  $Coded\_Ptable\_Len$ , 1 — количество битов  $Coded\_Ptable$  и остальные  $Ptable\_Len [ ]$  кодированные записи  $Ptable$  по 7 битов каждая.

#### 6.2.1.1.9.3 Coded\_P\_one

$Coded\_P\_one$  является 7-битовым целым числом без знака, которое содержит кодированное значение следующей записи текущей  $Ptable$ .

#### 6.2.1.1.9.4 PC\_Method

$PC\_Method$  является 2-битовым полем, которое идентифицируют метод кодирования  $Ptable$  для текущей  $Ptable$ .

#### 6.2.1.1.9.5 PCM

$PCM$  является 3-битовым целым числом без знака, которое содержит параметр  $Ptable Coding M$ , использующийся для декодирования по Райсу записей  $Ptable$  текущей  $Ptable$ . Минимальное возможное значение для  $PCM$  является нулем. Максимальное допустимое значение для  $PCM$  равно 4.

#### 6.2.1.1.9.6 RL\_Bit

$RL\_Bit$  содержит единственный код длины серии, который состоит из нулей с завершающей единицей. Самый короткий код длины серии равен '1'.

#### 6.2.1.1.9.7 LSB

Младшие значащие биты  $PCM$  абсолютного значения прогнозируемой записи сохраняются в  $LSB$ .

#### 6.2.1.1.9.8 Sign

$Sign$  (знак) представляет собой бит, который указывает, является ли предсказанная запись положительной ( $Sign = '0'$ ) или отрицательной ( $Sign = '1'$ ).

#### 6.2.1.1.10 Arithmetic\_Coded\_Data

Синтаксис  $Arithmetic\_Coded\_Data$  определяется в таблице 12. Длина  $Arithmetic\_Coded\_Data$  не кодируется.

### 6.2.1.1.10.1 A\_Data

*A\_Data* [] содержит арифметический код и биты заполнения.

Биты заполнения добавляются в конце арифметического кода, чтобы выровнять *Audio\_Frame* до границы байта. Число битов заполнения составляет 0...7. Значение битов заполнения должно быть нулем.

*A\_Data* [] используется функцией «*input next bit D*». Минимальная длина *A\_Data* составляет нуль битов. Если длина *A\_Data* не равна нулю, у *A\_Data* [0] должно быть значение нуль. Максимальная длина арифметического кода является числом битов, обработанных «*input next bit D*». Разрешается, чтобы конечные нули арифметического кода не были закодированы в *A\_Data* [].

## 7 Эталонная модель декодера DST

### 7.1 Процессы декодирования DST

Параметры, обработанные и извлеченные из потока, используются, чтобы декодировать кодированный фрейм DST. Этот подпункт объясняет процессы декодирования, необходимые для кодированных фреймов DST.

#### 7.1.1 Введение

Существуют три функции: арифметический декодер, мультиплексор/демультиплексор и ряд моделей источника. Арифметический декодер получает последовательность битов ( $D = A_{Data}$ ) и последовательность вероятностей ( $P$ ) и генерирует последовательность битов ( $E$ ). Последовательности  $E$  и  $P$  присваиваются моделям источника в циклическом порядке, которым управляет мультиплексор/демультиплексор. Каждая модель источника получает необходимые параметры, как коэффициенты фильтра прогноза и записи таблицы вероятности из потока, как определено в синтаксисе и семантике.

Модель источника  $S$  соответствует каналу  $S$ . Вывод  $X$  из модели источника  $S$  является сигналом  $DSD$  для канала  $S$ .

#### 7.1.2 Арифметический декодер

Арифметическое кодирование является методом кодирования переменной длины для сжатия данных близко к их энтропии. Кодированные данные представляются как число. Число использует только цифры, сколько требуется, чтобы однозначно определять исходные данные.

Таблица 23 определяет переменные, используемые на рисунке 2, рисунке 3, рисунке 4 и рисунке 5.

Таблица 23 — Переменные, используемые в рисунке 2, рисунке 3, рисунке 4 и рисунке 5

Имя	Характеристики	Описание
A	12-битовый регистр	Это целое число без знака представляет текущую величину интервала арифметического декодера
C	12-битовый регистр	Это целое число без знака содержит часть битов арифметического кода
K	4-битовая переменная	Это целое число без знака является 4-битовым приближением $A$
P	8-битовая переменная	Это целое число без знака является величиной вероятности, применяемой к арифметическому декодеру
Q	12-битовая переменная	Это целое число без знака является произведением $K$ и $P$

Рисунок 2 показывает полную блок-схему алгоритма декодирования DST.

Процесс инициализации показан на рисунке 3 и требуется в начале декодирования каждого фрейма. Он состоит из загрузки первых 13 битов арифметического кода в регистр C и переустановки регистра A в 4095. Первый бит, считываемый в C, будет перезаписан, это предусмотрено, потому что первый бит всегда 0.

Функция «*Input next bit D*» (ввод следующего бита  $D$ ) на рисунке 2, рисунке 3 и рисунке 5 означает, что бит  $D$  берется из *A\_Data* [], начиная с первого бита. После того как все биты из *A\_Data* [] были считаны, функция «*Input next bit D*» устанавливает бит  $D$  в 0.

Чтение информации о вероятности  $P$  получает из модели источника, за исключением первого бита, где вместо этого читается вероятность *DST\_X\_Bit*.

Рисунок 4 иллюстрирует декодирование одного бита  $E$  и обновление регистров  $A$  и  $C$ . Сначала вычисляется текущий аппроксимированный размер интервала  $K$ . Затем произведение аппроксимированного размера интервала  $K$  и применяемой величины вероятности  $P$  сохраняется в  $Q$ . Если  $C$  больше или равно  $A-Q$ , то арифметический код находится в верхней части интервала, что означает, что исходный закодированный бит  $E$  был битом '1'; иначе был передан бит '0'.  $A$  и  $C$  должны быть скорректированы таким же образом, как в кодере.

Процесс ренормализации показан на рисунке 5. Ренормализация требуется, когда значение  $A$  слишком маленькое. Если  $A$  является слишком маленьким, то  $A$  и  $C$  смещаются на один бит влево, и новый бит арифметического кода  $D$  читается в младший значащий бит  $C$ .

Возможно, что последние биты  $A\_Data$  не используются функцией «*Input next bit D*» для того, чтобы декодировать аудиофрейм. Эти неиспользованные биты являются битами заполнения для выравнивания аудиофрейма на границе байта.

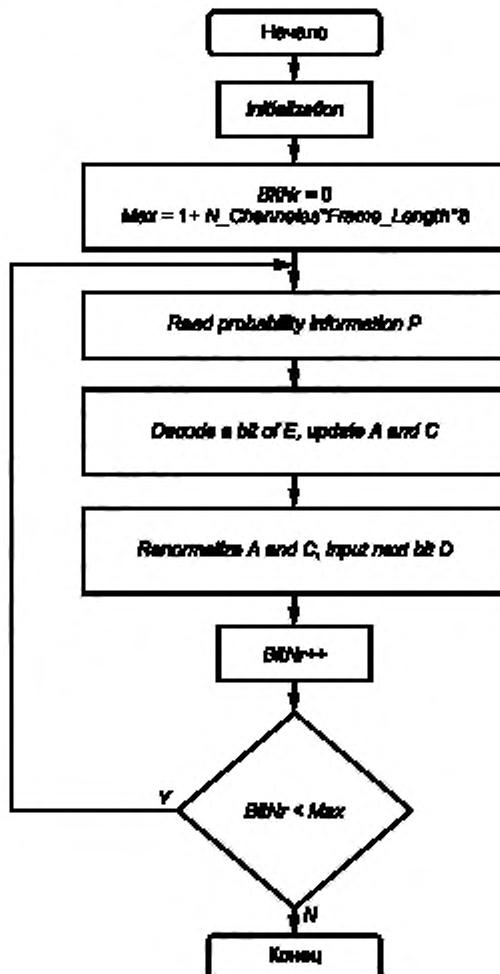


Рисунок 2 — Блок-схема арифметического дёкодера

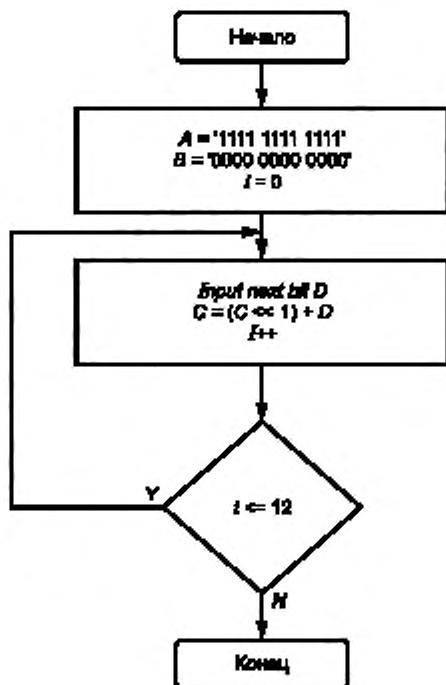
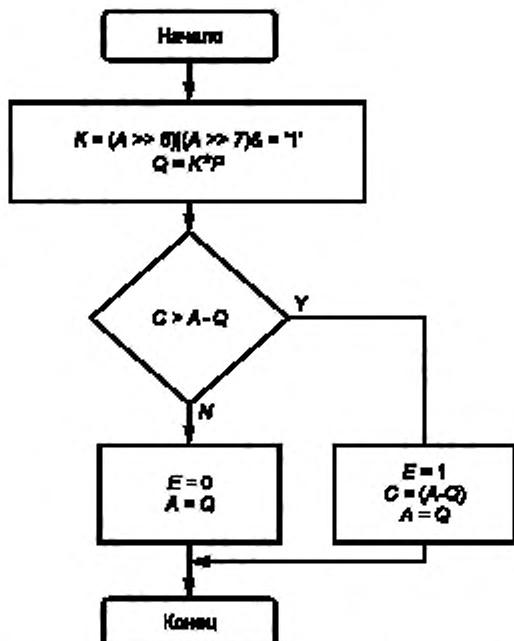
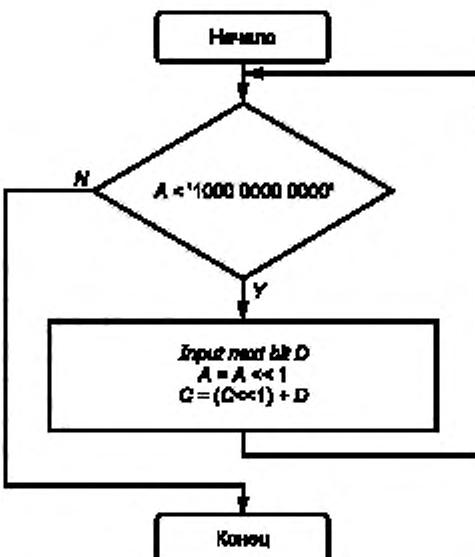


Рисунок 3 — Инициализация

Рисунок 4 — Декодирование бита  $E$ , обновление  $A$  и  $C$

Рисунок 5 — Ренормализация  $A$  и  $C$ , ввод следующего бита(ов)  $D$ 

### 7.1.3 Модель источника

Процесс декодирования в модели источника описывается только для одного канала, поскольку он эквивалентен для всех каналов.

Сегментация и отображение определяют, какой фильтр прогноза и какая  $Ptable$  должны использоваться, чтобы декодировать следующий бит звукового канала. Две функции  $Filter\_N(n)$  и  $Ptable\_N(n)$  возвращают номер фильтра прогноза и  $Ptable$ , используемых для того, чтобы декодировать бит  $n$  звукового канала. Функции  $Filter\_N(n)$  и  $Ptable\_N(n)$  используют информацию сегментации и отображения.  $Filter\_N(n)$  определяется следующим образом:

$$Filters.Start[1] = 0$$

$$Filters.Start[Seg+1] = Filters.Start[Seg] + Filters.Segment\_Length[Channel\_Nr][Seg],$$

где  $Seg = 1 \dots Filters.Nr\_Of\_Segments[Channel\_Nr]$  и  $Seg$  является номером сегмента звукового канала  $Channel\_Nr$ .

Для бита  $n$  переменный  $Seg$  может быть определен:

```

if ((n>>3) >= Filter.Start[Filters.Nr_of_Segments[Channel_Nr]])
{
    Seg = Filters.Nr_of_Segments[Channel_Nr]
}
else
{
    Filter.Start[Seg] <= (n<<3) < Filter.Start[Seg+1]
}

```

$Filter\_N(n)$  может быть найдено путем использования формулы  $Filter\_N(n) = Filter[Channel\_Nr][Seg]$ .

Функция  $Ptable\_N(n)$  определяется, используя тот же самый механизм путем замены  $Filters$  на  $Ptables$  и замены  $Filter\_N$  на  $Ptable\_N$ .

$Filters.Segment\_Length[]/[]$  и  $Ptables.Segment\_Length[]/[]$  определяются в 6.2.1.1.5.

$Filters.Nr\_Of\_Segments[]/[]$  и  $Ptables.Nr\_Of\_Segments[]/[]$  определяются в 6.2.1.1.5.

$Filter[]/[]$  и  $Ptable[]/[]$  определяется в 6.2.1.1.6.

$Filter\_N(n)$  и  $Ptable\_N(n)$  используются в следующих определениях:

*N Pred\_Order[Filter\_N(n)]* Порядок прогноза фильтра прогноза, который использует текущий сегмент.

*H[] Coef[Filter\_N(n)]* Коэффициенты фильтра прогноза, который использует текущий сегмент.

*L Ptable\_Len[Ptable\_N(n)]* Длина *Ptable*, которую использует текущий сегмент.

*T[] P\_one[Ptable\_N(n)]* Записи *Ptable*, которую использует текущий сегмент.

*n* Номер бита в диапазоне  $0..8*Frame\_Length - 1$  Переменная, которая проходит через все биты текущего аудиофрейма.

### 7.1.3.1 Инициализация

Инициализация фильтра прогноза в начале каждого фрейма определяется следующим образом:

$$Y[m] = (-1)^m \text{ для } -N \leq m < 0.$$

Выходное значение фильтра прогноза определяется как

$$Z[n] = \sum_{i=0}^{N-1} Y[n-1-i] H[i].$$

$\mathbf{Q}$ -функция преобразовывает *Z* в *F* следующим образом:

$$F[n] = \begin{cases} 1, & \text{если } Z[n] \geq 0 \\ 0, & \text{если } Z[n] < 0 \end{cases}$$

Есть два метода для того, чтобы применить значения вероятности к арифметическому декодеру. В случае *Half\_Prob [Channel\_N]* = '0' значение вероятности определяется так:

$$P[n] = T[\min(|Z(n)|) \gg 3, L - 1].$$

В случае если *Half\_Prob [Channel\_N]* = '1', значение вероятности определяется

$$P[n] = \begin{cases} 128, & \text{если } 0 \leq n < N \\ T[\min(|Z(n)|) \gg 3, L - 1], & \text{если } n > N \end{cases}$$

Значение *P[n]* применяется к арифметическому декодеру, который возвратит значение *E[n]*. Выходное значение *X[n]* (выборка DSD) является исключающим ИЛИ для *E[n]* и *F[n]*:

$$X[n] = E[n] \oplus F[n].$$

Логическое значение *X[n]* преобразовывается в численное значение *Y[n]* следующим образом:

$$Y[n] = \begin{cases} +1, & \text{если } X[n] = 1 \\ -1, & \text{если } X[n] = 0 \end{cases}$$

### 7.1.4 Мультиплексирование/демультиплексирование

Устройство мультиплексирования/демультиплексирования соединяет каждую модель источника с арифметическим декодером в соответствующий момент.

Для удобочитаемости уравнений вводятся некоторые новые переменные:

*C N\_Channels*, число используемых звуковых каналов.

*N* порядковый номер бита, диапазон:  $0..8*Frame\_Length - 1$ .

Для  $1 \leq i \leq C$  имеем:

*DST\_X\_Bit* = *E[0]*.

*P[0]* = *P(DST\_X\_Bit)*,

*E\_i[n]* = *E[C n + i]*,

*P[C n + i]* = *P\_i[n]*.

*P(DST\_X\_Bit)* должно быть взято из *Coef[0][0]* следующим образом. Если *Coef[0][0] = %c8c7c6c5c4c3c2c1c0*, то *P(DST\_X\_Bits)* должно быть равным  $%0c_0c_1c_2c_3c_4c_5c_6 + 1$ . сх представляет значение отдельных битов с *x*, находящимся в диапазоне  $0..8$ . Значение *P(DST\_X\_Bit)* находится в диапазоне  $1..128$ .

## 7.2 Ограничения на кодированный DST Audio\_Frames

Чтобы позволить оптимальный проект декодера DST, должны быть введены следующие ограничения для кодированных DST аудиофреймов.

### 7.2.1 Ограниченнное количество ошибочно предсказанных выборок

Максимальное разрешенное число ошибочно предсказанных выборок в кодированном *Audio\_Frame* DST является половиной числа выборок DSD во фрейме.

$$N_{\text{Errors}}_{\text{max}} = \frac{N_{\text{Channels}} * \text{Frame\_Length} * 8}{2}$$

Общее количество ошибочных предсказанных выборок (*N\_Errors*) во фрейме является суммой числа ошибочных предсказанных выборок на звуковой канал:

$$N_{\text{Errors}} = \sum_{j=1}^{N_{\text{Channels}}} \left( \sum_{n=0}^{\text{Frame\_Length} * 8 - 1} E_j[n] \right).$$

где  $E_j[n]$  равно 0 (хороший прогноз) или 1 (неправильный прогноз).

Для каждого кодированного DST аудиофрейма должно применяться следующее правило:  
 $N_{\text{Errors}} \leq N_{\text{Errors}}_{\text{max}}$ .

### 7.2.2 Требование к разработке таблицы вероятности

Определенные в этом пункте ограничения должны применяться к таблицам *Ptables*.

Для каждой *Ptable*, которая используется во фрейме, содержание определяется следующим алгоритмом.

Обратите внимание на все выборки фрейма, которые используют рассматриваемую *Ptable*, и подсчитайте для этих выборок, сколько раз используется запись *Ptable[CA[]][]* и сколько раз сигнал *E* равен 1 для записи *Ptable(CW[])*

```

for (Ptable_Nr=0; Ptable_Nr<Nr_Of_Ptables; Ptable_Nr++)
{
    for (Entry_Nr=0, Entry_Nr<Ptable_Len[Ptable_Nr]; Entry_Nr++)
    {
        CA[Ptable_Nr][Entry_Nr] = 0; CW[Ptable_Nr][Entry_Nr] = 0;
    }
}
for (Channel_Nr=1; Channel_Nr<=N_Channels; Channel_Nr++)
{
    if (Half_Prob[Channel_Nr]==0)
    {
        Start = 0;
    }
    else
    {
        Start = Pred_Order[Filter[Channel_Nr][1]];
    }
    Stop = 0;
    for (Seg_Nr=1; Seg_Nr<=Ptables.Nr_Of_Segments[Channel_Nr]; Seg_Nr++)
    {
        Stop += 8*Ptables.Segment_Length[Channel_Nr][Seg_Nr];
        for (Bit_Nr=Start; Bit_Nr<Stop; Bit_Nr++)
        {
            Ptable_Nr = Ptable[Channel_Nr][Seg_Nr];
            Entry_Nr = min([Z[Channel_Nr][Bit_Nr]>>3, Ptable_Len[Ptable_Nr]-1);
            CA[Ptable_Nr][Entry_Nr]++;
            CW[Ptable_Nr][Entry_Nr] += EChannel_Nr[Bit_Nr];
        }
        Start = Stop;
    }
}

```

Для каждой записи *Ptable* вероятность для ошибочного прогноза получается из этих чисел следующим образом:

```

for (Ptable_Nr=0; Ptable_Nr<Nr_Of_Ptables; Ptable_Nr++)
{
    for (Entry_Nr=0; Entry_Nr<Ptable_Len[Ptable_Nr]; Entry_Nr++)
    {
        if (CA[Ptable_Nr][Entry_Nr] == 0)
        {
            P_min[Ptable_Nr][Entry_Nr] = 1;
        }
        else
        {
            p = trunc((s12 * cw[Ptable_Nr][Entry_Nr] + CA[Ptable_Nr][Entry_Nr]) / 2 * CA[Ptable_Nr][Entry_Nr]);
            P_min[Ptable_Nr][Entry_Nr] = min(max(p, 1), 128);
        }
    }
}

```

*P\_min* являются минимальными разрешенными значениями вероятности для записей *Ptables*. Для каждой записи каждой *Ptable* вероятности, фактически используемые для кодирования (*P\_one* [] []), должны соответствовать следующему условию:

$$P_{\min}[Ptable\_Nr][Entry\_Nr] \leq P_{one}[Ptable\_Nr][Entry\_Nr] \leq 1.$$

## Приложение А (справочное)

### Описание кодера

#### A.1 Технический обзор

В алгоритме аудиокодирования без потерь имеют место три основных процесса: формирование фреймов (структуривание), прогноз и кодирование энтропии. В кодере входящие одноразрядные данные сначала структурируются, а затем передаются этапу прогноза. После этапа прогноза ошибочный сигнал, вычисленный на основе прогноза и исходного сигнала, передается этапу кодирования энтропии. На этом последнем этапе ошибочный сигнал кодируется, используя арифметическое кодирование, у которого производительность близка к (оптимальному) кодированию энтропии. Фильтры прогноза и таблицы вероятности могут быть сгруппированы для каналов с целью еще лучшей эффективности кодирования. Сгенерированные арифметическим кодером данные объединяются с коэффициентами прогноза и мультиплексируются в потоке битов.

#### A.1.1 Структурирование

Цель структурирования (формирования фрейма) является двойкой. Во-первых, структурирование необходимо, чтобы обеспечить легкий, «случайный» доступ к аудиоданным во время воспроизведения. По той же самой причине каждый фрейм должен быть независимо закодирован, что позволяет проигрывателю декодировать отдельные фреймы без знания о предыдущих фреймах. Во-вторых, структурирование позволяет аудиоконтенты во фрейме расценивать как стационарные (или по крайней мере квазистационарные). Это базовое предположение в процессе прогноза.

Процесс структурирования делит исходный одноразрядный аудиопоток, состоящий из выборок  $b \in \{0, 1\}$ , на фреймы длиной  $M = 37,632$  битов, соответствующие  $1/75$  с, принимая частоту дискретизации  $2,8224$  MS/с.

#### A.1.2 Прогноз

Фильтрация прогноза является первым необходимым шагом в процессе сжатия (аудио) данных. Шаг фильтрации прогноза пытается удалить избыточность из аудиопотока битов  $b$ , создавая новый поток битов  $e$ , который не избычен. Вместе с коэффициентами фильтра прогноза  $h$  поток с ошибками  $e$  переносит ту же самую информацию, что и  $b$ . Фильтр прогноза обозначается как  $z^{-1}H(z)$ , чтобы подчеркнуть тот факт, что передача фильтра содержит задержку, которая обязательна, чтобы создать кодер, который может быть инвертирован по времени.

Фильтр прогноза  $FIR$  может быть разработан согласно стандартным методам, самый известный из них основан на минимальной среднеквадратической ошибке (MMSE). Применение критерия MMSE приводит к уравнению ошибочного прогноза, которое должно быть минимизировано:

$$\sum_{n=1}^M e^2[n] = \sum_{n=1}^M \left( \sum_{i=1}^L h[i] b[n-i] - b[n] \right)^2,$$

где  $M$  является числом битов на фрейм и  $L$  — числом коэффициентов прогноза, которые кодируются как 'Coded\_Pred\_Order' 1'. После дальнейшего манипулирования это приводит к коэффициентам  $h$ . Фильтр  $FIR$ , найденный таким образом, будет минимальным фазовым фильтром. Чтобы получить оптимальный баланс между точностью прогноза и числом битов, взятым описанием фильтра прогноза, коэффициенты фильтра прогноза квантуются в 9-разрядные числа с фиксированной точкой первым масштабированием их на 256. Полученные коэффициенты сохраняются в 'Coef[Filter\_Nr]' [Coef\_Nr], где  $Filter\_Nr$  обозначает индекс фильтра, а  $Coef\_Nr$  — индекс коэффициента фильтра соответственно. Сигнал прогноза  $z$  является мультибитовым. Биты прогноза  $q$  получаются из мультибитовых значений  $z$  простым усечением, обозначенным блоком, который маркирован как  $Q(z)$ . Ошибка между потоком битов  $b$  и мультибитовым сигналом  $z$  минимизируется, тогда как в идеале было бы минимизировано различие между  $b$  и  $q$ , однобитовая квантованная версия  $z$ . Это, однако, приводит к трудоемким расчетам.

Ошибкачный сигнал  $e$  вычисляется операцией исключающего или ( $XOR$ ) между  $b$  и  $q$ . Цель фильтра прогноза состоит в том, чтобы создать столько нулей в  $e$ , сколько возможно, поскольку это обеспечит существенное снижение объема данных энтропийным кодированием.

#### A.1.3 Арифметическое кодирование

Когда используются надлежащие фильтры прогноза, сигнал  $e$  будет состоять из большего количества нулей, чем единиц, и может таким образом привести к возможному увеличению компрессии. Предположите, что вероятность '1' в  $e$  обозначается как  $p$ , тогда вероятность '0' равняется  $(1-p)$ . Минимальное число битов  $N_{bits}$ , которым может быть представлен в среднем единственный бит потока  $e$ , тогда равняется

$$N_{bits} = -(p \log_2[p] + (1-p) \log_2[1-p]).$$

Предположим, что 90 % всех прогнозов корректны, тогда  $p = 0,1$  и  $N_{bits} = 0,47$ . Как правило, возможна компрессия приблизительно с коэффициентом 2. В то время как этот подсчет, основанный на вычислении энтропии,

представляет верхний предел достижимого сжатия, алгоритм, который при практических обстоятельствах приближается к этому пределу, является алгоритмом арифметического кодирования.

Методы арифметического кодирования могут успешно использоваться только тогда, когда доступна точная информация о вероятностях символов «0» или «1».

Вероятности символов, необходимые для арифметического кодирования, вычисляются, составляя гистограмму (или таблицу). Обозначая вероятность, что прогноз корректен  $P(e = 0)$ , видим, что, так как  $P(e = 0) = 1 - P(e = 1)$ , нет необходимости вычислять две таблицы: для арифметического кодирования используется только таблица вероятности ошибки  $t$  для  $P(e = 1)$  и передается декодеру.

#### A.1.4 Мультиплексирование канала

В предыдущих разделах «модель источника», состоящая из фильтра прогноза и таблицы вероятности, обсуждалась для одного канала. В полном кодере у каждого канала имеется своя собственная модель источника, тогда как используется только единственный арифметический кодер. Однако, чтобы использовать корреляцию между каналами, также можно позволить каналам совместно использовать фильтры прогноза и/или таблицы вероятности. Совместное использование фильтров или таблиц вероятности выгодно, когда уменьшение числа битов метаданных, необходимых, чтобы передать информацию о фильтре или таблице с кодера на декодер, выше, чем увеличение числа битов арифметического кода. Последнее число обычно будет несколько больше, так как не всегда возможно создать фильтр прогноза (или таблицу вероятности), который приводит к оптимальному арифметическому кодированию для всех каналов, которые его используют.

Арифметический кодер получает для каждого канала потоки  $e$  и  $r$ , которые поставляются отдельными моделями источника.

### Библиография

- [1] ИСО/МЭК 14496-3:2009<sup>1)</sup> Информационные технологии. Кодирование аудиовизуальных объектов. Часть 3. Аудио (ИСО/МЭК14496-3:2009 *Information technology — Coding of audio-visual objects — Part 3: Audio*)

---

<sup>1)</sup> Заменен на ISO/IEC 14496-3:2019.

---

УДК 621.396:006.354

ОКС 33.170

Ключевые слова: звуковое вещание, электрические параметры, каналы и тракты, технологии MPEG-кодирования, синтетический звук, масштабирование, защита от ошибок, поток битов расширения, психоакустическая модель

---

Редактор переиздания *Е.В. Яковлева*  
Технические редакторы *В.Н. Прусакова, И.Е. Черепкова*  
Корректор *Е.М. Поляченко*  
Компьютерная верстка *Г.В. Струковой*

Сдано в набор 13.07.2020. Подписано в печать 30.09.2020. Формат 60 × 84 $\frac{1}{8}$ . Гарнитура Ариал.  
Усл. печ. л. 3,72. Уч.-изд. л. 3,40.

Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

---

ИД «Юриспруденция», 115419, Москва, ул. Орджоникидзе, 11.  
[www.jurisizdat.ru](http://www.jurisizdat.ru) [y-book@mail.ru](mailto:y-book@mail.ru)

Создано в единичном исполнении во ФГУП «СТАНДАРТИНФОРМ»  
для комплектования Федерального информационного фонда стандартов,  
117418 Москва, Нахимовский пр-т, д. 31, к. 2.  
[www.gostinfo.ru](http://www.gostinfo.ru) [info@gostinfo.ru](mailto:info@gostinfo.ru)