ФЕДЕРАЛЬНОЕ АГЕНТСТВО
ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р 53556.4— 2013

Звуковое вещание цифровое

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

Часть III (MPEG-4 AUDIO)

Основные методы кодирования звуковых сигналов (GA): усовершенствованное звуковое кодирование (AAC), взвешивающее векторное квантование (TwinVQ), побитовое арифметическое кодирование (BSAC)

ISO/IEC 14496-3:2009 (NEQ)

Издание официальное

Москва
Стандартинформ
2014

gost.e-ecolog.ru

Предисловие

1 РАЗРАБОТАН Санкт-Петербургским филиалом Центрального научно-исследовательского института связи «Ленинградское отделение» (ФГУП ЛО ЦНИИС)

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 480 «Связь»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 22 ноября 2013 г. № 2152-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)

5 ВВЕДЕН ВПЕРВЫЕ

Правила применения настоящего стандарта установлены в ГОСТ Р 1.0—2012 (раздел 8). 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 (gost.ru)

© Стандартинформ, 2014

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

Содержание

1 Область применения . . . 1
1.1 Технический обзор . . . 1
2 Нормативные ссылки . . . 6
3 Термины и определения . . . 6
4 Синтаксис . . . 6
4.1 Конфигурация декодера (*GASpecificConfig*) . . . 6
4.2 Полезные нагрузки потока битов *GA* . . . 8
5 Общая структура данных . . . 48
5.1 Декодирование специальной конфигурации *GA* . . . 48
5.2 Декодирование полезных нагрузок потока битов *GA* . . . 51
5.3 Требования к буферу . . . 103
5.4 Таблицы . . . 105
6 Описания инструмента *GA* . . . 120
6.1 Квантование . . . 120
6.2 Масштабные коэффициенты . . . 121
6.3 Бесшумное кодирование . . . 123
6.4 Бесшумное кодирование для мелкоструктурной масштабируемости . . . 130
6.5 Квантование чередующегося вектора . . . 138
6.6 Прогноз частотной области . . . 141
6.7 Долгосрочный прогноз (*LTP*) . . . 141
6.8 Объединенное кодирование . . . 143
6.9 Временное формирование шума (*TNS*) . . . 150
6.10 Нормализация спектра . . . 154
6.11 Гребенка фильтров и переключение блока . . . 164
6.12 Управление усилением . . . 169
6.13 Перцепционная шумовая замена (*PNS*) . . . 175
6.14 Модуль частотно-избирательного переключателя (*FSS*) . . . 177
6.15 Инструмент фильтра повышенной дискретизации . . . 179
6.16 Инструменты для устойчивости к ошибкам *AAC* . . . 180
6.17 Кодек с низкой задержкой . . . 187
6.18 Инструмент *SBR* . . . 189
6.19 *SBR* с низкой задержкой . . . 216
6.20 Расширенный кодек с малой задержкой . . . 221
Приложение А (обязательное) Нормативные таблицы . . . 224

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)

Основные методы кодирования звуковых сигналов (GA): усовершенствованное звуковое кодирование (AAC), взвешивающее векторное квантование (TwinVQ), побитовое арифметическое кодирование (BSAC)

Sound broadcasting digital. Coding of signals of sound broadcasting with reduction of redundancy for transfer on digital communication channels. A part III (MPEG-4 audio). General audio coding (GA): advanced audio coding (AAC), weighing vector quantization (TwinVQ), bit-by-bit arithmetic coding (BSAC)

Дата введения — 2014—09—01

1 Область применения

Подсистема кодирования общее аудио (*GA*) для MPEG-4 *Audio* предназначена для использования в качестве универсального аудиокодирования при самых низких скоростях передачи. Кодирование *GA* используется для сигналов сложного музыкального материала в моно от 6 Кбит/с на канал и для стерео от 12 Кбит/с на сигнал стерео до аудио вещательного качества при 64 Кбит/с или более на канал. Кодированный MPEG-4 материал может быть представлен или единственным набором данных, как в MPEG-1 и MPEG-2 *Audio*, или несколькими подмножествами, которые позволяют декодировать при различных уровнях качества, в зависимости от числа подмножеств, доступных на стороне декодера (масштабируемость скорости передачи).

Синтаксис усовершенствованного кодирования звука (*AAC*) MPEG-2 (включая поддержку многоканального аудио) полностью поддерживается кодированием MPEG-4 *Audio GA*. Все функции и возможности стандарта MPEG-2 *AAC* также применяются к MPEG-4. *AAC* был протестирован на предмет получения 'неотличимого' качества на скоростях передачи данных 320 Кбит/с для аудиосигналов с пятью полнодиапазонными каналами. В MPEG-4 инструменты, полученные из MPEG-2 *AAC*, доступны вместе с другими инструментами кодирования MPEG-4 *GA*, которые обеспечивают дополнительную функциональность, такую как масштабируемость скорости передачи и улучшенная эффективность кодирования на очень низких скоростях передачи. Масштабируемость скорости передачи достигается только инструментами кодирования *GA* или при использовании комбинации с внешним базовым кодером.

Кодирование MPEG-4 *GA* не ограничивается некоторыми фиксированными скоростям передачи и поддерживает широкий диапазон скоростей передачи и кодирование с переменной скоростью. В то время, как эффективное моно, стерео кодирование и многоканальное кодирование, возможно с использованием расширенных инструментов, получаемых из MPEG-2 *AAC*. Настоящий стандарт также обеспечивает расширение этого комплекта инструментов, которые позволяют масштабируемость моно/стерео, где сигнал моно может быть извлечен, декодируя только подмножества закодированного потока стерео.

1.1 Технический обзор

1.1.1 Кодер и декодер

Существуют связанные с *MPEG-2 AAC* инструменты с дополнениями *MPEG-4* для некоторых из них и инструменты, связанные с квантованием *TwinVQ* и кодированием. *TwinVQ* является альтернативным модулем для квантования *AAC*-типа, это основано на чередующемся векторном квантовании и спектральной оценке *LPC* (кодирование с линейным предсказанием). Это работает от 6 Кбит/с и рекомендуется к использованию при менее 16 Кбит/с с постоянной скоростью передачи.

В функции декодера входят поиск и описание квантованных спектров звука в полезной нагрузке потока битов, декодирование квантованных значений и другой информации о реконструкции, восстановление квантованных спектров, обработка восстановленных спектров любыми инструментами, которые актив-

Издание официальное

ны в полезной нагрузке потока битов, чтобы достигнуть фактических сигнальных спектров полезной нагрузкой входного потока битов и преобразовать спектры частотной области во временную область, с дополнительным инструментом регулировки усиления или без него. После начальной реконструкции и масштабирования реконструкции спектра есть множество дополнительных инструментов, которые изменяют один или более спектров, чтобы обеспечить более эффективное кодирование. Для каждого из дополнительных инструментов, которые работают в спектральной области, сохраняется опция "пройти", и во всех случаях, когда спектральная работа опускается, спектры при ее вводе передаются непосредственно через инструмент без модификации.

1.1.2 Краткий обзор инструментов кодера и декодера

Вводом в инструмент демультиплексора полезной нагрузки потока битов является полезная нагрузка потока битов *MPEG-4 GA*. Демультиплексор разделяет полезную нагрузку потока битов на части для каждого инструмента и обеспечивает каждый из инструментов информацией о полезной нагрузке потока битов, связанного с этим инструментом.

Выводы из инструмента демультиплексора полезной нагрузки потока битов таковы:

- квантованные (и дополнительно бесшумно кодированные) спектры, представленные с помощь информации о разделении и бесшумно кодированных спектров (*AAC*) или ряда индексов векторов кода (*TwinVQ*) информации об арифметической модели и бесшумно кодированного спектра (*BSAC*);
- информация о решении *M/S* (дополнительная);
- информация о стороне прогнозирующего устройства (дополнительная);
- информация о перцепционной замене шума (*PNS*) (дополнительная);
- информация об интенсивности управления стерео и информация об управлении связываниющего канала (обе дополнительные);
- информация о формировании временного шума (*TNS*) (дополнительная);
- информация об управлении гребенки фильтров;
- информация о регулировке усиления (дополнительная);
- дополнительная информация, связанная с масштабируемостью скорости передачи (дополнительная).

Инструмент декодирования бесшумности *AAC* берет информацию из демультиплексора полезной нагрузки потока битов, анализирует эту информацию, декодирует кодированные по Хафману данные и восстанавливает квантованные спектры и масштабные коэффициенты, кодированные по Хафману и *DPCM*.

Вводы в инструмент декодирования бесшумности таковы:

- информация о разделении для бесшумно кодированных спектров;
- бесшумно кодированные спектры.

Выводы инструмента декодирования бесшумности:

- декодированное целочисленное представление масштабных коэффициентов;
- к вантованные значения для спектров.

Инструмент инверсного квантователя принимает квантованные значения для спектров и преобразовывает целочисленные значения в немасштабируемые, восстановленные спектры. Этот квантователь является неоднородным квантователем.

Ввод в инструмент инверсного квантователя — квантованные значения для спектров.

Вывод инструмента инверсного квантователя — немасштабированные, инверсно квантованные спектры.

Инструмент перемасштабирования преобразовывает целочисленное представление масштабных коэффициентов в фактические значения и умножает немасштабированные инверсно квантованные спектры на соответствующие масштабные коэффициенты.

Вводы в инструмент масштабных коэффициентов следующие:

- декодированное целочисленное представление масштабных коэффициентов;
- немасштабированные, инверсно квантованные спектры.

Вывод из инструмента масштабных коэффициентов — масштабированные, инверсно квантованные спектры.

Инструмент *M/S* преобразовывает пары спектров из *Mid/Side* в *Left/Right* под управлением информации о решении *M/S*, улучшая качество отображения стерео и иногда обеспечивая эффективность кодирования.

Вводы в инструмент *M/S* такие:

- информация о решении *M/S*;
- масштабированные, инверсно квантованные спектры, связанные с парами каналов.

Вывод из инструмента *M/S* — масштабированные, инверсно квантованные спектры, связанные с парами каналов, после декодирования *M/S*.

Примечание — Масштабированные, инверсно квантованные спектры индивидуально кодированных каналов не обрабатываются блоком *M/S*, скорее их пропускают прямо через блок без модификации. Если блок *M/S* не является активным, то все спектры проходят через этот блок неизмененными.

Инструмент прогноза инвертирует процесс прогноза, выполненный в кодере. Этот процесс прогноза заново вводит избыточность, которая была извлечена инструментом прогноза в кодере, под управлением информации о состоянии прогнозирующего устройства. Этот инструмент реализуется как обратное адаптивное прогнозирующее устройство второго порядка.

Вводы в инструмент прогноза следующие:

- информация о состоянии прогнозирующего устройства;
- информация о стороне прогнозирующего устройства;
- масштабируемые, инверсно квантованные спектры

Вывод из инструмента прогноза — масштабируемые, инверсно квантованные спектры, после того, как применяется прогноз.

Примечание — Если прогноз отключается, масштабируемые, инверсно квантованные спектры передают непосредственно через блок без модификации.

Альтернативно предусмотрен адаптивный в прямом направлении инструмент долгосрочного прогноза.

Вводы в инструмент долгосрочного прогноза:

- восстановленный вывод временной области декодера;
- масштабируемые, инверсно квантованные спектры.

Вывод из инструмента долгосрочного прогноза — масштабируемые, инверсно квантованные спектры, после того, как применен прогноз.

Примечание — Если прогноз отключается, масштабируемые, инверсно квантованные спектры передают непосредственно через блок без модификации.

Инструмент замены перцепционного (воспринимаемого) шума (*PNS*) реализует декодирование шумовой замены в спектре канала, обеспечивая эффективное представление компонентов шумоподобного сигнала.

Вводы в инструмент замены перцепционного шума такие:

- инверсно квантованные спектры;
- управляющая информация замены перцепционного шума.

Вывод из инструмента замены перцепционного шума — инверсно квантованные спектры.

Примечание — Если какая-либо часть этого блока отключается, масштабируемые, инверсно квантованные спектры передаются непосредственно через эту часть без модификации. Если блок замены перцепционного шума неактивен, все спектры передают через этот блок неизмененными.

Инструмент интенсивности стерео реализует декодирование интенсивности стерео на парах спектров.

Вводы в инструмент интенсивности стерео следующие:

- инверсно квантованные спектры;
- управляющая информация интенсивности стерео.

Вывод из инструмента интенсивности стерео — инверсно квантованные спектры после декодирования канала интенсивности.

Примечание — Масштабированные, инверсно квантованные спектры индивидуально кодированных каналов передаются непосредственно через этот инструмент без модификации. Инструмент интенсивности стерео и инструмент *M/S* располагаются так, чтобы работа *M/S* и интенсивности стерео была взаимоисключающей на любой данной полосе коэффициента масштабирования и группе одной пары спектров.

Инструмент спаривания для взаимозависимо коммутируемых спаренных каналов добавляет соответствующие данные из зависимо коммутируемых спаренных каналов к спектрам, как назначено управляющей информацией спаривания.

Вводы в инструмент спаривания следующие:

- инверсно квантованные спектры;
- информация управления спариванием.

Вывод из инструмента спаривания — инверсно квантованные спектры, спаренные с взаимнозависимо коммутируемыми каналами спаривания.

Примечание — Масштабированные, инверсно квантованные спектры проходят непосредственно через этот инструмент без модификации, если спаривание не обозначено. В зависимости от управляющей информации спаривания зависимо переключаемые каналы спаривания могут быть спарены до или после обработки *TNS*.

Инструмент спаривания для независимо коммутируемых каналов спаривания добавляет соответствующие данные из независимо коммутируемых каналов спаривания к сигналу времени, как назначено управляющей информацией спаривания.

Вводы в инструмент спаривания:

- сигнал времени как выход гребенки фильтров;
- управляющая информация спаривания.

Вывод из инструмента спаривания — сигнал времени, спаренный с независимо коммутируемыми каналами спаривания.

Примечание — Сигнал времени проходит непосредственно через этот инструмент без модификации, если спаривание не обозначается.

Инструмент формирования временного шума (*TNS*) реализует управление точной временной структурой шума кодирования. В кодере процесс *TNS* сгладил временную огибающую сигнала, к которому он был применен. В декодере используется обратный процесс, чтобы восстановить фактическую временную огибающую(ие), под управлением информации *TNS*. Это делается, применяя процесс фильтрации к частям спектральных данных.

Вводы в инструмент *TNS* следующие:

- инверсно квантованные спектры;
- информация *TNS*

Вывод из блока *TNS* — инверсно квантованные спектры.

Примечание — Если этот блок отключается, инверсно квантованные спектры проходят без модификации.

Инструмент гребенка фильтров/переключение блока применяет инверсию отображения частоты, которая была выполнена в кодере. Инверсно модифицированное дискретное косинусное преобразование (*IMDCT*) используется для инструмента гребенка фильтров. *IMDCT* может быть сконфигурировано для поддержки или одного набора из 120, 128, 480, 512, 960, или 1024, или четырех наборов из 32 или 256 спектральных коэффициентов.

Вводы в инструмент гребенка фильтров следующие:

- инверсно квантованные спектры;
- управляющая информация гребенки фильтров/переключение блока.

Вывод(ы) из инструмента гребенка фильтров — восстановленный аудиосигнал(ы) временной области.

Когда существует инструмент гребенка фильтров регулировки усиления применяют отдельную регулировку усиления временной области к каждой из 4 полос частот, которые были созданы гребенкой фильтров *PQF* регулировки усиления в кодере. Затем он собирает эти 4 полосы частот и восстанавливает временную форму посредством гребенки фильтров инструмента регулировки усиления.

Вводы в инструмент регулировки усиления:

- восстановленные аудиосигнал(ы) временной области;
- информация о регулировке усиления.

Вывод(ы) из инструмента регулировки усиления — восстановленные аудиосигнал(ы) временной области.

Если инструмент регулировки усиления неактивен, восстановленный(е) аудиосигнал(ы) временной области передается непосредственно из инструмента гребенка фильтров на выход декодера. Этот инструмент используется только для типа аудио объекта с масштабируемой частотой дискретизации (*SSR*).

Инструмент *SBR* регенерирует верхнюю полосу аудиосигнала. Это основано на репликации последовательностей гармоник, усеченных во время кодирования. Он корректирует огибающую спектра сгенерированной верхней части полосы и применяет инверсную фильтрацию, а также добавляет шумовые и синусоидальные компоненты, чтобы воссоздать спектральные характеристики исходного сигнала.

Ввод в инструмент *SBR*:

- квантованные данные огибающей;
- различная управляющая информация;
- сигнал временной области из базового декодера *AAC*.

Вывод из инструмента *SBR* — сигнал временной области.

Инструмент нормализации спектра преобразовывает восстановленные равномерные спектры в фактические значения в декодере. Спектральная огибающая определяется коэффициентами *LPC*, огибающей шкалы Барка, периодическими пиковыми компонентами и усилением.

Ввод в инструмент нормализации спектра:

- восстановленные плоские спектры;

- информация о коэффициентах *LPC*, огибающей шкалы Барка, периодических пиковых компонентах и усилении.

Вывод из инструмента нормализации спектра — восстановленные фактические спектры.

Инструмент чередования *VQ* преобразовывает индекс вектора в сглаженные спектры в декодере *TwinVQ* посредством поиска в таблице кодовой книги и инверсного чередования спектров. Шум квантования минимизируется взвешенной мерой искажения в кодере вместо адаптивного распределения битов. Он является альтернативой инструмента квантования *AAC*.

Ввод в инструмент чередования *VQ* — набор индексов вектора кода.

Вывод из инструмента *TwinVQ* — восстановленные сглаженные спектры.

Инструмент частотно-избирательного коммутатора (*FSS*) используется для того, чтобы управлять комбинацией уровня кодирования *AAC* с *TwinVQ* и уровнем кодирования *CELP*, если они используются в качестве кодера базового уровня в масштабируемых конфигурациях. Во второй функции этот инструмент применяется, чтобы управлять комбинацией уровня кодирования моно и стерео в масштабируемых конфигурациях, где используется уровень кодирования и моно, и стерео, чтобы кодировать входной сигнал стерео.

Инструмент фильтра повышения дискретизации адаптирует частоту дискретизации кодера ядра *CELP*, который может использоваться в качестве кодера базового уровня в масштабируемых конфигурациях, к частоте дискретизации уровня расширения *AAC*.

Ввод в инструмент фильтра повышения дискретизации — выход кодера ядра *CELP*, работающего на более низкой частоте дискретизации, чем уровень расширения *AAC*.

Вывод из инструмента фильтр повышения дискретизации — выход кодера ядра *CELP* с повышенной дискретизацией, соответствующий частоте дискретизации уровня расширения *AAC*, преобразованный в частотную область с точно той же частотой и разрешением времени как уровень расширения *AAC*.

Инструмент бесшумного декодирования *BSAC* получает информацию от демультиплексора полезной нагрузки потока битов, анализирует эту информацию, декодирует арифметически кодированные данные и восстанавливает квантованные спектры, и арифметически кодированные масштабные коэффициенты. Модуль бесшумного кодирования *BSAC* является альтернативой модуля кодирования *AAC*. Бесшумное кодирование *BSAC* используется, чтобы сделать полезную нагрузку потока битов масштабируемой и устойчивой к ошибкам, и уменьшить избыточность масштабных коэффициентов и квантованного спектра.

Вводы в инструмент декодирования *BSAC* следующие:

- информация об арифметической модели для бесшумно кодированных спектров;
- бесшумно кодированные разрядно-модульные данные.

Выводы из инструмента декодирования *BSAC*:

- декодированное целочисленное представление масштабных коэффициентов;
- квантованное значение для спектров.

Инструмент виртуальных кодовых книг (*VCB*11) может расширить подсистему демультиплексора полезной нагрузки потока битов, который декодирует информацию о разделении. Инструмент *VCB*11 дает возможность обнаружить серьезные ошибки в пределах спектральных данных полезной нагрузки потока битов *AAC MPEG*-4.

Ввод в инструмент *VCB*11 — закодированные данные раздела, использующие виртуальные кодовые книги.

Вывод из инструмента *VCB*11 — декодированная информация о разделении.

Инструмент кодирования с реверсивной переменной длиной (*RVLC*) может заменить подсистему инструмента бесшумного кодирования, которая декодирует кодированные по Хафману и *DPCM* масштаные коэффициенты. Инструмент *RVLC* используется, чтобы увеличить устойчивость к ошибкам для данных масштабных коэффициентов в пределах полезной нагрузки потока битов *AAC MPEG*-4.

Ввод в инструмент *RVLC* — бесшумно кодированные масштабные коэффициенты, использующие *RVLC*.

Вывод из инструмента *RVLC* — декодированное целочисленное представление масштабных коэффициентов.

Инструмент переупорядочивания кодовой комбинации Хаффмана (*HCR*) может расширить подсистему инструмента бесшумного кодирования, который декодирует спектральные данные, кодированные по Хафману. Инструмент *HCR* используется, чтобы увеличить устойчивость к ошибкам для спектральных данных в пределах полезных нагрузки потока битов *AAC MPEG*-4.

Ввод в инструмент *HCR* следующий:

- информация о разделении для бесшумно кодированных спектров;
- бесшумно кодированные спектральные данные устойчивым к ошибкам способом переупорядочения;

- длина самой длинной кодовой комбинации в пределах спектральных данных;
- длина спектральных данных.

Вывод из инструмента *HCR* — квантованное значение спектров.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты:

ГОСТ Р 53556.0—2009 Звуковое вещание цифровое. Кодирование сигналов звукового вещания с сокращением избыточности для передачи по цифровым каналам связи. MPEG-4, часть III (MPEG-4 *audio*). Основные положения

ГОСТ Р 54711—2011 Звуковое вещание цифровое. Кодирование сигналов звукового вещания с сокращением избыточности для передачи по цифровым каналам связи. MPEG-1 часть III (MPEG-1 audio)

ГОСТ Р 54712—2011 Звуковое вещание цифровое. Кодирование сигналов звукового вещания с сокращением избыточности для передачи по цифровым каналам связи. MPEG-2, часть III (MPEG-2 audio)

ГОСТ Р 54713—2011 Звуковое вещание цифровое. Кодирование сигналов звукового вещания с сокращением избыточности для передачи по цифровым каналам связи. MPEG-2, часть VII. Усовершенствованное кодирование звука (MPEG-2 *AAC*)

Примечание — При пользовании настоящим стандартом целесообразно проверить действие ссылочных стандартов в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет или по ежегодно издаваемому информационному указателю «Национальные стандарты», который опубликован по состоянию на 1 января текущего года, и по соответствующим ежемесячно издаваемым информационным указателям, опубликованным в текущем году. Если заменен ссылочный стандарт, на который дана недатированная ссылка, то рекомендуется использовать действующую версию этого стандарта с учетом всех внесенных в данную версию изменений. Если заменен ссылочный стандарт, на который дана датированная ссылка, то рекомендуется использовать версию этого стандарта с указанным выше годом утверждения (принятия). Если после утверждения настоящего стандарта в ссылочный стандарт, на который дана датированная ссылка, внесено изменение, затрагивающее положение, на которое дана ссылка, это положение рекомендуется применять без учета данного изменения. Если ссылочный стандарт отменен без замены, то положение, в котором дана ссылка на него, рекомендуется применять в части, не затрагивающей эту ссылку.

3 Термины и определения

В настоящем стандарте применены термины по ГОСТ Р 53556.0.

4 Синтаксис

4.1 Конфигурация декодера (*GASpecificConfig*)

Таблица 1 — Синтаксис *GASpecificConfig* ()

Синтаксис	Количество битов	Мнемоника
GASpecificConfig (samplingFrequencyIndex, channelConfiguration, audioObjectType)		
{	1	*bslbf*
frameLengthFlag; dependsOnCoreCoder;	1	*bslbf*
if (dependsOnCoreCoder) {		
coreCoderDelay;		
}		
extensionFlag;	14	*uimsbf*
if (! channelConfiguration) {	1	*bslbf*
program_config_ element ();		
}		
if ((audioObjectType == 6) || *(audioObjectType* == 20)) {		
layerNr;		
}		
if (extensionFlag) {		
if (audioObjectType == 22) {	3	*uimsbf*
numOfSubFrame;		

Окончание таблицы 1

Синтаксис	Количество битов	Мнемоника
layer_length;		
}	5	*bslbf*
if (audioObjectType == 17 || *audioObjectType* == 19 || *audioObjectType* == 20 || *audioObjectType* == 23*) {*	11	*bslbf*
aacSectionDataResilienceFlag;		
aacScalefactorDataResilienceFlag;		
aacSpectralDataResilienceFlag;		
}	1	*bslbf*
extensionFlag3;	1	*bslbf*
if (extensionFlag3) {	1	*bslbf*
/ tbd in version 3 * /*	1	*bslbf*
}		
}		
}		

4.1.1 Элемент конфигурации программы

Таблица 2 — Синтаксис *program_config_element*()

Синтаксис	Количество битов	Мнемоника
program_config_element()		
{		
element_instance_tag;	4	*uimsbf*
object_type;	2	*uimsbf*
sampling_frequency_index;	4	*uimsbf*
num_front_channel_elements;	4	*uimsbf*
num_side_channel_elements;	4	*uimsbf*
num_back_channel_elements;	4	*uimsbf*
num_lfe_channel_elements;	2	*uimsbf*
num_assoc_data_elements;	3	*uimsbf*
num_valid_cc_elements;	4	*uimsbf*
mono_mixdown_present;	1	*uimsbf*
if (mono_mixdown_present == 1 *)*		
mono_mixdown_element_number;	4	*uimsbf*
stereo_mixdown_present;	1	*uimsbf*
if (stereo_mixdown_present == 1 *)*		
stereo_mixdown_element_number;	4	*uimsbf*
matrix_mixdown_idx_present;	1	*uimsbf*
if (matrix_mixdown_idx_present == 1 *) {*		
matrix_mixdown_idx ;	2	*uimsbf*
pseudo_surround_enable;	1	*uimsbf*
}		
for (i = 0*; i* < *num_front_channel_elements; i*++*) {*		
front_element_is_cpe[i];	1	*bslbf*
front_element_tag_select[i];	4	*uimsbf*
}		
for (i = 0*; i* < *num_front_channel_elements; i*++*) {*		
side_element_is_cpe[i];	1	*bslbf*
side_element_tag_select[i];	4	*uimsbf*
}		
for (i = 0*; i* < *num_back_channel_elements; i*++*) {*		
back_element_is_cpe[i];	1	*bslbf*
back_element_tag_select[i];	4	*uimsbf*
}		
for (i = 0*; i* < *num_lfe_channel_elements; i*++*)*		

Окончание таблицы 2

Синтаксис	Количество битов	Мнемоника
lfe_element_tag_select[i]; *for (i = 0; i < num_assoc_data_elements; i++)* *assoc_data_element_tag_select[i];* *for (i = 0; i < num_valid_cc_elements; i++) {* *cc_element_is_ind_sw[i];* *valid_cc_element_tag_select[i]; }* *byte_alignment();* *comment_field_bytes;* *for (i = 0; i < comment_field_bytes; i++)* *comment_field_data[i]; }*	4 4 1 4 8 8	*uimsbf* *uimsbf* *uimsbf* *uimsbf* Примечание *uimsbf* *uimsbf*
Примечание — Если вызов поступил из *AudioSpecificConfig* (), этот *byte_alignment* должен относиться к запуску *AudioSpecificConfig* ().		

4.2 Полезные нагрузки потока битов *GA*

4.2.1 Полезные нагрузки для аудио объекта типа основное *AAC, AAC SSR, LC AAC* и *AAC LTP*

Таблица 3 — Синтаксис высокоуровневой полезной нагрузки для аудио объектов типов основное *AAC*, *SSR*, *LC* и *LTP* *(raw_data_block())*

Синтаксис	Количество битов	Мнемоника
raw_data_block() *{* *while((id = id_syn_ele) != ID_END){* *switch (id) {* *case ID_SCE: single_channel_element();* *break;* *case ID_CPE: channel_pair_element();* *break;* *case ID_CCE: coupling_channel_element();* *break;* *case ID_LFE: lfe_channel_element();* *break;* *case ID_DSE: data_stream_element();* *break;* *case ID_PCE: program_config_element();* *break;* *case ID_FIL: fill_element();* *}* *}* *byte_alignment();* *}*	3	*uimsbf*

Таблица 4 — Синтаксис *single_channel_element ()*

Синтаксис	Количество битов	Мнемоника
single_channel_element() *{* *element_instance_tag;* *individual_channel_stream*(0,0); *}*	4	*uimsbf*

Таблица 5 — Синтаксис *channel_pair_element* ()

Синтаксис	Количество битов	Мнемоника
channel_pair_element()		
{		
element_instance_tag;	4	*uimsbf*
common_window;	1	*uimsbf*
if (common_window) {		
ics_info();		
ms_mask_present;	2	*uimsbf*
if (ms_mask_present == 1) {		
for (g = 0; g < num_window_groups; g++) {		
for (sfb = 0; sfb < max_sfb; sfb++) {		
ms_used[g][sfb];	1	*uimsbf*
}		
}		
}		
} individual_channel_stream(common_window,0);		
individual_channel_stream(common_window,0); }		

Таблица 6 — Синтаксис *ics_info ()*

Синтаксис	Количество битов	Мнемоника
ics_info()		
ics_reserved_bit;	1	*bslbf*
window_sequence;	2	*uimsbf*
window_shape;		*uimsbf*
if (window_sequence == EIGHT_SHORT_SEQUENCE) {	1	
max_sfb;	4	*uimsbf*
scale_factor_grouping;	7	*uimsbf*
}		
max_sfb;	6	*uimsbf*
predictor_data_present;	1	*uimsbf*
if (predictor_data_present) {		
predictor_reset;	1	*uimsbf*
if (predictor_reset) {		
predictor_reset_group_number;	5	*uimsbf*
}		
for (sfb = 0; sfb < min (max_sfb,		
PRED_SFB_MAX); sfb++) {		
prediction_used[sfb];	1	*uimsbf*
}		
}		
ltp_data_present;	1	*uimsbf*
if (ltp_data_present) {		
ltp_data();		
ltp_data_present;	1	*uimsbf*
if (ltp_data_present) {		
ltp_data();		
}		
}		
}		
}		
}		
}		

Таблица 7 — Синтаксис *pulse_data ()*

Синтаксис	Количество битов	Мнемоника
pulse_data() {		
number_pulse;	2	*uimsbf*
pulse_start_sfb;	6	*uimsbf*
for (i = 0; i < number_pulse+1; i++) {		
pulse_offset[i];	5	*uimsbf*
pulse_amp[i];	4	*uimsbf*
}		

Таблица 8 — Синтаксис *coupling_channel_element ()*

Синтаксис	Количество битов	Мнемоника
coupling_channel_element()		
{		
element_instance_tag;	4	*uimsbf*
ind_sw_cce_flag;	1	*uimsbf*
num_coupled_elements;	3	*uimsbf*
num_gain_element_lists = 0;		
for (c = 0; c < num_coupled_elements+1; c++) {		
num_gain_element_lists++;		
cc_target_is_cpe[c];	1	*uimsbf*
cc_target_tag_select[c];	4	*uimsbf*
if (cc_target_is_cpe[c]) {		
cc_l[c];	1	*uimsbf*
cc_r[c];	1	*uimsbf*
if (cc_l[c] && cc_r[c])		
num_gain_element_lists++;		
}		
}		
cc_domain;	1	*uimsbf*
gain_element_sign;	1	*uimsbf*
gain_element_scale;	2	*uimsbf*
individual_channel_stream(0,0);		
for (c=1; c<num_gain_element_lists; c++) {		
if (ind_sw_cce_flag) {		
cge = 1;		
} else {		
common_gain_element_present[c];	1	*uimsbf*
cge = common_gain_element_present[c];		
}		
if (cge)		
hcod_sf[common_gain_element[c]];	1…19	*Viclbf*
else {		
for (g = 0; g < num_window_groups; g++) { for		
for (sfb=0; sfb<max_sfb; sfb++) {		
if (sfb_cb[g][sfb] != ZERO_HCB)		
hcod_sf[dpcm_gain_element[c][g][sfb]];	1…19	*Viclbf*
}		
}		
}		
}		
}		

Таблица 9 — Синтаксис *lfe_channel_element ()*

Синтаксис	Количество битов	Мнемоника
lfe channel element() *{* *element_instance_tag;* *individual_channel_stream*(0,0); *}*	*4*	*uimsbf*

Таблица 10 — Синтаксис *data_stream_element ()*

Синтаксис	Количество битов	Мнемоника
data_stream_element() *{* *element_instance_tag;* *data_byte_align_flag;* *cnt = count;* *if (cnt == 255)* *cnt += esc_count;* *if (data_byte_align_flag)* *byte_alignment();* *for (i = 0; i < cnt; i++)* *data_stream_byte[element_instance_tag][i];* *}*	4 1 8 8 8	*uimsb* *uimsb* *uimsb* *uimsb* *uimsb*

Таблица 11 — Синтаксис *fill_element ()*

Синтаксис	Количество битов	Мнемоника
Fill_element() *{* *cnt = count;* *if (cnt == 15)* *cnt += esc_count – 1;* *while (cnt > 0) {*	*4* *8*	*uimsbf* *uimsbf*

Таблица 12 — Синтаксис *gain_control_data ()*

Синтаксис	Количество битов	Мнемоника
gain_control_data() *{* *max_band;* *if (window_sequence == ONLY_LONG_SEQUENCE) {* *for (bd = 1; bd <= max_band; bd++) {* *for (wd = 0; wd < 1; wd++) {* *adjust_num[bd][wd];* *for (ad = 0; ad < adjust_num[bd][wd]; ad++) {* *alevcode[bd][wd][ad];* *aloccode[bd][wd][ad];* *}* *}* *}* *}*	2 3 4 5	*uimsbf* *uimsbf* *uimsbf* *uimsbf*

Окончание таблицы 12

Синтаксис	Количество битов	Мнемоника
else if (window_sequence == LONG_START_SEQUENCE) { *for (bd = 1; bd <= max_band; bd++) {* *for (wd = 0; wd < 2; wd++) {* *adjust_num[bd][wd];* *for (ad = 0; ad < adjust_num[bd][wd]; ad++) {* *alevcode[bd][wd][ad];* *if (wd == 0)* *aloccode[bd][wd][ad];* *else* *aloccode[bd][wd][ad];* *}* *}* *}* *}* *else if (window_sequence == EIGHT_SHORT_SEQUENCE) {* *for (bd = 1; bd <= max_band; bd++) {* *for(wd = 0; wd < 8; wd++) {* *adjust_num[bd][wd];* *for (ad = 0; ad < adjust_num[bd][wd]; ad++) {* *alevcode[bd][wd][ad];* *aloccode[bd][wd][ad];* *}* *}* *}* *}* *else if (window_sequence == LONG_STOP_SEQUENCE) {* *for (bd = 1; bd <= max_band; bd++) {* *for (wd = 0; wd < 2; wd++) {* *adjust_num[bd][wd];* *for (ad = 0; ad < adjust_num[bd][wd]; ad++) {* *alevcode[bd][wd][ad];* *if (wd == 0)* *aloccode[bd][wd][ad];* *else* *aloccode[bd][wd][ad];* *}* *}* *}* *}* *}*	 3 4 4 2 3 4 2 2 3 4 4 5	 *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf*

4.2.2 Полезные нагрузки для аудио объектного типа масштабируемые *AAC*

Т а б л и ц а 13 — Синтаксис высокоуровневой полезной нагрузки *ASME* для аудио объектного типа масштабируемой *AAC (aac_scalable_main_element)*

Синтаксис	Количество битов	Мнемоника
aac_scalable_main_element() *{* *aac_scalable_main_header();* *for (ch=0; ch<(this_layer_stereo ? 2:1); ch++){* *individual_channel_stream(1, 1);* *}* *cnt = bits_to_decode() / 8;* *while (cnt >= 1) {* *cnt -= extension_payload(cnt);*		Примечание

Окончание таблицы 13

Синтаксис	Количество битов	Мнемоника

}
byte_alignment();
}		

Примечание — Для *ER AAC AOTs* высокоуровневая полезная нагрузка, описанная выше, обрабатывается как полезная нагрузка логического потока битов. Чтобы получить эту полезную нагрузку логического потока битов из полезной нагрузки физического потока битов требуются шаги предварительной обработки.		

Таблица 14 — Синтаксис высокоуровневой полезной нагрузки *ASEE* для аудио объектного типа масштабируемой *AAC (aac_scalable_extension_element)*

Синтаксис	Количество битов	Мнемоника

aac_scalable_main_element()
{
aac_scalable_main_header();
for (ch=0; ch<(this_layer_stereo ? 2:1); ch++){
individual_channel_stream(1, 1);
}
cnt = bits_to_decode() / 8;
while (cnt >= 1) {
cnt -= extension_payload(cnt);
}
byte_alignment();
}
		Примечание

Примечание — Для *ER AAC AOTs* высокоуровневая полезная нагрузка, описанная выше, обрабатывается как полезная нагрузка логического потока битов. Чтобы получить эту полезную нагрузку логического потока битов из полезной нагрузки физического потока битов требуются шаги предварительной обработки.		

Таблица 15 — Синтаксис *aac_scalable_main_header ()*

Синтаксис	Количество битов	Мнемоника
aac_scalable_main_header() {		
ics_reserved_bit;	1	bslbf
If (t+vq_layer_present == 0) {		
window_sequence;	2	uimsbf
window_shape;	1	uimsbf
} if (window_sequence == EIGHT_SHORT_SEQUENCE) {		
max_sfb;	4	uimsbf
scale_factor_grouping;	7	uimsbf
} else {		
max_sfb;	6	uimsbf
} if (this_layer_stereo) {		
ms_mask_present;	2	bslbf
if (ms_mask_present == 1) {		
ms_data();		
} }		

Окончание таблицы 15

Синтаксис	Количество битов	Мнемоника
if (mono_stereo_flag && (core_flag *|| (tvq_layer_present && tvq_mono_tns == 0)))* *tns_channel_mono_layer;* *for (ch = 0; ch < (this_layer_stereo ? 2:1); ch++) {* *if (!tvq_layer_present || (tns_aac_tvq_en[ch] == 1)) {* *tns_data_present;* *if (tns_data_present)* *tns_data();* *}* *if (core_flag || tvq_layer_present) {* *if ((ch == 0) || ((ch == 1) && (core_stereo ||* *diff_control_data();* *if (mono_stereo_flag)* *diff_control_data_lr();* *} else {* *ltp_data_present;* *if (ltp_data_present) {* *tp_data ();* *}* *}* *}*	1 1 1	*bslbf* *bslbf* *bslbf*

Таблица 16 — Синтаксис *aac_scalable_extension_header ()*

Синтаксис	Количество битов	Мнемоника
aac scalable extension header() *{* *if (window_sequence == EIGHT_SHORT_SEQUENCE) {* *max_sfb;* *} else {* *max_sfb;* *}* *if (this_layer_stereo) {* *ms_mask_present;* *If (ms_mask_present == 1) {* *ms_data0; }* *}* *if (mono_stereo_flag) {* *for (ch = 0; ch < 2; ch++) {* *tns_data_present;* *if (tns_data_present)* *tns_data0; }* *}* *if ((mono_layer_flag) && (this_layer_stereo)) {* *for (ch = 0; ch < 2; ch++) {* *diff_control_data_lr();* *}* *}* *}*	 4 6 2 1 	 *bslbf* *bslbf* *bslbf* *bslbf*

Таблица 17 — Синтаксис *diff_control_data ()*

Синтаксис	Количество битов	Мнемоника
diff_control_data() *{* *If (window_sequence == EIGHT_SHORT_SEQUENCE)* *for (win = 0; win < 8; w++)* *diff_control[win][0];* *else* *for (dc_group=0; dc_group<no_of_dc_groups;* *dc_group++);* *diff_control[0][dc_group];* *}*	1 2 ...5	*bslbf* *bslbf*

Таблица 18 — Синтаксис *diff_control_data_lr ()*

Синтаксис	Количество битов	Мнемоника
diff_control_data_lr() { *if (window_sequence != EIGHT_SHORT_SEQUENCE) {* *for (sfb = last_max_sfb_mc;* *sfb < min(last_mono_max_sfb;max_sfb); sfb++)* *if (!ms_used[0][sfb])* *diff_control_lr[0][sfb* *} else {* *if (last_max_sfb_ms == 0) /* only in the first stereo layer*/* *for (win = 0; win < 8; win++)* *diff_control_lr[win][0* *}* *}*	1 2 ...5	*bslbf* *bslbf*

4.2.3 Полезные нагрузки для аудио объектных типов *ER AAC LC*, *ER AAC LTP* и *ER AAC LD*

Таблица 19 — Синтаксис высокоуровневой полезной нагрузки для аудио объектных типов *ER AAC LC*, *ER AAC LTP* и *ER AAC LD (er_raw_data_block)*

Синтаксис	Количество битов	Мнемоника
er_raw_data_block() *{* *if (channelConfiguration == 0) {* */* reserved */* *if (channelConfiguration == 1) {* *single_channel_element();* *if (channelConfiguration == 2) {* *channel_pair_element();* *if (channelConfiguration == 3) {* *single_channel_element(); channel_pair_element();* *if (channelConfiguration == 4) {* *single_channel_element();* *channel_pair_element();* *single_channel_element();* *}* *if (channelConfiguration == 5) {* *single_channel_element();* *channel_pair_element();* *channel_pair_element();*		Примечание

Окончание таблицы 19

Синтаксис	Количество битов	Мнемоника
} *if (channelConfiguration == 6) {* *single_channel_element();* *channel_pair_element();* *channel_pair_element();* *lfe_channel_element();* *}* *if (channelConfiguration == 7) {* *single_channel_element();* *hannel_pair_element();* *channel_pair_element();* *channel_pair_element();* *lfe_channel_element();* */* reserved */* *}* *}* *if (channelConfiguration >= 8) {* *cnt = bits_to_decode() / 8;* *byte_alignment();* *}* *while (cnt >= 1) {*		
Примечание — Высокоуровневая полезная нагрузка, описанная выше, обрабатывается как полезная нагрузка логического потока битов. Чтобы получить эту полезную нагрузку логического потока битов из полезной нагрузки физического потока битов требуются шаги предварительной обработки.		

4.2.4 Полезные нагрузки для аудио объектного типа *Twin_VQ*

Таблица 20 — Синтаксис высокоуровневой полезной нагрузки *TSME* для аудио объектного типа *Twin_VQ (tvq_scalable_main_element)*

Синтаксис	Количество битов	Мнемоника
tvq_scalable_main_element() *{* *tvq_scalable_main_header();* *vq_single_element*(0); *}* *}*		

Таблица 21 — Синтаксис высокоуровневой полезной нагрузки *TSEE* для аудио объектного типа *Twin_VQ (tvq_scalable_extension_element)*

Синтаксис	Количество битов	Мнемоника
tvq_scalable_extension_element() *{* *tvq_scalable_extension_header();* *vq_single_element(lay);* *}*		

Таблица 22 — Синтаксис *tvq_scalable_main_header ()*

Синтаксис	Количество битов	Мнемоника
tvq_scalable_main_header() *{* *window_sequence;* *window_shape;* *if (this_layer_stereo) {* *ms_mask_present;* *if (ms_mask_present == 1) {* *if (window_sequence ==* *EIGHT_SHORT_SEQUENCE)* *scale_factor_grouping;* *for(ch = 0; ch < (this_layer_stereo ? 2:1); ch++) {* *ltp_data_present;* *if (ltp_data_present)* *ltp_data ();* *tns_data_present;* *if (tns_data_present)* *tns_data();* *}* *}*	 2 1 2 7 1 1 	 *bslbf* *bslbf* *bslbf* *bslbf* *bslbf* *bslbf*

Таблица 23 — Синтаксис *tvq_scalable_extension_header ()*

Синтаксис	Количество битов	Мнемоника
tvq_scalable_extension_header() *{* *if (this_layer_stereo) {* *ms_mask_present;* *if (ms_mask_present == 1) {* *ms_data();* *}* *}* *}*	 2 	 *bslbf*

Таблица 24 — Синтаксис *vq_single_element*

Синтаксис	Количество битов	Мнемоника
vq_single_element(lyr) *{* *if (lyr == 0);* *bandlimitpresent* *if (window_sequence != EIGHT_SHORT_SEQUENCE* *&& lyr == 0) {* *ppc_present;* *postprocess _present;* *}* *if (lyr >= 1)* *for (i_ch = 0; i_ch < n_ch; i_ch++) {* *fb_shift[i_ch];* *}* *if (lyr == 0 && bandlimit_present) {* *for (i_ch = 0; i_ch < n_ch; i_ch++) {* *index_blim_h[i_ch];* *index_blim_l[i_ch];*	 1 1 1 2 2 1	 *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf*

Окончание таблицы 24

Синтаксис	Количество битов	Мнемоника
}		
if (ppc_present) {		
for (idiv = 0; idiv < N_DIV_P; idiv++) {		
index_shape0_p[idiv];	7	
index_shape1_p[idiv];	7	
}		
for (i_ch = 0; i_ch < n_ch; i_ch++) {		
index_pit[i_ch];	8	
index_pgain[i_ch];	7	
}		
}		
for (idiv = 0; idiv < N_DIV; idiv++) {		
index_shape0[idiv];	5/6	
index_shape1[idiv];	5/6	
}		
for (i_ch = 0; i_ch < n_ch; i_ch++) {		
for (isb = 0; isb < N_SF; isb++) {		
for (ifdiv = 0; ifdiv < FW_N_DIV; ifdiv++) {		
index_env[i_ch][isb][ifdiv];	0,6	
}		
}		
}		
for (i_ch = 0; i_ch < n_ch; i_ch++) {		
for (isbm = 0; isbm < N_SF; isbm++){		
index_fw_alf[i_ch][isbm];	0,1	
}		
}		
for (i_ch = 0; i_ch < n_ch; i_ch++){		
index_gain[i_ch]	8...9	
if (N_SF[b_type] > 1){		
for (isbm = 0; isbm < N_SF[b_type]; isbm++) {		
index_gain_sb[i_ch][isbm]	4	
}		
}		
}		
for (i_ch = 0; i_ch < n_ch; i_ch++) {		
index_lsp0[i_ch]	1	
index_lsp1[i_ch]	6	
for (isplt = 0; isplt < LSP_SPLIT; isplit++) {		
index_lsp2[i_ch][isplt]	4	
}		
}		
}		

4.2.5 Полезные нагрузки для аудио объектного типа *ER TwinVQ*

Таблица 25 — Синтаксис объектного типа *ER TwinVQ* (базовый)

Синтаксис	Количество битов	Мнемоника
tvq_scalable_main_element() *{* *Error_Sensitivity_Category1();* *Error_Sensitivity_Category2();* *}*		

Таблица 26 — Синтаксис объектного типа *ER TwinVQ* (расширенный)

Синтаксис	Количество битов	Мнемоника
tvq_scalable_main_element() *{* *Error_Sensitivity_Category3();* *Error_Sensitivity_Category4();* *}*		

Таблица 27 — Синтаксис *Error_Sensitivity_Category1 ()*

Синтаксис	Количество битов	Мнемоника
Error_Sensitivity_Category1() *{* *window_sequence;* *window_shape;* *if (this_layer_stereo) {* *ms_mask_present;* *if (ms_mask_present == 1) {* *if (window_sequence ==* *EIGHT_SHORT_SEQUENCE)* *scale_factor_grouping;* *ms_data();* *}* *}* *for (ch = 0; ch < (this_layer_stereo ? 2:1); ch++) {* *ltp_data_present;* *if (ltp_data_present)* *ltp_data ();* *tns_data_present;* *if (tns_data_present)* *tns_data();* *}* *Bandlimit present;* *if (window_sequence != EIGHT_SHORT_SEQUENCE) {* *ppc_present;* *postprocess_present;* *}* *if (bandlimit_present)* *for (i_ch = 0; i_ch < n_ch; i_ch++) {* *index_blim_h[i_ch];* *index_blim_l[i_ch];* *}* *}* *if (ppc_present) {* *for (idiv = 0; idiv < N_DIV_P; idiv++) {* *index_shape0_p[idiv];* *index_shape1_p[idiv];* *}* *for (i_ch = 0; i_ch < n_ch; i_ch++) {* *index_pit[i_ch];* *index_pgain[i_ch];* *}* *}* *for (i_ch = 0; i_ch < n_ch; i_ch++) {* *index_gain[i_ch];*	 2 1 2 7 1 1 1 1 1 2 1 7 8 7 9	 *bslbf* *bslbf* *bslbf* *bslbf* *bslbf* *bslbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf*

Окончание таблицы 27

Синтаксис	Количество битов	Мнемоника
if (N_SF[b_type] > 1) { *for (isbm = 0; isbm < N_SF[b_type]; isbm++) {* *index_gain_sb[i_ch][isbm];* *}* *}* *}*	4	*uimsbf*
for (i_ch = 0; i_ch < n_ch; i_ch++) { *index_lsp0[i_ch];* *index_lsp1[i_ch];* *for (isplt = 0; isplt < LSP_SPLIT; isplit++) {* *index_lsp2[i_ch][isplt];* *}* *}*	1 6 4	*uimsbf* *uimsbf* *uimsbf*
for (i_ch = 0; i_ch < n_ch; i_ch++) { *for (isb = 0; isb < N_SF; isb++) {* *for (ifdiv = 0; ifdiv < FW_N_DIV; ifdiv++) {* *index_env[i_ch][isb][ifdiv];* *}* *}* *}*	0,6	*uimsbf*
for (i_ch = 0; i_ch < n_ch; i_ch++) { *for (isbm = 0; isbm < N_SF; isbm++) {* *index_fw_alf[i_ch][isbm];* *}* *}* *}*	0,1	*uimsbf*

Т а б л и ц а 28 — Синтаксис *Error_Sensitivity_Category* 2 *()*

Синтаксис	Количество битов	Мнемоника
Error_Sensitivity_Category 2*()* *{* *for (idiv = 0; idiv < N_DIV; idiv++) {* *index_shape*0*[idiv];* *index_shape*1*[idiv];* *}* *}*	5/6 5/6	*uimsbf* *uimsbf*

Т а б л и ц а 29 — Синтаксис *Error_Sensitivity_Category* 3 *()*

Синтаксис	Количество битов	Мнемоника
Error_Sensitivity_Category 3*() {* *if (this_layer_stereo) {* *ms_mask_present;* *if (ms_mask_present == 1) {* *ms_data();* *}* *}*	2	*bslbf*
for (i_ch = 0; i_ch < n_ch; i_ch++) { *fb_shift[i_ch];* *}*	2	*uimsbf*
for (i_ch = 0; i_ch < n_ch; i_ch++) { *index_gain[i_ch];*	8	*uimsbf*

Окончание таблицы 29

Синтаксис	Количество битов	Мнемоника
if (N_SF[b_type] > 1) { *for (isbm = 0; isbm < N_SF[b_type]; isbm++) {* *index_gain_sb[i_ch][isbm];* *}* *}* *}* *for (i_ch = 0; i_ch < n_ch; i_ch++) {* *for (isb = 0; isb < N_SF; isb++) {* *for (ifdiv = 0; ifdiv < FW_N_DIV; ifdiv++) {* *index_env[i_ch][isb][ifdiv];* *}* *}* *}* *for (i_ch = 0; i_ch < n_ch; i_ch++) {* *for (isbm = 0; isbm < N_SF; isbm++) {* *index_fw_alf[i_ch][isbm];* *}* *}* *}*	4 0,6 0,1	*uimsbf* *uimsbf* *uimsbf*

Таблица 30 — Синтаксис *Error_Sensitivity_Category4 ()*

Синтаксис	Количество битов	Мнемоника
Error_Sensitivity_Category4() *{* *for (idiv = 0; idiv < N_DIV; idiv++) {* *index_shape0[idiv];* *index_shape1[idiv];* *}*	 5/6 5/6	 *uimsbf* *uimsbf*

4.2.6 Полезные нагрузки для аудио объектного типа *ER BSAC*

Таблица 31 — Синтаксис высокоуровневой полезной нагрузки для аудио объектного типа *ER BSAC (bsac_payload())*

Синтаксис	Количество битов	Мнемоника
bsac_payload(lay) *{* *for (frm = 0; frm < numOfSubFrame; frm++) {* *bsac_lstep_element(frm, lay);* *}* *}*		

Таблица 32 — Синтаксис *bsac_lstep_element ()*

Синтаксис	Количество битов	Мнемоника
bsac_lstep_element(frm, lay) *{* *offset = LayerStartByte[frm][lay];* *for(i = 0;i < LayerLength[frm][lay]; i++)* *bsac_stream_byte[frm][offset+i];* *}*	8	*uimsbf*

Таблица 33 — Синтаксис *bsac_raw_data_block ()*

Синтаксис	Количество битов	Мнемоника
bsac raw data block() *{* *bsac_base_element();* *layer=slayer_size;* *while(data_available() && layer<(top_layer+slayer_size))* *{ bsac_layer_element(layer);* *layer++;* *}* *byte_alignment();* *if (data_available()) {* *zero_code* *sync_word* *while(bits_to_decode() > 4) {* *extension_type* *switch(extension_type) {* *case EXT_BSAC_CHANNEL :* *extended_bsac_raw_data_block();* *break;* *case EXT_BSAC_SBR_DATA :* *extended_bsac_sbr_data(nch, 0);* *break;* *case EXT_BSAC_SBR_DATA_CRC :* *extended_bsac_sbr_data(nch, 1);* *break;* *case EXT_BSAC_CHANNEL_SBR :* *extended_bsac_raw_data_block();* *extended_bsac_sbr_data(nch, 0);* *break;* *case EXT_BSAC_CHANNEL_SBR_CRC :* *extended_bsac_raw_data_block();* *extended_bsac_sbr_data(nch, 1);* *break;* *case EXT_BSAC_SAC_DATA :* *extended_bsac_sac_data();* *break;* *default :* *extended_bsac_data();* *break;* *}* *byte_alignment();* *}* *}* *}*	 32 4 4 <

Окончание таблицы 34

Синтаксис	Количество битов	Мнемоника
byte_alignment(); *for (slayer = 0; slayer < slayer_size; slayer++)* *bsac_layer_element(slayer);* *}*		

Таблица 35 — Синтаксис *bsac_header ()*

Синтаксис	Количество битов	Мнемоника
bsac_ header() *{* *header_length;* *sba_mode;* *top_layer;* *base_snf_thr;* *for (ch = 0;ch < nch; ch++)* *max_scalefactor[ch];* *base_band;* *for(ch = 0;ch < nch; ch++) {* *cband_si_type[ch];* *base_scf_model[ch];* *enh_scf_model[ch];* *max_sfb_si_len[ch];* *}* *}*	 4 1 6 2 8 5 5 3 3 4	 *uimbf* *uimbf* *uimbf* *uimbf* *uimbf* *uimbf* *uimbf* *uimbf* *uimbf* *uimbf*

Таблица 36 — Синтаксис *general_header ()*

Синтаксис	Количество битов	Мнемоника
general_header() *{* *reserved_bit;* *window_sequence;* *window_shape;* *if (window_sequence == EIGHT_SHORT_SEQUENCE) {* *max_sfb;* *scale_factor_grouping;* *} else {* *max_sfb;* *}* *pns_data_present;* *if (pns_data_present)* *pns_start_sfb;* *if (nch == 2)* *ms_mask_present;* *for (ch = 0 ch < nch; ch++) {* *tns_data_present[ch];* *if (tns_data_present[ch])* *tns_data();*	 1 2 1 4 7 6 1 6 2 1	 *bslbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimbf* *uimbf* *bslbf* *bslbf*

Окончание таблицы 36

Синтаксис	Количество битов	Мнемоника
ltp_data_present[ch]; *if (ltp_data_present[ch])* *ltp_data(last_max_sfb, max_sfb);* *}* *}*	1	*bslbf*

Таблица 37 — Синтаксис *bsac_layer_element ()*

Синтаксис	Количество битов	Мнемоника
bsac_layer_element(layer) *{* *layer_cband_si(layer);* *layer_sfb_si(layer);* *bsac_layer_spectra (layer);* *if (!sba_mode) {* *bsac_lower_spectra (layer);* *}* *else if (terminal_layer[layer]) {* *bsac_lower_spectra (layer);* *bsac_higher_spectra (layer);* *}*		

Таблица 38 — Синтаксис *layer_cband_si ()*

Синтаксис	Количество битов	Мнемоника
layer_cband_si(layer) *{* *g = layer_group[layer];* *for (ch = 0; ch < nch; ch++) {* *for (cband = layer_start_cband[layer];* *cband < layer_end_cband[layer]; cband++) {* *acode_cband_si[ch][g][cband];* *}* *}* *}*	1..14	*bslbf*

Таблица 39 — Синтаксис *layer_sfb_si ()*

Синтаксис	Количество битов	Мнемоника
layer_sfb_si (layer) *{* *g = layer_group[layer];* *for (ch = 0; ch < nch; ch++)* *for (sfb = layer_start_sfb[layer];sfb < layer_end_sfb[layer];* *sfb++) {* *if (nch == 1) {* *if (pns_data_present && sfb >= pns_start_sfb) {* *acode_noise_flag[g][sfb];*	1	*bslbf*

Окончание таблицы 39

Синтаксис	Количество битов	Мнемоника
}		
} else if (stereo_side_info_coded[g][sfb] == 0) {		
if (ms_mask_present !=2) {		
if (ms_mask_present == 1) {		
acode_ms_used[g][sfb];	0…2	*bslbf*
} else if (ms_mask_present == 3) {		
acode_stereo_info[g][sfb];	0…4	*bslbf*
}		
if (pns_data_present && sfb >= pns_start_sfb) {		
acode_noise_flag_l[g][sfb];	1	*bslbf*
acode_noise_flag_r[g][sfb];	1	*bslbf*
if (ms_mask_present == 3 && stereo_info == 3) {		
if (noise_flag_l && noise_flag_r){		
acode_noise_mode[g][sfb];	2	*bslbf*
}		
}		
}		
}		
stereo_side_info_coded[g][sfb] = 1;		
}		
if (noise_flag[ch][g][sfb]) {		
if (noise_pcm_flag[ch] == 1) {		
acode_max_noise_energy[ch];	9	*bslbf*
noise_pcm_flag[ch] = 0;		
}		
acode_dpcm_noise_energy_index[ch][g][sfb];	0...14	*bslbf*
} else if (stereo_info[g][sfb] >=2 && ch == 1) {		
acode_is_position_index[g][sfb];	0...14	*bslbf*
} else {		
acode_scf_index[ch][g][sfb];	0...14	*bslbf*
}		
}		
}		

Т а б л и ц а 40 — Синтаксис *bsac_layer_spectra ()*

Синтаксис	Количество битов	Мнемоника
bsac_layer_spectra(layer)		
{		
g = layer_group[layer];		
start_index[g] = layer_start_ index[layer];		
end_index[g] = layer_end_ index[layer];		
if (layer < slayer_size)		
thr_snf = base_snf_thr;		
else		
thr_snf = 0;		
bsac_spectral_data (g, g+1, thr_snf, cur_snf);		
}		

Таблица 41 — Синтаксис *bsac_lower_spectra ()*

Синтаксис	Количество битов	Мнемоника
bsac_lower_spectra(layer) *{* *for (g = 0; g < num_window_groups; g++) {* *start_index[g] = 0;* *end_index[g] = 0;* *}* *for (play = 0; play < layer; play++) {* *end_index[layer_group[play]] = layer_end_index[play];* *}* *bsac_spectral_data (0, num_window_groups, 0, unc_snf);* *}*		

Таблица 42 — Синтаксис *bsac_higher_spectra ()*

Синтаксис	Количество битов	Мнемоника
bsac_higher_spectra(layer) *{* *for (nlay = layer+1; nlay < top_layer+slayer_size; nlay++) {* *g = layer_group[nlay];* *start_index[g] = layer_start_index[nlay];* *end_index[g] = layer_end_index[nlay];* *bsac_spectral_data (g, g+1, 0, unc_snf);* *}* *}*		

Таблица 43 — Синтаксис *bsac_spectral_data ()*

Синтаксис	Количество битов	Мнемоника
bsac_spectral_data(start_g, end_g, thr_snf, cur_snf) *{* *if (!layer_data_available()) return;* *for (snf = maxsnf; snf > thr_snf; snf- -)* *for (g = start_g; g < end_g; g++)* *for (i = start_index[g]; i < end_index[g]; i++)* *for (ch = 0; ch < nch; ch++) {* *if (cur_snf[ch][g][i] < snf) continue;* *if (!sample[ch][g][i] || sign_is_coded[ch] [g][i])* *acod_sliced_bit[ch][g][i][snf];*	0...6	*bslbf*
if (sample[ch][g][i] && !sign_is_coded[ch] [g][i]) { *if (layer_data_available()) return;* *acod_sign[ch][g][i];* *sign_is_coded[ch][g][i] = 1;* *}* *cur_snf[ch][g][i]- -;* *if (layer_data_available()) return;* *}* *}*	1	*bslbf*

Таблица 44 — Синтаксис *extended_bsac_raw_data_block ()*

Синтаксис	Количество битов	Мнемоника
extended bsac raw data block() *{* *extended_bsac_base_element();* *layer=slayer_size;* *while(data_available() && layer<(top_layer+slayer_size)) {* *bsac_layer_element(layer);* *layer++;* *}* *byte_alignment();* *}*		

Таблица 45 — Синтаксис *extended_bsac_base_element ()*

Синтаксис	Количество битов	Мнемоника
extended_bsac_base_element() *{* *element_length* *channel_configuration_index* *reserved_bit* *bsac_header();* *general_header();* *byte_alignment();* *for (slayer = 0; slayer < slayer_size; slayer++)* *bsac_layer_element(slayer);* *}*	 11 3 1	 *uimbf* *uimbf* *uimbf*

Таблица 46 — Синтаксис *extended_bsac_sbr_data ()*

Синтаксис	Количество битов	Мнемоника
extended_bsac_sbr_data(nch, crc_flag) *{* *num_sbr_bits = 0;* *cnt = count;* *num_sbr_bits += 4;* *if (cnt == 15) {* *cnt += esc_count - 1;* *num_sbr_bits += 8;* *}* *if (crc_flag) {* *bs_sbr_crc_bits;* *num_sbr_bits += 10;* *}* *num_sbr_bits += 1; if (bs_header_flag)* *num_sbr_bits += sbr_header();* *num_sbr_bits += bsac_sbr_data(nch, bs_amp_res);* *num_align_bits = (8*cnt - num_sbr_bits);* *bs_fill_bits;* *}*	 4 8 10 1 Число битов выравнивания	 *uimsbf* *uimsbf* *uimsbf* *uimsbf*

Таблица 47 — Синтаксис *bsac_sbr_data ()*

Синтаксис	Количество битов	Мнемоника
bsac_sbr_data(nch, bs_amp_res) *{* *switch (nch) {* *case 1 :* *sbr_single_channel_element(bs_amp_res) break;* *case 2 :* *sbr_channel_pair_element(bs_amp_res) break;* *}* *}*		

Таблица 48 — Синтаксис *extended_bsac_data ()*

Синтаксис	Количество битов	Мнемоника
extended_bsac_data() *{* *cnt = count;* *if (cnt == 255) {* *cnt += esc_count - 1;* *}* *for (i=0 ; i < cnt-1 ; i++) {* *byte_payload* *}* *}*	 8 8 8 	 *uimsbf* *uimsbf* *uimsbf*

Таблица 49 — Синтаксис *extended_bsac_sac_data ()*

Синтаксис	Количество битов	Мнемоника
extended_bsac_sac_data() *{* *cnt = count;* *if (cnt == 255) {* *cnt += esc_count - 1;* *}* *ancType;* *ancStart;* *ancStop;* *bs_crc_flag* *bs_fill_bits* *if (bs_crc_flag) { ancCrcWord;* *cnt = cnt - 1;* *}* *for (i=0; i<cnt-2; i++) {* *ancDataSegmentByte[i];* *}* *}*	 8 8 2 1 1 1 8 8 	 *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *bslbf*

4.2.7 Вспомогательные полезные нагрузки

Таблица 50 — Синтаксис *individual_channel_stream ()*

Синтаксис	Количество битов	Мнемоника
individual_channel_stream(common_window, scale_flag) *{* *globalgain;* *if (! common_window && ! scale_flag) {* *ics_info ();* *}* *section_data ();* *scale_factor_data ();* *if (! scale_flag) {* *pulse_data_present;* *if (pulse_data_present) {* *pulse_data ();* *}* *tns_data_present;* *if (tns_data_present) {* *tns_data ();* *}* *gain_control_data_present;* *if (gain_control_data_present) {* *gain_control_data ();* *}* *}* *if (! aacSpectralDataResilienceFlag) {* *spectral_data ();* *}* *else {* *length_of_reordered_spectral_data;* *length_of_longest_codeword;* *reordered_spectral_data ();* *}* *}*	 8 1 1 1 14 6	 *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf*

Таблица 51 — Синтаксис *reordered_spectral_data ()*

Синтаксис	Количество битов	Мнемоника
reordered_spectral_data () *{* *}*		

Таблица 52 — Синтаксис *section_data ()*

Синтаксис	Количество битов	Мнемоника
section_data() *{* *if (window_sequence == EIGHT_SHORT_SEQUENCE) {* *sect_esc_val = (1 << 3) – 1;* *}* *else {* *sect_esc_val = (1 << 5) – 1; }* *for (g = 0; g < num_window_groups; g++) {* *k = 0;* *i = 0;*		

Окончание таблицы 52

Синтаксис	Количество битов	Мнемоника
while (k < max_sfb) { *if (aacSectionDataResilienceFlag)* *sect_cb[g][i];* *}* *else {* *sect_cb[g][i];* *}* *sect_len = 0;* *if (! aacSectionDataResilienceFlag ||* *sect_cb < 11 || (sect_cb > 11 && sect_cb < 16)) {* *while (sect_len_incr == sect_esc_val) {* *sect_len += sect_esc_val;* *}* *}* *else {* *sect_len_incr = 1;* *}* *sect_len += sect_len_incr;* *sect_start[g][i] = k;* *sect_end[g][i] = k + sect_len;* *for (sfb = k; sfb < k + sect_len; sfb++) {* *sfb_cb[g][sfb] = sect_cb[g][i];* *}* *k += sect_len;* *i++;* *}* *num_sec[g] = i;* *}* *}*	 5 4 3/5	 *uimsbf* *uimsbf* *uimsbf*

Т а б л и ц а 53 — Синтаксис *scale_factor_data ()*

Синтаксис	Количество битов	Мнемоника
scale_factor_data() *{* *if (! aacScalefactorDataResilienceFlag) {* *noise_pcm_flag = 1;* *for (g = 0; g < num_window_groups; g++) {* *for (sfb = 0; sfb < max_sfb; sfb++) {* *if (sfb_cb[g][sfb] != ZERO_HCB) {* *if (is_intensity (g, sfb)) {* *hcod_sf[dpcm_is_position[g][sfb]];* *}* *else {* *if (is_noise(g, sfb)) {* *if (noise_pcm_flag) {* *noise_pcm_flag = 0;* *dpcm_noise_nrg[g][sfb];* *}* *else {* *hcod_sf[dpcm_noise_nrg[g][sfb]];* *}* *}*	 1...19 9 1...19	 *vlclbf* *uimsbf* *vlclbf*

Продолжение таблицы 53

Синтаксис	Количество битов	Мнемоника

else {
hcod_sf[dpcm_sf[g][sfb]];
}
}
}
}
}
}
else {
intensity_used = 0;
noise_used = 0;
sf_concealment;
rev_global_gain;
length_of_rvlc_sf;
for (g = 0; *g* < *num_window_groups; g++) {*
for (sfb=0; *sfb* < *max_sfb; sfb++) {*
if (sfb_cb[g][sfb] != ZERO_HCB) {
if (is_intensity (g, sfb)) {
intensity_used = 1;
rvlc_cod_sf[dpcm_is_position[g][sfb]];
}
else {
if (is_noise(g,sfb)) {
if (! noise_used) {
noise_used = 1;
dpcm_noise_nrg[g][sfb];
}
else {
rvlc_cod_sf[dpcm_noise_nrg[g][sfb]];
}
}
else {
rvlc_cod_sf[dpcm_sf[g][sfb]];
}
}
}
}
if (intensity_used) {
rvlc_cod_sf[dpcm_is_last_position];
}
noise_used = 0;
sf_escapes_present;
if (sf_escapes_present) {
length_of_rvlc_escapes;
for (g = 0; *g* < *num_window_groups; g++) {*
for (sfb = 0; *sfb* < *max_sfb; sfb++) {*
if (sfb_cb[g][sfb] != ZERO_HCB) {
if (is_intensity (g, sfb) &&
rvlc_esc_sf[dpcm_is_position[g][sfb]];
dpcm_is_position[g][sfb] == ESC_FLAG) {
}
else {
if (is_noise (g, sfb) {
if (! noise_used) {
noise_used = 1;
}
	
	1
	8
	11/9
	
	1...9
	
	9
	
	1...9
	
	1...9
	
	1...9
	
	1
	8
	
	2...20
		
		uimsbf
		uimsbf
		uimsbf
		
		vlclbf
		
		uimsbf
		
		vlclbf
		
		vlclbf
		
		vlclbf
		
		uimsbf
		uimsbf
		
		vlclbf

Окончание таблицы 53

Синтаксис	Количество битов	Мнемоника
else { *if (dpcm_noise_nrg[g][sfb] == ESC_FLAG) {* *rvlc_esc_sf[dpcm_noise_nrg[g][sfb]];* *}* *}* *}* *else {* *if (dpcm_sf[g][sfb] == ESC_FLAG) {* *rvlc_esc_sf[dpcm_sf[g][sfb]];* *}* *}* *}* *}* *}* *}* *if (intensity_used &&* *dpcm_is_last_position == ESC_FLAG) {* *rvlc_esc_sf[dpcm_is_last_position];* *}* *}* *if (noise_used) {*	 2...20 2...20	 *vlclbf* *vlclbf*

Т а б л и ц а 54 — Синтаксис *tns_data ()*

Синтаксис	Количество битов	Мнемоника
tns_data() { *for (w = 0; w < num_windows; w++) {* *n_filt[w];* *if (n_filt[w])* *coef_res[w];* *for (filt = 0; filt < n_filt[w]; filt++) {* *length[w][filt];* *order[w][filt];* *if (order[w][filt]) {* *direction[w][filt];* *coef_compress[w][filt];* *for (i = 0; i < order[w][filt]; i++)* *coef[w][filt][i];* *}* *}* *}* *}*	 1...2 1 {4,6} {3,5} 1 1 2...4	 *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf*

Т а б л и ц а 55 — Синтаксис *ltp_data ()*

Синтаксис	Количество битов	Мнемоника
ltp_data() *{* *if (AudioObjectType == ER_AAC_LD) {* *ltp_lag_update;* *if (ltp_lag_update) {* *ltp_lag;*	 1 10	 *uimsbf* *uimsbf*

Окончание таблицы 55

Синтаксис	Количество битов	Мнемоника
} else {		
ltp_lag = ltp_prev_lag;		
}		
ltp_coef;	3	*uimsbf*
for (sfb = 0; sfb < min(max_sfb, MAX_LTP_LONG_SFB); sfb++) {		
ltp_long_used[sfb];	1	*uimsbf*
}		
}		
else {		
ltp_lag;		
ltp_coef;	11	*uimsbf*
if(window_sequence!=EIGHT_SHORT_SEQUENCE) {	3	*uimsbf*
for (sfb=0; sfb<min(max_sfb, MAX_LTP_LONG_SFB); sfb++) {		
ltp_long_used[sfb];	1	*uimsbf*
}		
}		
}		
}		

Таблица 56 — Синтаксис *spectral_data ()*

Синтаксис	Количество битов	Мнемоника
Spectral_data()		
{		
for (g = 0; g < num_window_groups; g++) {		
for (i = 0; i < num_sec[g]; i++) {		
if (sect_cb[g][i] != ZERO_HCB &&		
sect_cb[g][i] != NOISE_HCB &&		
sect_cb[g][i] != INTENSITY_HCB &&		
sect_cb[g][i] != INTENSITY_HCB2) {		
for (k = sect_sfb_offset[g][sect_start[g][i]];		
k< sect_sfb_offset[g][sect_end[g][i]];) {		
if (sect_cb[g][i]<FIRST_PAIR_HCB) {		
hcod[sect_cb[g][i]][w][x][y][z];	1...16	*vlclbf*
if (unsigned_cb[sect_cb[g][i]])		
quad_sign_bits;		
k += QUAD_LEN;	0...4	*bslbf*
}		
else {		
hcod[sect_cb[g][i]][y][z];	0...15	*vlclbf*
if (unsigned_cb[sect_cb[g][i]])	0...2	*bslbf*
pair_sign_bits;		
k += PAIR_LEN;		
if (sect_cb[g][i] == ESC_HCB) {	5...21	*vlclbf*
if (y == ESC_FLAG)	5...21	*vlclbf*
hcod_esc_y;		
}		
if (z == ESC_FLAG)		
hcod_esc_z;		
}		
}		
}		
}		
}		

Таблица 57 — Синтаксис *extension_payload ()*

Синтаксис	Количество битов	Мнемоника
extension_payload(cnt) *{* *extension_type;* *align = 4;* *switch(extension_type) {* *case EXT_DYNAMIC_RANGE:* *return dynamic_range_info();* *case EXT_SAC_DATA:* *return sac_extension_data(cnt);* *case EXT_SBR_DATA:* *return sbr_extension_data(id_aac, 0);* *case EXT_SBR_DATA_CRC:* *return sbr_extension_data(id_aac, 1);* *case EXT_FILL_DATA:* *fill_nibble* *for (i=0; i<cnt-1; i++) {* *fill_byte[i];* *}* *return cnt;* *case EXT_DATA_ELEMENT:* *data_element_version;* *switch(data_element_version) {* *case ANC_DATA:* *loopCounter = 0;* *dataElementLength = 0;* *do {* *dataElementLengthPart;* *dataElementLength += dataElementLengthPart;* *loopCounter++;* *} while (dataElementLengthPart == 255);* *for (i=0; i<dataElementLength; i++) {* *data_element_byte[i];* *}* *return (dataElementLength+loopCounter+1);* *default:* *align = 0;* *}* *case EXT_FIL:* *default:* *for (i=0; i<8*(cnt-1)+align; i++) {* *other_bits[i];* *}* *return cnt;* *}* *}*	 4 4 8 4 8 8 1 	 *uimsbf* Примечание Примечание *uimsbf* *uimsbf* *uimsbf* *uimsbf* <

Окончание таблицы 58

Синтаксис	Количество битов	Мнемоника
pce_tag_present ;	1	
if (pce_tag_present == 1) {		
pce_ instance_tag;	4	*uimsbf*
drc_tag_reserved_bits;	4	*uimsbf*
n++;		
}		
excluded_chns_present;	4	
if (excluded_chns_present == 1) {		
n += excluded_channels();		
}		
drc_bands_present;	1	*uimsbf*
if (drc_bands_present == 1) {		
drc_band_incr;	4	*uimsbf*
drc_interpolation_scheme;	4	*uimsbf*
n++;		
drc_num_bands = drc_num_bands + drc_band_incr;		
for (i = 0; i < drc_num_bands; i++) {		
drc_band_top[i];	8	
n++;		
}		
}		
prog_ref_level_present;		
if (prog_ref_level_present == 1) {		
prog_ref_level;	1	*uimsbf*
prog_ref_level_reserved_bits;		
n++;	7	*uimsbf*
}	1	*uimsbf*
for (i = 0; i < drc_num_bands; I++) {		
dyn_rng_sgn[i];		
dyn_rng_ctl[i];		
n++;	1	*uimsbf*
}	7	*uimsbf*
return n;		
}		

Т а б л и ц а 59 — Синтаксис *excluded_channels ()*

Синтаксис	Количество битов	Мнемоника
Excluded_channels( )		
{		
n = 0;		
num_excl_chan = 7;		
for (i = 0; i < 7; i++)		
exclude_mask[i];	1	*uimsbf*
n++;		
while (additional_excluded_chns[n-1] == 1) {	1	*uimsbf*
for (i = num_excl_chan; i < num_excl_chan+7; i++)		
exclude_mask[I];	1	*uimsbf*
n++;		
num_excl_chan += 7;		
}		
return n;		
}		

Таблица 60 — Синтаксис *ms_data ()*

Синтаксис	Количество битов	Мнемоника
ms_data() *{* *for (g = 0; g < num_window_groups; g++) {* *for (sfb = last_max_sfb_ms; sfb < max_sfb; sfb++) {* *ms_used[g][sfb];* *}* *}* *}*	1	*bslbf*

Таблица 61 — Синтаксис *sac_extension_data ()*

Синтаксис	Количество битов	Мнемоника
sac_extension_data(cnt) *{* *ancType;* *ancStart;* *ancStop;* *for (i=0; i<cnt-1; i++) {* *ancDataSegmentByte[i];* *}* *return (cnt);* *}*	2 1 1 8	*uimsbf* *uimsbf* *uimsbf* *bslbf*

4.2.8 Полезные нагрузки для аудио объектного типа *SBR*

Таблица 62 — Синтаксис *sbr_extension_data ()*

Синтаксис	Количество битов	Мнемоника
sbr_extension_data(id_aac, crc_flag) *{* *num_sbr_bits = 0;* *if (crc_flag) {* *bs_sbr_crc_bits;* *num_sbr_bits += 10;* *}* *if (sbr_layer != SBR_STEREO_ENHANCE) {* *num_sbr_bits += 1;* *if (bs_header_flag)* *num_sbr_bits += sbr_header();* *}* *num_sbr_bits += sbr_data(id_aac, bs_amp_res);* *num_align_bits = (8*cnt - 4 - num_sbr_bits)%8;* *bs_fill_bits;* *return ((num_sbr_bits + num_align_bits + 4) / 8)* *}*	 10 1 Число выравнивания битов	 *uimsbf* Примечание 1 Примечание 2 Примечание 2 *uimsbf*

Примечание 1 — Когда инструмент *SBR* используется с немасштабируемым базовым кодером *AAC*, значение переменной помощника *sbr_layer* является *SBR_NOT_SCALABLE*. Когда инструмент *SBR* используется с масштабируемым базовым кодером *AAC*, значение переменной помощника *sbr_layer* зависит от текущего уровня и конфигурации масштабируемости базового кодера *AAC*.

Примечание 2 — *sbr_header ()* и *sbr_data ()* возвращают число битов чтения (*cnt* является параметром в *extension_payload ()*).

Таблица 63 — Синтаксис *sbr_header ()*

Синтаксис	Количество битов	Мнемоника
sbr_header()		
{		
bs_amp_res;	1	
bs_start_freq;	4	*uimsbf* Примечание 1
bs_stop_freq;	4	*uimsbf* Примечание 1
bs_xover_band;	3	*uimsbf* Примечание 2
bs_reserved;	2	*uimsbf*
bs_header_extra_1;	1	
bs_header_extra_2;	1	
if (bs_header_extra_1) {		Примечание 3
bs_freq_scale;	2	
bs_alter_scale;	1	*uimsbf*
bs_noise_bands;	2	*uimsbf*
}		
if (bs_header_extra_2) {		Примечание 3
bs_limiter_bands;	2	*uimsbf*
bs_limiter_gains;	2	*uimsbf*
bs_interpol_freq;	1	
bs_smoothing_mode	1	
}		
}		

Примечание 1 — *bs_start_freq* и *bs_stop_freq* должны определять полосу частот, которая не превышает пределы, определенные в 6.18.3.6.

Примечание 2 — Индекс к таблице полосы задающей частоты, указывающий, где начинается текущий диапазон *SBR*.

Примечание 3 — Если этот бит не установлен, то для базовых элементов данных должны использоваться значения по умолчанию, игнорируя любое предыдущее значение.

Таблица 64 — Синтаксис *sbr_data ()*

Синтаксис	Количество битов	Мнемоника
sbr_data(id_aac, bs_amp_res)		
{		
switch (sbr_layer) {		
case SBR_NOT_SCALABLE		Примечание
switch (id_aac) {		
case ID_SCE		
sbr_single_channel_element(bs_amp_res)		
break;		
case ID_CPE		
sbr_channel_pair_element(bs_amp_res)		
break;		
}		
break;		
case SBR_MONO_BASE		
sbr_channel_pair_base_element(bs_amp_res)		
break;		
case SBR_STEREO_ENHANCE		
sbr_channel_pair_enhance_element(bs_amp_res)		
break;		

Окончание таблицы 64

Синтаксис	Количество битов	Мнемоника
case SBR_STEREO_BASE *sbr_channel_pair_element(bs_amp_res)* *break;* *}* *}*		
Примечание — Когда инструмент *SBR* используется с немасштабируемым базовым кодером *AAC*, значение переменной помощника *sbr_layer* является *SBR_NOT_SCALABLE*. Когда инструмент *SBR* используется с масштабируемым базовым кодером *AAC*, значение переменной помощника *sbr_layer* зависит от текущего уровня и конфигурации масштабируемости базового кодера *AAC* как определено в таблице 120 в 5.2.8.2.4.		

Таблица 65 — Синтаксис *sbr_single_channel_element ()*

Синтаксис	Количество битов	Мнемоника
sbr_single_channel_element (bs_amp_res) *{* *if(bs_data_extra)* *bs_reserved;* *sbr_grid (0);* *sbr_dtdf (0);* *sbr_invf (0);* *sbr_envelope (0, 0, bs_amp_res);* *sbr_noise (0, 0);* *if (bs_add_harmonic_flag[0])* *sbr_sinusoidal_coding(0);* *if (bs_extended_data) {* *cnt = bs_extension_size;* *if (cnt == 15)* *cnt += bs_esc_count;* *num_bits_left = 8 * cnt;* *while (num_bits_left > 7) {* *bs_extension_id;* *num_bits_left -= 2;* *sbr_extension(bs_extension_id, num_bits_left);* *}* *bs_fill_bits;* *}* *}*	 1 4 1 1 4 8 2 *Num_bits_left*	 *uimsbf* *uimsbf* Примечание
Примечание — *sbr_extension ()* должен уменьшить переменную *num_bits_left* на число битов, считанное из полезной нагрузки потока битов в *sbr_extension ().*		

Таблица 66 — Синтаксис *sbr_channel_pair_element ()*

Синтаксис	Количество битов	Мнемоника
sbr_channel_pair_element(bs_amp_res) *{* *if (bs_data_extra) {* *bs_reserved;* *bs_reserved;* *}*	 1 4 4	 *uimsbf* *uimsbf*

Окончание таблицы 66

Синтаксис	Количество битов	Мнемоника
if (bs_coupling) { *sbr_grid(0);* *sbr_dtdf(0);* *sbr_dtdf(1);* *sbr_invf(0);* *sbr_envelope(0,1, bs_amp_res);* *sbr_noise(0,1);* *sbr_envelope(1,1, bs_amp_res);* *sbr_noise(1,1);* *} else {* *sbr_grid(0);* *sbr_grid(1);* *sbr_dtdf(0);* *sbr_dtdf(1);* *sbr_invf(0);* *sbr_invf(1);* *sbr_envelope(0,0, bs_amp_res);* *sbr_envelope(1,0, bs_amp_res);* *sbr_noise(0,0);* *sbr_noise(1,0);* *}*	1	
if (bs_add_harmonic_flag[0]) *sbr_sinusoidal_coding(0);*	1	
if (bs_add_harmonic_flag[1]) *sbr_sinusoidal_coding(1);*	1	
if (bs_extended_data) { *cnt = bs_extension_size;* *if (cnt == 15)* *cnt += bs_esc_count;* *num_bits_left = 8 * cnt;* *while (num_bits_left > 7) {* *bs_extension_id;* *num_bits_left -= 2;* *sbr_extension(bs_extension_id, num_bits_left);* *}* *bs_fill_bits;* *}* *}*	1 4 8 2 *Num_bit_left*	 *uimsbf* *uimsbf* *uimsbf* Примечание
П р и м е ч а н и е — *sbr_extension ()* должен уменьшить переменную *num_bits_left* на число битов, считанное из полезной нагрузки потока битов в *sbr_extension ()*.		

Т а б л и ц а 67 — Синтаксис *sbr_channel_pair_base_element ()*

Синтаксис	Количество битов	Мнемоника
sbr_channel_pair_base_element(bs_amp_res) *{* *if (bs_data_extra) {* *bs_reserved;* *bs_reserved;* *}* *bs_coupling*	 1 4 4 1	 *uimsbf* *uimsbf* Примечание 1

Окончание таблицы 67

Синтаксис	Количество битов	Мнемоника
sbr_grid(0); *sbr_dtdf(0);* *sbr_invf(0);* *sbr_envelope(0,1, bs_amp_res);* *sbr_noise(0,1);* *if (bs_add_harmonic_flag[0])* *sbr_sinusoidal_coding(0);* *if (bs_extended_data) {* *cnt = bs_extension_size;* *if (cnt == 15)* *cnt += bs_esc_count;* *num_bits_left = 8 * cnt;* *while (num_bits_left > 7) {* *bs_extension_id;* *num_bits_left -= 2;* *sbr_extension(bs_extension_id, num_bits_left);* *}* *bs_fill_bits;* *}* *}*	1 1 4 8 2 *num_bits_left*	 *uimsbf* *uimsbf* *uimsbf* Примечание 2

Примечание 1 — У *bs_coupling* должно быть значение 1.

Примечание 2 — *sbr_extension ()* должен уменьшить переменную *num_bits_left* на число битов, считанных из полезной нагрузки потока битов в пределах *sbr_extension ()*.

Таблица 68 — Синтаксис *sbr_channel_pair_enhance_element ()*

Синтаксис	Количество битов	Мнемоника
sbr_channel_pair_enhance_element(bs_amp_res) *{* *sbr_dtdf(1);* *sbr_envelope(1,1, bs_amp_res);* *sbr_noise(1,1);* *if (bs_add_harmonic_flag[1])* *sbr_sinusoidal_coding(1);* *}*	 1	

Таблица 69 — Синтаксис *sbr_grid ()*

Синтаксис	Количество битов	Мнемоника
sbr_grid(ch) *{* *switch (bs_frame_class) {* *case FIXFIX* *bs_num_env[ch] = 2^ tmp;* *if (bs_num_env[ch] == 1)* *bs_amp_res = 0;* *bs_freq_res[ch][0];* *for (env = 1; env < bs_num_env[ch]; env++)* *bs_freq_res[ch][env] = bs_freq_res[ch][0];* *break;*	 2 2 1	 *uimsbf* *uimsbf* Примечание 1

Окончание таблицы 69

Синтаксис	Количество битов	Мнемоника
case FIXVAR		
bs_var_bord_1 [ch];	2	*uimsbf*
bs_num_env[ch] = bs_num_rel_1[ch] + 1;	2	*uimsbf*
for (rel = 0; rel < bs_num_env[ch]-1; rel++)		
bs_rel_bord_1[ch][rel] = 2 tmp + 2;*	2	*uimsbf*
ptr_bits = ceil (log (bs_num_env[ch] + 1) / log (2));		Примечание 2
bs_pointer[ch];	*ptr_bits*	*uimsbf*
for (env = 0; env < bs_num_env[ch]; env++)		
bs_freq_res[ch][bs_num_env[ch] – 1 – env];	1	
break;		
case VARFIX		
bs_var_bord_0[ch];		
bs_num_env[ch] = bs_num_rel_0[ch] + 1;	2	*uimsbf*
for (rel = 0; rel < bs_num_env[ch]-1; rel++)	2	*uimsbf*
bs_rel_bord_0[ch][rel] = 2 tmp + 2;*		
ptr_bits = ceil (log (bs_num_env[ch] + 1) / log (2));	2	*uimsbf*
bs_pointer[ch];	*ptr_bits*	Примечание 2
for (env = 0; env < bs_num_env[ch]; env++)		*uimsbf*
bs_freq_res[ch] [env];	1	
break;		
case VARVAR		
bs_var_bord_0[ch];	2	*uimsbf*
bs_var_bord_1 [ch];	2	*uimsbf*
bs_num_rel_0[ch];	2	*uimsbf*
bs_num_rel_1 [ch];	2	*uimsbf*
bs_num_env[ch] = bs_num_rel_0[ch] +		Примечание 1
bs_num_rel_1[ch] + 1;		
for (rel = 0; rel < bs_num_rel_0[ch]; rel++)		
bs_rel_bord_0[ch][rel] = 2 tmp + 2;*	2	*uimsbf*
for (rel = 0; rel < bs_num_rel_1[ch]; rel++)		
bs_rel_bord_1[ch][rel] = 2 tmp + 2;*	2	*uimsbf*
ptr_bits = ceil (log(bs_num_env[ch] + 1) / log (2));		Примечание 2
bs_pointer[ch];		
for (env = 0; env < bs_num_env[ch]; env++)		
ptr_bits = ceil (log(bs_num_env[ch] + 1) / log (2));		
bs_pointer[ch];	*ptr_bits*	*uimsbf*
for (env = 0; env < bs_num_env[ch]; env++)		
bs_freq_res[ch][env];	1	
break;		
}		
if (bs_num_env[ch] > 1)		
bs_num_noise[ch] = 2;		
else		
bs_num_noise[ch] = 1;		
}		

Примечание 1 — *bs_num_env* ограничивается согласно 6.18.3.6.

Примечание 2 — Деление (/) является плавающим делением, без округления или усечения.

Таблица 70 — Синтаксис *sbr_dtdf ()*

Синтаксис	Количество битов	Мнемоника
sbr_dtdf(ch) *{* *for (env = 0; env < bs_num_env[ch]; env++)* *bs_df_env[ch][env];* *for (noise = 0; noise < bs_num_noise[ch]; noise++)* *bs_df_noise[ch][noise];* *}*	 1 1	

Таблица 71 — Синтаксис *sbr_invf ()*

Синтаксис	Количество битов	Мнемоника
sbr_invf(ch) *{* *for (n = 0; n<num_noise_bands[ch]; n++)* *bs_invf_mode[ch][n];* *}*	 2	 Примечание *uimsbf*
Примечание — *num_noise_bands [ch]* получают из заголовка, согласно 6.18.3 и называется он N_Q.		

Таблица 72 — Синтаксис *sbr_envelope ()*

Синтаксис	Количество битов	Мнемоника
sbr_envelope(ch, bs_coupling, bs_amp_res) *{* *if (bs_coupling) {* *if (ch) {* *if (bs_amp_res) {* *t_huff = t_huffman_env_bal_3_0dB;* *f_huff = f_huffman_env_bal_3_0dB;* *} else {* *t_huff = t_huffman_env_bal_1_5dB;* *f_huff = f_huffman_env_bal_1_5dB;* *}* *} else {* *if (bs_amp_res) {* *t_huff = t_huffman_env_3_0dB;* *f_huff = f_huffman_env_3_0dB;* *} else {* *t_huff = t_huffman_env_1_5dB;* *f_huff = f_huffman_env_1_5dB;* *}* *} else {* *if (bs_amp_res) {* *t_huff = t_huffman_env_3_0dB;* *f_huff = f_huffman_env_3_0dB;* *} else {* *t_huff = t_huffman_env_1_5dB;* *f_huff = f_huffman_env_1_5dB;* *}* *}* *} else {* *if (bs_amp_res) {* *t_huff = t_huffman_env_3_0dB;* *f_huff = f_huffman_env_3_0dB;*		

Окончание таблицы 72

Синтаксис	Количество битов	Мнемоника
} else { *t_huff = t_huffman_env_1_5dB;* *f_huff = f_huffman_env_1_5dB;* *}* *}* *for (env = 0; env < bs_num_env[ch]; env++) {* *if (bs_df_env[ch][env] == 0) {* *if (bs_coupling && ch) {* *if (bs_amp_res)* *bs_data_env[ch][env][0] = bs_env_start_value_balance;* *bs_data_env[ch][env][0] = bs_env_start_value_balance;* *} else {* *if (bs_amp_res)* *bs_data_env[ch][env][0] = bs_env_start_value_level;* *bs_data_env[ch][env][0] = bs_env_start_value_level;* *}* *for (band = 1; band < num_env_bands[bs_freq_res[ch][env]];* *band++)* *bs_data_env[ch][env][band] = sbr_huff_dec(f_huff,* *bs_codeword);* *} else {* *for (band = 0; band < num_env_bands[bs_freq_res[ch][env]];* *band++)* *bs_data_env[ch][env][band] = sbr_huff_dec(t_huff,* *bs_codeword);* *}* *}* *}*	 5 6 6 7 1...18 1...18	 *uimsbf* *uimsbf* *uimsbf* *uimsbf* Примечание 1 Примечание 2 Примечание 1 Примечание 2

Примечание 1 — *num_env_bands [bs_freq_res [ch] [env]]* получается из заголовка согласно 6.18.3 и называется *n*.

Примечание 2 — *sbr_huff_dec ()* определяется в Приложении А.6.1.

Таблица 73 — Синтаксис *sbr_noise ()*

Синтаксис	Количество битов	Мнемоника
sbr_noise(ch,bs_coupling) { *if (bs_coupling) {* *if (ch) {* *t_huff = t_huffman_noise_bal_3_0dB;* *f_huff = f_huffman_noise_bal_3_0dB;* *} else {* *t_huff = t_huffman_noise_3_0dB;* *f_huff = f_huffman_noise_3_0dB;* *}* *} else {* *t_huff = t_huffman_noise_3_0dB;* *f_huff = f_huffman_noise_3_0dB;* *}* *for (noise = 0; noise < bs_num_noise[ch]; noise++) {* *if (bs_df_noise[ch][noise] == 0) {* *if (bs_coupling && ch)* *bs_data_noise[ch][noise][0] =* *bs_noise_start_value_balance;*	 5	 *uimsbf*

Окончание таблицы 73

Синтаксис	Количество битов	Мнемоника
else *bs_data_noise[ch][noise][0] =* *bs_noise_start_value_level;* *for (band = 1; band < num_noise_bands[ch]; band++)* *bs_data_noise[ch][noise][band] =* *sbr_huff_dec(f_huff,bs_codeword);* *} else {* *for (band = 0; band < num_noise_bands[ch];* *band++)* *bs_data_noise[ch][noise][band] =* *sbr_huff_dec(t_huff,bs_codeword);* *}* *}* *}*	5 1...18 1...18	*uimsbf* Примечание 1 Примечание 2 Примечание 1 Примечание 2 Примечание 2
Примечание 1 — *num_noise_bands* [*ch*] получается из заголовка согласно 6.18.3 и называется N_Q. Примечание 2 — *sbr_huff_dec* () определяется в приложении А.6.1.		

Таблица 74 — Синтаксис *sbr_sinusoidal_coding()*

Синтаксис	Количество битов	Мнемоника
sbr_sinusoidal_coding(ch) *{* *for (n = 0; n<num_high_res[ch]; n++)* *bs_add_harmonic[ch][n]* *}*	1	Примечание
Примечание — *num_high_res* [ch] получается из заголовка согласно 6.18.3 и называется N_{High}.		

4.2.9 Полезные нагрузки для аудио объектного типа *ER AAC ELD*

Таблица 75 — Синтаксис высокоуровневой полезной нагрузки для аудио объектного типа *ER AAC ELD (er_raw_data_block_eld)*

Синтаксис	Количество битов	Мнемоника
er_raw_data_block_eld(channelConfiguration) *{* *switch(channelConfiguration) {* *case 1:* *single_channel_element_eld();* *break;* *case 2:* *channel_pair_element_eld();* *break;* *case 3:* *single_channel_element_eld();* *channel_pair_element_eld();* *break;* *case 4:* *single_channel_element_eld();* *channel_pair_element_eld();*		Примечание

Окончание таблицы 75

Синтаксис	Количество битов	Мнемоника
single_channel_element_eld(); *break;* *case* 5*:* *single_channel_element_eld();* *channel_pair_element_eld();* *channel_pair_element_eld();* *break;* *case* 6*:* *single_channel_element_eld();* *channel_pair_element_eld();* *channel_pair_element_eld();* *lfe_channel_element_eld();* *break;* *case* 7*:* *single_channel_element_eld();* *channel_pair_element_eld();* *channel_pair_element_eld();* *channel_pair_element_eld();* *lfe_channel_element_eld();* *break;* *default:* */* reserved */* *break;* *}* *if (ldSbrPresentFlag) {* *er_low_delay_sbr_block(channelConfiguration);* *}* *cnt = bits_to_decode() / 8;* *while (cnt >= 1) {* *cnt -= extension_payload(cnt);* *}* *byte_alignment();* *}*		
Примечание — Описанная выше высокоуровневая полезная нагрузка обрабатывается как логическая полезная нагрузка потока битов. Чтобы получить эту логическую полезную нагрузку потока битов из физической полезной нагрузки потока битов, требуются шаги предварительной обработки, как описано в 5.2.4.		

Таблица 76 — Синтаксис *single_channel_element_eld ()*

Синтаксис	Количество битов	Мнемоника
single channel element eld() *{* *individual_channel_stream_eld (*0*);* *}*		

Таблица 77 — Синтаксис *lfe_channel_element_eld ()*

Синтаксис	Количество битов	Мнемоника
lfe channel element eld() *{* *individual_channel_stream_eld (*0*);* *}*		

Таблица 78 — Синтаксис *channel_pair_element_eld ()*

Синтаксис	Количество битов	Мнемоника
channel_pair_element_eld() *{* *common_window* = 1; *max_sfb;* *ms_mask_present;* *if (ms_mask_present* == 1 *) {* *for (sfb* = 0; *sfb* < *max_sfb; sfb++) {* *ms_used[0][sfb];* *}* *}* *individual_channel_stream_eld (common_window);* *individual_channel_stream_eld (common_window);* *}*	 6 2 1	 *uimsbf* *uimsbf* *uimsbf*

Таблица 79 — Синтаксис *individual_channel_stream_eld ()*

Синтаксис	Количество битов	Мнемоника
individual_channel_stream_eld (common_window) *{* *global_gain;* *if (! common_window) {* *max_sfb;* *section_data ();* *scale_factor_data ();* *tns_data_present;* *if (tns_data_present) {* *tns_data ();* *}* *if (! aacSpectralDataResilienceFlag) {* *spectral_data ();* *}* *else {* *length_of_reordered_spectral_data;* *ength_of_longest_codeword;* *reordered_spectral_data ();* *}* *}*	 8 6 1 14 6	 *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf*

Таблица 80 — Синтаксис *er_low_delay_sbr_block*

Синтаксис	Количество битов	Мнемоника
er_low_delay_sbr_block(channelConfiguration) *{* *switch (channelConfiguration) {* *case* 1*:* *low_delay_sbr_data(ID_SCE, ldSbrCrcFlag, bs_amp_res);* *break;* *case* 2*:* *low_delay_sbr_data(ID_CPE, ldSbrCrcFlag, bs_amp_res);* *break;* *case* 3*:* *low_delay_sbr_data(ID_SCE, ldSbrCrcFlag, bs_amp_res);* *low_delay_sbr_data(ID_CPE, ldSbrCrcFlag, bs_amp_res);* *break;*		

Окончание таблицы 80

Синтаксис	Количество битов	Мнемоника
case 4*:* *low_delay_sbr_data(ID_SCE, ldSbrCrcFlag, bs_amp_res);* *low_delay_sbr_data(ID_CPE, ldSbrCrcFlag, bs_amp_res);* *low_delay_sbr_data(ID_SCE, ldSbrCrcFlag, bs_amp_res);* *break;* *case* 5*:* *low_delay_sbr_data(ID_SCE, ldSbrCrcFlag, bs_amp_res);* *low_delay_sbr_data(ID_CPE, ldSbrCrcFlag, bs_amp_res);* *low_delay_sbr_data(ID_CPE, ldSbrCrcFlag, bs_amp_res);* *break;* *case* 6*:* *low_delay_sbr_data(ID_SCE, ldSbrCrcFlag, bs_amp_res);* *low_delay_sbr_data(ID_CPE, ldSbrCrcFlag, bs_amp_res);* *low_delay_sbr_data(ID_CPE, ldSbrCrcFlag, bs_amp_res);* *break;* *case* 7*:* *low_delay_sbr_data(ID_SCE, ldSbrCrcFlag, bs_amp_res);* *low_delay_sbr_data(ID_CPE, ldSbrCrcFlag, bs_amp_res);* *low_delay_sbr_data(ID_CPE, ldSbrCrcFlag, bs_amp_res);* *low_delay_sbr_data(ID_CPE, ldSbrCrcFlag, bs_amp_res);* *break;* *default:* */* reserved */* *break;* *}* *}*		

Т а б л и ц а 81 — Синтаксис *low_delay_sbr_data ()*

Синтаксис	Количество битов	Мнемоника
low_delay_sbr_data(id_aac, ldSbrCrcFlag, bs_amp_res) *{* *if (ldSbrCrcFlag) {* *bs_sbr_crc_bits;* *}* *if (bs_header_flag) {* *sbr_header();* *}* *If (id_aac==ID_SCE) {* *sbr_single_channel_element(bs_amp_res);* *} else if (id_aac==ID_CPE) {* *sbr_channel_pair_element(bs_amp_res);* *}* *}*	 10 1	 *uimsbf*
П р и м е ч а н и е — *bs_amp_res* обычно передается внутри функции *sbr_header ()*, включенным в состав *ELDSpecificConfig ()*. Этот параметр может быть обновлен функцией *sbr_header ().*		

5 Общая структура данных

5.1 Декодирование специальной конфигурации *GA*

5.1.1 *GASpecificConfig ()*

Параметры вызовов '*samplingFrequencyIndex*', '*channelConfiguration*', '*audioObjectType*' передаются из специального элемента конфигурации аудио. Информация, содержавшаяся в этих параметрах, обязательна для процесса декодирования.

Если частота дискретизации не является одной из величин, перечисленных в правой графе таблицы 82, анализирующей полезную нагрузку потока битов, то должны быть выведены показатели зависимые от частоты дискретизации таблицы (таблицы кода, таблицы масштабного коэффициента полосы и т. д.). Так как данная частота дискретизации сопоставляется только с одной таблицей частоты дискретизации и так как требуется максимальная гибкость в диапазоне возможных частот дискретизации, чтобы связать подразумеваемую частоту дискретизации с требующимися таблицами зависимости от частоты дискретизации должна использоваться следующая таблица.

Таблица 82 — Отображение частоты дискретизации

Частотный диапазон, Гц	Частоты дискретизации, Гц
$f \geq 92017$	96000
$92017 > f \geq 75132$	88200
$75132 > f \geq 55426$	64000
$55426 > f \geq 46009$	48000
$46009 > f \geq 37566$	44100
$37566 > f \geq 27713$	32000
$27713 > f \geq 23004$	24000
$23004 > f \geq 18783$	22050
$18783 > f \geq 13856$	16000
$13856 > f \geq 11502$	12000
$11502 > f \geq 9391$	11025
$9391 > f$	8000

Если в таблице 82 частота дискретизации приведенная в правой графе, не будет определена, то должна использоваться самая близкая из определенных таблиц.

frameLengthFlag Длина фрейма, число спектральных линий соответственно.

Для всех типов *General Audio Object Types* (общих аудио объектных типов), кроме *AAC SSR* и *ER AAC LD*:

если установлено в "0", используется *IMDCT* с 1024/128 линиями и *frameLength* устанавливается в 1024, если установлено в "1", используется *IMDCT* с 960/120 линиями и *frameLength* устанавливается в 960.

Для *ER AAC LD*: если установлено в "0", используется *IMDCT* с 512 линиями и *frameLength* устанавливается в 512, если установлено в "1", используется *IMDCT* с 480 линиями и *frameLength* устанавливается в 480.

Для *AAC SSR*: должно быть установлено в "0". Используется *IMDCT* с 256/32 линиями.

Примечание — Фактическое число линий для *IMDCT* (первое или второе значение) отличается значением *window_sequence.*

DependsOnCoreCoder Сигнализирует о том, что в лежащей в основе базового уровня масштабируемой конфигурации *AAC* использовался базовый кодер.

CoreCoderDelay — Задержка в выборках, которая должна быть применена к сверхдискретизированному (если необходимо) выводу базового декодера, перед вычислением *IMDCT*.

extensionFlag — Должно быть '0' для аудио объектных типов 1, 2, 3, 4, 6, 7. Должно быть '1' для аудио объектных типов 17, 19, 20, 21, 22, 23.

layerNr — 3-разрядное поле, указывающее номер уровня *AAC* в масштабируемой конфигурации. Первый уровень *AAC* обозначается значением 0.

numOfSubFrame — 5-разрядное целочисленное значение без знака, представляющее число подфреймов, которые группируются и передаются в суперкадре.

layer_length — 11-разрядное целочисленное значение без знака, представляющее среднюю длину уровней большого шага в байтах.

aacSectionDataResilienceFlag — Этот флаг сигнализирует о различных схемах кодирования данных раздела *AAC*. Если используется кодовая книга 11, эта схема передает дополнительную информацию о максимальном абсолютном значении для линий спектра. Это позволяет обнаруживать ошибки спектральных линий, которые больше, чем указанное значение.

aacScalefactorDataResilienceFlag — Этот флаг сигнализирует о различных схемах кодирования данных масштабного фактора *AAC*, которые более устойчивы к ошибкам, чем исходные.

aacSpectralDataResilienceFlag — Этот флаг сигнализирует о различных схемах кодирования (*HCR*) спектральных данных *AAC*, которые более устойчивы к ошибкам, чем исходные.

extensionFlag3 — Флаг расширения для будущего использования. Должен быть '0'.

Ограничения: *program_config_element ()* должен использоваться только для основных типов аудио объекта *AAC, AAC SSR, AAC LC* и *AAC LTP*.

5.1.2 Элемент конфигурации программы (*PCE*)

Следующие изменения применяются в контексте MPEG-4:

program_config_element () может встретиться вне полезной нагрузки *AAC*, например, как часть *GASpecificConfig ()* или adif_header (), но также и в составе полезной нагрузки *AAC* как синтаксический элемент в *raw_data_block ()*.

Конфигурация канала, данная в *program_config_element* () в полезной нагрузке *AAC*, оценивается, если никакая конфигурация канала не дается вне полезной нагрузки *AAC*. Это имеет место только для MPEG-4 *ADTS* с *channel_configation* == 0.

sampling_frequency_index, данный в *program_config_element ()*, может указывать номинальную частоту выборок, которая отличается от фактической частоты дискретизации, то есть, намеченной частоты дискретизации выходного сигнала декодера. Это является случаем, когда используется фактическая частота дискретизации, которая не может быть представлена посредством *sampling_frequency_index* в *program_config_element* (). Фактическая частота дискретизации сообщается в *AudioSpecificConfig ()* или неявно известна системе. Отношение между фактической частотой дискретизации и номинальной частотой дискретизации определяется в таблице 82.

В любом случае в определенное время может быть сконфигурирована только одна программа.

object_type — Двухбитовый индекс объектного типа из таблицы 83.

Таблица 83 — Индекс типа объекта

Индекс	Объектный тип
0	*AAC Main*
1	*AAC LC*
2	*AAC SSR*
3	*AAC LTP*

sampling_frequency_index — Указывает частоту дискретизации программы.

5.1.2.1 Конфигурация канала

Аудио синтаксис *AAC* обеспечивает три способа передачи отображения каналов в пределах ряда синтаксических элементов в физическом расположении динамиков.

5.1.2.1.1 Явное отображение канала, используя настройки канала по умолчанию

Если MPEG-4 *Audio* используется вместе с *MPEG-4 Systems*, должны использоваться настройки канала по умолчанию.

5.1.2.1.2 Явное отображение канала, используя *program_config_element ()*

Любая возможная конфигурация канала может быть определена, используя *program_config_element ()*. Чтобы анализировать любой *program_config_element ()* в полезной нагрузке *AAC*, всегда требуется декодер *MPEG-4*. Если никакая конфигурация канала не дается вне полезной нагрузки *AAC*, декодер используется только для оценки *program_config_element ()*.

5.1.2.1.3 Неявное отображение канала

Этот вид отображения канала не разрешается в ГОСТ Р 53556.4—2014.

5.1.2.2 *Matrix-mixdown* метод

5.1.2.2.1 Описание

Метод *matrix-mixdown* (матричное сведение) применяется только для того, чтобы сводить 5-канальную программу с конфигурацией динамиков 3-спереди/2-сзади к стерео или моно программе. Это не применимо ни к какой программе кроме конфигурации 3/2.

5.1.2.2.2 Процесс *matrix-mixdown*

Производный сигнал стерео может быть сгенерирован в декодере *matrix-mixdown* при помощи одного из двух следующих наборов уравнений.

Набор 1:

$$L' = \frac{1}{1 + 1/\sqrt{2} + A}\left[L + C/\sqrt{2} + AL_s\right]$$

$$R' = \frac{1}{1 + 1/\sqrt{2} + A}\left[R + C/\sqrt{2} + AR_s\right]$$

Набор 2:

$$L' = \frac{1}{1 + 1/\sqrt{2} + 2A}\left[L + C/\sqrt{2} - A(L_s + R_s)\right]$$

$$R' = \frac{1}{1 + 1/\sqrt{2} + 2A}\left[R + C/\sqrt{2} + A(L_s + R_s)\right],$$

где L, C, R, L_s и R_s являются исходными сигналами, L' и R' являются производными сигналами стерео, и A является матричным коэффициентом, указанным *matrix_mixdown_idx*. Каналы *LFE* в *mixdown* опускаются.

Если *pseudo_surround_enable* не устанавлено, то должна использоваться только установка в 1. Если *pseudo_surround_enable* устанавливается, то могут использоваться уравнения или набора 1, или набора 2 в зависимости от того, есть ли у ресивера средства, чтобы реализовать некоторую форму синтеза окружения.

В качестве дополнительной информации нужно отметить, что можно получить моно сигнал, используя следующее уравнение:

$$M = \frac{1}{3 + 2A}\left[L + C + R + A(L_s + R_s)\right].$$

5.1.2.2.3 Консультация

Обеспечение *matrix-mixdown* позволяет использовать режим работы, который может быть выгодным при некоторых обстоятельствах. Психоакустические принципы, на которых базируется аудиокодирование, нарушаются этой формой постобработки, и перцепционно правильная реконструкция сигнала не может быть гарантирована. Предпочтительный метод состоит в использовании каналов стерео или моно в синтаксисе *AAC*, чтобы обеспечить стерео или моно программирование, которое специально создается стандартным студийным смешиванием до снижения битовой скорости.

Стерео и моно каналы дополнительно позволяют провайдеру контента отдельно оптимизировать стерео и многоканальные смеси программы. Это невозможно при использовании *matrix-mixdown* метода.

Дополнительно необходимо отметить, что из-за алгоритмов, используемых для кодирования смеси многоканального и стерео, лучшая комбинация качества и скорости передачи обычно обеспечивается при помощи каналов стерео *mixdown*, это может быть обеспечено процессом *matrix-mixdown*.

5.1.2.2.4 Таблицы

Таблица 84 — Коэффициенты *matrix-mixdown*

matrix_mixdown_idx	*A*
0	$1/\sqrt{2}$
1	1/2
2	$1/(2\sqrt{2})$
3	0

5.2 Декодирование полезных нагрузок потока битов *GA*

5.2.1 Полезные нагрузки высшего уровня для аудио объекта типов *AAC main, AAC SSR, LC AAC* и *AAC LTP*

5.2.1.1 Определения

raw_data_block () Блок необработанных данных, который содержит аудиоданные для временного периода 1024 или 960 выборок, соответствующая информация и другие данные. Существует семь синтаксических элементов, идентифицированных элементом данных *id_syn_ele*. У элементов *audio_channel_element()'s* в одном *raw_data_block ()* должна быть только одна частота дискретизации. В *raw_data_block ()* могут быть несколько экземпляров того же самого синтаксического элемента, но они должны иметь различные 4 бита *element_instance_tag*, за исключением *data_stream_element ()'s* и *fill_element ()'s*. Поэтому в одном *raw_data_block ()* может быть от 0 до максимум 16 экземпляров любого синтаксического элемента, за исключением *data_stream_element ()'s* и *fill_element ()'s*, где это ограничение не применяется. Если имеется несколько *data_stream_element ()'s*, которые имеют тот же самый *element_instance_tag*, тогда они — часть того же самого потока данных. У *fill_element()* нет никакого тега *element_instance_tag* (так как контент-не требует последующей ссылки) и он может встретиться любое число раз. Конец *raw_data_block ()* обозначается специальным *id_syn_ele (TTERM)*, который может иметь место только однажды в *raw_data_block ()*.

id_syn_ele Элемент данных, который идентифицирует один из следующих синтаксических элементов (таблица 85).

Таблица 85 — Синтаксические элементы

имя *ID*	Кодирование	Сокращение	Синтаксический элемент
ID_SCE	0x0	*SCE*	*single_channel_element()*
ID_CPE	0x1	*CPE*	*channel_pair_element ()*
ID_CCE	0x2	*CCE*	*coupling_channel_element ()*
ID_LFE	0x3	*LFE*	*lfe_channel_element ()*
ID_DSE	0x4	*DSE*	*data_stream_element ()*
ID_PCE	0x5	*PCE*	*program_config_element ()*
ID_FIL	0x6	*FIL*	*fill_element ()*
ID_END	0x7	*TERM*	

single_channel_element ()	Аббревиатура *SCE*. Синтаксический элемент потока битов, содержащий кодированные данные для единственного звукового канала. *single_channel_element ()* в основном состоит из *individual_channel_stream ()*. Может быть до 16 таких элементов на блок необработанных данных, у каждого должен быть уникальный *element_instance_tag*.
channel_pair_element ()	Сокращение *CPE*. Синтаксический элемент полезной нагрузки потока битов, содержащий данные для пары каналов. *channel_pair_element()* состоит из двух *individual_channel_streams* и дополнительной объединенной информации о кодировании канала. Эти два канала могут совместно использовать общую информацию о стороне. У *channel_pair_element ()* имеются те же самые ограничения, что и у элемента единственного канала, касающиеся *element_instance_tag* и числа экземпляров.
coupling_channel_element ()	Сокращение *CCE*. Синтаксический элемент, который содержит аудиоданные для спаривания канала. Спаренный канал представляет собой информацию для степени многоканальности для одного блока или альтернативно для диалога для многоязычного программирования. Правила для числа *coupling_channel_element()'s* и тегов экземпляра такие же, как для *single_channel_element ()*.
lfe_channel_element ()	Сокращение *LFE*. Синтаксический элемент, который содержит канал расширения с малой частотой дискретизации. Правила для числа *lfe_channel_element () 's* и теги экземпляра те же, что для *single_channel_element () 's*.
program_config_element ()	Сокращение *PCE*. Синтаксический элемент, который содержит данные о конфигурации программы. Правила для числа *program_config_element () 's* и теги экземпляра элемента являются теми же самыми, что для *single_channel_element () 's*. *PCEs* должны поступать перед всеми другими синтаксическими элементами в *raw_data_block ()*.
fill_element ()	Сокращение *FIL*. Синтаксический элемент, который содержит данные заполнения. В нем может быть любое число элементов заполнения, которые могут приходить в любом порядке в блоке необработанных данных.
data_stream_element ()	*DSE* сокращения. Синтаксический элемент, который содержит данные. Снова там 16 *element_instance_tags*. Однако нет никакого ограничения на число *data_stream_element () 's* ни с любым тегом экземпляра, поскольку единственный поток данных может продолжаться через несколько *data_stream_element () 's* с тем же самым тегом экземпляра.
element_instance_tag	Уникальный тег экземпляра для синтаксических элементов кроме *fill_element ()*. Все синтаксические элементы, содержащие теги экземпляра, могут появляться не один раз, но, за исключением *data_stream_element ()* 's, должны иметь уникальный *element_instance_tag в каждом raw_data_block ()*. Этот тег также используется для ссылочных аудио синтаксических элементов в *single_ channel_element ()* 's, *channel_pair_element ()* 's, *lfe_channel_element ()* 's, *data_channel_element ()* 's и *coupling_channel_element ()* 's в *program_config_element ()* и обеспечивает возможность до 16 независимых *program_config_element ()* 's.
audio_channel_element	Общее обозначение для *single_channel_element ()*, *channel_pair_element*, *coupling_channel_element ()* и *lfe_channel_element ()*.
common_window	Флаг указывающий используются совместно *individual_channel_streams* и *ics_info* или нет. В случае совместного использования *ics_info* является частью *channel_pair_element ()* и должен использоваться для обоих каналов. Иначе *ics_info* является частью каждого *individual_channel_stream*.

5.2.1.2 Процесс декодирования

Предполагая, что начало *raw_data_block* известно, он может декодироваться без какой-либо дополнительной информации "о транспортном уровне" кроме частоты дискретизации (необходимой для выбора *sfb* и других таблиц) и производит 1024 или 960 аудиопробов на канал вывода.

Таблица 86 — Примеры самых простых полезных нагрузок потока битов

Сегмент полезной нагрузки потока битов	Выходной сигнал
<SCE> <TERM> <SCE> < TERM >	Моно сигнал
<CPE> < TERM > <CPE> < TERM >	Сигнал стерео
<SCE><CPE><CPE><LFE><TERM><SCE><CPE><CPE><LFE><TERM>	Сигнал канала 5.1

Угловые скобки (< >) используются, чтобы разграничить синтаксические элементы. Для моно сигнала у каждого *SCE* должно быть то же самое значение в его *element_instance_tag* и точно так же для сигнала стерео у каждого *CPE* должно быть то же самое значение в его *element_instance_tag*. Для сигналов канала 5.1 у каждого *SCE* должно быть то же самое значение в его *element_instance_tag*, у каждого *CPE*, связанного с передней парой каналов, должно быть то же самое значение в его *element_instance_tag* и у каждого *CPE*, связанного с парой обратных каналов, должно быть то же самое значение в его *element_instance_tag*.

Если эти полезные нагрузки потока битов должны быть переданы по каналу с постоянной скоростью, тогда они могут включать *fill_element* (), чтобы корректировать мгновенную скорость передачи. В этом случае пример кодированного сигнала стерео имеет вид

<CPE> <FIL> <TERM> <CPE> <FIL> < TERM > ...

Если полезные нагрузки потока битов должны переносить вспомогательные данные и работать на основе канала с постоянной скоростью, тогда пример кодированного сигнала стерео будет

<CPE><DSE><FIL><TERM><CPE><DSE><FIL><TERM>...

Все *data_stream_element () 's* имеют тот же самый *element_instance_tag*, если они являются частью того же самого потока данных.

single_channel_element () составляется из *element_instance_tag* и *individual_channel_stream*. В этом случае *ics_info* всегда располагается в *individual_channel_stream*.

channel_pair_element () начинается с *element_instance_tag* и флага *common_window*. Если *common_window* равняется '1', то *ics_info* совместно используется среди двух *individual_channel_stream* элементов и информация о *MS* передается. Если *common_window* равняется '0', то в пределах каждого *individual_channel_stream* есть *ics_info* и нет никакой информации о *MS*.

5.2.1.3 Элемент канала с низкочастоным улучшением (*LFE*)

5.2.1.3.1 Общее

Чтобы поддержать регулярную структуру в декодере, *lfe_channel_element ()* определяется как стандартный элемент *individual_channel_stream (0)*, то есть равный *single_channel_element ()*. Таким образом, декодирование может быть выполнено, используя стандартную процедуру для декодирования *single_channel_element* ().

Чтобы обеспечить большую скорость передачи и эффективную реализацию аппаратных средств декодера *LFE* к опциям, используемым для кодирования этого элемента, применяются несколько ограничений:

- поле *window_shape* всегда устанавливается в 0, то есть синусоидальное окно;
- поле *window_sequence* всегда устанавливается в 0 (*ONLY_LONG_SEQUENCE*);
- только самые низкие 12 спектральных коэффициентов любого *LFE* могут быть ненулевыми;
- не используется никакое временное шумовое формирование, то есть tns_data_present устанавливается в 0;
- не используется никакой прогноз, то есть *predictor_data_present* устанавливается в 0.

5.2.1.4 Элемент потока данных (*DSE*)

См. ГОСТ Р 54713—2011.

5.2.1.5 Элемент заполнения (*FIL*)

5.2.1.5.1 Элементы данных

count Начальное значение для длины данных заполнения.

esc_count Инкрементное значение длины данных заполнения.

5.2.1.5.2 Элементы помощника

cnt Величина, равная полной длине в байтах всех последующих *extensioin_payload () 's*.

Разрешено любое число элементов заполнения.

Элементы заполнения, содержащие *extension_payload()* с *extension_type* для *EXT_SBR_DATA* или *EXT_SBR_DATA_CRC* не должны содержать никаких других *extension_payload* любого *другого extension_type*.

5.2.1.5.3 Процесс декодирования

Синтаксическое элемент *count* дает начальное значение длины последующего *extension_payload()'s*. Для элемента данных это значение постепенно увеличивается со значением *esc_count*, если *count* равняется 15. Получающееся число (*cnt*) дает число байтов, которые будут считаны.

5.2.2 Полезные нагрузки для аудио объектного типа масштабируемого *AAC*

5.2.2.1 Определения

aac_scalable_main_element () — Сокращение *ASME*. Синтаксический элемент полезной нагрузки потока битов содержит кодированные данные для первого уровня кодирования *AAC* в масштабируемой конфигурации. Этот тип синтаксиса может использоваться также для приложений единственного уровня кодирования (немасштабируемые). В этом случае только один *ASME* содержит всю кодированную информацию. *ASME* состоит из *aac_scalable_main_header ()* и для каждого кодированного выхода канала аудио один *individual_channel_stream ()*. Максимальное количество таких элементов ограничивается 1 для каждого аудиообъекта.

aac_scalable_main_header () — Содержит всю дополнительную информацию, необходимую для первого уровня кодирования *AAC*, за исключением дополнительной информации, которая передается в отдельных потоках канала. Подблок *ics_info () AAC* здесь не используется. Вместо этого информация, которую обычно передают в *ics_info ()*, содержится непосредственно в этом элементе.

aac_scalable_extension_element () — Сокращение *ASEE*; синтаксический элемент полезной нагрузки потока битов, содержащий кодированные данные для всех, кроме первого уровня кодирования *AAC* в масштабируемой конфигурации. *ASEE* состоит из *aac_scalable_extension_header ()* и для каждого кодированного канала аудиовыхода одного *individual_channel_stream ()*. Максимальное количество таких элементов ограничивается 7 для одного аудиообъекта.

aac_scalable_extension_header () — Содержит всю дополнительную информацию, требующуюся для уровня расширения *AAC*, за исключением дополнительной информации, которая передается в отдельных потоках канала. Расширение в этом контексте означает, что это не первый уровень кодирования *AAC*.

bits_to_decode () — Функция помощи; возвращает число битов еще не декодированных в текущей высокоуровневой полезной нагрузке, если длина этой полезной нагрузки сообщается системой/транспортным уровнем. Если длина высокоуровневой полезной нагрузки неизвестна, *bits_to_decode ()* возвращает 0.

diff_control [w] [dc_group] — Для каждого окна это — управляющая информация *FSS* для одного *dc_group*. Если тип окна не является *SHORT_WINDOW.diff_control[w] [dc_group]* закодированная по Хаффману с использованием таблицы 165 в полезной нагрузке потока битов.

diff_control_lr [w] [sfb] — Элемент, используемый в конфигурации моно *GA*/стерео *GA*, для управления взаимодействием канала *M* с каналом *L* и *R*.

5.2.2.1.1 Элементы справки

this_layer_stereo — Устанавливается в '1', если текущий уровень является уровнем стерео, иначе устанавливается в '0'.

mono_layer_flag — Устанавливается в '1', если имеется какой-либо уровень моно, иначе устанавливается в '0'.

mono_stereo_flag — Устанавливается в '1', если есть один уровень моно и это — первый уровень стерео; иначе устанавливается в '0'.

last_max_sfb — *max_sfb* предыдущего уровня кодирования. Если предыдущий уровень работает при другой части дискретизации или не является кодером *GA*, *last_max_sfb* устанавливается в '4* *no_of_dc_groups*-1', если тип окна не является *SHORT_WINDOW*, иначе это устанавливается в самый низкий *sfb*, покрывающий все *diff_short_lines*.

last_max_sfb_ms	*max_sfb* предыдущего уровня стерео. Устанавливается в '0', если предыдущий уровень является уровнем моно.
last_mono_max_sfb	*max_sfb hightst* самого верхнего уровня моно.
core_flag	Устанавливается в '1', если присутствует базовый кодер. В рамках стандарта *MPEG-4 Audio* единственным допустимым базовым кодером является *MPEG-4 Celp*, хотя могут использоваться механизмы, например, для других кодеров формы сигнала или кодера *AAC*, работающего на более низкой частоте дискретизации, чем уровень расширения *AAC*; если нет никакого базового кодера, устанавливается в '0'.
tvq_layer_present	Устанавливается в '1', если в предыдущем уровне присутствует кодер *TwinVQ*; иначе устанавливается в '0'.
tvq_tns_present	Устанавливается в '1', если *tns_present* устанавливается в '1' в предыдущем уровне *TwinVQ*; иначе устанавливается в '0'.
tvq_stereo	Устанавливается в '1', если предыдущий уровень *TwinVQ* является стерео; иначе устанавливается в '0'.
tvq_mono_tns	Устанавливается в '1', если существует моно уровень *tvq*, который использует *tns*; устанавливается в '0', если существует моно уровень *tvq*, который не использует *tns*.
tvq_stereo_tns_left	Устанавливается в '1', если существует уровень стерео *tvq*, который использует *tns* в левом канале; устанавливается в '0', если есть уровень стерео *tvq*, который не использует *tns* в левом канале.
tvq_stereo_tns_right	Устанавливается в '1', если существует уровень стерео *tvq*, который использует *tns* в правом канале; устанавливается в '0', если есть уровень стерео *tvq*, который не использует *tns* в правом канале.
tns_aac_tvq_en [ch]	Устанавливается в '1', если бит *tns_data_present* передается в *ASME*, если присутствует уровень *TwinVQ* (см. таблицу 90).

5.2.2.2 Процесс декодирования

Предполагая, что начало *ASME* или *ASEE* известно, процесс может декодироваться с дополнительной информацией, данной в *GASpecificConfig()*. Для каждого элемента производятся 1024 или 960 аудиовыборок на канал вывода.

Кодирование с поддержкой *ASME* или *ASEE* для каналов как с постоянной скоростью, так и с изменяемой скоростью. В каждом случае структура полезной нагрузки потока битов и работа декодера идентичны.

В декодере вывод *ASME* и вывод всех доступных *ASEE*, где доступен вывод предыдущего уровня кодирования, должны быть объединены, чтобы сформировать выходной сигнал, дающий максимальное качество звука. Если есть промежуточное отсутствие уровня, например, из-за ошибок передачи, информация в последующих уровнях не может использоваться. Кроме того, если есть базовый кодер *CELP* либо один или более уровней *TwinVQ*, о которых сообщено для масштабируемого аудио объекта, вывод этих кодеров должен быть объединен с уровнями кодирования *AAC*.

5.2.2.3 Допустимые комбинации *AAC* с *TwinVQ* или с *CELP*

Масштабируемый кодер *AAC* в комбинации с кодером базового уровня *TwinVq* или *CELP* обеспечивает один способ достижения масштабируемости скорости передачи. Он основан на вычислении разностного сигнала между выходным сигналом кодера базового уровня и исходным входным сигналом. Используется один уровень расширения *AAC*. Возможны конфигурации объединенного кодирования стерео и смешанного моно/стерео. Все объединенные режимы стерео *AAC* доступны в объединенном кодере.

Три главных класса масштабируемых конфигураций с *AAC* существуют в зависимости от используемых типов кодера:

1. Только уровни *AAC*.
2. Узкополосный базовый уровень *CELP* плюс *AAC*.
3. Базовый уровень *TwinVQ* плюс *AAC*.

В любой конфигурации переданная полоса пропускания (посредством *max_sfb* в случае *AAC* и *TwinVQ* и посредством *no_of_dc_groups* или *diff_short_lines* в случае *CELP*) определенного уровня не должна быть меньшей, чем полоса предыдущего уровня.

Уровень кодирования *AAC* или *TwinVQ* может быть закодирован в моно, или стерео вида стерео/объединенный. Таблица 87 суммирует возможные переходы между двумя уровнями в отношении комбинаций кодера, и конфигураций канала для каждого из трех типов кодера в моно и стерео.

Таблица 87 — Допустимые масштабируемые комбинации

	Уровень *N*+1				
Уровень *N*	*NB CELP* моно	*TwinVQ* моно	*TwinVQ* стерео	*AAC* моно	*AAC* стерео
Узкополосный (*NB*) *CELP* моно	X			X	X
TwinVQ моно		X		X	X
TwinVQ стерео			X		X
AAC моно				X	X
AAC стерео					X

5.2.2.4 Декодирование комбинаций только *AAC*

Комбинации только *AAC* в основном рассчитаны на кодирование различия спектра в модуле *Scalable Inverse AAC Quantization Module* (модуль квантования масштабируемого инверсного *AAC*) (*SIAQ*). В модуле *SIAQ* после инверсного квантования добавляются восстановленные спектры всех *individual_channel_streams*. Число уровней ограничивается одним основным уровнем *AAC* и до 7 уровней расширения *AAC*. Скорость передачи основного уровня и дополнительных уровней может быть любой скоростью передачи, возможной для *AAC*. Однако применяются ограничения буфера полезной нагрузки потока битов. *SIAQ* является также основным конструктивным блоком для комбинаций *CELP* и *TwinVQ* с *AAC*.

Возможны три комбинации каналов аудио масштабируемого кодера, основанные только на *AAC*.

AAC-Only-M:	Основной уровень *AAC* моно	плюс
	От 0 до 7 моно уровней расширения *AAC*	
AAC-Only-S:	Основной уровень *AAC* стерео	плюс
	От 0 до 7 уровней расширения стерео *AAC*	
AAC-Only-M/S:	Основной уровень *AAC* моно	плюс
	От 0 до 7 уровней расширения *AAC* моно	плюс
	От 1 до 7 уровней расширения *AAC* стерео	(общее количество уровней ≤ 8)

В этом режиме *AAC-Only-M/S* присутствуют три модуля *SIAQ*. Первый — для звукового канала левый (*L*), или средний (*M*), или интенсивный (*I*) соответственно, второй — для звукового канала правый (*R*) или сторонний (*S*) и третий — для моно уровня (*M*), который является сокращенной смесью левого и правого, сгенерированной в кодере согласно $M = 0{,}5(L + R)$. Для всех полос с масштабным коэффициентом, где выбирается кодирование *M/S*, *M*-сигнал вычисляется, добавляя *M"* и *M'* (ограничения, данные в 5.2.2.7, должны сопровождаться сведениями о том, что запрещает дополнение при определенных обстоятельствах). Для всех полос, где выбирается кодирование *L/R*, *L* и *R* сгенерированы из *L'*, *R'*, и $2{,}0M"$, используя *diff_control_lr*, чтобы управлять соответствующим модулем *FSS*. *M*, *S*, *L* и *R* тогда подаются в модуль инверсной обработки объединенного стерео *AAC*, который вырабатывает спектры *L* и *R*. После обработки в последовательной комбинации фильтров *TNS*, как описано в от 6.9 до 6.9.5, временные сигналы могут быть вычислены посредством гребенки фильтров *IMDCT*.

Чтобы декодировать режим *AAC-Only-S*, модули *FSS* и инверсные фильтры *TNS-M* обходятся. *SIAQ M"*-канала в этом режиме отсутствует. Для декодирования режима *AAC-Only-M* инверсный модуль обработки объединенного стерео и модуль правый/сторонний *SIAQ*, а также соответствующая комбинация *TNS/IMDCT* не требуются. Только один инверсный фильтр *TNS* применяется к спектру перед *IMDCT*.

5.2.2.5 Декодирование комбинаций *CELP/AAC*

Эта комбинация прежде всего используется, чтобы добавить базовый уровень очень малой битовой скорости в масштабируемой системе. В то время как в комбинациях *TwinVQ/AAC* все уровни кодирования работают в частотной области, уровень кодирования *CELP* является кодером временной области, который работает с другой частотой дискретизации. Используя инструмент повышающей дискретизации и модуль *FSS*, комбинация делает усиление кодирования базового кодера *CELP* доступным для последующих уровней расширения. Отношение частоты дискретизации базового кодера и частоты дискретизации кодера *AAC* всегда является целым числом, чтобы сделать возможным использование инструмента повышающей дискретизации.

5.2.2.5.1 Постфильтр *CELP*

Постфильтр базового кодера *CELP* должен быть выключен для выходного сигнала, который тогда объединяется с этапами улучшения *AAC* в масштабируемом кодере. Если требуется только вывод кодера *CELP*, постфильтр может использоваться чтобы генерировать этот вывод.

5.2.2.5.2 Адаптация длины фрейма / суперфрейма

Посколку длина фрейма *AAC* не обязательно должна быть кратной базовой длине фрейма, наименьшее общее кратное этих двух длин может быть очень большим. Чтобы избежать таких ситуаций, разрешена альтернативная продолжительность фрейма *AAC* 960 выборок вместо 1024. Эта длина фрейма позволяет легко интегрировать ядро MPEG-4 *CELP* и *AAC*, создавая суперфреймы разумного размера. Некоторые комбинации частот дискретизации кодеров CELP и GA требуют другого числа фреймов базового кодера и фреймов *AAC* для создания общих суперфреймов. Таблица 88 дает обзор длины фрейма *AAC* с 960 выборками на различных частотах дискретизации и длины наименьших возможных суперфреймов для каждой частоты дискретизации для всех вариантов длины базового фрейма MPEG-4 *CELP* (40, 30, 20 и 10 мс). В этой таблице также перечислено количество фреймов кодеров *AAC* и *CELP* в этих наименьших возможных суперфреймах. Эти суперфреймы не существуют в MPEG-4 *System*, поскольку только единичные фреймы каждого уровня кодирования адресуются системами MPEG-4, делая возможными различные решения. Однако декодер MPEG-4 *Audio* обязан поддерживать только эти суперфреймы минимальной длины фрейма. Буферизовать целый суперфрейм не обязательно. Декодер может начать вырабатывать объединенные выходные выборки всякий раз, когда доступно достаточное число сверхдискретизированных выходных выборок ядра *CELP* и полный фрейм уровня *AAC*.

Базовый сигнал *CELP* должен быть задержан после *zeo*-вставки в инструменте повышения дискретизации (6.15) перед гребенкой фильтров *MDCT*. Значение «*coreCoderDelay*», данное в *GASpecificConfig ()* первого уровня кодирования *AAC*, является числом выборок, на которые должен быть задержан базовый сигнал. У кодера есть выбор оптимизировать полную системную задержку базового потока *Celp* (никакая дополнительная задержка не добавлена в кодере *CELP*), или скорректировать сигнал ядра и уровня *AAC* в некоторой степени, чтобы не требовалось никакой задержки масштабируемого декодера или никакого значения между этими двумя экстремальными значениями. В любом случае *Composition Time Stamp* первого ядра и его соответствующего первого фрейма *AAC* в суперфрейме, должны быть идентичны. Если требуется только вывод ядра *Celp*, представленного *compositor*, и минимальная задержка (например, для двухсторонней связи), приемный терминал, который соответствует *CoreCoderDelay*, может вычесть время *td* из *Composition Time*. Это верно если нет никакого ассоциированного видео потока, для которого нужно гарантировать синхронный состав.

td вычисляется согласно:

$$td = CoreCoderDelay/fs,$$

где *fs* является частотой дискретизации уровня(ней) *AAC*.

Таблица 88 — Длины фрейма *AAC* для 960 выборок на фрейм и длина суперфрейма комбинации *AAC/CELP*

Частота дискретизации, кГц	96	64	48	32	24	16	8
Длина фрейма *AAC*, мс	10	15	20	30	40	60	120
Длина суперфрейма (40 мс базовый фрейм), мс	40	120	40	120	40	120	120
AAC/CELP фреймов на суперфрейм	4/1	8/3	2/1	4/3	1/1	2/3	1/3
Длина суперфрейма (30 мс базовый фрейм), мс	30	30	60	30	120	60	120
AAC/CELP фреймов на суперфрейм	3/1	2/1	3/2	1/1	3/4	1/2	1/4
Длина суперфрейма (20 мс базовый фрейм), мс	20	60	20	60	40	60	120
AAC/CELP фреймов на суперфрейм	2 / 1	4/3	1/1	2/3	1/2	1/3	1/6
Длина суперфрейма (15 мс базовый фрейм), мс	30	15	60	30	120	60	120
AAC/CELP фреймов на суперфрейм	3/2	1/1	3/4	1/2	3/8	1/4	1/8
Длина суперфрейма (10 мс базовый фрейм), мс	10	30	20	30	40	60	120
AAC/CELP фреймов на суперфрейм	1/1	2/3	1/2	1/3	1/4	1/6	1/12

5.2.2.5.3 Кодер ядра *CELP* с *AAC*, работающий при частотах дискретизации 88,2 кГц, 44,1 кГц, или 22,05 кГц

Фреймы *AAC*, использующие частоты дискретизации 88,2 кГц, 44,1 кГц или 88,2 кГц, 22,05 кГц могут быть получены, корректируя частоту дискретизации базового кодера *CELP* так, что достигается целочисленное отношение между этими двумя частотами дискретизации. Таблица 131 показывает отображение частот дискретизации *AAC* на частоты дискретизации кодера ядра *CELP*. Кодер ядра *CELP* работает с частотами дискретизации, перечисленными в этой таблице. Процесс декодирования *CELP* абсолютно идентичен методам, определенным для частоты дискретизации 8 кГц для узкополосного кодера *CELP*. Таблица 89 показывает параметры суперфрейма для частот дискретизации *AAC* 88,2 кГц, 44,1 кГц и 88,2 кГц, 22,05 кГц.

Таблица 89 — Параметры суперфрейма комбинаций *AAC/CELP* на частотах дискретизации *AAC* 88,2 кГц, 44,1 кГц и 22,05 кГц

Частота дискретизации *AAC*, кГц	88,2	44,1	22,05
Длина фрейма *AAC*, мс	10,884	21,768	43,537
Длина суперфрейма (43,537 мс базовый фрейм), мс	43,537	43,537	43,537
AAC/CELP фреймов на суперфрейм	4/1	2/1	1/1
Длина суперфрейма (32,653 мс базовый фрейм), мс	32,653	65,306	130,612
AAC/CELP фреймов на суперфрейм	3/1	3/2	3/4
Длина суперфрейма (21,768 мс базовый фрейм), мс	21,768	21,768	43,537
AAC/CELP фреймов на суперфрейм	2/1	1/1	1/2
Длина суперфрейма (16,326 мс базовый фрейм), мс	32,653	65,306	130,612
AAC/CELP фреймов на суперфрейм	3/2	3/4	3/8
Длина суперфрейма (10,884 мс базовый фрейм), мс	10,884	21,768	43,537
AAC/CELP фреймов на суперфрейм	1/1	1/2	1/4

Возможные частоты дискретизации для ядра *CELP* в этом случае составляют 7350 Гц (узкополосное ядро) или 14700 Гц (широкополосное ядро).

5.2.2.5.4 Конфигурации *CELP/AAC*

Определяются несколько комбинаций *CELP/AAC*, которые отличаются по числу звуковых каналов в каждом уровне:

CELP-AAC-M	*CELP* моно уровень	плюс
	AAC моно основной уровень	плюс
	От 0 до 7 моно уровней расширения *AAC*	
CELP-AAC-MC	*CELP* моно уровень	плюс
	AAC моно основной уровень	плюс
	От 0 до 7 уровней расширения *AAC*	плюс
	От 0 до 7 уровней расширения стерео *AAC*	общее количество уровней *AAC* ≤ 8
CELP-AAC-S	*CELP* моно уровень	плюс
	Стерео *AAC* основной уровень	плюс
	От 0 до 7 уровней расширения стерео *AAC*	

У кодера *CELP* также есть внутренняя функция масштабируемости. Это означает, что уровень кодирования *CELP* может сам состоять из нескольких узкополосных уровней кодирования *CELP*.

Ядро *CELP* декодируется и вывод сверхдискретизируется, задерживается и преобразуется в частотную область как описано в 6.15. Этот сигнал затем объединяется с восстановленным спектром уровня *AAC* в инструменте *Frequency Selective Switch* (частотно-избирательном коммутаторе), чтобы получить полный выходной спектр. Если используется *TNS*, фильтр *TNS* кодера применяется к спектру базового кодера. Затем перед вычислением инверсной гребенки фильтров *IMDCT Filterbank* как обычно применяется нормальный фильтр *TNS* декодера.

Фильтр *TNS* применяется к коэффициентам *MDCT*, вычисленным из сверхдискретизированного вывода декодера *CELP*. Инверсный фильтр *TNS* необходим только для вывода уровня расширения.

Фильтр *TNS* для сигнала *Mid* применяется к коэффициентам *MDCT*, восстановленным из ядра *CELP*. Оба инверсных фильтра *TNS*, инверсный фильтр для сигнала *M* и инверсный фильтр *TNS* для сигнала *L/R* применяются к заключительному выводу уровня как описано в от 6.9—6.9.5.

5.2.2.5.5 Альтернативный базовый кодер

В пределах *MPEG*-4 только кодер *MPEG*-4 *CELP* в конфигурации единственного уровня доступен как базовый кодер. Однако нет никакого основного ограничения относительно того, какой базовый кодер может использоваться, хотя базовый кодер должен кодировать форму входного сигнала, чтобы позволить вычислять полезный разностный сигнал. Особенно легко может быть интегрирован кодер *CELP* с длиной фрейма кратной 10 мс.

5.2.2.6 Декодирование комбинаций *TwinVQ/AAC*

Возможно создать масштабируемый кодер, комбинируя *TwinVQ* и *AAC*. Эта конфигурация обеспечивает преимущества обоих кодеров: *TwinVQ* может уменьшить усредненное искажение в диапазоне малой битовой скорости и *AAC* способен управлять шумом квантования для высококачественного кодирования. Масштабируемая полезная нагрузка потока битов сначала демультиплексируется, и извлекается спектральная информация уровня *TwinVQ* и *AAC*, а также дополнительная информация. После инверсного квантования *AAC* производится добавление этих двух спектров, учитывая управляющую информацию *FSS*. Наконец, посредством *IMDCT* генерируется сигнал временной области.

Одно из преимуществ объединенного аудиокодирования *TwinVQ* и *AAC* состоит в том, что обе схемы квантования работают на одном и том же сигнальном пространстве, то есть коэффициентах *MDCT*. Оба кодера совместно используют ту же самую частоту дискретизации. Поэтому никакое «взаимодействие через интерфейс» не требуется, чтобы взаимно преобразовывать внутренние представления аудиоданных этих двух кодеров, как для кодера *CELP*, где должна быть вычислена дополнительная гребенка фильтров повышения дискретизации.

5.2.2.6.1 Комбинация с инструментами *AAC*

Ориентированные на *AAC* инструменты, такие как *TNS*, *LTP* и объединенное кодирование стерео, могут быть объединены с *TwinVQ* и масштабируемым кодером. *LTP* применяется только к базовому уровню, а другие инструменты обычно используются и для уровня *TwinVQ* и для уровня *AAC*. Есть ряд возможностей того, как объединить эти инструменты. Передача *tns_data_present* и *tns_data* () в первом уровне *AAC* зависит от фактически используемой конфигурации. Таблица 90 показывает условия, когда передаются имеющие отношение к *TNS* элементы данных и когда нет.

Таблица 90 — Значения элемента помощника *tns_aac_tvq_en []* в зависимости от *tvq_mono_tns, tvq_stereo_tns_left* и *tvq_stereo_tns_right*

	tns_aac_tvq_en [0]	*tns_aac_tvq_en [1]*
(tvq_mono_tns = =1*) && (this_layer_stereo* = =1*)*	1	1
(tvq_mono_tns = =1*) && (this_layer_stereo* = =0*)*	0	X
(tvq_mono_tns = = 0*)*	1	1
(tvq_stereo_tns_left = = 1*) && (tvq_stereo_tns_right* = =1*)*	0	0
(tvq_stereo_tns_left = =1*) && (tvq_stereo_tns_right* = =0*)*	0	1
(tvq_stereo_tns_left = =0*) && (tvq_stereo_tns_right* = =1*)*	1	0
(tvq_stereo_tns_left = =0*) && (tvq_stereo_tns_right* = =0*)*	1	1

5.2.2.6.2 *max_sfb* и *scale_factor_grouping* для *TwinVQ*

Для большинства инструментов связанных с *AAC*, таких как *LTP, TNS, FSS (diff_control)* и объединенного кодирования стерео необходимы таблицы полосы с масштабным фактором и параметр *max_sfb*. Если *TwinVQ* объединяется с этими инструментами, используется полоса с масштабным фактором как определено в таблицах 129—147. С другой стороны, *TwinVQ* обычно не использует параметр *max_sfb*. Однако он может быть вычислен из частотного диапазона, который используется в векторном квантовании с чередованием как показано в следующем псевдокоде:

tvq_main_element ():
*max_line = (int) (frame_length/num_windows * qsample);*
for (iw = 0, i_sfb = 0; ((iw <num_swb) && (swb_offset [iw] <max_line)); iw ++) {
i_sfb = iw+1;}
max_sfb = i_sfb;
tvq_extension_element ():
max_line = (int) (frame_length/num_windows (qsample+bias*3));*
for (iw = 0, i_sfb = 0; ((iw <num_swb) && (swb_offset [iw] <max_line)); iw ++) {
i_sfb = iw+1;}
max_sfb [lyr] = i_sfb;.

Параметр *max_line* является наивысшей частотнй составляющей, которую квантует чередующийся *VQ. Frame_length* равна 1024 или 960. В случае *EIGHT_SHORT_SEQUENCES num_windows* равно 8, иначе равно 1. Значения *qsample* и *bias* определяются в разделе адаптивного активного выбора полосы (6.5.4), и зависят только от частоты дискретизации и скорости передачи. Таблицы *swb_offset* перечисляются в 5.4. Параметр *scale_factor_grouping* передается в элементе *TwinVQ,* если это необходимо для объединенного кодирования стерео с *TwinVQ.*

5.2.2.6.3 Конфигурации звукового канала

TVQ-AAC-MM	*TwinVQ* моно уровень	плюс
	AAC основной уровень моно	плюс
	От 0 до 7 моно уровней расширения *AAC*	
TVQ-AAC-MMC	*TwinVQ* уровень моно	плюс
	AAC моно основной уровень	плюс
	От 0 до 7 моно уровней расширения *AAC*	плюс
	От 0 до 7 уровней расширения стерео *AAC*	(общее количество уровней *AAC* ≤ 8)
TVQ-AAC-MC	*TwinVQ* уровень моно	плюс
	Стерео *AAC* основной уровень	плюс
	От 0 до 7 уровней расширения стерео *AAC*	плюс
TVQ-AAC-SS	Уровень стерео *TwinVQ*	плюс
	Стерео *AAC* основной уровень	плюс
	От 0 до 7 уровней расширения стерео *AAC*	плюс

5.2.2.7 Комбинирование уровней *AAC,* если *PNS, MS* или инструменты интенсивности используются в определенной полосе масштабного коэффициента

Таблицы 91, 92, 93 определяют выходной спектр определенной полосы масштабного коэффициента объединенных уровней *N* и *N*+1 для различных комбинаций *PNS,* интенсивности и инструментов кодирования *MS* в уровне *N* и уровне *N*+1 для различных комбинаций уровней:

Таблица 91 — Комбинация уровней моно-моно

Инструмент, используемый в уровне *N*	Инструмент, используемый в уровне *N*+1	Вывод объединенного уровня
Никакого инструмента	Никакого инструмента	Сумма уровня *N* и уровня *N*+1
Никакого инструмента	*PNS*	Недопустимая комбинация
PNS	Никакого инструмента	См.6.13.6
PNS	*PNS*	Уровень *N*+1

Таблица 92 — Комбинация уровней стерео-стерео

Инструмент, используемый в уровне *N*	Инструмент, используемый в уровне *N*+1	Вывод объединенного уровня
Никакого инструмента или *MS*	Никакого инструмента или *MS*	Сумма уровня *N* и уровня *N*+1
Никакого инструмента или *MS*	*PNS*	Недопустимая комбинация
Никакого инструмента или *MS*	Интенсивность	Недопустимая комбинация

Окончание таблицы 92

Инструмент, используемый в уровне *N*	Инструмент, используемый в уровне *N*+1	Вывод объединенного уровня
Никакого инструмента или *MS*	*PNS* и интенсивность	Недопустимая комбинация
PNS	Никакого инструмента	См.6.13.6
PNS	*MS*	См.6.13.6
PNS	Интенсивность	Уровень *N*+1
PNS	*PNS*	Уровень *N*+1
PNS	*PNS* и интенсивность	Уровень *N*+1
Интенсивность	Никакого инструмента или *MS*	Уровень *N*+1
Интенсивность	*PNS*	Недопустимая комбинация
Интенсивность	Интенсивность	Сумма уровня *N* и уровня *N*+1, только канал *M/L*; Взять позиции из уровня *N*+1
Интенсивность	*PNS* и интенсивность	Недопустимая комбинация
PNS и интенсивность	Никакого инструмента или *MS*	Уровень *N*+1
PNS и интенсивность	*PNS*	Недопустимая комбинация
PNS и интенсивность	Интенсивность	Уровень *N*+1
PNS и интенсивность	*PNS* и интенсивность	Уровень *N*+1

Т а б л и ц а 93 — Комбинация уровней моно-стерео

Инструмент, используемый в уровне *N*	Инструмент, используемый в уровне *N*+1	Вывод объединенного уровня
Никакого инструмента или *MS*	Никакого инструмента	Сумма уровня *N* и уровня *N*+1 (*FSS-Tools*)
Никакого инструмента или *MS*	*MS*	Сумма уровня *N* и уровня *N*+1
Никакого инструмента или *MS*	*PNS*	Недопустимая комбинация
Никакого инструмента или *MS*	Интенсивность	Уровень *N*+1
Никакого инструмента или *MS*	*PNS* и интенсивность	Уровень *N*+1
PNS	Никакого инструмента	Уровень *N*+1
PNS	*MS*	Уровень *N*+1
PNS	Интенсивность	Уровень *N*+1
PNS	*PNS*	Уровень *N*+1
PNS	*PNS* и интенсивность	Уровень *N*+1

5.2.3 Декодирование *individual_channel_stream (ICS)* и *ics_info*

Элементы *Individual Channel Stream* (отдельный поток канала) *(ICS)* являются подэлементами обоих объектных типов аудио стиля *MPEG-2 AAC* (*AAC Main, LC, SSR, LTP*), и объектного типа аудио масштабируемого *AAC* (*AAC Scalable*).

5.2.3.1 Определения

5.2.3.1.1 Элементы данных

individual_channel_stream () Содержит данные, необходимые, чтобы декодировать один канал.

ics_info () — Содержит дополнительную информацию, необходимую, чтобы декодировать *individual_channel_stream* для элементов *SCE* и *CPE*. *individual_channel_streams* элемента *channel_pair_element ()* могут совместно использовать один общий *ics_info*.

ics_reserved_bit — Флаг зарезервирован для будущего использования. Должен быть равен '0'.

window_sequence — Указывает последовательность окон как, определено в таблице 128.

window_shape — 1-битовое поле, которое определяет, какое окно используется для правой части из этого окна анализа.

max_sfb — Количество полос масштабного коэффициента, переданных на любую группу в *ics_info (), aac_scalable_main_element ()* или *aac_scalable_extension_element ()*. Для элементов *TwinVQ max_sfb* не передается непосредственно количество полос, но вычисляется в декодере из другой переданной информации как определено в 5.2.2.6.1.

scale_factor_grouping — Битовое поле, которое содержит информацию о группировке коротких спектральных данных.

5.2.3.1.2 Элементы справки

scalefactor window band — Термин для полос масштабных коэффициентов в пределах окна, данный в таблицах 129—147.

scalefactor band — Элемент для полосы масштабного коэффициента в пределах группы. В случае *EIGHT_SHORT_SEQUENCE* и группировки полоса масштабного коэффициента может содержать несколько полос окна масштабного коэффициента соответствующей частоты. Для всех других *window_sequences* полосы масштабного коэффициента и полосы окна масштабного коэффициента идентичны.

g — Групповой индекс.

win — Индекс окна в пределах группы.

sfb — Индекс полосы масштабного коэффициента в пределах группы.

swb — Индекс полосы окна масштабного коэффициента в пределах окна.

bin — Индекс коэффициента.

num_window_groups — Количество групп окон, которые совместно используют один набор масштабных коэффициентов.

window_group_length [g] — Количество окон в каждой группе.

bit_set (bit_field, bit_num) — Функция, которая возвращает значение количества битов *bit_num* поля *bit_field* (самый правый бит является битом 0).

num_windows — Количество фактической последовательности окон.

num_swb_long_window — Количество полос масштабного коэффициента для длинных окон. Это число должно быть выбрано в зависимости от частоты дискретизации. См. 5.4.

num_swb_short_window — Количество полос окна масштабного коэффициента для коротких окон. Это число должно быть выбрано в зависимости от частоты дискретизации. См. 5.4.

num_swb — Количество полос окна масштабного коэффициента в случае *EIGHT_SHORT_SEQUENCE* для коротких окон, в других случаях количество полос окна масштабного коэффициента для длинных окон.

swb_offset_long_window [swb] — Таблица, содержащая индекс самого низкого спектрального коэффициента полосы масштабного коэффициента *sfb* для длинных окон. Эта таблица должна быть выбрана в зависимости от частоты дискретизации.

swb_offset_short_window [swb] — Таблица, содержащая индекс самого низкого спектрального коэффициента полосы масштабного коэффициента *sfb* для коротких окон. Эта таблица должна быть выбрана в зависимости от частоты дискретизации.

swb_offset [swb] — Таблица, содержащая индекс самого низкого спектрального коэффициента полосы масштабного коэффициента *sfb* для коротких окон в случае *EIGHT_SHORT_SEQUENCE*, в других случаях для длинных окон.

sect_sfb_offset [g] [section] — Таблица, которая дает число стартовых коэффициентов для *section_data ()* в пределах группы. Это смещение зависит от *window_sequence* и *scale_factor_grouping*.

5.2.3.2 Процесс декодирования

Декодирование *individual_channel_stream (ICS).*

В *individual_channel_stream*, порядок декодирования следующий:

- получить *global_gain;*
- получить *ics_info* (анализ полезной нагрузки потока битов, если общая информация отсутствует);
- получить *section_data,* если присутствует;
- получить *scale_factor_data,* если присутствует;
- получить *pulse_data,* если присутствует;
- получить *tns_data,* если присутствует;
- получить *gain_control_data,* если присутствует;

Если инструмент *HCR* не используется:

- получить *spectral_data,* если присутствует.

Если инструмент *HCR* используется:

- получить *length_of_reordered_spectral_data,* если присутствует;
- получить *length_of_longest_codeword,* если присутствует;
- получить *reordered_spectral_data,* если присутствует.

Процесс восстановления *pulse_data* описывается в 6.3, *tns_data* в 6.9 и *gain_control* данных в 6.12.

Восстановление *ics_info ().*

Для *single_channel_element () 's ics_info* всегда располагается сразу после *global_gain* в *inidividual_channel_stream.* Для элемента канальной пары есть два возможных варианта расположения *ics_info.*

В случае *channel_pair_element (),* если флаг *common_window* устанавливается в 1, оба канала совместно используют то же самое *ics_info ()* (то есть у обоих имеется тот же самый *window_sequence,* тот же самый *window_shape,* тот же самый *scale_factor_grouping,* тот же самый *max_sfb* и т. д.). В ином случае (то есть, когда *common_window* устанавливается в 0) сразу после *global_gain* для каждого из двух *individual_channel_stream* () есть *ics_info.*

ics_info () переносит информацию об окне, связанную с *ICS,* и таким образом позволяет каналам в паре *channel_pair* переключиться отдельно при необходимости. Кроме того, он переносит *max_sfb,* который устанавливает верхнюю границу числа *ms_used []* и биты *predictor_used [],* которые должны быть переданы. Если *window_sequence* является *EIGHT_SHORT_SEQUENCE,* то передается *scale_factor_grouping.* Если ряд коротких окон формирует группу, тогда они совместно используют масштабные коэффициенты, а также позиции интенсивности стерео и информацию *PNS* и имеют чередование своих спектральных коэффициентов. Первое короткое окно всегда является новой группой, таким образом никакой бит группировки не передается. Последующие короткие окна находятся в той же самой группе, если соответствующий бит группировки равен 1. Новая группа запускается, если соответствующий бит группировки равен 0. Предполагается, что у сгруппированных коротких окон имеются подобные статистики сигналов. Поэтому их спектры чередуются, чтобы разместить коррелированные коэффициенты друг за другом.

Восстановление *section_data ().*

В *ICS* восстанавливается информация об одном длинном окне или восьми коротких окнах. *section_data ()* является первым полем, которое будет декодироваться, и описывает коды Хаффмана, которые применяются к полосам коэффициента масштабирования в *ICS.* Данные раздела имеют следующую форму:

sect_cb — кодовая книга для раздела и *sect_len* — длина раздела.

Эта длина восстанавливается путем чтения полезной нагрузки потока битов последовательно для длины раздела, добавляя значение *escape* к полной длине раздела, пока не будет найдено значение не-*escape*, которое добавляется, чтобы установить полную длину раздела. Если устанавливается флаг *aacSectionDataResilienceFlag, sect_len_incr* не передается, но устанавливается в одно из значений по умолчанию в случае кодовой книги для раздела 11 или в диапазоне 16 и 31. Заметим, что в пределах каждой группы разделы должны описывать полосы масштабного коэффициента от нуля до *max_sfb* так, чтобы первый раздел в пределах каждой группы начинался в нулевых полосах, а последний раздел в пределах каждой группы заканчивался в *max_sfb.*

Данные секционирования описывают кодовую книгу и затем длину раздела, используя эту кодовую книгу, начиная с первой полосы масштабного коэффициента и продолжая пока не будет достигнуто общее количество полос масштабного коэффициента.

После того, как это описание обеспечивается, весь *scalefactors* и спектральные данные, соответствующие кодовой книге ноль, обнуляются и никакие значения, соответствующие этим масштабным коэффици-

ентам или спектральным данным, не будут переданы. Сканируя данные масштабного коэффициента важно отметить, что масштабные коэффициенты для любых полос масштабного коэффициента, кодовые книги Хафмана которых являются нулем, будут опущены. Точно так же опускаются все спектральные данные, связанные с кодовой книгой Хафмана нуль.

Кроме того, спектральные данные, связанные с полосами масштабного коэффициента, у которых имеется кодовая книга интенсивности, не будут переданы, но позиции интенсивности стерео будут переданы вместо масштабных коэффициентов, как описано в 6.8.1.4.

Анализ и декодирование *scalefactor_data()*.

Для каждой полосы масштабного коэффициента, которая не находится в разделе, кодированном нулевой книгой шифров (*ZERO_HCB*), передается масштабный коэффициент. Они будут обозначены как 'активные' полосы масштабного коэффициента, а соответствующие масштабные коэффициенты как активные масштабные коэффициенты. Первый элемент данных в *ICS* глобальное усиление, обычно является значением первого активного масштабного коэффициента. Все масштабные коэффициенты (а также позиции стерео и энергии pns) передаются, используя кодированный по Хаффману *DPCM*, соответственно предыдущему активному масштабному коэффициенту (соответственно предыдущей позиции стерео или предыдущей энергии *pns*. См. 6.2 и 6.3). Первый активный масштабный коэффициент дифференцированно кодируется относительно глобального усиления. Если какие-либо позиции стерео интенсивности приняты с вкрапленными элементами масштабного коэффициента *DPCM*, они отправляются модулю интенсивности стерео и не включаются в кодирование *DPCM* значений масштабного коэффициента. Это также применяется к энергиям *PNS*. Значение первого активного масштабного коэффициента обычно передается как *global_gain* с первым масштабным коэффициентом *DPCM*, имеющим нулевое значение. Как только масштабные коэффициенты декодированы, фактические значения находятся через функцию мощности.

Анализ и декодирование *spectral_data ()*.

Спектральные данные восстанавливаются как последняя часть анализа *ICS*. Они состоят из всех необнуленных коэффициентов, остающихся в спектре или спектрах, упорядоченных как описано в *ICS_info*. Для каждой ненулевой, кодовой книги не интенсивности данные восстанавливаются с применением декодирования Хафмана в четверках или парах, как обозначено в инструменте бесшумного кодирования (См. 6.9). Если спектральные данные ассоциированы со сборником кодов Хафмана без знака, необходимые биты знака следуют за кодовой комбинацией Хафмана. В случае кодовой книги *ESCAPE*, если получается какое-либо значение *escape*, соответствующая *escape*-последовательность появится после этого кода Хаффмана. Может быть ноль, одна или две escape-последовательности для каждой кодовой комбинации в сборнике кодов *ESCAPE*, как обозначено присутствием значений *escape* в этой декодируемой кодовой комбинации. Для каждого раздела декодирование Хаффмана продолжается, пока не будут декодированы все спектральные значения в этом разделе. Как только все разделы будут декодированы, данные умножаются на декодированные масштабные коэффициенты и в случае необходимости устраняется чередование.

Если используется инструмент *HCR*, спектральные данные больше не состоят из последовательных кодовых комбинаций. Относительно *HCR* две или четыре строки всех данных, которые необходимы чтобы декодировать, рассматриваются как кодовая комбинация. Они включают кодовую комбинацию Хаффмана, биты знака и *escape*-последовательности.

5.2.3.3 Окна и последовательности окон

Квантование и кодирование производится в частотной области. С этой целью временной сигнал отображается в кодере в частотную область. Декодер выполняет инверсное отображение как описано в 6.11. В зависимости от сигнала кодер может изменить временное/частотное разрешение, используя два различных окна: *LONG_WINDOW* и *SHORT_WINDOW*. Чтобы переключиться между окнами, используются окна перехода, *LONG_START_WINDOW* и *LONG_STOP_WINDOW*. Таблица 127 перечисляет окна, определяет соответствующую длину преобразования и схематично показывает форму окон. Используются две длины трансформации: 1024 (или 960) (называемая длинным преобразованием) и 128 (или 120) коэффициентов (называемая коротким преобразованием).

Последовательности окон составляются из окон таким способом, что *raw_data_block* всегда содержит данные, представляющие 1024 (или 960) выходных выборок. Элемент данных *window_sequence* указывает последовательность окон, которая фактически использует списки того, как последовательности окон составляются из отдельных окон.

5.2.3.4 Полосы масштабного коэффициента и группировка

Многие инструменты декодера выполняют операции на группах последовательных спектральных значений, названных полосами масштабного коэффициента (сокращенно *'sfb'*). Ширина полос масштабного

коэффициента создается в имитации критических полос человеческой слуховой системы. По этой причине число полос масштабного коэффициента в спектре и их ширина зависят от длины преобразования и частоты дискретизации. Таблицы от 129—147 перечисляют смещение к началу каждой полосы масштабного коэффициента на длинах преобразования 1024 (960) и 128 (120) и на частотах дискретизации.

Чтобы уменьшать количество дополнительной информации в случае последовательностей, которые содержат *SHORT_WINDOWS,* последовательные *SHORT_WINDOWs* могут быть сгруппированы. Информация о группировке содержится вэлементе данных *scale_factor_grouping.* Группировка означает, что передается только один набор масштабных коэффициентов для всех сгруппированных окон, как будто имеется только одно окно. Масштабные коэффициенты применяются к соответствующим спектральным данным во всех сгруппированных окнах. Чтобы увеличить эффективность бесшумного кодирования (см. 6.3), спектральные данные группы передаются в чередующемся порядке, приведенном в 5.2.3.5. Чередование производится на основе полосы масштабного коэффициента полосой масштабного коэффициента так, чтобы спектральные данные могли быть сгруппированы, образуя виртуальную полосу масштабного коэффициента, к которой может быть применен общий масштабный коэффициент. Выражение 'полоса масштабного коэффициента' (сокращение '*sfb*') обозначает эти виртуальные полосы масштабного коэффициента. Если упоминаются полосы масштабного коэффициента единственных окон, используется выражение 'полоса окна масштабного коэффициента' (сокращение '*swb*'). Из-за ее влияния на полосы масштабного коэффициента группировка влияет на значение *section_data* (см. 6.3), порядок спектральных данных (см. 5.2.3.5), и общее количество полос масштабного коэффициента. Для *LONG_WINDOW* полосы масштабного коэффициента и полосы окна масштабного коэффициента идентичны, так как есть только одна группа с одним окном.

Чтобы уменьшать количество информации, необходимой для передачи дополнительной информации, специфической для каждой полосы масштабного коэффициента, передается элемент данных *max_sfb.* Его значение является на единицу большим, чем самая высокая активная полоса масштабного коэффициента во всех группах.*max_sfb* оказывает влияние на интерпретацию данных раздела (см. 6.3), передачу масштабных коэффициентов (см.6.3 и 6.2), передачу данных прогнозирующего устройства (см. 6.6 и 6.7), управляющую информацию *FSS* в режимах кодирования моно-стерео (см. 6.14 и 5.2.2), и передачу *ms_mask* (см. 6.8.1).

Поскольку полосы масштабного коэффициента являются базовым элементом алгоритма кодирования, некоторые переменные и массивы справки должны описывать процесс декодирования во всех инструментах, использующих полосы масштабного коэффициента. Эти переменные справки зависят от *sampling_frequency, window_sequence, scalefactor_grouping* и *max_sfb* и должны быть созданы для каждого *raw_data_block.* Псевдокод, показанный ниже, описывает:

- как определить число окон в *window_sequence,* названное *num_windows;*
- как определить число *window_groups,* названное *num_window_groups;*
- как определить число окон в каждой группе, названное *window_group_length [g] ;*
- как определить общее количество полос окна масштабного коэффициента, названных *num_swb* для фактического типа окна;
- как определить *swb_offset [swb],* смещение первого коэффициента в полосе окна масштабного коэффициента, названного *swb* фактически используемого окна;
- как определить *sect_sfb_offset [g] [section],* смещение первого коэффициента в разделе, названного *section.* Это смещение зависит *от window_sequence* и *scale_factor_grouping* и необходимо, чтобы декодировать *spectral_data ().*

Окно длинного преобразования всегда описывается как *window_group,* содержащее единственное окно. Так как число полос масштабного коэффициента и их ширина зависят от частоты дискретизации, переменные, на которые оказывается влияние, индексируются индексом *sampling_frequency_index,* чтобы выбрать соответствующую таблицу.

```
fs_index = sampling_frequency_index;
switch (window_sequence) {
case ONLY_LONG_SEQUENCE:
case LONG_START_SEQUENCE:
case LONG_STOP_SEQUENCE:
num_windows = 1;
num_window_groups = 1;
window_group_length[num_window_groups-1] = 1;
num_swb = num_swb_long_window[fs_index];
```

```
/* preparation of sect_sfb_offset for long blocks */
/* also copy the last value! */
for(i = 0; i < max_sfb + 1; i++) {
sect_sfb_offset[0][i] = swb_offset_long_window[fs_index][i];
swb_offset[i] = swb_offset_long_window[fs_index][i];
}
break;
case EIGHT_SHORT_SEQUENCE:
num_windows = 8;
num_window_groups = 1;
window_group_length[num_window_groups-1] = 1;
num_swb = num_swb_short_window[fs_index];
for (i = 0; i < num_swb_short_window[fs_index] + 1; i++)
swb_offset[i] = swb_offset_short_window[fs_index][i];
for (i = 0; i < num_windows-1; i++) {
if (bit_set(scale_factor_grouping,6-i)) == 0) {
num_window_groups += 1;
window_group_length[num_window_groups-1] = 1;
}
else {
window_group_length[num_window_groups-1] += 1;
}
}
/* preparation of sect_sfb_offset for short blocks */
for (g = 0; g < num_window_groups; g++) {
sect_sfb = 0;
offset = 0;
for (i = 0; i < max_sfb; i++) {
width = swb_offset_short_window[fs_index][i+1] -
swb_offset_short_window[fs_index][i];
width *= window_group_length[g];
sect_sfb_offset[g][sect_sfb++] = offset;
offset += width;
}
sect_sfb_offset[g][sect_sfb] = offset;
 }
break;
default:
break;
}
```

5.2.3.5 Порядок спектральных коэффициентов в *spectral_data*

Если используется инструмент переупорядочивания кодовой комбинации Хаффмана (*HCR*), применяется подпункт 6.16.3.3. Иначе спектральные коэффициенты упорядочиваются следующим образом:

- для окон *ONLY_LONG_SEQUENCE* (*num_window_groups* = 1, *window_group_length* [0]=1) спектральные данные находятся в возрастающем спектральном порядке;

- для окна *EIGHT_SHORT_SEQUENCE* спектральный порядок зависит от группировки следующим образом:

а) группы упорядочиваются последовательно;

б) в пределах группы полоса масштабного коэффициента состоит из спектральных данных всех сгруппированных *SHORT_WINDOWs* для ассоциированной полосы окна масштабного коэффициента.

5.2.3.6 Длина слова вывода

Глобальное усиление для каждого звукового канала масштабируется так, что целочисленная часть вывода *IMDCT* может использоваться непосредственно в качестве 16-разрядного аудиовыхода *PCM* к цифро-аналоговому преобразователю (*D/A*). Это — режим работы по умолчанию и приведет он к корректным уровням громкости. Если у декодера есть цифроаналоговый преобразователь с разрешением больше чем

16 разрядов, тогда масштаб вывода *IMDCT* может быть увеличен так, что включается соответствующее число дробных битов, чтобы сформировать требуемый размер слова *D/A*. В этом случае уровень вывода преобразователя может соответствовать уровню из 16-разрядного *D/A*, но будет иметь преимущество большего динамического диапазона сигнала и пониженного минимального уровня шума преобразователя. Точно так же могут быть обеспечены более короткие длины слова *D/A*.

5.2.4 Полезные нагрузки для аудио объекта типов *ER AAC LC, ER AAC LTP, ER AAC LD, ER AAC ELD* и масштабируемого *ER AAC*

Для AAC существуют два вида синтаксиса полезной нагрузки потока битов: масштабируемый и многоканальный. Чтобы получить их устойчивые к ошибкам аналоги нужно внести следующие изменения:

- многоканальный *AAC:* синтаксис высокоуровневой полезной нагрузки был изменений. *raw_data_block () —* в зависимости от *AOT*, замененный либо на *er_raw_data_block ()* как описано в таблице 19 или *er_raw_data_block_eld* () как описано в таблице 75. Из-за этой модификации *coupling_channel_element* (), *data_stream_element* (), *program_config_element* () и *fill_element* () не поддерживаются в пределах устойчивого к ошибкам синтаксиса полезной нагрузки потока битов. Для определения функции помощника *bits_to_decode ()* см. 5.2.1;

- масштабируемый *AAC*: *синтаксис aac_scalable_main_element()* и *aac_scalable_extension_element*() не изменяется для устойчивости к ошибкам.

Относительно *ER AAC AudioObjectTypes* вышеупомянутые высокоуровневые олезные нагрузки, описанные *er_raw_data_block ()* таблице 19, *er_raw_data_block_eld ()* в таблице 75, *aac_scalable_main_element ()* в таблице 13, *aac_scalable_extension_element ()* в таблице 14 обрабатываются как логические полезные нагрузки потока битов.

Чтобы получить эту логическую полезную нагрузку потока битов из полезной нагрузки физического потока битов, требуются шаги предварительной обработки.

На стороне кодера элементы данных подразделяются на различные категории в зависимости от их чувствительности к ошибкам и собираются в экземплярах этих категорий чувствительности к ошибкам.

5.2.4.1 Присвоение категории чувствительности к ошибкам

Следующая таблица дает краткий обзор используемых для *AAC* категорий чувствительности к ошибкам *(channel_pair_element () = CPE, individual_channel_stream () = ICS, extension_payload* () = *EPL*):

Таблица 94 — Присвоение категории чувствительности к ошибкам

Категория	Полезная нагрузка	Обязательная	Ведет/может привести к одному экземпляру	Описание
0	Основная	Да	*CPE* /уровень стерео	Обычно используемая дополнительная информация
1	Основная	Да	*ICS*	Зависимая от канала дополнительная информация
2	Основная	Нет	*ICS*	Устойчивые к ошибкам данные масштабного коэффициента
3	Основная	Нет	*ICS*	Данные *TNS*
4	Основная	Да	*ICS*	Спектральные данные
5	Расширенная	Нет	*EPL*	Тип расширения / *data_element_version*
6	Расширенная	Нет	*EPL*	Данные *DRC*
7	Расширенная	Нет	*EPL*	Заполнение битами
8	расширенная	Нет	*EPL*	Данные *ANC*
9	Расширенная	Нет	*EPL*	Данные *SBR*

Таблица 148 показывает присвоение категорий для основной полезной нагрузки (поддерживаемыми элементами являются *SCE, LFE,* и *CPE*). В пределах этой таблицы "-" означает, что этот элемент данных не появляется в пределах этой конфигурации.

Таблица 149 показывает присвоение категорий для расширенной полезной нагрузки.

5.2.4.2 Образцы категории и структура ее зависимости

Подразделение на образцы производится на основе фрейма в случае масштабируемого синтаксиса в дополнение к основанию уровня. Порядок образцов в пределах устойчивого к ошибкам фрейма/уровня *AAC* так же как структура зависимости в случае нескольких элементарных потоков присваиваются согласно следующим правилам:

Таблица 95 — Структура зависимости в пределах полезной нагрузки *ER AAC*

Уровень иерархии	Устойчивый к ошибкам многоканальный синтаксис	Устойчивый к ошибкам масштабируемый синтаксис
Фрейм/уровень	Основная полезная нагрузка, сопровождаемая полезной нагрузкой расширения	
Основная полезная нагрузка	Порядок синтаксических элементов следует порядку, установленному в таблице 19	Обычно используемая дополнительная информация, сопровождаемая зависимой от канала информацией
Расширенная полезная нагрузка	Никаких правил относительно порядка нескольких *EPLs* не дается, вид полезной нагрузки расширения может быть идентифицирована *extension_type*	
Синтаксический элемент в основной полезной нагрузке	Обычно используемая дополнительная информация, сопровождаемая зависимой от канала информацией	—
Зависимая от канала информация	Левый канал, сопровождаемый правым каналом	
Зависимая от канала информация/*EPL*	Структура зависимости согласно номерам категорий	

Примечание — Зависимая от канала информация состоит из потоков *individual_channel_stream () (ICS).* Исключения:

- *ltp_data_present* и *ltp_data* () обрабатываются как зависимая от канала информация, даже если они не часть *ICS;*

- для масштабируемого объектного типа *ER AAC, tns_data_present, tns_data (), diff_control_data ()* и *diff_control_data_lr ()* обрабатываются как зависимая от канала информация, даже если они не часть *ICS.*

5.2.5 Полезные нагрузки для аудио объекта типов *TwinVQ* и *TwinVQ ER*

5.2.5.1 Определения

tvq_scalable_main_element () — Элемент данных для декодирования базовых данных *TwinVQ*.

tvq_scalable_extension_element () — Элемент данных для декодирования масштабируемых данных *TwinVQ*.

vq_single_element () — Блок данных, содержащий один фрейм синтаксических элементов для элемента базового уровня *TwinVQ* и элемента уровня расширения *TwinVQ*.

ppc_present — Указывает на активацию периодического пикового кодирования компонент.

postprocess_present — Указывает на активацию постпроцессора в спектральной нормализации.

bandlimit_present — Указывает на активацию управления полосой пропускания коэффициентов *MDCT*.

fb_shift — Указывает расположение активной полосы частот в уровне расширения.

index_blim_h — Указывает верхнюю границу для процесса управления полосой пропускания инструмента нормализации спектра.

index_blim_l — Указывает нижнюю границу для процесса управления полосой пропускания инструмента нормализации спектра.

*index_shape*0 — Указывает номер кодового вектора кодовой книги формы 1 и полярность кодового вектора для инструмента квантования чередующегося вектора.

*index_shape*1 — Указывает номер кодового вектора кодовой книги формы 1 инструмента квантования чередующегося вектора.

index_env — Указывает номер кодового вектора кодовой книги, огибающей *Bark-scale* инструмента нормализации спектра.

index_fw_alf — Указывает переключатель прогноза кодирования огибающей *Bark-scale* инструмента нормализации спектра.

index_gain — Указывает коэффициент усиления инструмента нормализации спектра.

index_gain_sb — Указывает коэффициент усиления подблока инструмента нормализации спектра.

*index_lsp*0 — Указывает переключатель прогноза *MA LSP* инструмента нормализации спектра.

*index_lsp*1 — Указывает номер кодового вектора *LSP VQ* первой стадии в инструменте нормализации спектра.

*index_lsp*2 — Указывает номер кодового вектора *LSP VQ* второй стадии в инструменте нормализации спектра.

*index_shape*0_*p* — Указывает номер кодового вектора кодовой книги 0 кодирования периодического пикового компонентного в инструменте нормализации спектра.

*index_shape*1_*p* — Указывает номер кодового вектора кодовой книги 1 кодирования периодического пикового компонентного в инструменте нормализации спектра.

index_pit — Указывает основную частоту периодического пикового компонента в инструменте нормализации спектра.

index_pgain — Указывает коэффициент усиления периодического пикового компонента в инструменте нормализации спектра.

5.2.5.2 Установка параметров

Для декодирования элемента базового уровня *TwinVQ* и элемента уровня расширения *TwinVQ* используют следующие параметры:

N_CH — число каналов, определенное системным уровнем;

N_DIV — число субвекторных разделов для чередующегося векторного квантования. Вычисляется согласно состоянию декодера;

N_SF — число подблоков группы фильтров в фрейме;

FW_N_DIV — число разделов кодовой книги для квантования огибающей *Bark-scale*;

LSP_SPLIT — число подвекторов для *LSP VQ;*

N_DIV_P — число субвекторных разделов для кодирования периодического пикового компонента.

На эти параметры также делается ссылка из инструмента квантования чередующегося вектора и инструмента нормализации спектра. Следующие параметры имеют постоянные величины как показано ниже.

Таблица 96 — Параметры, которые имеют постоянные величины

N_SF(SHORT) = 8
N_SF(LONG) = 1
FW_N_DIV = 7
LSP_SPLIT = 3
N_DIV_P = 2*N_CH

5.2.5.3 Распределение битов

5.2.5.3.1 Инструмент нормализации спектра

Для упомянутых ниже элементов синтаксиса число битов определяется согласно параметрам *lyr* и *window_sequence:*

Таблица 97 — Распределение битов элементов синтаксиса

Наименование переменных	Наименование числа битов	*lyr* = 0 короткий	*lyr* = 0 длинный	lyr ≥ 1 короткий	lyr ≥ 1 длинный	Времена на фрейм
fb_shift	—	0	0	2	2	*N_CH*
index_blim h	—	2/0	2/0	0	0	*N_CH*
index_blim l	—	1/0	1/0	0	0	*N_CH*
index_env	*FW_N_BIT*	6	0	6	0	*FW_N_DIV*N_CH*
index_fw_alf	—	1	0	1	0	*N_CH*
index_gain	*GAIN_BIT*	9	9	8	8	*N_CH*
index_gain_sb	*SUB_GAIN_BIT*	0	4	0	4	*N_SF*N_CH*
*index_lsp*0	*LSP*0_*BIT*	1	1	1	1	*N_CH*
*index_lsp*1	*LSP*1_*BIT*	6	6	6	6	*N_CH*
*index_lsp*2	*LSP*2_*BIT*	4	4	4	4	*LSP_SPLIT*N_CH*
*index_shape*0_*p*	*MAXBIT_P*+1	7/0	0	0	0	*N_DIV_P*
*index_shape*1_*p*	*MAXBIT_P*+1	7/0	0	0	0	*N_DIV_P*
index_pit	*BASF_IT*	8/0	0	0	0	*N_CH*
index_pgain	*PGAIN_BIT*	7/0	0	0	0	*N_CH*

5.2.5.3.2 Инструмент чередующегося *VQ*

Параметр *N_DIV*, число битов индекса формы кода 0, *bits0* и число битов индекса кода формы 1, *bits*1, вычисляются следующим образом.

Число битов для дополнительной информации *bits_for_side_information* вычисляется следующим образом:

bits_for_side_information = WS_TBIT + OPT_TBIT + LSP_TBIT + GAIN_TBIT + FW_TBIT + PIT_TBIT + + BAND_TBIT+used_bits,

где *WS_TBIT* — число битов для флагов последовательности окон, формы окна, объединенного кодирования стерео (*MS_present*), *LTP* и инструментов *TNS.*

OPT_TBIT является числом битов для флагов полосы пропускания, *ppc*, постобработки и выбора активной полосы.

Если *ppc_present* активируется, используются 43 бита на канал (*PIT_TBIT*) и *bandlimit_present* активируется, используются 3 бита на канал (*BAND_TBIT*).

used_bit является числом битов, используемых инструментами помимо инструмента нормализации спектра такими, как объединенные инструменты стерео, инструменты *LTP* и инструменты *TNS.*

LSP_TBIT, GAIN_TBIT, FW_TBIT и *PIT_TBIT* являются числом битов для опционной информации, *lsp* кодирования, кодирования усиления, кодирования огибающей *Bark-scale* и кодирования периодических пиковых компонентов соответственно. Они устанавливаются следующим образом:

```
LSP_TBIT = (LSP_BIT0+LSP_BIT1 + (LSP_BIT2*LSP_SPLIT)) * N_CH;
if (FW_N_BIT> 0) {
FW_TBIT = ((FW_N_BIT * FW_N_DIV + 1) * N_SF) * N_CH;
}
 else {
FW_TBIT = 0;
}
switch (window_sequence) {
case EIGHT_SHORT_SEQUENCE:
GAIN_TBIT = (GAIN_BIT + SUB_GAIN_BIT * N_SF) * N_CH;
PIT_TBIT = 0;
break;
```

default:
*GAIN_TBIT = GAIN_BIT * N_CH;*
if (ppc_present = = TRUE}
*PIT_TBIT = (MAXBIT_P + 1) *N_DIV_P*2 + (BASF_BIT + PGAIN_BIT) * N_CH;*
else
PIT_TBIT = 0;
break;
}
Числа битов для дополнительной информации перечисляются в таблице 98.

Таблица 98 — Распределение битов дополнительной информации

Масштабируемый уровень *lyr* *Window_sequence*	0 длинный	0 длинный	0 короткий	0 короткий	≥1 длинный	≥1 длинный	≥1 короткий	≥1 короткий
N_CH	1	2	1	2	1	2	1	2
WS_TBIT	5	9	5	9	0	2	0	2
OPT_TBIT	3	3	1	1	2	4	2	4
LSP_TBIT	19	38	19	38	19	38	19	38
GAIN_TBIT	9	18	41	82	8	16	40	80
FW_TBIT	43	86	0	0	43	86	0	0
BAND_TBIT	3/0	6/0	3/0	6/0	0	0	0	0
PIT_TBIT	43/0	86/0	0	0	0	0	0	0

Число доступных битов, *bits_available_vq* вычисляется следующим образом:
*bits_available_vq = (int) (((FRAME_SIZE * bitrate/sampling_frequency)*/8+0,5) *8) - *bits_for_side_information*,* где скорость передачи дается системным параметром в бит/с и частота дискретизации дается в правой графе таблицы 82.

Число подвектора формы, *N_DIV* и число битов для индексов кода формы, *bits*0 и *bits*1, вычисляются следующим образом:
N_DIV = ((интервал) ((*bits_available_vq* + *MAXBIT**2-1)/(*MAXBIT**2)));
bits = (bits_available_vq + N_DIV - 1 - idiv)/N_DIV;
*bits*0 = *(int) (bits*+1)/2;*
*bits*1 = *(int) bits /2,*
где *MAXBIT* является максимальным количеством битов кода формы.

5.2.5.3.3 Масштабируемый кодер посредством *TwinVQ*

Масштабируемый кодер может быть создан, располагая каскадом квантователя ядра *TwinVQ* и расширения *TwinVQ*. Процесс декодирования является прямым. Коэффициенты *MDCT* восстанавливаются при использовании демультиплексированной полезной нагрузки потока битов для каждого уровня. Каждый уровень расширения охватывает другую часть коэффициентов, области которых адаптивно определяются параметрами сдвига. Выполняется суммирование этих восстановленных коэффициентов. Набор фильтров *IMDCT* преобразует обратно спектр (коэффициенты *MDCT*) в сигнал временной области, используя окна синтеза и методы перекрытия/добавления. Если декодировать только базовые коэффициенты *MDCT*, декодируемые базовые коэффициенты должны быть переданы непосредственно набору фильтров. Базовый уровень этого масштабируемого кодера может быть объединен с инструментами *AAC*, такими как *TNS*, *LTP* и объединенным кодированием стерео. Этот кодер не поддерживает ни кодер каскадирования моно-стерео, ни управление *FSS*.

Если в *tvq_scalable_element* () истина следующее условие:
(*MS_mask* == 1) && *(this_layer_stereo == 1)* &&
(window_sequence == EIGHT_SHORT_SEQUENCE) &&
и в предыдущем уровне (уровнях) нет никакой информации о *scale_factor_grouping*, тогда *num_window_group* должен быть установлен в '1' (*scale_factor_grouping* = 0x7F).

5.2.6 Полезные нагрузки для аудио объектного типа *ER BSAC*

5.2.6.1 Декодирование полезной нагрузки для аудио объектного типа *ER BSAC (bsac_payload ())*

Мелкоструктурная масштабируемость может создать большое количество служебной информации, если попытаться передать мелкоструктурные уровни по нескольким элементарным потокам (*ES*). Чтобы уменьшить служебную нагрузку и эффективно реализовать мелкоструктурную масштабируемость в текущей системе MPEG-4, сервер может организовать мелкоструктурные аудиоданные в полезной нагрузке, деля мелкоструктурные аудиоданные на уровни большого шага и связывая уровни большого шага нескольких подфреймов. Тогда полезная нагрузка передается по *ES*.

Таким образом, полезная нагрузка, переданная по *ES*, требует наличия процесса перестановки для фактического декодирования.

5.2.6.1.1 Определения

bsac_payload (lay) — последовательность элементов *bsac_lstep_element()s*. Синтаксический элемент полезной нагрузки, переданный по *ES* уровня *layth*. *bsac_payload(lay)* в основном состоит из потоков битов нескольких уровней *layth, bsac_lstep_element ()* нескольких подфреймов.

bsac_lstep_element (frm, lay) — синтаксический элемент для потока битов уровня большого шага *layth* подфрейма *frmth*.

bsac_stream_byte [frm] [offset+i] — *(offset+i)*-ый байт, который извлекается из полезной нагрузки. После того как из всех полезных нагрузок, которые были переданы к получателю, извлекаются потоковые байты *bsac*, эти данные связываются и сохраняются в массиве *bsac_stream_byte* [*frm*] [], который является потоком битов подфрейма *frmth*. Затем мы продолжаем декодировать связанный поток *bsac_stream_byte*[*frm*][], используя синтаксис мелкоструктурной масштабируемости *BSAC*.

5.2.6.1.1.1 Элементы справки

data_available () — функция, которая возвращает "1" пока данные доступны, иначе "0".

LayerStartByte [frm] [lay] — стартовая позиция уровня большого шага *layth* в байтах, которая располагается в потоке битов подфрейма *frmth*. Процесс вычисления этого значения См. в 5.2.6.1.2.

LayerLength [frm] [lay] — длина уровня большого шага в байтах, который располагается в *ES* полезной нагрузки уровня *layth* и связан с потоком битов подфрейма *frmth*. Процесс вычисления этого значения См. в 5.2.6.1.2.

LayerOffset [frm] [lay] — стартовая позиция уровня большого шага фрейма *frmth* в байтах, который располагается в полезной нагрузке *ES* уровня *layth*. Процесс вычисления этого значения См. в 5.2.6.1.2.

frm — индекс фрейма, в котором сохраняются потоковые байты *bsa.c*

lay — индекс уровня большого шага, в котором передаются точно отрегулированные аудиоданные.

numOfSubFrame — количество подфреймов, которые группируются и передаются в суперфрейме, чтобы уменьшить передачу служебной информации.

layer_length — средняя длина уровней большого шага в байтах, которые собираются в полезной нагрузке.

numOfLayer — число уровней большого шага, на которые разделенны мелкоструктурные аудиоданные.

5.2.6.1.2 Процесс декодирования

На уровне *sync (SL)* системы *MPEG*-4 элементарный поток упакован в блоки доступа или их части. Такой пакет называют пакетом *SL*. Блоки доступа *(AU)s* являются единственными семантическими объектами в уровне *sync (SL)* системы MPEG-4, которые должны быть полностью сохранены. *AU* используются в качестве основной единицы для синхронизации и составляются из одного или более пакетов *SL*.

Динамические данные для *BSAC* передаются как полезная нагрузка *SL_Packet* в *Elementary Stream (ES)* базового уровня и *(ES)s* уровня расширения. Динамические данные составляются из уровней большого шага одного или более последующих подфреймов.

Когда пакеты *SL* из *AU* прибывают к получателю, последовательность пакета отображается в полезную нагрузку, которая разделяется в уровни большого шага *bsac_lstep_layer (frm, lay)* для последующих подфреймов. Уровни разделения должны быть связаны с уровнями большого шага, которые передаются в другом *ES*.

В выделенных уровнях большого шага устраняется чередование, и они связываются, чтобы полностью отобразить все мелкоструктурные данные *BSAC*. Затем декодируются связанные потоки битов, используя синтаксис (*bsac_raw_data_block*) для мелкоструктурной масштабируемости, чтобы создать восстановленный сигнал.

Чтобы описать процесс реконструкции, переданной по *ES* полезной нагрузки, необходимы некоторые переменные справки и массивы. Эти переменные справки зависят от уровней, *numOfLayer, numOfSubFrame, layer_length* и *frame_length* и должны быть созданы для того, чтобы отобразить *bsac_raw_data_block* () каждого подфрейма из полезных нагрузок.

Псевдокод, показанный ниже, описывает:

- как вычислить *LayerLength [i] [k]*, длину уровня большого шага, который располагается в точно отрегулированных аудиоданных *bsac_raw_data_block () i*-го подфрейма;

- как подсчитать *LayerOffset [i] [k]*, который указывает стартовую позицию уровня большого шага *i*-го фрейма, который располагается в полезной нагрузке *k*-го *ES (bsac_payload());*

- как вычислить *LayerStartByte [i] [k]*, который указывает стартовую позицию уровня большого шага, который располагается в точно отрегулированных аудиоданных *bsac_raw_data_block* () *i*-го подфрейма

```
for (k = 0; к  < numOfLayer; к  ++) {
LayerStartByte[0][k] = 0;
for (i = 0; i < numOfSubFrame; i++) {
if (k == (numOfLayer-1) ) {
LayerEndByte[i][k] = frame_length[i];
} else {
LayerEndByte[i][k] = LayerStartByte[i][k] + layer_length[k];
if (frame_length[i] < LayerEndByte[i][k])
LayerEndByte [i][k] = frame_length[i];
}
LayerStartByte[i + 1][k] = LayerEndByte[i][k];
LayerLength[i][k] = LayerEndByte[i][k] - LayerStartByte[i][k];
}
}
for (k = 0; к  < numOfLayer; к  ++) {
LayerOffset[0][k] = 0;
for (i = 0; i < numOfSubFrame; i++) {
LayerOffset[i + 1][k] = LayerOffset[i][k] + LayerLength[i][k];
}
}
```

где *frame_length [i]* — длина потока битов *i*-го фрейма, который получается из элемента синтаксиса *frame_length* и *layer_length [i]* — средняя длина уровней большого шага в полезной нагрузке *ES i*-го уровня и получается из *Audio DecoderSpecificInfo*.

5.2.6.2 Декодирование *bsac_raw_data_block ()*

5.2.6.2.1 Определения

5.2.6.2.1.1 Элементы данных

bsac_raw_data_block () — блок необработанных данных, который содержит кодированные аудиоданные, соответствующую информацию и другие данные. *bsac_raw_data_block* () в основном состоит из *bsac_base_element* () и нескольких *bsac_layer_element* (). Там существует модуль, который определяет, есть ли у потока битов *BSAC* расширенная часть.

bsac_base_element () — синтаксический элемент потока битов базового уровня, содержащего кодированные аудио данные, соответствующая информация и другие данные.

frame_length — длина фрейма включая заголовки в байтах.

bsac_header () — содержит общую информацию, используемую для *BSAC*.

header_length — длина заголовков, включая *frame_length*, *bsac_header* () и *general_header* () в байтах. Фактическая длина равна (*header_length+7*) байтов. Однако, если *header_length* равно 0, это значит, что фактическая длина меньше или равна 7 байтам. Если *header_length* равно 15, это значит, что фактическая длина больше или равна (15+7) байтов и должна быть вычислена посредством декодирования заголовков.

sba_mode	указывает, что используется схема сегментированного двоичного арифметического кодирования (*SBA*), если этот элемент равен 1. Иначе используется общая схема двоичного арифметического кодирования.
top_layer	главный индекс уровня масштабируемости.
base_snf_thr	порог значения, используемый для кодирования секционированных данных базового уровня.
base_band	указывает максимальную линию спектра базового уровня. Если *window_sequence* является *SHORT_WINDOW*, 4**base_band* является максимальной линией спектра. Иначе максимальной линией спектра является 32**base_band.*
max_scalefactor [ch]	максимальное значение масштабных коэффициентов.
cband_si_type [ch]	дополнительная информации о типе полосы кодирования. Используя этот элемент, может быть установлено наибольшое значение *cband_si's* и арифметической модели для декодирования *cband_si*, как показано в таблице А.31.
base_scf_model [ch]	арифметическая модель для декодирования масштабных коэффициентов в базовом уровне.
enh_scf_model [ch]	арифметическая модель, используемая для декодирования масштабных коэффициентов в других уровнях расширения.
max_sfb_si_len [ch]	максимальная длина, которая может использоваться на канал для того, чтобы кодировать дополнительную информацию о полосе масштабного коэффициента, включая масштабный коэффициент и соответствующую стерео информацию в пределах полосы масштабного коэффициента. Фактическая максимальная длина равна (*max_sfb_si_len*+5). Это значение используется для того, чтобы определить размер потока битов каждого уровня.
general_header ()	содержит данные заголовка для общего аудиокодирования.
reserved_bit	бит зарезервированн для будущего использования.
window_sequence	указывает последовательность окон.
window_shape	однобитовое поле, которое определяет, какое окно используется для замыкающей части этого аналитического окна.
max_sfb	число полос масштабного коэффициента, переданных на группу.
scale_factor_grouping	битовое поле, которое содержит информацию о группировке коротких спектральных данных.
pns_data_present	флаг, указывающий будет ли используйтоваться (1) или не будет (0) перцепционная шумовая замена.
pns_start_sfb	полоса масштабного коэффициента, с которой стартует инструмент *pcns*.
MS_mask_present	двухбитовое поле указывает, что маска стерео: - 00 независимая; - 01 однобитовая маска *MS_used* располагается в части дополнительной информации уровня *sfb, (layer_sfb_si);* - 10 все *MS_used* равны единице; - 11 двухбитовая маска *stereo_info* располагается в части дополнительной информации уровня *sfb, layer_sfb_si* ()).
layer_cband_si ()	содержит дополнительную информацию кодирования, необходимую для арифметического кодирования/декодирования секционированных данных в пределах полосы кодирования.
layer_sfb_si ()	содержит дополнительную информацию полосы масштабного коэффициента, такую как стерео, - *pns* и информацию масштабного коэффициента.
bsac_layer_element ()	синтаксический элемент потока битов уровня расширения, содержащий кодированные аудиоданные для временного периода 1024 (960) выборок, соответствующую информацию и другие данные.
bsac_layer_spectra ()	содержит арифметически кодированные аудиоданные квантованных спектральных коэффициентов, которые вновь добавляются к каждому уровню. См. 5.2.6.2.5 о новых спектральных коэффициентах.
bsac_lower_spectra ()	содержит арифметически кодированные аудиоданные квантованных спектральных коэффициентов, которые ниже, чем спектры, добавленные к каждому уровню.

bsac_higher_spectra () содержит арифметически кодированные аудиоданные квантованных спектральных коэффициентов, которые выше, чем спектры, добавленные к каждому уровню.

bsac_spectral_data () содержит арифметически кодированные аудиоданные квантованных спектральных коэффициентов.

zero_code 32-разрядные нулевые величины, чтобы завершить арифметическое декодирование для части стерео.

sync_word четырехбитовый код, который идентифицирует начало расширенной части. Битовая строка '1111'.

bits_to_decode () функция помощника; возвращает число битов, еще декодированных в текущем *bsac_raw_data_block ()*.

extension_type четырехбитовый код, который идентифицирует тип расширения согласно таблице 99.

Таблица 99 — *extension_type* для *BSAC*

Символ	Значение *extension_type*	Цель
EXT_BSAC_CHANNEL	'1111'	Расширение канала *BSAC*
EXT_BSAC_SBR_DATA	'0000'	Расширение *SBR BSAC*
EXT_BSAC_SBR_DATA_CRC	'0001'	Расширение *SBR* за счет *CRC*
EXT_BSAC_SAC_DATA	'0010'	Расширение окружения *MPEG*
EXT_BSAC_CHANNEL_SBR	'1110'	Расширение канала *BSAC* за счет *SBR*
EXT_BSAC_CHANNEL_SBR_CRC	'1101'	Расширение канала *BSAC* за счет *SBR_CRC*
RESERVED	'0011' ~ '1100'	Зарезервировано

extended_bsac_raw_data_block () блок необработанных данных, который содержит кодированные аудиоданные, соответствующую информацию и другие данные для расширенной части. *extended_bsac_raw_data_block* () в основном состоит из *extended_bsac_base_element*() и нескольких *bsac_layer_element*().

extended_bsac_base_element () синтаксический элемент потока битов базового уровня, содержащий кодированные аудиоданные, соответствующую информацию и другие данные для расширенной части *BSAC*.

element_length длина *extended_bsac_raw_data_block* () в байтах. Это используется для надлежащего арифметического декодирования.

channel_configuration_index трехбитовое поле, которые указывают конфигурацию канала аудиовыхода в расширенной части. Каждый индекс определяет число каналов, данных отображению канала в динамик.

Таблица 100 — *channel_configuration_index*

Индекс	Отображение канала в динамиках	Число каналов (*nch*)
0	Центриральный фронтальный динамик	1
1	Левый, правый фронтальные динамики	2
2	Задние объемные динамики	1
3	Левый объемный, правый объемный задние динамики	2
4	Фронтальный динамик низкочастотных эффектов	1
5	Левый, прямо внешние фронтальные динамики	2
6—7	Зарезервировано	—

reserved_bit — бит зарезервированный для будущего использования.

extended_bsac_sbr_data () — синтаксический элемент, который содержит данные расширения *SBR* для *ER BSAC.*

count — начальная длина *extended_bsac_sbr_data*() или *extended_bsac_data*().

esc_count — инкрементная длина *extended_bsac_sbr_data* () или *extended_bsac_data* ().

bs_sbr_crc_bits — циклическая контрольная сумма избыточности для данных расширения *SBR*. Код *CRC* определяется генератором полинома $G^{10}(x) = x^{10} + x^9 + x^5 + x^4 + x + 1$ и начальное значение для вычисления *CRC* является нулем.

bs_header_flag — указывает, существует ли заголовок *SBR.*

sbr_header () — синтаксический элемент, который содержит заголовок *SBR*. См. таблицу 63

bsac_sbr_data () — синтаксический элемент, который содержит данные *SBR* для *ER BSAC.*

bs_fill_bits — биты выравнивания байта.

sbr_single_channel_element () — синтаксический элемент, который содержит данные для единственного элемента канала *SBR*. См. таблицу 65

sbr_channel_pair_element () — синтаксический элемент, который содержит данные для парного элемента канала *SBR*. См. таблицу 66

extended_bsac_data () — синтаксический элемент, содержащий полезную нагрузку, которая должна быть отброшена декодером. Это для дальнейших расширений.

byte_payload — байт, который будет отброшен декодером.

extended_bsac_sac_data () — элемент синтаксиса *extended_bsac_sac_data* () содержит дополнительную информацию пространственного аудиокодирования для декодирования *MPEG Surround.*

bs_crc_flag — указывает, присутствует ли слово *CRC.*

5.2.6.2.1.2 Элементы справки

data_available () — функция, которая возвращается в «1» пока поток битов доступен, иначе «0».

nch — элемент данных, который идентифицирует номер канала.

Scalefactor window band — элемент для полос масштабного коэффициента в пределах окна.

Scalefactor band — элемент для полосы масштабного коэффициента в пределах группы. В случае *EIGHT_SHORT_SEQUENCE* и группировки полоса масштабного коэффициента может содержать несколько полос окна масштабного коэффициента соответствующей частоты. Для всех других *window_sequences scalefactor* полосы масштабного коэффициента и полосы окна масштабного коэффициента идентичны.

G — групповой индекс.

win — индекс окна в пределах группы.

sfb — индекс полосы масштабного коэффициента в пределах группы.

swb — индекс полосы окна масштабного коэффициента в пределах окна.

num_windows_groups — число групп окон, которые совместно используют один набор масштабных коэффициентов. См. 5.2.6.2.4.

window_group_length [g] — число окон в каждой группе. См. 5.2.6.2.4.

bit_set (bit_field, bit_num) — функция, которая возвращает значение номера бита *bit_num для bit_field* (самый правый бит является битом 0.)

num_windows — число окон фактической последовательности окон. См. 5.2.6.2.4.

num_swb_long_window — число полос масштабного коэффициента для длинных окон. Это число должно быть выбрано в зависимости от частоты дискретизации.

Num_swb_short_window — число полос окна масштабного коэффициента для коротких окон. Это число должно быть выбрано в зависимости от частоты дискретизации.

Num_swb — число полос окна масштабного коэффициента для коротких окон в случае *EIGHT_SHORT_SEQUENCE*, иначе число полос окна масштабного коэффициента для длинных окон. См. 5.2.6.2.4

swb_offset_long_window [swb] — таблица, содержащая индекс самого низкого спектрального коэффициента полосы масштабного коэффициента *sfb* для длинных окон. Эта таблица должна быть выбрана в зависимости от частоты дискретизации.

Swb_offset_short_window [swb]	таблица, содержащая индекс самого низкого спектрального коэффициента полосы масштабного коэффициента *sfb* для коротких окон. Эта таблица должна быть выбрана в зависимости от частоты дискретизации.
Swb_offset [g] [swb]	таблица, содержащая индекс самого низкого спектрального коэффициента полосы масштабного коэффициента *sfb* для коротких окон в случае EIGHT_SHORT_SEQUENCE, иначе для длинных окон. См. 5.2.6.2.4.
layer_group [layer]	указывает групповой индекс спектральных данных, которые вновь будут добавлены в уровень масштабируемости.
layer_start_sfb [layer]	указывает индекс самого низкого индекса полосы масштабного коэффициента, который будет вновь добавлен в уровень масштабируемости.
layer_ end _sfb [layer]	указывает самый высокий индекс полосы масштабного коэффициента, который будет вновь добавлен в уровень масштабируемости.
layer_start_cband [layer]	указывает самый низкий индекс полосы кодирования, который будет вновь добавлен в уровень масштабируемости.
layer_ end _cband [layer]	указывает самый высокий индекс полосы кодирования, который будет вновь добавлен в уровень масштабируемости.
layer_start_index [layer]	указывает индекс самого низкого спектрального компонента, который будет вновь добавлен в уровень масштабируемости.
layer_end_index [layer]	указывает индекс самого высокого спектрального компонента, который будет вновь добавлен в уровень масштабируемости.
start_index [g]	указывает индекс самого низкого спектрального компонента, который будет кодирован в группе *g*.
end_index [g]	указывает индекс самого высокого спектрального компонента, который будет кодирован в группе *g*.
layer_data_available ()	функция, которая возвращает «1» пока доступен поток битов каждого уровня, иначе «0». Другими словами, эта функция указывает, доступен ли оставшийся поток битов каждого уровня.
Terminal_layer [layer]	указывает, является ли уровень терминальным уровнем сегмента, который составлен из одного или более уровней масштабируемости. Если сегментированное двоичное арифметическое кодирование не активируется, все эти значения всегда устанавливаются в 0, за исключением значения верхнего слоя. Иначе эти значения определяются в 6.4.6.3.

5.2.6.2.2 Процесс декодирования

5.2.6.2.2.1 Заполнение нулем

Чтобы полностью выполнить арифметическое декодирование, к потоку битов должно быть привязано 32-разрядное нулевое значение. В случае режима *SBA* поток битов фрейма разделяется на несколько сегментов. Таким образом, нулевое значение должно быть привязано ко всем сегментам. Однако в случае режима не *SBA* достаточно одного нулевого заполнения, так как поток битов фрейма не разделяется.

5.2.6.2.2.2 *bsac_raw_data_block*

Суммарноый поток *BSAC, bsac_raw_data_block,* имеет многоуровневую структуру. Во-первых, анализируется и декодируется *bsac_base_element* (), который является потоком битов для основного уровня масштабируемости. Затем анализируется и декодируется *bsac_layer_element* () для следующего уровня расширения. Процедура декодирования *bsac_layer_element* () повторяется пока доступны данные декодируемого потока битов и уровень расширения будет меньше верхнего уровня *top_layer* или равен ему.

5.2.6.2.2.3 *bsac_base_element*

bsac_base_element () состоит из *frame_length, bsac_header, general_header* и *bsac_layer_element* ().

Вначалеиз синтаксиса разыскивается *frame_length.* Он представляет длину фрейма, включая заголовки, в байтах.

Затем элементы синтаксиса для базового уровня, которые составляются из *bsac_header* (), потом *general_header* (), *layer_cband_si* (), *layer_sfb_si* () и *bsac_layer_element*, анализируются. У *bsac_base_element* () есть несколько элементов *bsac_layer_element* (), поскольку базовый слой разделяется на эти несколько подуровней для устойчивости базового уровня к ошибкам. Число подуровней *slayer_size* вычисляется, используя групповой индекс и полосу кодирования как показано в 5.2.6.2.5.

5.2.6.2.2.4 Восстановление *bsac_header*

BSAC обеспечивает мелкоструктурную масштабируемость, которая имеет многоуровневую структуру, один базовый уровень и несколько уровней расширения. Базовый уровень содержит общую допол-

нительную информацию для всех уровней, специальную дополнительную информацию для базового уровня и аудиоданные. Общая дополнительная информация передается в синтаксисах *bsac_header* () и *general_header* ().

Bsac_header состоит из *top_layer*, *header_length*, *sba_mode*, *base_band*, *max_scalefactor*, *cband_si_type*, *base_scf_model* и *enh_scf_model*. Все элементы данных включаются в форму целого числа без знака.

Вначале анализируются 4 бита *header_length*, которые представляют длину заголовков, включая *frame_length*, *bsac_header* и *general_header* в байтах. Длина заголовков равна (*header_length* + 7) *8. Затем анализируется 1-битовый *sba_mode*, который показывает используется ли сегментированное двоичное арифметическое кодирование (*SBA*) или двоичное арифметическое кодирование.

Затем анализируются 6 битов *top_layer*, которые представляют главный индекс уровня масштабируемости, который будет закодирован. Далее, анализируются 2 бита *base_snf_thr*, которые представляют порог значения, используемый для кодирования секционированных (разрядно-модульных) данных базового уровня.

Затем анализируются 8 битов *max_scalefactor*, которые представляет максимальное значение масштабных коэффициентов. Если число каналов не равно 1, это значение анализируется еще раз.

Затем анализируется 5-разрядный *base_band*, который представляют минимальную линию спектра, которая кодируется в базовом уровне. Если последовательность окон является *SHORT_WINDOW*, то 4**base_band* указывает минимальную линию спектра. Иначе минимальную линию спектра указывает 32**base_band*.

Анализируются 5 битов *cband_si_type*, который представляет арифметическую модель *cband_si* и наибольший *cband_si*, который может декодироваться как показано в таблице 31. Анализируются 3 бита *base_scf_model* и *enh_scf_model*, которые представляют таблицу арифметической модели для масштабных коэффициентов базового уровня и других уровней расширения, соответственно. Затем анализируются 4 бита *max_sfb_si_len*, который представляет максимальную длину дополнительной информации полосы масштабного коэффициента, чтобы обеспечить возможность использования в каждом уровне. Максимальная длина равна (*max_sfb_si_len+5*).

5.2.6.2.2.5 Восстановление *general_header*

Порядок декодирования синтаксиса *bsac_header* заключается в получении:

- *reserved_bit*;
- *window_sequence*;
- *window_shape*;
- *max_sfb*;
- *scale_factor_grouping*, если *window_sequence* является *EIGHT_SHORT_SEQUENCE;*
- *pns_present*;
- *pns_start_sfb* если существует;
- флаг *MS_mask_present*, если номер канала равен 2;
- *tns_data_present;*
- *данные TNS*, если присутствуют;
- *ltp_data_present*;
- данные *ltp*, если существуют.

Если номер канала не 1, декодирование другого канала выполняется следующим образом:

- получить *tns_data_present*;
- получить данные *TNS*, если существует;
- получить *ltp_data_present*;
- получить данные *ltp*, если существуют.

Процесс восстановления *tns_data* и *ltp_data* описывается в 6.9.

5.2.6.2.2.6 *bsac_layer_element*

bsac_layer_element () является потоком битов уровня расширения и состоит из *layer_cband_si* (), *layer_sfb_si* (), *bsac_layer_spectra* (), *bsac_lower_spectra* () и *bsac_higher_spectra* ().Процесс декодирования *bsac_layer_element* () следующий:

- декодировать *layer_cband_si*;
- декодировать *layer_sfb_si*;
- декодировать *bsac_layer_spectra*;
- декодировать *bsac_lower_spectra*;
- декодировать *bsac_higher_spectra*.

5.2.6.2.2.7 Декодирование дополнительной информации о кодирования полосы (*layer_cband_si*)

Спектральные коэффициенты делятся на полосы кодирования, которые содержат 32 квантованных спектральных коэффициента для бесшумного кодирования. Полосы кодирования (сокращение *'cband'*) являются основными единицами, используемыми для бесшумного кодирования.

cband_si представляет плоскость *MCB* и таблицу вероятности секционированных битов в пределах полосы кодирования, как показано в таблице А.33. Используя этот *cband_si*, арифметически кодируются секционированные данные каждой полосы кодирования.

cband_si является моделью *arithmetic_coded*, которая дается в элементе синтаксиса *cband_si_type*, как показано в таблице А.31.

5.2.6.2.2.8 Декодирование дополнительной информации полосы масштабных коэффициентов (*layer_sfb_si*)

layer_sfb_si составляется следующим образом:

- декодирование *stereo_info*, *ms_used* или *noise_flag*;
- декодирование масштабных коэффициентов.

5.2.6.2.2.9 Декодирование *stereo_info, noise_flag* или *ms_used*

Процесс декодирования *stereo_info*, *noise_flag* или *ms_used* зависит от *pns_data_present*, номера канала и *ms_mask_present*.

Если данные *pns* не существуют, процесс декодируя происходит следующим образом:

- если *ms_mask_present* равен 0, арифметическое декодирование *stereo_info* или *ms_used* не требуется;
- если *ms_mask_present* равен 2, все значения *ms_used* в этом случае являются единицами. Так, обработка стерео *M/S* для *AAC* производится во всей полосе масштабного коэффициента;
- если *ms_mask_present* равен 1, в этом случае передается однобитовая маска полос *max_sfb* для *ms_used*. Таким образом, *ms_used* арифметически декодируется. Обработка стерео *M/S AAC* производится согласно декодируемому *ms_used*;
- если *ms_mask_present* равен 3, *stereo_info* является арифметически декодированным. *stereo_info* является двухбитовым флагом на полосу масштабного коэффициента, указывающим на кодирование *M/S* или режим кодирования интенсивности. Если *stereo_info* не равно 0, стерео *M/S* или интенсивность стерео *AAC* производятся с этими декодируемыми данными.

Если данные *pns* присутствуют и номер канала 1, процесс декодирования следующий:

- Если номер канала 1 и данные *pns* существуют, шумовой флаг полос масштабного коэффициента между *pns_start_sfb* и *max_sfb* является арифметически декодируемым. Перцепционная шумовая замена выполняется согласно декодируемому шумовому флагу.

Если данные *pns* присутствуют, и номер канала 2, процесс декодирования следующий:

- Если *ms_mask_present* равен 0, шумовой флаг для *pns* является арифметически декодируемым. Перцепционная шумовая замена независимого режима производится согласно декодируемому шумовому флагу;
- если *ms_mask_present* равен 2, все значения *ms_used* являются в этом случае единицами. Так, обработка стерео *M/S* для *AAC* выполняется во всей полосе масштабного коэффициента. Однако обработка pns независимо от флага *pns_data_present* отсутствует;
- если *ms_mask_present* равен 1, в этом случае передается однобитовая маска *max_sfb полос ms_used*. Таким образом, арифметически декодируется *ms_used*. Обработка стерео *M/S* для *AAC* производится согласно декодируемому *ms_used*. Если *ms_used* равен 1, обработка *pns* отсутствует;
- если *ms_mask_present* равен 3, арифметически декодируется *stereo_info*. Если *stereo_info* равен 1 или 2, выполняется обработка стерео *M/S* или стерео интенсивности *AAC* с этими декодируемыми данными, а обработка *pns* отсутствует. Если *stereo_info* равен 3 и полоса масштабного коэффициента меньше чем *pns_start_sfb*, выполняется обработка стерео интенсивности *out_of_phase*. Если *stereo_info* равен 3 и полоса масштабного коэффициента больше или равна *pns_start_sfb*, шумовой флаг для *pns* арифметически декодируется. И затем, если оба шумовых флага двух каналов равны 1, режим замены шума арифметически декодируется. Перцепционным шум заменяется, или производится обработка стерео интенсивности *out_of_phase* согласно режиму замены. Иначе перцепционный шум заменяется, только если шумовой флаг равен 1.

5.2.6.2.2.10 Декодирование масштабных коэффициентов

Спектральные коэффициенты делятся на полосы масштабного коэффициента, которые содержат кратную четырем совокупность квантованных спектральных коэффициентов. Каждая полоса масштабного

коэффициента имеет масштабный коэффициент. Для всех масштабных коэффициентов отличие от максимального значения масштабного коэффициента *max_scalefactor* арифметически кодируется, используя арифметическую модель, данную в таблице А.32. Арифметическая модель, необходимая для того, чтобы кодировать дифференциальные масштаные коэффициенты в базовом уровне, дается как 3-разрядное целое число без знака в элементе данных *base_scf_model*. Арифметическая модель, необходимая для того чтобы кодировать дифференциальные масштабные коэффициенты в других уровнях расширения, дается как 3-разрядное целое число без знака в элементе данных *enh_scf_model*. Максимальное значение масштабного коэффициента дается как 8-битовая PCM в элементе данных *max_scalefactor*.

5.2.6.2.2.11 Секционированные (разрядно-модульные) спектральные данные

В кодере *BSAC* абсолютные значения квантованных спектральных коэффициентов отображаются в секционированную последовательность. Эти нарезанные биты являются символами арифметического кодирования. Каждый нарезанный бит является двоично арифметически кодированным с соответствующей вероятностью (арифметическая модель) от нижнечастотного коэффициента до верхнечастотного коэффициента уровня масштабируемости, начиная с плоскости *Most Significant Bit* (старший значащий бит) (*MCB*) и продолжаясь до плоскости *Least Significant Bit* (младший значащий бит) (*LSB*). Арифметическое кодирование битов знака, связанных с ненулевым коэффициентом, следует за кодированием нарезанного бита, когда нарезанный бит в первый раз равен 1.

Чтобы арифметически кодировать символы (нарезанные биты) должно быть определено значение вероятности. Двоичная таблица вероятности составляется из значений вероятности символа "0". Прежде всего, таблица вероятности выбирается, используя *cband_si,* как показано в таблице А.33. Значение вероятности выбирается среди нескольких значений в выбранной таблице согласно такому контексту, как размер остающегося доступного бита и нарезанных битов последовательных неналоженых 4 спектральных данных.

Для случая нескольких окон на блок связанный и возможно сгруппированный, чередующийся набор спектральных коэффициентов обрабатывается как единственный набор коэффициентов, которые растут от низких до высоких как описано в 5.2.6.2.6. После декодирования, должно быть устранено чередование в наборе спектральных коэффициентов. Набор секционированной последовательности делится на полосы кодирования. Индекс таблицы вероятности (таблица А.33), используемый для кодирования секционированных данных в пределах каждой полосы кодирования, включается в элемент потока битов *cband_si* и передается начиная с полосы кодирования самой низкой частоты и продолжая до полосы кодирования самой высокой частоты. Спектральная информация для всех полос масштабного коэффициента, равных *max_sfb* или больше, обнуляется.

5.2.6.2.2.12 Декодирование нарезанных битов спектральных данных

Спектральная ширина полосы увеличивается пропорционально уровню масштабируемости. Новые спектральные данные добавляются к каждому уровню. Прежде всего, эти новые спектральные данные кодируются в каждом уровне (*bsac_layer_spectra*). Процесс кодирования продолжается пока данные каждого уровня доступны, или все нарезанные биты новых спектров кодируются. Длина доступного потока битов (*available_len []*) инициализируется в начале каждого уровня как описано в 5.2.6.2.5. Предполагаемая длина кодовой комбинации (*est_cw_len*), которая будет декодироваться, вычисляется из процесса арифметического декодирования как описано в 5.2.6.2.7. После арифметического декодирования символа длина доступного потока битов должна быть обновлена, вычитая из нее предполагаемую длину кодовой комбинации. Мы можем определить доступен ли оставшийся поток битов каждого уровня или нет, проверяя массив *available_len [].*

От самого низкого уровня до верхнего уровня новые спектры являются арифметически кодированными уровень за уровнем в вышеупомянутом первом процессе (*bsac_layer_spectra*). Некоторые нарезанные биты не могут быть кодированы из-за отсутствия кодовой комбинации, выделенной уровню. После того как первый процесс кодирования заканчивается текущие значения (*cur_snf*) сохраняются для вторичных процессов кодирования (*bsac_lower_spectra* () и *bsac_higher_spectra* ()). Нарезанные биты, которые остаются некодированными, кодируются, используя сохраненные значения (*unc_snf*) во вторичном процессе кодирования.

Если после первого кодирования остаются доступные кодовые комбинации, следующий символ, который будет декодироваться с этими избыточными кодовыми комбинациями, зависит от того, является ли активным режим сегментированного двоичного арифметического кодирования (*SBA*).

В случае режима не *SBA* некодированные символы спектров в уровнях более низких, чем текущий уровень кодируются во вторичном процессе кодирования (*bsac_lower_spectra*).

В случае режима *SBA* для устойчивости к ошибкам следующий символ зависит от того, является ли уровень терминальным уровнем сегмента или нет. Если уровень не является терминалом сегмента, спектральные данные следующего уровня (*bsac_layer_spectra (layer*±1)) должны декодироваться. То есть избыточная длина уровня добавляется к доступной длине потока битов (*available_len [layer*+1*]*) следующего уровня в первом процессе кодирования.

Если уровень является терминалом сегмента, некодированными символы спектров в уровнях более низких, чем текущий уровень кодируются во вторичном процессе кодирования (*bsac_lower_spectra*). Некодированный символ спектров в уровнях более высоких, чем текущий уровень кодируется во вторичном процессе кодирования (*bsac_higher_spectra*), если кодовая комбинация уровня доступна несмотря на то, что более низкие спектры кодировались. Остающиеся символы непрерывно кодируются в уровнях, кодовая комбинация которых доступна, начиная с самого низкого уровня и продолжая до верхнего слоя.

Если есть избыточные биты после вторичного кодирования, размер избыточных битов добавляется к длине доступного потока битов (*available_len [layer*+1*]*) следующего уровня и избыточные биты используются в первом кодировании следующего уровня.

5.2.6.2.2.13 Реконструкция декодируемой выборки из секционированных данных

Чтобы восстановить спектральные данные, секционированная последовательность, которая была декодирована, должна быть отображена в значения квантованного спектра. Декодируемый арифметический символ является нарезанным битом. Декодируемый символ преобразовывается в битовые значения квантованных спектральных коэффициентов, как определено в следующем псевдо коде *C*:

```
snf = the significance of the symbol (the sliced bit) to be decoded;
sliced_bit[ch][g][i][snf] = the decoded symbol (the sliced bits of the quantized spectrum);
sample[ch][g][i] = buffer for quantized spectral coefficients to be recontructed;
scaled_bit = sliced_bit[ch][g][i][snf] << (snf-1);
if (sample[ch][g][i] < 0)
sample[ch][g][i] -= scaled_bit;
else
sample[ch][g][i] += scaled_bit;.
```

Если бит знака декодируемой выборки равен 1, декодируемый образец выборки *[ch] [g] [i]* имеет отрицательное значение следующим образом:

```
if (sample[ch][g][i] != 0) {
if (sign_bit[ch][g][i]==1) sample[ch][g][i] = -sample[ch][g][i];
}
```

5.2.6.2.2.14 Декодирование расширенной части

Функции *extended_bsac_raw_data_block*, *extended_bsac_base_element*, *extended_bsac_sbr_data*, и *extended_bsac_sac_data* определяются в 5.2.11.

5.2.6.2.3 Окна и последовательности окна для *BSAC*

Квантование и кодирование производятся в частотной области. С этой целью сигнал времени отображается в частотную область в кодере. В зависимости от сигнала кодер может изменить разрешающую способность по времени/частоте путем использования двух различных окон: *LONG_WINDOW* и *SHORT_WINDOW*. Чтобы переключиться между окнами, используются окна перехода, *LONG_START_WINDOW* и *LONG_STOP_WINDOW*.

5.2.6.2.4 Полосы масштабного коэффициента, группировка и полосы кодирования для *BSAC*

Многие инструменты декодера *AAC/BSAC* выполняют операции на группах последовательных спектральных значений, называемых полосами масштабного коэффициента (сокращение "*sfb*"). Ширина полос масштабного коэффициента создается в имитации критических полос человеческой слуховой системы. По этой причине число полос масштабного коэффициента в спектре и их ширина зависят от длины преобразования и частоты дискретизации.

Инструмент декодирования *BSAC* выполняет операции на группах последовательных спектральных значений, называемых полосами кодирования (сокращение "*cband*"). Чтобы увеличить эффективность бесшумного кодирования, ширина полос кодирования фиксируется как 32 независимо от длины преобразования и частоты дискретизации. В случае последовательностей, которые содержат *LONG_WINDOW*, 32 спектральных данных просто группируются в полосу кодирования. Так как спектральные данные в пределах группы чередуются в возрастающем спектральном порядке в случае *SHORT_WINDOW*, чередующиеся спектральные данные группируются в полосу кодирования. Каждый спектральный индекс в пределах группы отображается в полосу кодирования с функцией отображения *cband* = *spectral_index*/32.

Так как полосы масштабного коэффициента и полосы кодирования являются основным элементом алгоритма кодирования *BSAC*, необходимы некоторые вспомогательные переменные и массивы, чтобы описать процесс декодирования во всех инструментах, используя полосы масштабного коэффициента и полосы кодирования. Эти переменные справки должны быть определены для декодирования *BSAC*. Эти вспомогательные переменные зависят от *sampling_frequency*, *window_sequence*, *scalefactor_grouping* и *max_sfb* и должны быть созданы для каждого *bsac_raw_data_block*. Псевдокод, показанный ниже, описывает:

- как определить число окон *num_windows* в последовательности *window_sequence;*
- как определить число групп *num_window_groups* в *window_groups;*
- как определить число окон в каждой группе *window_group_length [g];*
- как определить общее количество полос окна масштабного коэффициента *num_swb* для фактического типа окна ;
- как определить *swb_offset [г] [swb]*, смещение первого коэффициента в полосе окна масштабного коэффициента *swb* фактически используемого окна.

Длинное окно преобразования всегда описывается как *window_group*, содержащее единственное окно. Поскольку число полос масштабного коэффицента и их ширина зависят от частоты дискретизации, переменные, на которые это влияет, индексируются с помощью *sampling_frequency_index*, чтобы выбрать соответствующую таблицу.

```
fs_index = sampling_frequency_index;
switch (window_sequence) {
case ONLY_LONG_SEQUENCE:
case LONG_START_SEQUENCE:
case LONG_STOP_SEQUENCE:
num_windows = 1;
num_window_groups = 1;
window_group_length[num_window_groups-1] = 1;
num_swb = num_swb_long_window[fs_index];
for (sfb = 0; sfb < max_sfb+1; sfb++) {
swb_offset[0][sfb] = swb_offset_long_window[fs_index][sfb];
}
break;
case EIGHT_SHORT_SEQUENCE:
num_windows = 8;
num_window_groups = 1;
window_group_length[num_window_groups-1] = 1;
num_swb = num_swb_short_window[fs_index];
for(i = 0; i < num_windows-1; i++) {
if (bit_set(scale_factor_grouping,6-i)) == 0 ) {
num_window_groups += 1;
window_group_length[num_window_groups-1] = 1;
} else
{
window_group_length[num_window_groups-1] += 1;
}
}
for (g = 0; g < num_window_groups; g++)
swb_offset[g][0] = 0;
for (sfb = 0; sfb < max_sfb+1; sfb++) {
for (g = 0; g < num_window_groups; g++) {
swb_offset[g][sfb] = swb_offset_short_window[fs_index][sfb];
swb_offset[g][sfb] = swb_offset[g][sfb] * window_group_length[g];
}
}
break;
default:
break;
}
```

5.2.6.2.5 Уровень мелкоструктурной масштабируемости *BSAC*

BSAC обеспечивает мелкоструктурную масштабируемость, которая имеет многоуровневый поток битов, один базовый уровень *BSAC* и различные уровни расширения. Базовый уровень *BSAC* составляется из общей дополнительной информации для всех мелкоструктурных уровней, специальной дополнительной информации только для базового уровня и аудиоданных. Уровни расширения *BSAC* содержат дополнительную информацию уровня и аудиоданные.

Масштабируемая схема кодирования *BSAC* имеет масштабируемый предел полосы согласно мелкоструктурному уровню. Прежде всего устанавливается базовый предел полосы. Базовый предел полосы зависит от сигнала, который будет закодирован, и находится в элементе синтаксиса *base_band*. Фактически ограниченной линией спектра является 4**base_band*, если последовательность окон является *SHORT_WINDOW*. Иначе ограниченной линией спектра будет 32**base_band*. Чтобы обеспечить мелкоструктурную масштабируемость, *BSAC* расширяет предел полосы согласно мелкоструктурному уровню. Предел полосы каждого уровня зависит от базового предела полосы, длин преобразования 1024 (960) и 128 (120) и частоты дискретизации. Полоса спектра расширяется все больше и больше, поскольку число уровней расширения увеличивается. Так, к каждому уровню добавляются новые спектральные компоненты.

Чтобы описать разрядно-модульный процесс декодирования дополнительной информации и спектральных данных в каждом мелкоструктурном уровне, *BSAC* необходимы некоторые вспомогательные переменные и массивы. Эти вспомогательные переменные зависят от *sampling_frequency*, уровня, *nch*, *frame_length*, *top_layer*, *window_sequence* и *max_sfb* и должны быть созданы для каждого *bsac_layer_element*. Псевдокод, показанный ниже, описывает:

- как определить *slayer_size*, число подуровней, на которых разделяется базовый уровень:

```
slayer_size = 0;
for (g = 0; g < num_window_groups; g++) {
if (window_sequence == EIGHT_SHORT_SEQUENCE) {
end_index[g] = base_band * 4 * window_group_length[g];
if (fs == 44100 || fs == 48000) { if (end_index[g]%32>=16)
end_index[g] = (int)(end_index[g]/32)*32 + 20;
else if (end_index[g]%32 >= 4)
end_index[g] = (int)(end_index[g]/32)*32 + 8;
}
else if (fs == 22050 || fs == 24000 || fs == 32000)
end_index[g] = (int)(end_index[g]/16)*16;
else if (fs == 11025 || fs == 12000 || fs == 16000)
end_index[g] = (int)(end_index[g]/32)*32; else end_index[g] = (int)(end_index[g]/64)*64; end_cband[g] =
(end_index[g] + 31) / 32;
}
else
end_cband[g] = base_band;
slayer_size += end_cband[g];
};
```

- как определить *layer_group []*, групповой индекс спектральных компонентов, которые будут вновь добавлены в уровень масштабируемости:

```
layer = 0
for (g = 0; g < num_window_groups; g++)
for (cband = 1; cband <= end_cband[g]; layer++, cband++)
layer_group[layer] = g;
layer = slayer_size;
for (g = 0; g < num_window_groups; g++)
for (w = 0; w < window_group_length[g]; w++)
layer_group[layer++] = g;
for (layer = slayer_size+8; layer < (top_layer+slayer_size); layer++)
layer_group[layer] = layer_group[layer-8];
```

- как определить *layer_end_index []*, конечное смещение спектральных компонентов, которые будут вновь добавлены в каждом уровне масштабируемости;

- как определить *layer_end_cband []*, конечную полосу кодирования, которая будет вновь добавлена в каждом уровне масштабируемости;

- как определить *layer_start_index []*, стартовое смещение спектральных компонентов, которые будут вновь добавлены в каждом уровне масштабируемости;

- как определить *layer_start_cband []*, стартовую полосу кодирования, которая будет вновь добавлена в каждом уровне масштабируемости:

```
layer = 0;
for (g = 0; g < num_window_groups; g++) {
for (cband = 0; cband < end_cband[g]; cband++) {
layer_start_cband[layer] = cband;
end_cband[g] = layer_end_cband[layer] = cband+1;
layer_start_index[layer] = cband * 32;
end_index[g] = layer_end_index[layer++] = (cband+1) * 32;
}
if (window_sequence == EIGHT_SHORT_SEQUENCE)
last_index[g] = swb_offset_short_window[fs_index][max_sfb] *
window_group_length[g];
else
last_index[g] = swb_offset_long_window[fs_index][max_sfb];
}
for (layer = slayer_size; layer < (top_layer+slayer_size); layer++) {
g = layer_group[layer];
layer_start_index[layer] = end_index[g]; if (fs == 44100 || fs == 48000) {
if (end_index[g]%32 == 0)
end_index[g] += 8;
else
end_index[g] += 12;
}
else if (fs == 22050 || fs == 24000 || fs == 32000)
end_index[g] += 16;
else if (fs == 11025 || fs == 12000 || fs == 16000)
end_index[g] += 32;
else
end_index[g] += 64;
if (end_index[g] > last_index[g])
end_index[g] = last_index[g];
layer_end_index[layer] = end_index[g];
layer_start_cband[layer] = end_cband[g];
end_cband[g] = layer_end_cband[layer]=(end_index[g] + 31) / 32;
},
```

где *fs* является частотой дискретизации;

- как определить *layer_end_sfb []*, конечную полосу масштабного коэффициента, которая будет вновь добавлена в каждом уровне масштабируемости;

- как определить *layer_start_sfb []*, стартовую полосу масштабного коэффициента, которая будет вновь добавлена в каждом уровне масштабируемости:

```
for (g = 0; g < num_window_groups; g++)
end_sfb[g] = 0;
for (layer = 0; layer < (top_layer+slayer_size); layer++) {
g = layer_group[layer];
layer_start_sfb[layer] = end_sfb[g];
layer_end_sfb[layer] = max_sfb;
for (sfb = 0; sfb < max_sfb; sfb++) {
if (layer_end_index[layer] <=
swb_offset_short_window[fs_index][sfb] * window_group_length[g]) {
layer_end_sfb[layer] = sfb + 1;
break;
}
}
```

```
end_sfb[g] = layer_end_sfb[layer];
};
```

- как определить *available_len [i]*, доступный максимальный размер потока битов *i*-го уровня. Если арифметическое кодирование было инициализировано в начале уровня, из *available_len [i]* нужно вычесть 1, так как дополнительный 1 бит требуется при завершении арифметического кодирования. Максимальная длина дополнительной информации нулевой полосы кодирования (*max_cband*0_*si_len*) определяется как 11:

```
for (layer = 0; layer <(top_layer+slayer_size); layer++) {
layer_si_maxlen[layer] = 0;
for (cband = layer_start_cband[layer]; cband < layer_end_cband[layer]; cband++) {
for (ch=0; ch <nch; ch++) {
if (cband == 0)
layer_si_maxlen[layer] += max_cband0_si_len;
else
layer_si_maxlen[layer] += max_cband_si_len[cband_si_type[ch]];
}
}
for (sfb = layer_start_sfb[layer]; sfb < layer_end_sfb[layer]; sfb++)
for (ch = 0; ch < nch; ch++)
layer_si_maxlen[layer] += max_sfb_si_len[ch] + 5;
}
for (layer = slayer_size; layer <= (top_layer + slayer_size); layer++) {
layer_bitrate = nch * ( (layer-slayer_size) * 1000 + 16000);
layer_bit_offset[layer] = layer_bitrate * BLOCK_SIZE_SAMPLES_IN_FRAME;
layer_bit_offset[layer] = (int)(layer_bit_offset[layer] / SAMPLING_FREQUENCY / 8 ) * 8;
if (layer_bit_offset[layer] > frame_length*8)
layer_bit_offset[layer] = frame_length*8;
}
for (layer = (top_layer + slayer_size -1); layer >= slayer_size; layer- -) {
bit_offset = layer_bit_offset[layer+1] - layer_si_maxlen[layer]
if ( bit_offset < layer_bit_offset[layer] )
layer_bit_offset[layer] = bit_offset
}
for (layer = slayer_size – 1; layer >= 0; slayer--)
layer_bit_offset[layer] = layer_bit_offset[layer+1] – layer_si_maxlen[layer];
overflow_size = (header_length + 7) * 8 – layer_bit_offset[0];
layer_bit_offset[0] = (header_length + 7) * 8;
if (overflow_size > 0) {
for (layer = (top_layer+slayer_size-1); layer >= slayer_size; layer- -) {
layer_bit_size = layer_bit_offset[layer+1] – layer_bit_offset[layer];
layer_bit_size -= layer_si_maxlen[layer];
if (layer_bit_size >= overflow_size) {
layer_bit_size = overflow_size;
overflow_size = 0;
}
else
overflow_size = overflow_size – layer_bit_size;
for (m = 1; m <= layer; m++)
layer_bit_offset[m] += layer_bit_size;
if (overflow_size <= 0)
break;
}
}
else {
underflow_size = -overflow_size;
```

```
for (m = 1; m < slayer_size; m++) {
layer_bit_offset[m]=layer_bit_offset[m-1]+layer_si_maxlen[m-1];
layer_bit_offset[m] += underflow_size / slayer_size;
if (layer <= (underflow_size%slayer_size);
layer_bit_offset[m] += 1;
}
}
for (layer = 0; layer < (top_layer+slayer_size); layer++)
available_len[layer] = layer_bit_offset[layer+1]–layer_bit_offset[layer].
```

Чтобы описать разрядно-модульный процесс декодирования спектральных значений в каждом мелкоструктурном уровне *BSAC*, необходимы некоторые вспомогательные переменные и массивы. *cur_snf [ch] [g] [i]* инициализируется как плоскость *MCB (MCBplane [ch] [g] [cband])*, назначенная полосе кодирования *cband*, где мы можем получить *MCBplane [] []* из *cband_si [ch] [g] [cband]* используя таблицу А.33. Декодирование секционированных данных в каждом уровне запускается с максимального значения *maxsnf*.

Эти вспомогательные переменные и массивы должны быть созданы для каждого *bsac_spectral_data ()*. Показанный ниже псевдокод описывает:

- как инициализировать *cur_snf [] [] []*, текущее значение спектров, которые будут вновь добавлены из-за расширения спектральной полосы в каждом уровне масштабируемости расширения:

```
/* set current snf */
g = layer_group[layer];
for (ch = 0; ch < nch; ch++) {
for (i = layer_start_index[layer]; i<layer_end_index[layer]; i++) {
cband = i/32;
cur_snf[ch][g][i] = MCBplane[ch][g][cband];
}
};
```

- как определить *maxsnf*, максимальное значение всех векторов, которые будут декодироваться:

```
maxsnf = 0;
for (g = start_g; g < end_g; g++)
for (ch = 0; ch < nch; ch++) {
for (i = start_index[g]; i< end_index[g]; i++)
if (maxsnf < cur_snf[ch][g][i])
maxsnf = cur_snf[ch][g][i];
};
```

- как сохранить *cur_snf [] [] []* для вторичного кодирования (*bsac_lower_spectra* () и *bsac_higher_spectra* ()) после того, как секционированные биты новых спектров были кодированы в каждом уровне (*bsac_layer_spectra*()):

```
/* store current snf */
for (g = 0; g < no_window_groups;g++)
for (ch = 0; ch < nch; ch++) {
for (i = layer_start_index[layer]; i < layer_end_index[layer]; i++) {
unc_snf[ch][g][i] = cur_snf[ch][g][i];
}
}/
```

5.2.6.2.6 Порядок спектральных коэффициентов в *spectral_data*

Для окон *ONLY_LONG_SEQUENCE (num_window_groups = 1, window_group_length [0] = 1)* спектральные данные находятся в возрастающем спектральном порядке, как показано на рисунке 1.

Спектральные коэффициенты →

*sfb*0	*sfb*1	*sfb*2	.....	*sfb(num_sfb*-1)

Рисунок 1 — Порядок полос спектрального коэффициента для *ONLY_LONG_SEQUENCE*

Для окна *EIGHT_SHORT_SEQUENCE* каждые 4 спектральных коэффициента блоков в пределах каждой группы чередуются в возрастающем спектральном порядке и чередующиеся спектральные коэффициенты чередуются в порядке возрастания номера группы, как показано на рисунке 2.

Спектральные коэффициенты →

Спектральные коэфициенты для группы 0	Спектральные коэфициенты для группы 1	.....	Спектральные коэфициенты для группы *nun_win_groups-1*

Спектральные коэффициенты для группы →

WS 0..3	...	*WE* 0..3	*WS* 4..7	...	*WE* 4..7	...	*WS* 80..83	...	*WE* 80..83	...

где *WS* — индекс стартового окна, а *WE* — индекс окончания группы *g*

Рисунок 2 — Порядок спектральных данных для *EIGHT_SHORT_SEQUENCE*

5.2.6.2.7 Процедура арифметического кодирования

Арифметическое кодирование состоит из следующих 2 шагов:

- Инициализация, которая выполняется до кодирования первого символа;
- Кодирование самих символов.

5.2.6.2.7.1 Регистры, символы и константы

Чтобы описать арифметический декодер определяются несколько регистров, символов и констант:

- *half []*: 32-разрядный массив с фиксированной запятой, равный 1/2 ;
- *range*: 32-разрядный регистр с фиксированной запятой. Содержит диапазон интервала;
- *value*: 32-разрядный регистр с фиксированной запятой. Содержит значение арифметического кода;
- *est_cw_len*: 16-разрядный регистр с фиксированной запятой. Содержит предполагаемую длину арифметической кодовой комбинации, которая будет декодироваться;
- *p* 0: 16-разрядный регистр с фиксированной запятой (доступны верхние 6 *MCB*, другие *LSB* равны 0). Вероятность символа "0";
- *p* 1: 16-разрядный регистр с фиксированной запятой (доступны верхние 6 *MCB*, другие *LSB* равны 0). Вероятность символа "1";
- *cum_freq*: 16-разрядные регистры с фиксированной запятой. Совокупные вероятности символов.

5.2.6.2.7.2 Инициализация

Потоки битов каждого сегмента читаются в буфере каждого сегмента. 32-разрядный ноль присоединяется к буферу каждого сегмента. Если не сегментированное арифметическое кодирование, все потоки битов фрейма являются сегментом и используется нолевое заполнение.

Регистр *value* устанавливается в 0, *range* в 1 и *est_cw_len* в 30. Используя эти инициализированные регистры, в регистре value читаются 30 битов и регистры обновляются, когда декодируется первый символ.

5.2.6.2.7.3 Декодирование символа

Процедуры арифметического декодирования будут варьироваться когда будет декодироваться символ. Если символ является секционированным битом спектральных данных, используется двоичное арифметическое декодирование. Иначе используется общее арифметическое декодирование.

Когда символ является двоично-арифметически декодированным, вероятность *p*0 "0"-го символа обеспечивается согласно вычисленному контексту и использованию таблицы вероятности. *p*0 использует 6-разрядное представление числа фиксированной запятой. Так как декодер является двоичным, вероятность символа "1" определяется как 1 минус вероятность символа "0", то есть $p1 = 1 - p0$.

Когда символ является арифметически декодированным, обеспечиваются значения совокупной вероятности нескольких символов. Значения вероятности расцениваются как арифметическая модель. Арифметическая модель для декодирования символа дается в элементах данных. Например, арифметические модели масштабного коэффициента и *cband_si* даются в элементах данных *base_scf_model*, *enh_scf_model* и *cband_si_type*. Каждое значение арифметической модели использует 14-разрядное представление с фиксированной запятой.

5.2.6.2.7.3 Программное обеспечение

```
unsigned long half[16] =
{
0x20000000, 0x10000000, 0x08000000, 0x04000000,
0x02000000, 0x01000000, 0x00800000, 0x00400000,
0x00200000, 0x00100000, 0x00080000, 0x00040000,
0x00020000, 0x00010000, 0x00008000, 0x00004000
};
/* Initialize the Parameteres of the Arithmetic Decoder */
void initArDecode()
{
value = 0;
range = 1;
est_cw_len = 30;
}
/* GENEARL ARITHMETIC DECODE */
int decode_symbol (buf_idx, cum_freq, symbol)
int buf_idx;         /* buffer index to save the arithmetic code word */
int cum_freq[];   /* Cumulative symbol freqencies */
int *symbol;      /* Symbol decoded */
{
if (est_cw_len) {
range = (range << est_cw_len);
value = (value << est_cw_len) | readBits(buf_idx, est_cw_len);
/* read bitstream from the buffer */
}
range >>= 14;
cum = value/range;                    /* Find cum freq */
/* Find symbol */
for (sym = 0; cum_freq[sym]>cum; sym++);
*symbol = sym;
/* Narrow the code region to that allotted to this symbol. */
value -= (range * cum_freq[sym]);
if (sym > 0) {
range = range * (cum_freq[sym-1]-cum_freq[sym]);
}
else {
range = range * (16384-cum_freq[sym]);
}
for (est_cw_len = 0; range < half[est_cw_len]; est_cw_len++) ;
return est_cw_len;
}
/* BINARY ARITHMETIC-DECODE THE NEXT SYMBOL. */
int decode_symbol2 (buf_idx, freq0, symbol)
int buf_idx; /* buffer index to save the arithmetic code word */
int p0;      /* Normalized probability of symbol 0 */
int *symbol; /* Symbol decoded */
{
if (est_cw_len) {
range = (range << est_cw_len);
value = (value << est_cw_len) | readBits(buf_idx, est_cw_len);
/* read bitstream from the buffer */
}
range >>= 14;
```

```
/* Find symbol */
if ( (p0 * range) <= value ) {
*symbol = 1;
/* Narrow the code region to that allotted to this symbol. */
value -= range * p0;
p1 = 16384 – p0;
range = range * p1;
}
else {
*symbol = 0;
/* Narrow the code region to that allotted to this symbol. */
range = range * p0;
}
for (est_cw_len = 0; range < half[est_cw_len]; est_cw_len++);
return est_cw_len;
}
```

5.2.6.3 Присвоения категории чувствительности к ошибкам

BSAC имеет многоуровневую структуру, где синтаксис располагается в порядке важности, чтобы поддерживать мелкоструктурную масштабируемость и устойчивость к ошибкам. Поэтому синтаксис *BSAC* может быть эффективным кодированным каналом, без потока битов переупорядочения для усовершенствованных методов кодирования каналов, таких как неравномерная защита от ошибок (*UEP*), так как устойчивый к ошибкам синтаксис включен в 4.2.6. Для усовершенствованного кодирования канала должны быть определены категории чувствительности к ошибкам (*ESC*) элементов данных. Элемент данных может быть классифицирован в категории чувствительности к ошибкам в зависимости от его чувствительности к ошибкам следующим образом:

Таблица 101 — Присвоение категории чувствительности к ошибкам *BSAC*

Категория	Элементы данных	Описание
0	*frame_length, bsac_header() and general_header()*	Обычно используемая дополнительная информация
1	*bsac_layer_element(0)*	Базовый уровень *BSAC*, кроме общей стороны
2	*bsac_layer_element(1)* –	Уровни 1-го квартернарного расширения
3	*bsac_layer_element(top_layer/4+1)* –	Уровни 2-го квартернарного расширения
4	*bsac_layer_element(top_layer/2+1)* ~	Уровни 3-го квартернарного расширения
5	*basc_layer_element(top_layer*3/4+1)* ~	Уровни 4-го квартернарного расширения

Более низкая категория указывает на класс с более высокой чувствительностью к ошибкам, тогда как более высокая категория указывает на класс с более низкой чувствительностью.

5.2.7 Управление динамическим диапазоном (*DRC*)

5.2.7.1 Определения

pce_tag_present Однобитовая индикация, указывающая, что тег элемента программы присутствует.

pce_instance_tag Поле тега, которое указывает, с какой программой связана информация о динамическом диапазоне.

drc_tag_reserved_bits Зарезервировано.

excluded_chns_present Один бит, указывающий, что исключенные каналы присутствуют.

drc_bands_present Один бит, указывающий, что многополосная информация *DRC* присутствует.

drc_band_incr Число полос *DRC*, содержащих информацию о *DRC*, больше чем 1.

drc_interpolation_scheme Указывает, какая используется схема интерполяции для данных *DRC* в домене *SBR QMF* согласно таблице 102.

Таблица 102 — *drc_interploation_scheme*

drc_interpolation_scheme	Значение
0	Интерполяция по умолчанию
1	Интерполяция с крутым наклоном в позиции 0
2	Интерполяция с крутым наклоном в позиции 1
3	Интерполяция с крутым наклоном в позиции 2
4	Интерполяция с крутым наклоном в позиции 3
5	Интерполяция с крутым наклоном в позиции 4
6	Интерполяция с крутым наклоном в позиции 5
7	Интерполяция с крутым наклоном в позиции 6
8	Интерполяция с крутым наклоном в позиции 7
9—15	Зарезервировано

drc_band_top [i] Указывает вершину *i*-ой полосы *DRC* в модулях 4 линий спектра. Если *drc_band_top [i] =k*, то индекс (*w.r.t.* ноль) самого высокого спектрального коэффициента, который находится в *i*-ой полосе *DRC*, будет = 4*k*+3. В случае *EIGHT_SHORT_SEQUENCE window_sequence*, индекс интерпретируется как указывающий связанный массив 8*128 (де-чередующийся) точек частоты, соответствующий 8 коротким преобразованиям.

prog_ref_level_present Однобитовая индикация, указывающая, что контрольный уровень присутствует.

prog_ref_level Контрольный уровень. Мера долгосрочного уровня аудио программы для всех объединенных каналов.

prog_ref_level_reserved_bits Зарезервировано.

dyn_rng_sgn [i] Информация о знаке управления динамическим диапазоном. Один бит, указывающий знак *dyn_rng_ctl* (0 если положительный, 1, если отрицательный).

dyn_rng_ctl [i] Информация о величине управления динамическим диапазоном.

exclude_mask [i] Булев массив, указывающий звуковые каналы программы, которые исключены из обработки *DRC*, используя эту информацию *DRC*.

additional_excluded_chns [i] Один бит, указывающий, что присутствуют дополнительные исключенные каналы.

5.2.7.2 Процесс декодирования

Транспорт информации *DRC* не включает системный уровень MPEG–4, но вместо этого обрабатывается полностью в пределах элементов данных *GA*. Кроме того, опционной является оценка информации управляющей потенциально доступным динамическим диапазоном в декодере *GA*. Никакая соответствующая информация *DRC* не передается последующему наборщику аудио.

prog_ref_level_present указывает, что передается *prog_ref_level*. Это разрешает отправлять *prog_ref_level* так редко, как это требуется (например, однажды), хотя периодическая передача разрешила бы вмешательство.

prog_ref_level квантуется в шагах 0,25 дБ, используя 7 битов, и поэтому имеет диапазон приблизительно 32 дБ. Это указывает на уровень программы относительно полной шкалы (то есть дБ ниже полной шкалы) и восстанавливается как:

$$level = 32767 * 2^{\frac{-prog_ref_lebel}{24}},$$

где уровень полной шкалы равен 32767 (*prog_ref_level* равен 0).

pce_tag_present указывает, что *pce_instance_tag* передается. Это разрешает передавать *pce_instance_tag* так редко, как требуется (например, однажды), хотя периодическая передача разрешила бы вмешательство.

pce_instance_tag указывает, с какой программой связывается информация о динамическом диапазоне. Если он отсутствует, тогда указывается программа по умолчанию. Так как каждая полезная нагрузка

потока битов *AAC* обычно имеет только одну программу, это может быть наиболее распространенным режимом. Каждая программа в полезной нагрузке мультипрограммного потока битов может отправить информацию о своем динамическом диапазоне в отдельном *extension_payload* () элемента *fill_element* (). В случае многих программ всегда должен сообщаться *pce_instance_tag.*

drc_tag_reserved_bits заполняют дополнительные поля до интегрального числа байтов по длине.

Бит *excluded_chns_present* указывает, что образуемые каналы, которые должны быть исключены из обработки динамического диапазона, будут сразу указаны после этого бита. Информация о маске исключенного канала должна передаваться в каждом фрейме, где исключаются каналы. Используются следующие принципы упорядочивания, чтобы присвоить *exclude_mask* выводам каналов:

- если *PCE* присутствует, биты *exclude_mask* соответствуют звуковым каналам в элементах синтаксиса *SCE, CPE, CCE* и *LFE* в порядке их появления в *PCE*. В случае *CPE* первый переданный бит маски соответствует первому каналу в *CPE*, второй переданный бит маски соответствует второму каналу. В случае *CCE* бит маски передается только если связывающийся канал определяется как независимо коммутируемый канал связи;

- если *PCE* отсутствует, биты *exclude_mask* соответствуют звуковым каналам в элементах синтаксиса *SCE, CPE* и *LFE* в порядке их появления в полезной нагрузке потока битов, сопровождаемой звуковыми каналами в элементах синтаксиса *CCE* в порядке их появления в полезной нагрузке потока битов. В случае *CPE* первый переданный бит маски соответствует первому каналу в *CPE*, второй переданный бит маски соответствует второму каналу. В случае *CCE* бит маски передается, если связывающийся канал определяется как независимо коммутируемый канал связи.

drc_band_incr является числом полос бóльшим, чем единица, если имеется многополосная информация *DRC.*

dyn_rng_ctl квантуется с шагом 0,25 дБ, используя 7-разрядное целое число без знака, и поэтому в комбинации с *dyn_rng_sgn*, имеет диапазон +/–31,75 дБ. Это интерпретируется как значение усиления, которое должно быть применено к декодируемым выборкам аудиовыхода текущего фрейма.

Диапазон, поддерживаемый информацией о динамическом диапазоне, суммируется в таблице 103.

Таблица 103 — Информация о динамическом диапазоне

Поле	Биты	Шаги	Размер шага, дБ	Диапазон, дБ
Prog_ref_level	7	128	0,25	31,75
dyn_rng_sgn and dyn_rng_ctl	1 и 7	+/– 127	0,25	+/– 31,75

Процесс управления динамическим диапазоном применяется к спектральным данным *spec[i]* одного фрейма непосредственно перед блоком фильтров синтеза. В случае *window_sequence EIGHT_SHORT_SEQUENCE* индекс *i* интерпретируется как указывающий в связанном массиве 8*128 (устраненное чередование) точек частоты, соответствующих 8 коротким преобразованиям.

Следующий псевдокод служит только для целей иллюстрации, показывая один метод для применения одного набора управляющей информации динамического диапазона к фрейму целевого звукового канала. Константы *ctrl*1 и *ctrl*2 являются константами компрессии (обычно между 0 и 1, ноль означает отсутствие компрессии), которые могут дополнительно использоваться, чтобы масштабировать характеристики сжатия динамического диапазона для уровней, бóльших или меньших, чем контрольный уровень программы, соответственно. Постоянный *target_level* описывает уровень на выходе, требующийся пользователю, выраженный в том же самом масштабе как *prog_ref_level.*

```
#define FRAME_SIZE 1024 /* Change to 960 for 960-framing*/
bottom = 0;
drc_num_bands = 1;
if (drc_bands_present)
drc_num_bands += drc_band_incr;
else
drc_band_top[0] = FRAME_SIZE/4 - 1;
for (bd = 0; bd < drc_num_bands; bd++) {
top = 4 * (drc_band_top[bd] + 1);
/* Decode DRC gain factor */
```

```
if (dyn_rng_sgn[bd])
factor = 2^(-ctrl1*dyn_rng_ctl[bd]/24);  /* compress */
else
factor = 2^(ctrl2*dyn_rng_ctl[bd]/24);   /* boost */
/* If program reference normalization is done in the digital domain, modify
* factor to perform normalization.
*  prog_ref_level can alternatively be passed to the system for modification
*  of the level in the analog domain. Analog level modification avoids problems
*  with reduced DAC SNR (if signal is attenuated) or clipping (if signal is boosted)
*/
factor *= 0,5^((target_level-prog_ref_level)/24);
/* Apply gain factor */
for (i = bottom; i < top; i++)
spec[i] *= factor;
bottom = top;
}
```

Зависимо коммутируемые каналы связи всегда связываются в их целевые каналы как спектральные коэффициенты перед *DRC* и фильтрацией синтеза этих каналов. Поэтому сигнал зависимо коммутируемого канала связи, который связывается в определенный целевой канал, подвергнется обработке *DRC* этого целевого канала.

С момента, когда независимо коммутируемые каналы связи связываются со своими целевыми каналами во временно́й области, каждый независимо коммутируемый канал связи подвергается *DRC* и последующей фильтрации синтеза отдельно от своего целевого канала. Это позволяет независимо коммутируемому каналу связи иметь отдельную обработку *DRC*.

5.2.7.3 Персистентность информации *DRC*

В начале потока вся информация *DRC* для всех каналов устанавливается в ее значение по умолчанию: контрольный уровень программы, равный целевому контрольному уровню декодера, одна полоса *DRC* без модификации усиления *DRC* для этой полосы. Пока эти данные специально не перезаписываются, это остается в силе.

Есть два случая для персистентности информации *DRC*, которая была передана:

- контрольный уровень программы назначен для аудио программы и сохраняется, пока не передается новое значение, в этой точке новые данные перезаписывают старые и вступают в силу для этого фрейма;

- другая информация *DRC* сохраняется на поканальной основе. Если канал исключается посредством соответствующего бита *exclude_mask []*, тогда в вызове *dynamic_range_info* () для этого канала никакая информация не передается. Информация о маске исключенного канала должна быть передана в каждом фрейме, где исключаются каналы.

Правила сохранения поканальной информации о *DRC* следующие:

- если в данном фрейме для данного канала нет никакой информации *DRC*, используется информация, которая использовалась в предыдущем фрейме (это означает, что одна регулировка может сохраняться в течение долгого времени, хотя может быть уместно передавать информацию *DRC* периодически, чтобы разрешить вмешательство);

- если в текущем фрейме появляется какая-либо информация *DRC* для этого канала, имеет место следующая последовательность: во-первых, перезапись всей информации о *DRC* для этого канала с заменой значений по умолчанию (одна полоса *DRC*, без модификации усиления *DRC* для этой полосы), затем замена информации о *DRC* на канал переданными значениями.

5.2.7.4 Использование *DRC* с масштабируемым *AAC* аудио объектного типа

Если *DRC* используется с масштабируемым *AAC* аудио объектного типа, применяются следующие дополнительные ограничения и информация:

1). Поле *pce_tag_present* должно быть '0' (никаких ссылок на *PCE*).

2). Поле *excluded_chns_present* должно быть '0' (общий контроль всех звуковых каналов).

3). Информация о *DRC* может быть передана в нескольких уровнях масштабируемого аудио объекта. Информацией *DRC*, которая должна использоваться для обработки *DRC*, является информация, которая переносится в самом высоком уровне, доступном декодеру.

5.2.7.5 Использование *DRC* с аудио объектным типом *SBR*

Если *DRC* будет использоваться с аудио объектным типом *SBR*, то процесс должен быть применен к спектральным данным в домене *SBR QMF*. Декодер *High Efficiency AAC Profile* (профиль *AAC* высокой

производительности) должен быть в состоянии проанализировать элемент расширения *DRC*. Возможность декодировать и применить данные *DRC* является дополнительной для декодера *High Efficiency AAC Profile*. Если это будет реализовано, то обрисованная здесь реализация должна использоваться.

Следующий псевдо код и уравнения показывают, как факторы *DRC* сохраняются для использования в домене *SBR QMF*. Границы полос *DRC* квантуются, чтобы соответствовать разрешающей способности по частоте блока фильтров *SBR QMF*. Чтобы гарантировать надлежащую обратную совместимость, нужно рассмотреть задержку между синтезом *MDCT* и синтезом *QMF*. Параметры *DRC*, применявшиеся к подвыборкам *SBR QMF*, должны быть задержаны на то же самое количество времени как сигнал между синтезом *MDCT* и синтезом *QMF*.

Факторы *DRC* сохранены в матрице *factorQMF[l][k]*, где '*l*' указывает какая подвыборка *QMF* соответствует значениям. Так как границы полосы *DRC* относятся к 1024 линиям *MDCT* (или 960 линий *MDCT* для с 960-кадрированием), границы отображаются в соответствующие границы в 32 поддиапазонах более низкой части *SBR QMF*.

Для коротких последовательностей окон *AAC* не используется никакая интерполяция факторов *DRC*. Для других последовательностей окон *AAC* факторы *DRC* интерполируются в течение долгого времени, чтобы избежать шума. Для *DRC*, используемого с декодером *High Efficiency AAC*, возможно сообщить временную границу между факторами *DRC*. Следовательно существует возможность управлять переходным поведением *DRC* без необходимости полагаться на короткие последовательности окон *AAC*. Это выполняется с элементом данных *drc_interpolation_scheme*.

Полосы, покрытые данными *DRC*, охватывают только частотный диапазон *AAC MDCT*, то есть, до половины частоты дискретизации *AAC*. Данные *DRC* для частотного диапазона, которые выше этой половины частоты дискретизации выходного сигнала, те же, что и для самой высокой переданной полосы *DRC*.

```
#define FRAME_SIZE 1024 /* Change to 960 for 960-framing.*/
#define NUM_QMF_SUBSAMPLES (FRAME_SIZE/32)
#define NUM_QMF_SUBSAMPLES_2 (FRAME_SIZE/64)
#if 1 /* 1024 FRAMING */
static float offset[8] = {0, 4, 8, 12, 16, 20, 24, 28};
#else /* 960 FRAMING */
static float offset[8] = {0, 4, 8, 11, 15, 19, 23, 26};
#endif
for (i =  0 ; i < 64; i++){
for (j = 0; j < NUM_QMF_SUBSAMPLES; j++){
factorQMF[j][i] = factorQMF[NUM_QMF_SUBSAMPLES + j][i];
}
previousFactors[i] = factorQMF[2*NUM_QMF_SUBSAMPLES-1][i]; }
bottom = 0;
drc_num_bands = 1;
if (drc_bands_present)
drc_num_bands += drc_band_incr;
if (!drc_bands_present)
drc_band_top[0] = FRAME_SIZE/4 - 1;
for (bd = 0; bd < drc_num_bands; bd++)  {
top   = 4 * (drc_band_top[bd] + 1);
/* Decode DRC gain factor */
if (dyn_rng_sgn[bd])
factor = 2^(-ctrl1*dyn_rng_ctl[bd]/24);  /* compress */
else
factor = 2^(ctrl2*dyn_rng_ctl[bd]/24);   /* boost */
/* If program reference normalization is done in the digital domain, modify
* factor to perform normalization.
*  prog_ref_level can alternatively be passed to the system for modification
*  of the level in the analog domain. Analog level modification avoids problems
*  with reduced DAC SNR (if signal is attenuated) or clipping (if signal is boosted) */
factor *= 0,5^((target_level-prog_ref_level)/24);
/* Truncate bottom and top to be a multiple of NUM_QMF_SUBSAMPLES.*/
```

```
if(bT != EIGHT_SHORT_WINDOW){
bottom = NUM_QMF_SUBSAMPLES*(int)(bottom/NUM_QMF_SUBSAMPLES);
top   = NUM_QMF_SUBSAMPLES*(int)(top/NUM_QMF_SUBSAMPLES);
}
else{
bottom = (int)(NUM_QMF_SUBSAMPLES/8*(int)(bottom*8/NUM_QMF_SUBSAMPLES));
top   = (int)(NUM_QMF_SUBSAMPLES/8*(int)(top*8/NUM_QMF_SUBSAMPLES));
}
/* Apply gain factor */
 /* for (i = bottom; i < top; i++)
 spec[i] *= factor;
*/
if(bT != EIGHT_SHORT_WINDOW){ /* bt indicates the AAC window sequence for the current frame. */
bottomQMF = bottom*32/FRAME_SIZE;
for (j = -NUM_QMF_SUBSAMPLES_2; j < NUM_QMF_SUBSAMPLES; j++){
alphaValue = 0;
if(j+NUM_QMF_SUBSAMPLES_2 < NUM_QMF_SUBSAMPLES){
if(p->drc_interp_scheme == 0){
k = 1,0f / ((float) NUM_QMF_SUBSAMPLES);
alphaValue = (j+NUM_QMF_SUBSAMPLES_2)*k;
}
else{
if (j+NUM_QMF_SUBSAMPLES_2 >= offset[p->drc_interp_scheme - 1])
alphaValue = 1;
}
}
 else
alphaValue = 1;
for (i = bottomQMF ; i < 64 ; i++) {
factorQMF[NUM_QMF_SUBSAMPLES + j ][i] = alphaValue*factor +
(1 - alphaValue)*previousFactors[i];
}
}
}
else{
```

/* - *StartSample* и *stopSample* указывают на границы в подвыборках *QMF* для текущих данных *DRC*. Они получаются первым удалением всей целочисленной сети магазинов короткой длины окна. Всегда есть 32 аналитических поддиапазона *QMF* для входного типа телосложения *FRAME_SIZE*, число подвыборок *QMF* является *FRAME_SIZE*/32 (*NUM_QMF_SUBSAMPLES*) для каждого фрейма. Следовательно число подвыборок *QMF* для короткого окна (*NUM_QMF_SUBSAMPLES*)/8. *bottomQMF* вычисляется от остатка, когда целочисленная сеть магазинов короткой длины окна была удалена. Этот индекс в строке *MDCT* отображается на соответствующий поддиапазон в *QMF*, умножаясь на 32/(*FRAME_SIZE*/8). Если границы между двумя короткими окнами пересекаются, то есть *startSample* располагается в коротком окне *n*-1, и *stopSample* располагается в коротком окне *n*, *bottomQMF* должен быть обнулен, когда граница между фреймами пересекается.

```
/*
/* startSample is truncated to the corresponding start subsample in the
QMF of the short window bottom is present in.*/
startSample = floor((float)
bottom/(FRAME_SIZE/8,0f))*(NUM_QMF_SUBSAMPLES)/8;
/* stopSample is rounded upwards to the nearest corresponding stop subsample in the QMF of the short
window top is present in.*/
stopSample = ceil ((float) top/(FRAME_SIZE/8,0f))*(NUM_QMF_SUBSAMPLES)/8;
bottomQMF  = (bottom%(FRAME_SIZE/8))*32/(FRAME_SIZE/8);
```

```
for (j = startSample; j < stopSample; j++){
if(j > startSample && j%4 == 0){
bottomQMF = 0;
}
for (i = bottomQMF ; i < 64 ; i++) {
factorQMF[NUM_QMF_SUBSAMPLES + j][i] = factor;
}
}
}
bottom = top;
}
```

Для всех значений *drc_interpolation_scheme* кроме нуля используется ступенчатая функция. Для первого фрейма, который не является фреймом последовательности восьми коротких окон *AAC*, после последовательности восьми коротких окон *AAC drc_interpolation_sheme* должен быть равен пяти.

DRC применяется непосредственно перед синтезом *SBR QMF*, умножая каждый элемент в матрице, содержащей выборки подполос, которые будут синтезироваться соответствующим элементом в *factor(k, l)*, где

$$factor(l,k) = factorQMF[RATE*numTameSlots - l_{border} + l][k],$$

и l_{border} вычисляется следующим образом:

$$l_{border} = \frac{\frac{FRAME_SAZE}{2} + \frac{L_{Analisisfilter}}{2} - N_{analisisChannels} + Delay_{buffer}}{N_{analisisChannels}} = \begin{cases} 26 для FRAME_SAZE = 1024 \\ 25 для FRAME_SAZE = 960 \end{cases},$$

где

$N_{analisisChannels} = 32$, $L_{Analisisfilter} = 320$ и $Delay_{buffer} = 6*32$

5.2.8 Полезные нагрузки для *SBR* аудио объектного типа

5.2.8.1 Определения

bs_reserved — Биты, зарезервированные для будущего использования, значение по умолчанию является нулем.

sbr_extension_data () — Синтаксический элемент, который содержит данные расширения *SBR*.

bs_sbr_crc_bits — Циклическая контрольная сумма избыточности для данных расширения SBR. Код CRC определяется полиномиальным генератором $G^{10}(x) = x^{10} + x^9 + x^5 + x^4 + x + 1$ и начальное значение для вычисления *CRC* равно нулю.

bs_header_flag — Указывает, присутствует ли заголовок *SBR*.

sbr_header() — Синтаксический элемент, который содержит заголовок *SBR*.

sbr_data () — Синтаксический элемент, который содержит данные *SBR*.

bs_fill_bits — Биты выравнивания байта.

bs_amp_res — Определяет разрешение оценок огибающей согласно таблицы 104.

Таблица 104 — Векторный элемент *bs_amp_res*

bs_amp_res	Значение
0	1,5 дБ
1	3,0 дБ

bs_start_freq — Входной параметр для функции, которая вычисляет начало таблицы задающей частоты.

bs_stop_freq — Входной параметр для функции, которая вычисляет конец таблицы задающей частоты.

bs_xover_band — Индекс для таблицы задающей частоты.

bs_header_extra_1 — Указывает, присутствует ли дополнительная часть 1 заголовка.

bs_header_extra_2 — Указывает, присутствует ли дополнительная часть 2 заголовка.

bs_freq_scale Входной параметр для функции, которая вычисляет таблицу задающей частоты, определяемый с помощью таблицы 105.

Таблица 105 — Векторный элемент *bs_freq_scale*

bs_freq_scale	Значение
0 1 2 (по умолчанию) 3	Линейный 12 полос/октава 10 полос/октава 8 полос/октава

bs_alter_scale Входной параметр для функции, которая вычисляет таблицу задающей частоты, определяемый с помощью таблицы 106.

Таблица 106 — Векторный элемент *bs_alter_scale*

bs_alter_scale	Значение для *bs_freq_scale* = 0	Значение для *bs_freq_scale* > 0
0 1 (по умолчанию)	Никаких группировок каналов Группы из 2 каналов	Никаких изменений Сверхширокие полосы в самом высоком диапазоне

bs_noise_bands Входной параметр для функции, которая вычисляет таблицу полосы шумов, определяемый с помощью таблицы 107.

Таблица 107 — Векторный элемент *bs_noise_bands*

bs_noise_bands	Значение
0 1 2 (по умолчанию) 3	1 полоса 1 полоса/октава 2 полосы/октавы 3 полосы/октавы

bs_limiter_bands Входной параметр для функции, которая вычисляет таблицу полосы ограничителя, определяемый с помощью таблицы 108.

Таблица 108 — Векторный элемент *bs_limiter_bands*

bs_limiter_bands	Значение	Примечание
0 1 2 (по умолчанию) 3	1 полоса 1,2 полосы/октава 2,0 полосы/октава 3,0 полосы/октава	Единственная полоса Многополосный Многополосный Многополосный

bs_limiter_gains Определяет максимальное усиление ограничителей согласно таблицы 109.

Таблица 109 — Векторный элемент *bs_limiter_gains*

bs_limiter_gains	Значение
0 1 2 значений по умолчанию 3	-3 дБ Максимальное усиление 0 дБ Максимальное усиление 3 дБ Максимальное усиление *Inf.* дБ Максимальное усиление (то есть ограничитель отключен)

bs_interpol_freq Определяет, должна ли быть применена интерполяция частоты согласно таблицы 110.

Таблица 110 — Векторный элемент *bs_interpol_freq*

bs_interpol_freq	Значение
0 1 (по умолчанию)	Выключено Включено

bs_smoothing_mode Определяет, должно ли применяться сглаживание согласно таблицы 111.

Таблица 111 — Векторный элемент *bs_smoothing_mode*

bs_smoothing_mode	Значение
0 1 (по умолчанию)	Включено Выключено

sbr_single_channel_element () Синтаксический элемент, который содержит данные для элемента *SBR* с единственным каналом.

sbr_channel_pair_element () Синтаксический элемент, который содержит данные для элемента *SBR* с парой каналов.

sbr_channel_pair_base_element () Синтаксический элемент, который содержит данные базового уровня для элемента *SBR* с парой каналов, в масштабируемой системе.

sbr_channel_pair_enhance_element () Синтаксический элемент, который содержит данные уровня расширения для элемента *SBR* с парой каналов, в масштабируемой системе.

bs_data_extra Указывает, присутствуют ли зарезервированные биты в части данных *SBR*.

sbr_grid () Синтаксический элемент, который содержит частотную сетку времени.

sbr_dtdf () Синтаксический элемент, который содержит информацию о том, как кодированы дельта данные об огибающей и шуме.

sbr_invf () Синтаксический элемент, который содержит инверсные данные фильтрации.

sbr_envelope () Синтаксический элемент, который содержит кодированные по Хаффману данные огибающей.

sbr_noise () Синтаксический элемент, который содержит кодированные по Хаффману данные о минимальном уровне шума.

bs_add_harmonic_flag Указывает, присутствует ли информация о синусоидальном кодировании.

sbr_sinusoidal_coding () Синтаксический элемент, который содержит данные синусоидального кодирования

bs_extended_data Указывает, присутствует ли расширенный элемент данных *SBR*.

bs_extension_size Определяет размер расширенного элемента данных *SBR* в байтах

bs_esc_count Дополнительно определяет размер расширенного элемента данных *SBR* в случаях, когда размер больше чем 14 байтов.

bs_extension_id Определяет *ID* расширенного элемента данных *SBR*, для будущего использования согласно таблицы 112.

Таблица 112 — Векторный элемент *bs_extension_id*

bs_extension_id	Значение
0	Зарезервировано
1	Зарезервировано
2	*EXTENSION_ID_PS*
3	Зарезервировано

sbr_extension () Синтаксический элемент для будущих расширений.
bs_coupling Указывает, связана ли информация стерео между двумя каналами или нет согласно таблицы 113.

Таблица 113 — Векторный элемент *bs_coupling*

bs_coupling	Значение
0	Каналы не связываются
1	Каналы связываются

bs_frame_class Указывает на класс фрейма текущего фрейма *SBR* согласно таблицы 114.

Таблица 114 — Векторный элемент *bs_frame_class*

bs_frame_class	Значение
0	*FIXFIX*
1	*FIXVAR*
2	*VARFIX*
3	*VARVAR*

tmp Переменная помощника, используемая в *sbr_grid ()*
bs_num_env Указывает на число огибающих *SBR* в текущем фрейме *SBR*
bs_freq_res Указывает разрешающую способность по частоте для каждого канала и огибающей *SBR* согласно таблицы 115.

Таблица 115 — Векторный элемент *bs_freq_res*

bs_freq_res []	Значение
0	Низкочастотное разрешение
1	Высокочастотное разрешение

bs_pointer Указатель на определенную границу
bs_var_bord_0 Указывает позицию ведущей границы переменной для класса *VARVAR* и *VARFIX*
bs_var_bord_1 Указывает позицию задней границы переменной для класса *VARVAR* и *FIXVAR*
bs_num_rel_0 Указывает число относительных границ, начиная с *bs_var_bord_0*
bs_num_rel_1 Указывает на число относительных границ, начиная с *bs_var_bord_1*
bs_rel_bord_0 Указывает длины относительных границ, начиная с *bs_var_bord_0*
bs_rel_bord_1 Указывает длины относительных границ, начиная с *bs_var_bord_1*
bs_df_env Указывает направление дельта-кодирования для каждой огибающей *SBR* согласно таблицы 116.

Таблица 116 — Векторный элемент *bs_df_env*

bs_df_env []	Значение
0	Применить дельта-декодирование в частотном направлении для обозначенной полосы частот
1	Применить дельта-декодирование во временном направлении для обозначенной полосы частот

bs_df_noise Указывает направление дельта-кодирования для каждого минимального уровня шума согласно таблицы 117.

Таблица 117 — Векторный элемент *bs_df_noise*

bs_df_noise []	Значение
0	Применить дельта-декодирование в частотном направлении для обозначенной полосы частот
1	Применить дельта-декодирование во временном направлении для обозначенной полосы частот

bs_invf_mode Указывает на уровень инверсной фильтрации для каждой полосы частот соответственно:

Таблица 118 — Векторный элемент *bs_invf_mode:*

bs_invf_mode []	Значение
0	Инверсная фильтрация отсутствует
1	Низкоуровневая инверсная фильтрация
2	Промежуточная инверсная фильтрация
3	Строгая инверсная фильтрация

bs_env_start_value_balance Содержит первый масштабный коэффициент огибающей в случае связанной полезной нагрузки потока битов стерео.

bs_data_env Содержит необработанные масштабные коэффициенты огибающей для каждого канала, огибающей *SBR* и полосы.

bs_env_start_value_level Содержит первый масштабный коэффициент огибающей в случае полезной нагрузки потока битов несвязанного стерео или моно.

sbr_huff_dec () Декодер Хаффмана.

bs_codeword Кодовое слово Хаффмана.

bs_noise_start_value_balance Содержит первое значение минимального уровня шума в случае полезной нагрузки связанного потока битов стерео.

bs_data_noise Содержит необработанные данные минимального уровня шума для каждой огибающей и полосы.

bs_noise_start_value_level Содержит первое значение минимального уровня шума в случае полезной нагрузки потока битов несвязанного стерео или моно.

bs_add_harmonic Указывает, должно ли быть синусоидальное добавление к определенной полосе частот согласно таблице 119.

Таблица 119 — Векторный элемент *bs_add_harmonic*

bs_add_harmonic []	Значение
0	Не добавлять синусоиду к указанной полосе частот
1	Добавить синусоиду к указанной полосе частот

5.2.8.2 Процесс декодирования

5.2.8.2.1 Обзор фрейма *SBR*

Поле *CRC* (если применяется) содержит контрольную сумму циклического избыточного кода длиной 10 битов. Контрольная сумма должна быть вычислена, охватывая весь диапазон данных *SBR*, включая возможный *bs_fill_bits.*

Флаг *bs_header_flag*, если установлен, указывает, что присутствует часть заголовка *SBR*. Часть заголовка *SBR* содержит фундаментальную информацию, такую как частотный диапазон *SBR*, а также

управляющие сигналы, которые не требуют частых изменений (обозначенный как настройка (*tuning*)). Перед декодированием *SBR* должна присутствовать часть заголовка *SBR*. Пока никакая часть заголовка *SBR* не присутствует, декодер *SBR* выполняет только повышающую дискретизацию и корректировку задержки. В непрерывных широковещательных приложениях элементы данных расширения *SBR* с частью заголовка *SBR* обычно отправляются дважды в секунду. Кроме того, если требуется, в любое время мгновенно может быть вставлена часть заголовка *SBR* и изменены параметры заголовка.

Часть данных *SBR* может быть подразделена на дополнительную информацию и необработанные данные, где дополнительная информация определяется как сигналы, необходимые, чтобы декодировать необработанные данные и некоторые сигналы настройки декодера. Необработанные данные состоят из масштабных коэффициентов кодированной по Хаффману огибающей и оценок минимального уровня шума. Часть сетки описывает, как текущий фрейм *SBR* подразделяется во времени на временные сегменты и разрешающую способность по частоте этих временных сегментов. Часть *dtdf* сигнализирует, как данные кодируются (дельта-кодирование во временном или частотном направлении).

5.2.8.2.2 Полезная нагрузка расширения *SBR* для аудио объекта типов *AAC main*, *AAC SSR*, *AAC LC* и *AAC LTP*

На синтаксический элемент *AAC*, который должен быть расширен с помощью *SBR*, используется один элемент заполнения *SBR*. Элементы *SBR* вставляются в *raw_data_block ()* после соответствующих элементов *AAC*. За каждым *AAC SCE, CPE* или независимо переключаемым *CCE* должен следовать соответствующий элемент *SBR*. Элементы *LFE* декодируются согласно стандартным процедурам *AAC*, но должны быть с корректированной задержкой и передискретизированы, чтобы соответствовать выходной частоте выборок.

Соединение временного интервала независимо от коммутируемого *CCE* выполняется после декодирования *SBR*. Сначала к целевому *SCE* или каналам *CPE* добавляется зависимо коммутируемый *CCE* и после этого добавления применяется *SBR*.

5.2.8.2.3 Полезная нагрузка расширения *SBR* для аудио объекта типов *ER AAC LC* и *ER AAC LTP*

Число и порядок элементов данных расширения *SBR* (если присутствуют) даются с помощью *channelConfiguration*. К каждому *SCE* или *CPE* в одном *er_raw_data_block ()* есть соответствующий *SBR extension_payload ()*, содержащий или *sbr_extension_data* (*ID_SCE*), или *sbr_extension_data* (*ID_CPE*). Для *LFE* нет никакого *SBR extension_payload* (). Элементы *LFE* декодируются согласно стандартным процедурам *AAC*, но должны иметь скорректированную задержку и быть передискретизированы, чтобы соответствовать по выходной частоте дискретизации. Только элементы данных расширения *SBR* без проверки *CRC* допускаются для типов аудио объекта *ER AAC LC* и *ER AAC LTP*. Элементы расширения *SBR* должны быть помещены после любых других элементов расширения.

5.2.8.2.4 Полезная нагрузка расширения *SBR* для аудио объекта типов масштабируемый *AAC* и масштабируемый *ER AAC*

Масштабируемость пропускной способности и масштабируемость моно/стерео базового кодера *AAC* поддерживаются инструментом расширения полосы пропускания *SBR*. Для масштабируемости пропускной способности базового кодера данные *SBR* передаются в самом низком уровне базового кодера *AAC* и они охватывают самый большой частотный диапазон *SBR*, используемый в масштабируемой системе. Стартовая частота, передаваемая для частотного диапазона *SBR*, устанавливается в самую низкую верхнюю частоту частотного диапазона базового кодера *AAC*. Поэтому такая система масштабируемой полосы пропускания базового кодера с *SBR* обеспечивает постоянную полосу пропускания аудиосигнала для всех уровней. Если для декодирования доступен только самый низкий уровень, высокочастотный диапазон охватывается *SBR*. Если доступны более высокие уровни, высокочастотный диапазон частично (или полностью) охватывается *AAC* и сигнал *SBR* только частично необходим для остающейся верхней части высокочастотного диапазона, которая не охватывается *AAC*.

Масштабируемые данные *SBR* встраиваются в поток *MPEG*-4 так же, как в случае немасштабируемых элементов данных *SBR*, посредством использования *extension_payload ()*. Элементы расширения *SBR* должны быть помещены после любых других элементов расширения. Различные типы уровней масштабируемого *SBR* для всех возможных конфигураций масштабируемости полосы пропускания и масштабируемости моно/стерео базового кодера *AAC* описываются в таблице 120.

Таблица 120 — Данные *SBR* в масштабируемых уровнях *SBR*

Описание	*sbr_layer*	*sbr_extension_data() в extension_payload()*
Немасштабируемый базовый кодер *AAC*	*SBR_NOT_SCALABLE*	Да
Первый уровень масштабируемого базового кодера (моно) *AAC*	*SBR_MONO_BASE*	Да
Первый уровень *AAC* стерео в конфигурации масштабируемого базового кодера AAC моно/стерео	*SBR_STEREO_ENHANCE*	Да
Первый уровень масштабируемого базового кодера *AAC* (стерео)	*SBR_STEREO_BASE*	Да
Все другие уровни масштабируемого базового кодера *AAC*		Нет

5.2.9 Полезная нагрузка расширения

5.2.9.1 Элементы данных

extension_type Четырехбитовое поле, указывающее тип контента элемента заполнения. См. таблицу 121.

fill_nibble Четырехбитовое поле для заливки, должно быть установлено в '0000'.

fill_byte Байт, который будет отброшен декодером, должен быть установлен в '10100101' (для гарантии того, что самосинхронизирующиеся потоки данных, такие как в радио-модемах, могут выполнять надежное восстановление тактового сигнала).

data_element_version Четырехбитовое поле, указывающее версию элемента данных. См. таблицу 122.

dataElementLengthPart Поле, указывающее длину элемента данных полезной нагрузки расширения. Значение 255 используется в качестве значения *escape* (переход) и указывает, что следует, еще одно значение *dataElementLengthPart*. Полная длина передаваемого элемента данных вычисляется суммированием частичных значений.

data_element_byte Переменная, указывающая частичные значения 'элемента данных' полезной нагрузки расширения с типом *'ANC_DATA'* в байтах

other_bits Биты, которые будут отброшены декодером.

5.2.9.2 Элементы помощника

align Переменная помощника, указывающая количество битов, которые уже обработаны, чтобы выполнить требования выравнивания байта в полезной нагрузке расширения.

loopCounter Переменная помощника, указывающая длину переменной *dataElementLength* в байтах.

dataElementLength Переменная помощника, указывающая длину элемента данных полезной нагрузки расширения.

5.2.9.3 Таблицы

Таблица 121 — Значения поля *extension_type*

Символ	Значение *extension_type*	Назначение
EXT_FILL	'0000'	Заполнитель полезной нагрузки потока битов
EXT_FILL_DATA	'0001'	Данные полезной нагрузки потока битов как заполнитель
EXT_DATA_ELEMENT	'0010'	Элемент данных
EXT_DYNAMIC_RANGE	'1011'	Управление динамическим диапазоном
EXT_SAC_DATA	'1100'	*MPEG Surround*
EXT_SBR_DATA	'1101'	Расширение *SBR*
EXT_SBR_DATA_CRC	'1110'	Расширение *SBR* с *CRC*

Окончание таблицы 121

Символ	Значение *extension_type*	Назначение
—	Все другие значения	Зарезервировано. Эти значения могут быть использованы для дальнейшего расширения синтаксиса совместимым способом.
Примечание — К полезной нагрузке потока битов должны быть добавлены полезные нагрузки расширения типа *EXT_FILL* или *EXT_FILL_DATA*, если суммарное количество битов для всех аудиоданных вместе со всеми дополнительными данными меньше, чем минимально разрешенное количество битов в этом фрейме, необходимое для достижения целевой скорости передачи. Полезные нагрузки расширения избегают при нормальных условиях, и свободные биты используются, чтобы заполнить разрядный резервуар. Полезные нагрузки расширения записываются только если разрядный резервуар полон.		

5.2.9.4 Процесс декодирования

Таблица 122 — Значения *data_element_version*

Символ	Значение *data_element_version*	Назначение
ANC DATA	'0000'	Вспомогательный элемент данных
—	Все другие значения	Зарезервировано

5.2.10 Окружение *MPEG* (пространственное аудиокодирование)

Элемент синтаксиса *sac_extension_data* () используется, чтобы встроить дополнительную информацию пространственного аудиокодирования для декодирования *MPEG Surround.*

5.2.11 Полезные нагрузки для расширений аудиообъектного типа *ER BSAC*

5.2.11.1.1 Введение

Расширенный *ER BSAC* применяется, чтобы улучшить функционирование с полезными нагрузками расширения, такими как многоканальная, *SBR*, и *MPEG Surround.* Это расширение разрабатано, чтобы поддержать функциональность *FGS*, когда поток битов передается по системе *MPEG*-4. *ER BSAC* может обеспечить несколько рабочих режимов для различных сценариев приложения, как показано в таблице 123.

Таблица 123 — Рабочие режимы расширений *ER BSAC*

Рабочий режим	Полезная нагрузка расширения
BSAC моно/стерео	Недоступна
BSAC многоканальный	Многоканальная полезная нагрузка расширения *BSAC*
BSAC моно/стерео с *SBR*	Полезная нагрузка *BSAC SBR*
BSAC многоканальный с *SBR*	Многоканальная полезная нагрузка расширения *BSAC* и полезная нагрузка *BSAC SBR*
BSAC с *MPEG Surround*	Полезная нагрузка *MPEG Surround*

Интеграция *ER BSAC* и *SBR* должна обеспечить мелкоструктурное масштабируемое воспроизведение с расширением полосы пропускания для стерео и многоканальности. В привилегированных режимах работы интеграция *ER BSAC* и *SBR*, компоненты высокой частоты звука могут быть или сигналом уровня расширения *ER BSAC*, или синтезируемым сигналом *SBR*.

5.2.11.1.2 Процесс декодирования

5.2.11.1.2.1 *zero_code* и *sync_word*

Элементы синтаксиса *zero_code* и *sync_word* анализируются, в то время как данные доступны после декодирования моно или стерео части *BSAC*. Элемент синтаксиса *zero_code* используется для арифметического завершения моно или стерео части *BSAC*, и *sync_word* используется, чтобы указать начало расширенной части.

5.2.11.1.2.2 *extended_bsac_raw_data_block*

extended_bsac_raw_data_block содержит многоканальную информацию и имеет многоуровневую структуру как *bsac_raw_data_block*.

5.2.11.1.2.3 *extended_bsac_base_element*

Расширенный *bsac_base_element* состоит из *element_length*, *channel_configuration_index*, *reserved_bit*, *bsac_header*, *general_header* и *bsac_layer_element*. Для части стерео значение *nch* получается из *channelConfiguration* и это ограничивается 1 или 2 (левый и правый фронтальные динамики). Для расширенной части, параметр *nch* связан с остальными динамиками, и точное значение определяется *channel_configuration_index*, определенным в таблице 100. Каждый индекс указывает число каналов, учитывая канал для отображения динамиков.

5.2.11.1.2.4 *extended_bsac_sbr_data*

Элементы *SBR* вставляются в *bsac_raw_data_block* () после данных элемента уровня. Комбинируя базовый кодер *BSAC* и инструмент расширения полосы пропускания *SBR*, поддерживается функциональность мелкоструктурной масштабируемости с полной пропускной способностью для всех уровней. Стартовой частотой для частотного диапазона *SBR* является частота *base_band* базового уровня *BSAC*. Часть *SBR* охватывает высокочастоные области вне частотного диапазона базового уровня. Если передаются уровни расширения ядра *BSAC*, перекрытая область данных *SBR* заменяется базовыми данными.

5.2.11.1.2.5 *extended_bsac_sac_data*

Полезная нагрузка *MPEG Surround* встраивается в *bsac_raw_data_block* () после данных элемента уровня. Для полезной нагрузки *MPEG Surround* определен тип расширения *'EXT_BSAC_SAC_DATA'*. В случае, когда для полезной нагрузки *MPEG Surround* необходима проверка на ошибки *CRC*, флаг *bs_crc_flag* устанавливается в 1 и передается *ancCrcWord*. Интерфейс между *ER BSAC* и декодером *MPEG Surround* может быть или временным доменом или доменом *QMF*. В случае объединения *ER BSAC* с инструментом *SBR* представление *QMF* доступно как ввод декодера *MPEG Surround*.

5.2.11.1.3 Технический обзор интеграции *ER BSAC* и инструмента *SBR*

Масштабируемый инструмент *SBR* образуется добавлением модуля *HF-Overlap*. Процесс декодирования инструмента масштабируемого *SBR* идентичен процессу декодирования инструмента *SBR*, кроме модуля *HF-Overlap*.

5.2.11.1.4 Перекрытие *HF*

Масштабируемый поток битов *BSAC* и декодирование *SBR* выполняются параллельно. В базовом процессе декодирования *BSAC* значение *layer_max_freq* извлекается из *layer_end_index [layer]*. *layer_end_index [layer]* представляет верхнюю частоту самого высокого декодируемого уровня. Значение *layer_max_freq* доставляется для инструмента декодирования *SBR*.

В *SBR*, декодирующем процесс, вычисляется индекс поддиапазона *QMF*, соответствующий *layer_max_freq*. Индекс поддиапазона, обозначенный как *core_max_band*, определяется выражением

$$core_max_band = INT\left(64\frac{layer_max_freq}{1024}\right).$$

Если *core_max_band* больше чем исправленная подполоса *HF*, обозначенная как *HFPatchStartBand*, перекрытая область данных *SBR* заменяется базовыми данными *BSAC* в масштабируемой части *BSAC*.

5.3 Требования к буферу

5.3.1 Минимальный входной буфер декодера

Чтобы вычислить максимальное количество битов во входном буфере как для полезной нагрузки потока битов в целом, так и для любой данной программы или для любого данного *SCE/CPE/CCE* используются следующие правила.

Размер входного буфера составляет 6144 бита на *SCE* или на независимо переключаемый *CCE* плюс 12288 битов на *CPE*. Как размер общего буфера, так и размеры отдельных буферов ограничены таким образом чтобы предел буферизации мог быть вычислен для всей полезной нагрузки потока битов или для

отдельных аудио элементов, разрешая декодеру разбить многоканальную полезную нагрузку потока битов в отдельные полезные нагрузки потока битов моно и стерео, которые декодируются отдельными декодерами моно и стерео, соответственно. Все биты для *LFE's* или зависимого *CCE's* должны быть предоставлены из требований к общему буферу, основанных на независимых *CCE's*, *CPE's* и *SCE's*. Кроме того, все биты, требующиеся для любого *DSE's*, *PCE's*, *FIL's*, или фиксированные заголовки, переменные заголовки, *byte_alignment* и *CRC* должны быть предоставлены из тех же самых требований к общему буферу.

Для защиты полезной нагрузки от любой ошибки определяется дополнительный входной буфер декодера. Он на (*N* + 5) % больше, чем входной буфер для незащищенной полезной, где *N* является значением максимальной избыточности класса *FEC*. Все биты, требующиеся для любого *DSE's*, *PCE's*, *FIL's* или фиксированных заголовков, переменных заголовков, b*yte_alignment,* и *CRC*, должны быть предоставлены из тех же самых требований к общему буферу.

Для аудио объектного типа масштабируемого *AAC* применяются те же самые ограничения, однако здесь они применяются для объединенного размера входных буферов всех *ASME* и *ASEE*. Это означает, что, если кодируется программа моно, требуется размер буфера 6144=1024*6 битов, а для программы стерео доступен полный размер буфера 12288 битов. В случае масштабируемых конфигураций с обоими, моно- и стереоуровнями максимальный размер буфера для всех моноуровней составляет 6144 бита. Полный размер буфера для всех уровней составляет 12288 битов.

5.3.2 Разрядный резервуар

Разрядный резервуар управляется в кодере. Максимальный разрядный резервуар в кодере зависит от *NCC* и средней скорости передачи. Максимальный размер разрядного резервуара для каналов с постоянной скоростью может быть вычислен, вычитая среднее число битов на блок из минимального размера входного буфера декодера. Например, при 96 Кбит/с для сигнала стерео при частоте дискретизации 44,1 кГц среднее число битов на блок (*mean_framelength*) равно (96000 бит/с / 44100 1/*s* * 1024) = = 2229,1156….. Это приводит к максимальному размеру разрядного резервуара (*max_bit_reservoir*) *INT*(12288 битов-2229,1156….) = 10058. Для каналов с переменной скоростью передачи кодер должен работать таким способом, чтобы требования к входному буферу не превышали минимальный входной буфер декодера.

Состояние разрядного резервуара (*bit_reservoir_state*) передается в поле *buffer_fullness* либо как состояние разрядного резервуара, усеченное до целочисленного значения (*adif_buffer_fullness*), или как состояние разрядного резервуара, разделенное на *NCC*, разделенное на 32 и усеченное до целочисленного значения (*adts_buffer_fullness*).

bit_reservoir_state последующих фреймов может быть получено следующим образом:

bit_reservoir _ state[frame] = bit_reservoir_state[frame-1] + mean_ framelength – framelength [frame].

Длины фреймов должны быть скорректированы так, чтобы удовлетворять следующему ограничению:

$0 \le bit_reservoir_state[frame] \le max_bit_reservoir.$

5.3.3 Максимальная скорость передачи

Максимальная скорость передачи зависит от частоты дискретизации аудио. Она может быть подсчитана, основываясь на минимальном размере входного буфера согласно формуле:

$$maximum\ bitrate = \frac{6144\frac{bit}{blok}}{1024\frac{samples}{blok}} * sampling_frequency * NCC.$$

Таблица 126 дает некоторые примеры максимальных скоростей передачи на канал в зависимости от используемой частоты дискретизации.

Таблица 126 — Максимальная скорость передачи в зависимости от частоты дискретизации

Частоты дискретизации	Максимальная скорость передачи / *NCC*
48 кГц	288 Кбит/с
44,1 кГц	264,6 Кбит/с
32 кГц	192 Кбит/с

5.4 Таблицы

Таблица 127 — Окна преобразования

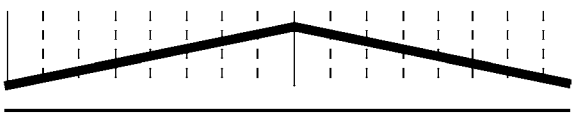
Окно	*num_swb*	Коэффициент	Выглядит как
LONG_WINDOW	49	1024/960	
SHORT_WINDOW	14	128/120	
LONG_START_WINDOW	49	1024/960	
LONG_STOP_WINDOW	49	1024/960	

Таблица 128 — Последовательности окон

Значение	*Window_sequence*	*Num_windows*	Выглядит как
1	*ONLY_LONG_SEQUENCE= LONG_WINDOW*	1	
2	*LONG_START_SEQUENC= LONG_START_WINDOW*	1	
3	*EIGHT_SHORT_SEQUENCE=* 8 * *SHORT_WINDOW*	8	
4	*LONG_STOP_SEQUENCE= LONG_STOP_WINDOW*	1	

Таблица 129 — Полосы масштабного коэффициента для длины окна 256 и 240 (значения для 240 в скобках) для *SHORT_WINDOW* при *fs* =32, *fs* =44,1 и *fs* =48 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_short_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_short_window*
0	0	8	44
1	4	9	56
2	8	10	68
3	12	11	80
4	16	12	96
5	20	13	112
6	28		128 (120)
7	36		

Таблица 130 — Полосы масштабного коэффициента для длины окна 2048 и 1920 (значения для 1920 в скобках) для *LONG_WINDOW*, *LONG_START_WINDOW*, *LONG_STOP_WINDOW* при *fs* = 44,1 и *fs* = 48 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
0	0	25	216
1	4	26	240
2	8	27	264
3	12	28	292
4	16	29	320
5	20	30	352
6	24	31	384
7	28	32	416
8	32	33	448
9	36	34	480
10	40	35	512
11	48	36	544
12	56	37	576
13	64	38	608
14	72	39	640
15	80	40	672
16	88	41	704
17	96	42	736
18	108	43	768
19	120	44	800
20	132	45	832
21	144	46	864
22	160	47	896
23	176	48	928
24	196		1024 (960)

Таблица 131 — Полосы масштабного коэффициента для длины окна 2048 и 1920 (значения для 1920 в скобках) для *LONG_WINDOW*, *LONG_START_WINDOW*, *LONG_STOP_WINDOW* при *fs* = 32 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
0	0	7	28
1	4	8	32
2	8	9	36
3	12	10	40
4	16	11	48
5	20	12	56
6	24	13	64

Окончание таблицы 131

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
14	72	33	448
15	80	34	480
16	88	35	512
17	96	36	544
18	108	37	576
19	120	38	608
20	132	39	640
21	144	40	672
22	160	41	704
23	176	42	736
24	196	43	768
25	216	44	800
26	240	45	832
27	264	46	864
28	292	47	896
29	320	48	928
30	352	49	960
31	384	50	992(-)
32	416		1024 (-)

Т а б л и ц а 132 — Полосы масштабного коэффициента для длины окна 2048 и 1920 (значения для 1920 в скобках) для *LONG_WINDOW*, *LONG_START_WINDOW*, *LONG_STOP_WINDOW* при *fs* = 8 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
0	0	13	156
1	12	14	172
2	24	15	188
3	36	16	204
4	48	17	220
5	60	18	236
6	72	19	252
7	84	20	268
8	96	21	288
9	108	22	308
10	120	23	328
11	132	24	348
12	144	25	372

Окончание таблицы 132

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
26	396	34	664
27	420	35	712
28	448	36	764
29	476	37	820
30	508	38	880
31	544	39	944
32	580		1024 (960)
33	620		

Таблица 133 — Полосы масштабного коэффициента для длины окна 256 и 240 (значения для 240 в скобках) для *SHORT_WINDOW* при *fs* = 8 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_short_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_short_window*
0	0	8	36
1	4	9	44
2	8	10	52
3	12	11	60
4	16	12	72
5	20	13	88
6	24	14	108
7	28		128 (120)

Таблица 134 — Полосы масштабного коэффициента для длины окна 256 и 240 (значения для 240 в скобках) для *SHORT_WINDOW* при *fs* = 11,025, *fs* = 12 и *fs* = 16 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_short_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_short_window*
0	0	8	36
1	4	9	40
2	8	10	48
3	12	11	60
4	16	12	72
5	20	13	88
6	24	14	108
7	28		128 (120)

Таблица 135 — Полосы масштабного коэффициента для длины окна 2048 и 1920 (значения для 1920 в скобках) для *LONG_WINDOW*, *LONG_START_WINDOW*, *LONG_STOP_WINDOW* при *fs* = 11,025, *fs* = 12 и *fs* = 16 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
0	0	22	228
1	8	23	244
2	16	24	260
3	24	25	280
4	32	26	300
5	40	27	320
6	48	28	344
7	56	29	368
8	64	30	396
9	72	31	424
10	80	32	456
11	88	33	492
12	100	34	532
13	112	35	572
14	124	36	616
15	136	37	664
16	148	38	716
17	160	39	772
18	172	40	832
19	184	41	896
20	196	42	960
21	212		1024 (-)

Таблица 136 — Полосы масштабного коэффициента для длины окна 2048 и 1920 (значения для 1920 в скобках) для *LONG_WINDOW*, *LONG_START_WINDOW*, *LONG_STOP_WINDOW* при *fs* = 22,05 и *fs* = 24 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
0	0	10	40
1	4	11	44
2	8	12	52
3	12	13	60
4	16	14	68
5	20	15	76
6	24	16	84
7	28	17	92
8	32	18	100
9	36	19	108

Окончание таблицы 136

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
20	116	34	364
21	124	35	396
22	136	36	432
23	148	37	468
24	160	38	508
25	172	39	552
26	188	40	600
27	204	41	652
28	220	42	704
29	240	43	768
30	260	44	832
31	284	45	896
32	308	46	960
33	336		1024 (-)

Таблица 137 — Полосы масштабного коэффициента для длины окна 256 и 240 (значения для 240 в скобках) для *SHORT_WINDOW* при *fs* =22,05 и *fs* =24 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_short_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_short_window*
0	0	8	36
1	4	9	44
2	8	10	52
3	12	11	64
4	16	12	76
5	20	13	92
6	24	14	108
7	28		128 (120)

Таблица 138 — Полосы масштабного коэффициента для длины окна 256 и 240 (значения для 240 в скобках) для *SHORT_WINDOW* при *fs* = 64 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_short_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_short_window*
0	0	7	32
1	4	8	40
2	8	9	48
3	12	10	64
4	16	11	92
5	20		128 (120)
6	24		

Таблица 139 — Полосы масштабного коэффициента для длины окна 2048 и 1920 (значения для 1920 в скобках) для *LONG_WINDOW*, *LONG_START_WINDOW*, *LONG_STOP_WINDOW* при *fs* = 64 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
0	0	24	172
1	4	25	192
2	8	26	216
3	12	27	240
4	16	28	268
5	20	29	304
6	24	30	344
7	28	31	384
8	32	32	424
9	36	33	464
10	40	34	504
11	44	35	544
12	48	36	584
13	52	37	624
14	56	38	664
15	64	39	704
16	72	40	744
17	80	41	784
18	88	42	824
19	100	43	864
20	112	44	904
21	124	45	944
22	140	46	984 (960)
23	156		1024 (-)

Таблица 140 — Полосы масштабного коэффициента для длины окна 2048 и 1920 (значения для 1920 в скобках) для *LONG_WINDOW*, *LONG_START_WINDOW*, *LONG_STOP_WINDOW* при *fs* = 88,2 и *fs* = 96 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
0	0	8	32
1	4	9	36
2	8	10	40
3	12	11	44
4	16	12	48
5	20	13	52
6	24	14	56
7	28	15	64

Окончание таблицы 140

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
16	72	29	276
17	80	30	320
18	88	31	384
19	96	32	448
20	108	33	512
21	120	34	576
22	132	35	640
23	144	36	704
24	156	37	768
25	172	38	832
26	188	39	896
27	212	40	960
28	240		1024 (-)

Таблица 141 — Полосы масштабного коэффициента для длины окна 256 и 240 (значения для 240 в скобках) для *SHORT_WINDOW* при *fs* = 88,2 и *fs* = 96 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_short_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_short_window*
0	0	7	32
1	4	8	40
2	8	9	48
3	12	10	64
4	16	11	92
5	20		128 (120)
6	24		

Таблица 142 — Полосы масштабного коэффициента для длины окна 960 при *fs* = 44,1 и *fs* = 48 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
0	0	9	36
1	4	10	40
2	8	11	44
3	12	12	48
4	16	13	52
5	20	14	56
6	24	15	64
7	28	16	72
8	32	17	80

Окончание таблицы 142

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
18	88	27	212
19	96	28	240
20	108	29	272
21	120	30	304
22	132	31	336
23	144	32	368
24	156	33	400
25	172	34	432
26	188		480

Таблица 143 — Полосы масштабного коэффициента для длины окна 1024 при *fs* = 4,1 и *fs* = 48 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
0	0	19	92
1	4	20	100
2	8	21	112
3	12	22	124
4	16	23	136
5	20	24	148
6	24	25	164
7	28	26	184
8	32	27	208
9	36	28	236
10	40	29	268
11	44	30	300
12	48	31	332
13	52	32	364
14	56	33	396
15	60	34	428
16	68	35	460
17	76		512
18	84		

Таблица 144 — Полосы масштабного коэффициента для длины окна 960 при *fs* = 32 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
0	0	19	88
1	4	20	96
2	8	21	104
3	12	22	112
4	16	23	124
5	20	24	136
6	24	25	148
7	28	26	164
8	32	27	180
9	36	28	200
10	40	29	224
11	44	30	256
12	48	31	288
13	52	32	320
14	56	33	352
15	60	34	384
16	64	35	416
17	72	36	448
18	80		480

Таблица 145 — Полосы масштабного коэффициента для длины окна 1024 при *fs* = 32 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
0	0	14	56
1	4	15	60
2	8	16	64
3	12	17	72
4	16	18	80
5	20	19	88
6	24	20	96
7	28	21	104
8	32	22	112
9	36	23	124
10	40	24	136
11	44	25	148
12	48	26	164
13	52	27	180

Окончание таблицы 145

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
28	200	33	352
29	224	34	384
30	256	35	416
31	288	36	448
32	320		480

Таблица 146 — Полосы масштабного коэффициента для длины окна 960 при *fs* = 22,05 и *fs* = 24 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
0	0	16	92
1	4	17	104
2	8	18	120
3	12	19	140
4	16	20	164
5	20	21	192
6	24	22	224
7	28	23	256
8	32	24	288
9	36	25	320
10	40	26	352
11	44	27	384
12	52	28	416
13	60	29	448
14	68		480
15	80		

Таблица 147 — Полосы масштабного коэффициента для длины окна 1024 при *fs* = 22,05 и *fs* = 24 кГц

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
0	0	9	36
1	4	10	40
2	8	11	44
3	12	12	52
4	16	13	60
5	20	14	68
6	24	15	80
7	28	16	92
8	32	17	104

Окончание таблицы 147

Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*	Номер окна *swb*	Полосы масштабного коэффициента *swb_offset_long_window*
18	120	25	320
19	140	26	352
20	164	27	384
21	192	28	416
22	224	29	448
23	256	30	480
24	288		512

Таблица 148 — Присвоение категории чувствительности к ошибкам *AAC* для основной полезной нагрузки

SCE/LFE/mono layer	*CPE, common_window= =0*	*CPE, common_window= =1/слой стерео*	*Data_element*	Функция
1	—	0	*max_sfb*	*aac_scalable_extension_header()*
—	—	0	*ms_mask_present*	*aac_scalable_extension_header()*
1	—	1	*tns_data_present*	*aac_scalable_extension_header()*
1	—	0	*ics reserved bit*	*aac_scalable_main_header()*
1	—	1	*ltp_data_present*	*aac_scalable_main_header()*
1	—	0	*max sfb*	*aac_scalable_main_header()*
—	—	0	*ms_mask_present*	*aac_scalable_main_header()*
1	—	0	*scale_factor_grouping*	*aac_scalable_main_header()*
—	—	0	*tns_channel_mono_layer*	*aac_scalable_main_header()*
1	—	1	*tns_data_present*	*aac_scalable_main_header()*
1	—	0	*window_sequence*	*aac_scalable_main_header()*
1	—	0	*window_shape*	*aac_scalable_main_header()*
—	0	0	*common window*	*channel_pair_element()*
—	0	0	*element_instance_tag*	*channel_pair_element()*
—	0	0	*ms_mask_present*	*channel_pair_element()*
—	0	0	*ms used*	*channel_pair_element()*
1	1	1	*diff control*	*diff_control_data()*
1	1	1	*diff control lr*	*diff_control_data_lr()*
1	1	0	*ics reserved bit*	*ics_info()*
1	1	1	*ltp_data_present*	*ics_info()*
1	1	0	*max sfb*	*ics_info()*
1	1	0	*predictor_data_present*	*ics_info()*
1	1	0	*scale_factor_grouping*	*ics_info()*

Продолжение таблицы 148

SCE/LFE/mono layer	*CPE, common_ window= =0*	*CPE, common_window= =1/слой стерео*	*Data_element*	Функция
1	1	0	*window_sequence*	*ics_info()*
1	1	0	*window_shape*	*ics_info()*
1	1	1	*gain_control_data_present*	*individual_channel_stream()*
1	1	1	*global_gain*	*individual_channel_stream()*
1	1	1	*length_of_longest_codeword*	*individual_channel_stream()*
1	1	1	*length_of_reordered_spectral_data*	*individual_channel_stream()*
1	1	1	*pulse_data_present*	*individual_channel_stream()*
1	1	1	*tns_data_present*	*individual_channel_stream()*
1	—	—	*element_instance_tag*	*lfe_channel_element()*
1	1	1	*ltp_coef*	*ltp_data()*
1	1	1	*ltp_lag*	*ltp_data()*
1	1	1	*ltp_lag_update*	*ltp_data()*
1	1	1	*ltp_long_used*	*ltp_data()*
—	—	0	*ms used*	*ms_data()*
1	1	1	*number_pulse*	*pulse_data()*
1	1	1	*pulse_amp*	*pulse_data()*
1	1	1	*pulse_offset*	*pulse_data()*
1	1	1	*pulse_start_sfb*	*pulse_data()*
4	4	4	*reordered_spectral_data*	*reordered_spectral_data()*
1	1	1	*dpcm_noise_last_position*	*scale_factor_data()*
1	1	1	*dpcm_noise_nrg*	*scale_factor_data()*
1	1	1	*hcod sf*	*scale_factor_data()*
1	1	1	*length_of_rvlc_escapes*	*scale_factor_data()*
1	1	1	*length_of_rvlc_sf*	*scale_factor_data()*
1	1	1	*rev_global_gain*	*scale_factor_data()*
2	2	2	*rvlc cod sf*	*scale_factor_data()*
2	2	2	*rvlc esc sf*	*scale_factor_data()*
1	1	1	*sf concealment*	*scale_factor_data()*
1	1	1	*sf_escapes_present*	*scale_factor_data()*
1	1	1	*sect cb*	*section_data()*
1	1	1	*sect len incr*	*section_data()*
1	—	—	*element_instance_tag*	*single_channel_element()*
4	4	4	*hcod*	*spectral_data()*
4	4	4	*hcod_esc_y*	*spectral_data()*

Окончание таблицы 148

SCE/LFE/mono layer	*CPE, common_ window= =0*	*CPE, common_window= =1/слой стерео*	*Data_element*	Функция
4	4	4	*hcod esc z*	*spectral_data()*
4	4	4	*pair_sign_bits*	*spectral_data()*
4	4	4	*quad_sign_bits*	*spectral_data()*
3	3	3	*coef*	*tns_data()*
3	3	3	*coef_compress*	*tns_data()*
3	3	3	*coef res*	*tns_data()*
3	3	3	*direction*	*tns_data()*
3	3	3	*length*	*tns_data()*
3	3	3	*n filt*	*tns_data()*
3	3	3	*order*	*tns_data()*

Таблица 149 — Присвоение категории чувствительности к ошибкам *AAC* для расширенной полезной нагрузки и полезной нагрузки *sbr* с малой задержкой

extension_payload	Полезна нагрузка *sbr* с малой задержкой	*data_element*	Функция
6	—	*drc_band_top*	*dynamic_range_info()*
6	—	*drc bands incr*	*dynamic_range_info()*
6	—	*drc_bands_present*	*dynamic_range_info()*
6	—	*drc bands reserved bits*	*dynamic_range_info()*
6	—	*drc_tag_reserved_bits*	*dynamic_range_info()*
6	—	*dyn_rng_ct*	*dynamic_range_info()*
6	—	*dyn_rng_sgn*	*dynamic_range_info()*
6	—	*excluded_chns_present*	*dynamic_range_info()*
6	—	*pce_instance_tag*	*dynamic_range_info()*
6	—	*pce_tag_present*	*dynamic_range_info()*
6	—	*prog_ref_level*	*dynamic_range_info()*
6	—	*prog_ref_level_present*	*dynamic_range_info()*
6	—	*prog_ref_level_reserved_bits*	*dynamic_range_info()*
6	—	*additional excluded chns*	*excluded_channels()*
6	—	*exclude mask*	*excluded_channels()*
5	—	*extension_type*	*extension_payload()*

Продолжение таблицы 149

extension_payload	Полезна нагрузка *sbr* с малой задержкой	*data_element*	Функция
5	—	*data element version*	*extension_payload()*
7	—	*fill_byte*	*extension_payload()*
7	—	*fill nibble*	*extension_payload()*
7	—	*other bits*	*extension_payload()*
8	—	*dataElementLengthPart*	*extension_payload()*
8	—	*data_element_byte*	*extension_payload()*
9	—	*bs sbr crc bits*	*sbr_extension_data()*
9	—	*bs_header_flag*	*sbr_extension_data()*
9	—	*bs fill bits*	*sbr_extension_data()*
9	9	*bs_amp_res*	*sbr_header()*
9	9	*bs_start_freq*	*sbr_header()*
9	9	*bs_stop_freq*	*sbr_header()*
9	9	*bs xover band*	*sbr_header()*
9	9	*bs reserved*	*sbr_header()*
9	9	*bs header extra 1*	*sbr_header()*
9	9	*bs header extra 2*	*sbr_header()*
9	9	*bs_freq_scale*	*sbr_header()*
9	9	*bs alter scale*	*sbr_header()*
9	9	*bs noise bands*	*sbr_header()*
9	9	*bs limiter bands*	*sbr_header()*
9	9	*bs_limiter_gains*	*sbr_header()*
9	9	*bs_interpol_freq*	*sbr_header()*
9	9	*bs_smoothing_mode*	*sbr_header()*
9	9	*bs data extra*	*sbr_single_channel_element()*
9	9	*bs reserved*	*sbr_single_channel_element()*
9	10	*bs_add_harmonic_flag*	*sbr_single_channel_element()*
9	10	*bs extended data*	*sbr_single_channel_element()*
9	10	*bs extension size*	*sbr_single_channel_element()*
9	10	*bs esc count*	*sbr_single_channel_element()*
9	10	*bs extension id*	*sbr_single_channel_element()*
9	9	*bs data extra*	*sbr_channel_pair_element()*
9	9	*bs reserved*	*sbr_channel_pair_element()*
9	9	*bs_coupling*	*sbr_channel_pair_element()*
9	10	*bs_add_harmonic_flag*	*sbr_channel_pair_element()*

Окончание таблицы 149

extension_payload	Полезна нагрузка *sbr* с малой задержкой	*data_element*	Функция
9	10	*bs extended data*	*sbr_channel_pair_element()*
9	10	*bs extension size*	*sbr_channel_pair_element()*
9	10	*bs esc count*	*sbr_channel_pair_element()*
9	10	*bs extension id*	*sbr_channel_pair_element()*
9	—	*bs frame class*	*sbr_grid()*
9	—	*tmp*	*sbr_grid()*
9	—	*bs_freq_res*	*sbr_grid()*
9	—	*bs_pointer*	*sbr_grid()*
9	—	*bs var bord 0*	*sbr_grid()*
9	—	*bs var bord 1*	*sbr_grid()*
9	—	*bs num rel 0*	*sbr_grid()*
9	—	*bs num rel 1*	*sbr_grid()*
9	9	*bs df env*	*sbr_dtdf()*
9	9	*bs df noise*	*sbr_dtdf()*
9	10	*bs invf mode*	*sbr_invf()*
9	10	*bs env start value balance*	*sbr_envelope()*
9	10	*bs env start value level*	*sbr_envelope()*
9	10	*bs codeword*	*sbr_envelope()*
9	10	*bs noise start value balance*	*sbr_noise()*
9	10	*bs noise start value level*	*sbr_noise()*
9	10	*bs codeword*	*sbr_noise()*
9	10	*bs add harmonic*	*sbr_sinusoidal_coding()*
—	9	*bs frame class*	*sbr_ld_grid()*
—	9	*tmp*	*sbr_ld_grid()*
—	9	*bs_freq_res*	*sbr_ld_grid()*
—	9	*bs_transient_position*	*sbr_ld_grid()*
—	9	*bs sbr crc bits*	*low_delay_sbr_data()*
—	9	*bs_header_flag*	*low_delay_sbr_data()*

6 Описания инструмента *GA*

6.1 Квантование

6.1.1 Описание инструмента

Для квантования спектральных коэффициентов в кодере используется неравномерный квантователь. Поэтому декодер после декодирования масштабных коэффициентов и спектральных данных по Хаффману должен выполнить инверсное неравномерное квантование.

6.1.2 Определения

Элементы справки:

x_quant [g] [win] [sfb] [bin] Квантованный спектральный коэффициент для группы *g*, окна *win*, полосы масштабного коэффициента *sfb*, коэффициента *bin*.

x_invquant [g] [win] [sfb] [bin] Спектральный коэффициент для группы *g*, окна *win*, полосы масштабного коэффициента *sfb*, коэффициента *bin* после инверсного квантования.

6.1.3 Процесс декодирования

Инверсное квантование описывается следующей формулой:

$$x_invquant = Sign(x_quant) \cdot |x_quant|^{\frac{4}{3}}.$$

Максимальная разрешенная абсолютная амплитуда для *x_quant* составляет 8191. Инверсное квантование применяется следующим образом:

```
for (g = 0; g < num_window_groups; g++) {
for (sfb = 0; sfb < max_sfb; sfb++) {
width = (swb_offset [sfb+1] - swb_offset [sfb]);
for (win = 0; win < window_group_len[g]; win++) {
for (bin = 0; bin < width; bin++) {
x_invquant[g][win][sfb][bin] = sign(x_quant[g][win][sfb][bin]) *
abs(x_quant[g][win][sfb][bin]) ^(4/3);
}
}
}
}
```

6.2 Масштабные коэффициенты

6.2.1 Описание инструмента

Основным методом корректировки шума квантования в частотной области является формирование шума, используя масштабные коэффициенты. С этой целью спектр делится на несколько групп спектральных коэффициентов, называемых полосами масштабного коэффициента, которые совместно используют один масштабный коэффициент. Масштабный коэффициент представляет собой значение усиления, которое используется для изменения амплитуды всех спектральных коэффициентов в этой полосе масштабного коэффициента. Этот механизм применяется для того, чтобы изменить распределение шума квантования в спектральной области, сгенерированного неравномерным квантователем.

Для *window_sequences*, которые содержат *SHORT_WINDOWs*, может быть применена группировка, то есть заданное количество последовательных *SHORT_WINDOWs* может иметь только один набор масштабных коэффициентов. Каждый масштабный коэффициент тогда применяется к группе полос масштабного коэффициента, соответствующих по частоте.

В этом инструменте масштабные коэффициенты применяются к инверсно квантованным коэффициентам, чтобы восстановить спектральные значения.

6.2.2 Определения

6.2.2.1 Элементы данных

global_gain 8-разрядное целочисленное значение без знака, представляющее значение первого масштабного коэффициента. Это также стартовое значение для следующих дифференциально кодированных масштабных коэффициентов.

scale_factor_data () Часть полезной нагрузки потока битов, которая содержит дифференциально кодированные масштабные коэффициенты

hcod_sf [] Кодовая комбинация Хаффмана из таблицы кодов Хаффмана, используемой для кодирования масштабных коэффициентов.

6.2.2.2 Элементы справки

dpcm_sf [g] [sfb] Дифференциально кодированный масштабный коэффициент группы *g*, полоса масштабного коэффициента *sfb*.

x_rescal [] Повторно масштабированные спектральные коэффициенты.

sf [g] [sfb] Массив для масштабных коэффициентов каждой группы.

get_scale_factor_gain () Функция, которая возвращает значение усиления, соответствующее масштабному коэффициенту.

6.2.3 Процесс декодирования

6.2.3.1 Полосы масштабного коэффициента

Масштабные коэффициенты используются, чтобы сформировать шум квантования в спектральном домене. С этой целью спектр делится на несколько полос масштабного коэффициента. У каждой полосы масштабного коэффициента есть масштабный коэффициент, который представляет определенное значение усиления, которое должно быть применено ко всем спектральным коэффициентам в этой полосе масштабного коэффициента. В случае *EIGHT_SHORT_SEQUENCE* полоса масштабного коэффициента может содержать несколько полос окна масштабного коэффициента последующего *SHORT_WINDOWs*.

6.2.3.2 Декодирование масштабных коэффициентов

Для всех масштабных коэффициентов разница с предыдущим значением кодируется, используя книгу кодов Хаффмана. Стартовое значение дается как 8 битов *PCM* в элементе данных *global_gain*. Масштабный коэффициент не передается для полос масштабного коэффициента, которые кодируются с помощью кодовой книги Хаффмана *ZERO_HCB*. Если кодовая книга Хаффмана для полосы масштабного коэффициента кодируется с применением *INTENSITY_HCB* или *INTENSITY_HCB2*, масштабный коэффициент используется для стерео-интенсивности. В этом случае обычный масштабный коэффициент не существует (но инициализируется обнуляясь, чтобы иметь действующий доступ в массив).

Следующий псевдокод описывает, как декодировать масштабные коэффициенты *sf [g] [sfb]:*

```
last_sf = global_gain;
for (g = 0; g < num_window_groups; g++) {
for (sfb = 0; sfb < max_sfb; sfb++) {
if (sfb_cb[g][sfb] != ZERO_HCB && sfb_cb[g][sfb] != INTENSITY_HCB
&& sfb_cb[g][sfb] != INTENSITY_HCB2 ) {
dpcm_sf = decode_huffman() - index_offset;
sf[g][sfb] = dpcm_sf + last_sf;
last_sf = sf[g][sfb];
}
else {
sf[g][sfb] = 0;
}
}
}
```

Масштабные коэффициенты *sf [g] [sfb]* должны быть в пределах диапазона от нуля до 255, оба включительно.

В случае устойчивого к ошибкам кодирования масштабного коэффициента вместо кода Хаффмана использовался *RVLC*. Процесс декодирования слов *RVLC* является тем же самым как для кодовых комбинаций Хаффмана, только должна использоваться другая кодовая книга. Этот сборник кодов использует симметричные кодовые комбинации. Благодаря этому возможно обнаружить ошибки, поскольку асимметричные кодовые комбинации недопустимы. Кроме того, декодирование может быть запущено на обеих сторонах. Чтобы позволить обратное декодирование, доступно дополнительное значение в пределах полезной нагрузки потока битов, которое содержит последнее значение масштабного коэффициента. В случае интенсивности доступна дополнительная кодовая комбинация, которая позволяет обратное декодирование. В случае *PNS* по той же самой причине доступно дополнительное значение *DPCM*.

В случае *sf_escapes_present* == 1, в качестве *ESC_FLAG* используется декодированное значение ± 7. Это сигнализирует, что существует значение *escape (переход)*, которое должно быть добавлено к + 7 или вычтено из – 7, чтобы найти фактическое значение масштабного коэффициента. Это значение *escape* является закодированным по Хаффману.

6.2.3.3 Применение масштабных коэффициентов

Спектральные коэффициенты всех полос масштабного коэффициента, которые соответствуют масштабному коэффициенту, должны быть повторно масштабированы согласно их масштабному коэффициенту. В случае последовательности окон, которая содержит группы коротких окон, все коэффициенты в сгруппированных полосах окон масштабного коэффициента должны масштабироваться, используя тот же самый масштабный коэффициент.

В случае *window_sequences* только с одним окном полосы масштабного коэффициента и соответствующие их коэффициенты находятся в спектральном порядке по возрастанию. В случае *EIGHT_SHORT_SEQUENCE* и группировки спектральные коэффициенты сгруппированных коротких окон чередуются полосами окна масштабного коэффициента.

Операция перемасштабирования производится согласно следующему псевдокоду:

```
for (g = 0; g < num_window_groups; g++) {
for (sfb = 0; sfb < max_sfb; sfb++) {
width = (swb_offset [sfb+1] - swb_offset [sfb] );
for (win = 0; win < window_group_len[g]; win++) {
gain = get_scale_factor_gain( sf[g][sfb] );
for (k = 0; k < width; k++) {
x_rescal[g][window][sfb][k] =
x_invquant[g][window][sfb][k] * gain;
}
}
}
}
```

Функция *get_scale_factor_gain (sf [g] [sfb])* возвращает коэффициент усиления, который соответствует масштабному коэффициенту. Возвращаемое значение следует уравнению:

$$gain = 2^{0,25 \cdot (\, sf[g][sfb] \, - \, SF_OFFSET)}.$$

Константа *SF_OFFSET* должна быть установлена в 100.

Следующий псевдокод описывает эту работу:

```
get_scale_factor_gain( sf[g][sfb] ) {
SF_OFFSET = 100;
gain = 2^(0,25 * ( sf[g][sfb] - SF_OFFSET));
return( gain );
}
```

6.3 Бесшумное кодирование

6.3.1 Описание инструмента

Бесшумное кодирование используется, чтобы дополнительно уменьшить избыточность масштабных коэффициентов и квантованный спектр каждого звукового канала.

global_gain кодируется как 8-битовое целое число без знака. Первый масштабный коэффициент ассоциированный с квантованным спектром дифференцированно кодируется относительно значения *global_gain*, а затем кодируется по Хаффману, используя сборник кодов масштабного коэффициента. Остающиеся масштабные коэффициенты дифференцированно кодируются относительно предыдущего масштабного коэффициента и затем кодируются по Хаффману, используя сборник кодов масштабного коэффициента.

Бесшумное кодирование квантованного спектра делится на два подразделения спектральных коэффициентов. Первое является подразделением полос масштабного коэффициента, которые содержат кратное 4 количество квантованных спектральных коэффициентов.

Второе подразделение, которое зависит от квантованных спектральных данных, является подразделением полос масштабного коэффициента на разделы формы. Значение раздела состоит в том, что квантованный спектр в пределах раздела представляется, используя единственный сборник кодов Хаффмана, выбранным из комплекта одиннадцати возможных сборников кодов. Длина раздела и связанного с ним сборника кодов Хафмана должна быть передана как дополнительная информация в дополнение к кодированному по Хаффману спектру раздела. Длина раздела дается в полосах масштабного коэффициента, а не в полосах окна масштабного коэффициента. Чтобы максимизировать соответствие статистики квантованного спектра этим сборникам кодов Хаффмана, разрешается столь же большое число разделов как число полос масштабного коэффициента. Максимальный размер раздела равен *max_sfb* полос масштабного коэффициента.

Как обозначено в таблице 151, спектральные сборники кодов Хаффмана могут представлять *n*-кортежи коэффициентов со знаком или без знака. Для сборников кодов без знака биты знака для каждого ненулевого коэффициента в *n*-кортеже следуют непосредственно за соответствующей кодовой комбинацией.

У бесшумного кодирования есть два способа представить большие квантованные спектры. Один способ состоит в том, чтобы отправить флаг перехода из сборника кодов Хаффмана переходов (*ESC*), который сигнализирует, что биты, следующие сразу после этой кодовой комбинации, плюс дополнительные биты знака являются *escape*-последовательностью, которая кодирует значения, большие чем те, которые представлены сборником кодов Хаффмана *ESC*. Вторым способом является импульсный метод перехода, в котором коэффициенты относительно большой амплитуды могут быть заменены коэффициентами с мень-

шими амплитудами, чтобы позволить использование таблиц кода Хаффмана с более высокой эффективностью кодирования. Эта замена корректируется отправкой позиции спектрального коэффициента и различий по амплитуде как дополнительной информации. Информация о частоте представляется комбинацией номера полосы масштабного коэффициента, чтобы указать на базовую частоту, и смещения в этой полосе масштабного коэффициента.

6.3.2 Определения

sect_cb [g] i]	Сборник кодов спектр Хаффмана, используемый для раздела *i* в группе *g*.
sect_len_incr	Используется для того, чтобы вычислить длину раздела, измеряет число полос масштабного коэффициента с начала раздела. Длина *sect_len_incr* составляет 3 бита, если *window_sequence* является *EIGHT_SHORT_SEQUENCE* и 5 битов в другом случае.
global_gain	Глобальное усиление квантованного спектра, отправленное как целочисленное значение без знака.
hcod_sf []	Кодовая комбинация Хаффмана из таблицы кодов Хаффмана, используемая для кодирования масштабных коэффициентов.
hcod [sect_cb [g] [i]] [w] [x] [y] [z]	Кодовая комбинация Хаффмана из сборника кодов *sect_cb [g] [i]*, которая кодирует следующие 4 кортежа (*w, x, y, z*) спектральных коэффициентов, где *w, x, y, z* являются квантованными спектральными коэффициентами. В пределах *n*-кортежа *w, x, y, z* упорядочиваются так, чтобы *x_quant [win] [sfb] [sfb] [bin] = w, x_quant [group] [win] [sfb] [bin+1] = x, x_quant [group] [win] [sfb] [bin+2] = y* и *x_quant [group] [win] [sfb] [bin+3] = z. n* -кортежи продвигаются от низкой до высокой частоты в пределах текущего раздела.
hcod [sect_cb [g] [i]] [y] [z]	Кодовая комбинация Хаффмана из сборника кодов *sect_cb [g] [i]*, которая кодирует следующий 2-кортеж (*y, z*) спектральных коэффициентов, где *y, z* — квантованные спектральные коэффициенты. В пределах *n*-кортежа *y, z* упорядочиваются так, чтобы *x_quant [group] [win] [sfb] [bin] = y* и *x_quant [group] [win] [sfb] [bin+1] = z. n*-кортежи прогрессируют от низкой до высокой частоты в пределах текущего раздела.
quad_sign_bits	Биты знака для ненулевых коэффициентов в спектральном 4-кортеже. '1' указывает отрицательный коэффициент, '0' — положительный. Биты, ассоциированные с коэффициентами более низкой частоты, отправляются первыми.
pair_sign_bits	Биты знака для ненулевых коэффициентов в спектральном 2-кортеже. '1' указывает отрицательный коэффициент, '0' — положительный. Биты, ассоциированные с коэффициентами более низкой частоты, отправляются первыми.
hcod_esc_y	*escape*-последовательность для квантованного спектрального коэффициента *y* 2-кортежа (y, z), ассоциированная с предшествовавшей кодовой комбинацией Хаффмана.
hcod_esc_z	*escape*-последовательность для квантованного спектрального коэффициента *z* 2-кортежа (*y, z*), ассоциированная с предшествовавшей кодовой комбинацией Хаффмана.
pulse_data_present	1 бит, указывающий, используется ли импульсный переход (1) или нет (0). Отметим, что *pulse_data_present* должно быть 0, если *window_sequence* = = *EIGHT_SHORT_SEQUENCE.*
number_pulse	2 бита, указывающие, сколько используется импульсных переходов. Число импульсных переходов лежит в пределах от 1 до 4.
pulse_start_sfb	6 битов, указывающих индекс самой низкой полосы масштабного коэффициента, где достигается импульсный переход.
pulse_offset [i]	5 битов, указывающих смещение.
pulse_amp [i]	4 бита, указывающие на величину импульса без знака.
sect_start [g] [i]	Смещение к первой полосе масштабного коэффициента в разделе *i* группы *g*.
sect_end [g] [i]	Смещение на единицу более высокой чем последняя полосе масштабного коэффициента в разделе *i* группы *g*.
num_sec [g]	Число разделов в группе *g*.
escape_flag	Значение 16 в сборнике кодов Хаффмана *ESC.*
escape_prefix	Разрядная последовательность *N* единиц.
escape_separator	Один нулевой бит.

escape_word	*N*+4 разрядное целочисленное слово без знака, сначала *msb*.
escape_sequence	Последовательность *escape_prefix*, *escape_separator* и *escape_word*
escape_code	2^ (*N*+4) + *escape_word*
x_quant [g] [win] [sfb] [bin]	Декодированное по Хаффману значение для группы *g*, окна *win*, полосы масштабного коэффициента *sfb*, коэффициента *bin*.
spec[w] [k]	Спектр с устраненным чередованием *w* в диапазоне от 0 до *num_windows*-1 и *k* в диапазе от 0 до *swb_offset* [*num_swb*]–1.

Инструмент бесшумного кодирования требует этих констант (см. таблицу 56).

ZERO_HCB	0
FIRST_PAIR_HCB	5
ESC_HCB	11
QUAD_LEN	4
PAIR_LEN	2
NOISE_HCB	13
INTENSITY_HCB2	14
INTENSITY_HCB	15
ESC_FLAG	16

6.3.3 Процесс декодирования

4-кортежи или 2-кортежи квантованных спектральных коэффициентов являются кодированными по Хаффману и передаются начиная с коэффициента самой низкой частоты и продвигаясь до коэффициента самой высокой частоты. Для случая нескольких окон на блок (*EIGHT_SHORT_SEQUENCE*) сгруппированный и чередующийся набор спектральных коэффициентов обрабатывается как единственный набор коэффициентов, которые следуют от низких до высоких. Набор коэффициентов, возможно, должен быть дечередован после того, как они декодируются. Коэффициенты сохраняются в массиве *x_quant* [*g*][*win*] [*sfb*] [*bin*] и порядок передачи кодовых комбинаций Хаффмана таков, что, когда они декодируются в порядке получения и сохраняются в массиве, *bin* является наиболее быстро увеличивающимся индексом, и *g* является наиболее медленно увеличивающимся индексом. Для кодовых комбинаций, связанных со спектральными 4-кортежами, порядок декодирования является *w*, *x*, *y*, *z*; для кодовых комбинаций, связанных со спектральными двойными кортежами, порядком декодирования является *y*, *z*. Набор коэффициентов делится на разделы, и информация о разделении передается начиная с самого низкого раздела частоты и продвигаясь до самого высокого раздела частоты. Спектральная информация для разделов, которые кодируются с "нулевым" сборником кодов, не отправляется, поскольку эта спектральная информация является нулем. Точно так же не отправляется спектральная информация для разделов, кодированных со сборниками кодов "интенсивности". Спектральной информацией для всех полос масштабного коэффициента при *max_sfb* и выше, для которых нет никаких данных о разделе, является нулем.

Есть единственный сборник кодов дифференциального масштабного коэффициента, который представляет диапазон значений, как показано в таблице 150. Сборник кодов дифференциального масштабного коэффициента показан в таблице А.1. Имеется одиннадцать сборников кодов Хаффмана для спектральных данных, как показано в таблице 151. Сборники кодов показаны в таблицах А.2—А.12. Существуют четыре других сборника кодов выше и вне фактических кодовых книг Хаффмана, а именно "нулевой" сборник кодов, указывающий, что ни масштабные коэффициенты, ни квантованные данные не будут переданы, и сборники кодов "интенсивности", указывающие, что этот отдельный канал является частью пары каналов, и что вместо данных, которые обычно были бы масштабными коэффициентами, даются данные управления для стереоинтенсивности. Точно так же сборник кодов „замена шума" указывает, что спектральные коэффициенты получаются из случайных чисел, а не путем квантования спектральных значений, и что вместо данных, которые обычно были бы спектральными коэффициентами, даются данные энергии шума. В этих случаях никакие квантованные спектральные данные не передаются. Индекс 12 сборника кодов резервируется.

Спектральные сборники кодов Хаффмана кодируют 2-ые или 4-ые кортежи квантованных спектральных коэффициентов без знака или со знаком, как показано в таблице 151. Эта таблица также показывает самое большое абсолютное значение (*LAV*), которое может быть закодировано каждым сборником кодов, и определяет булев массив переменной помощника *unsigned_cb []*, который является 1, если сборник кодов без знака, и 0, если со знаком.

Результатом декодирования по Хаффману каждой кодовой комбинации дифференциального масштабного коэффициента является индекс кодовой комбинации, приведенный в первом столбце таблицы А.1. Это преобразовывается в требуемый дифференциальный масштабный коэффициент, добавляя *index_offset*

к индексу. *Index_offset* имеет значение –60, как показано в таблице 150. Аналогично результатом декодирования по Хаффману каждого *n*-кортежа спектра является индекс кодовой комбинации, приведенный в первом столбце таблицы А.2 через таблицу А.12. Этот индекс преобразовывается в спектральные значения *n*-кортежа, как определено в следующем псевдо *C*-коде:

unsigned Булево значение *unsigned_cb [i]*, приведенное во второй графе таблицы 151.
dim Размерность сборника кодов, приведенная во второй графе таблицы 151.
lav *LAV*, приведенный в четвертой графе таблицы 151.
idx Индекс кодовой комбинации.

```
if (unsigned) {
mod = lav + 1;
off = 0;
}
else {
mod = 2*lav + 1;
off = lav;
}
if (dim == 4) {
w = INT(idx/(mod*mod*mod)) - off;
idx -= (w+off)*(mod*mod*mod);
x = INT(idx/(mod*mod)) - off;
idx -= (x+off)*(mod*mod);
y = INT(idx/mod) - off;
idx -= (y+off)*mod;
z = idx - off;
}
else {
y = INT(idx/mod) - off;
idx -= (y+off)*mod;
z = idx - off;
}
```

Если сборник кодов Хаффмана представляет значения со знаком, декодирование квантованного спектрального *n*-кортежа заканчивается после декодирования Хаффмана и преобразования индекса кодовой комбинации в квантованные спектральные коэффициенты. Если сборник кодов представляет значения без знака, тогда биты знака, ассоциированные с ненулевыми коэффициентами следуют сразу за кодовой комбинацией Хаффмана с '1', указывающей на отрицательный коэффициент, и '0', указывающим на положительный. Например, если кодовая комбинация Хаффмана из сборника кодов 7 *hcod [7] [y] [z]* была проанализирована, тогда сразу после этого в полезной нагрузке потока битов имеется *pair_sign_bits,* который является полем переменной длины от 0 до 2 битов. Это может быть проанализировано непосредственно из полезной нагрузки потока битов как:

```
if (y != 0)
if (one_sign_bit == 1)
y = -y ;
if (z != 0)
if (one_sign_bit == 1)
z = -z,
```

где *one_sign_bit* является следующим битом в полезной нагрузке потока битов, и *pair_sign_bits* является связью полей *one_sign_bit.*

Сборник кодов *ESC* является особым случаем. Он представляет значения от 0 до 16 включительно, но значения от 0 до 15 кодируют фактические значения данных, а значение 16 является *escape_flag*, который сигнализирует о присутствии *hcod_esc_y* или *hcod_esc_z*, любой из которых будет обозначен как *escape_sequence.* Этот *escape_sequence* разрешает закодировать квантованные спектральные элементы *LAV* > 15. Он состоит из *escape_prefix N* 1's, сопровождаемый *escape_separator* одного нуля, сопровождаемого *escape_word N*+4 битов, представляющих целочисленное значение без знака. У *escape_sequence* имеется декодированное значение 2 ^ (*N*+4) + *escape_word.* Требуемый квантованный спектральный коэффициент является тогда знаком, указанным *pair_sign_bits*, применяемым к значению *escape_sequence.* Другими словами, *escape_sequence* 00000 будет декодироваться как 16, *escape_sequence* 01111 как 31,

escape_sequence 1000000 как 32, а 1011111 как 63, и так далее. Для кодовых комбинаций Хаффмана *escape* упорядочивание элементов данных является кодовой комбинацией Хаффмана, сопровождаемой от 0 до 2 знаковыми битами и от 0 до 2 *escape*-последовательностей.

Когда *pulse_data_present* равен 1 (импульсный *escape* используется), один или несколько квантованных коэффициентов были замененны в кодере коэффициентами с меньшими амплитудами. Число замененных коэффициентов указывает *number_pulse.* В восстановлении квантованных спектральных коэффициентов *x_quant* эта замена компенсируется добавлением *pulse_amp* или вычитаем *pulse_amp* из ранее определенных коэффициентов, индексы частоты которых указываются *pulse_start_sfb* и *pulse_offset.* Импульсный метод *escape* недопустим для блока, *window_sequence* которого является *EIGHT_SHORT_SEQUENCE.* Процесс декодирования определяется в следующем псевдокоде *C*:

```
if (pulse_data_present) {
g = 0;
win = 0;
k = swb_offset[pulse_start_sfb];
for (j = 0; j < number_pulse+1; j++) {
k = pulse_offset[j];
/* translate_pulse_parameters(); */
for (sfb = pulse_start_sfb; sfb<num_swb;sfb++) {
if (k < swb_offset[sfb+1]) {
bin = k - swb_offset[sfb] ;
break;
}
}
/* restore coefficients */
if (x_quant[g][win][sfb][bin] > 0 )
x_quant[g][win][sfb][bin] += pulse_amp[j];
else
x_quant[g][win][sfb][bin] -= pulse_amp[j];
}
}
```

Несколько инструментов декодера (*TNS, filterbank*) получают доступ к спектральным коэффициентам нечередующимся способом, то есть все спектральные коэффициенты упорядочиваются согласно номеру окна и частоте в пределах окна. Это указывается путем использования нотации *spec[w] [k]*, а не *x_quant[g] [w] [sfb] [bin].*

Следующий псевдокод *C* указывает на соответствие между четырехмерной или чередующейся структурой массива *x_quant [] [] [] []* и двумерной или без чередования структурой *spec[][]* массива. В последнем массиве первый индекс постепенно увеличивается по отдельным окнам в последовательности окон, и второй индекс увеличивается по спектральным коэффициентам, которые соответствуют каждому окну, где коэффициенты линейно проходят от низкой до высокой частоты.

```
quant_to_spec() {
k = 0;
for (g = 0; g < num_window_groups; g++ ) {
j = 0;
for (sfb = 0; sfb < num_swb; sfb ++) {
width = swb_offset[sfb+1] - swb_offset[sfb];
for (win = 0; win < window_group_length[g]; win++) {
for (bin=0; bin < width; bin++) {
spec[win+k][bin+j] = x_quant[g][win][sfb][bin] ;
}
}
j+=width;
}
k+=window_group_length[g];
}
}
```

Декодирование переупорядоченных спектральных данных не может быть сделано напрямую. Следующее описание в стиле языка *C* показывает процесс декодирования:

```
/* helper functions */
void InitReordering(void);
/* Initializes variables used by the reordering functions like the segment
widths and the used offsets in segments and codewords */
void InitRemainingBitsInSegment(void);
/* Initializes remainingBitsInSegment[] array for each segment with the
total size of the segment */
int DecodeCodeword(codewordNr, segmentNr, direction);
/* Try to decode the codeword indexed by codewordNr using data already read
for this codeword and using data from the segment index by segmentNr.
The read direction in the segment is given by direction. DecodeCodeword
returns the number of bits read from the indexed segment. */
void MoveFromsegmentToCodeword(codewordNr, segmentNr, bitLen, direction); /*
Move bitLen bits from the segment indexed by segmentNr to the codeword indexed
by codewordNr using direction as read direction in the segment.
The bits are appended to existing bits for the codeword and the codeword length is adjusted. */
void AdjustOffsetsInSegment(segmentNr, bitLen, direction);
/* Like MoveFromsegmentToCodeword(), but no bits are moved. Only the offsets
for the segment indexed by segmentNr are adjusted according bitLen and direction. */
void MarkCodewordAsDecoded(codewordNr);
/* Marks the codeword indexed by codewordNr as decoded. */
bool CodewordIsNotDecoded(codewordNr);
/* Returns TRUE if the codeword indexed by codewordNr is not decoded. */
void ToggleReadDirection(void);
/* Toggles the read direction in the segments between forward and backward. */
/* (input) variables */
numberOfCodewords;
numberOfSegments;
numberOfSets;
DecodeReorderedSpectralData()
{
InitReordering();
InitRemainingBitsInSegment();
/* first step: decode PCWs (set 0) */
readDirection = forward;
for (codeword = 0; codeword < numberOfSegments; codeword++) {
cwLen = DecodeCodeword(codeword, codeword, readDirection);
if (cwLen <= remainingBitsInSegment[codeword]) {
AdjustOffsetsInSegment(codeword, cwLen, readDirection);
MarkCodewordAsDecoded(codeword);
remainingBitsInSegment[codeword] -= cwLen;
}
else {
/* error !!! (PCWs do always fit into segments) */
}
}
/* second step: decode nonPCWs */
for (set = 1; set < numberOfSets; set++) {
ToggleReadDirection();
for (trial = 0; trial < numberOfSegments; trial++) {
for (codewordBase = 0; codewordBase < numberOfSegments; codewordBase++) {
segment  = (trial + codewordBase) % numberOfSegments;
codeword = codewordBase + set*numberOfSegments;
```

```
if (CodewordIsNotDecoded(codeword) &&
(remainingBitsInSegment[segment] > 0)) {
cwLenInSegment = DecodeCodeword(codeword, segment, readDirection);
if (cwLenInSegment <= remainingBitsInSegment[segment]) {
AdjustOffsetsInSegment(segment, cwLenInSegment, readDirection);
MarkCodewordAsDecoded(codeword);
remainingBitsInSegment[segment] -= cwLenInSegment;
}
else { /* only part of codeword in segment */
MoveFromsegmentToCodeword(codeword,
segment,
remainingBitsInSegment[segment],
readDirection);
remainingBitsInSegment[segment] = 0;
}
}
}
}
}
}
```

4.6.3.4 Таблицы

Т а б л и ц а 150 — Параметры масштабного коэффициента сборника кодов Хаффмана

Номер сборника кодов	Размерность сборника кодов	*index_offset*	Диапазон значений	Сборник кодов отображен в таблице
0	1	–60	от –60 до +60	Таблица А.1

Т а б л и ц а 151 — Параметры спектра сборников кодов Хаффмана

Номер сборника кодов, *i*	*unsigned_cb[i]*	Размерность сборника кодов	*LAV* для сборника кодов	Сборник кодов отображен в таблице
0	—	—	0	—
1	0	4	1	Таблица А.2
2	0	4	1	Таблица А.3
3	1	4	2	Таблица А.4
4	1	4	2	Таблица А.5
5	0	2	4	Таблица А.6
6	0	2	4	Таблица А.7
7	1	2	7	Таблица А.8
8	1	2	7	Таблица А.9
9	1	2	12	Таблица А.10
10	1	2	12	Таблица А.11
11	1	2	16 (с *ESC* 8191)	Таблица А.12
12	—	—	Зарезервировано	—
13	—	—	Перцепционная шумовая замена	—
14	—	—	Несовпадение по фазе интенсивность	—
15	—	—	Интенсивность синфазно	—

Окончание таблицы 151

Номер сборника кодов, *i*	*unsigned_cb[i]*	Размерность сборника кодов	*LAV* для сборника кодов	Сборник кодов отображен в таблице
16	1	2	16 (*w/o ESC* 15)	Таблица А.12
17	1	2	16 (с *ESC* 31)	Таблица А.12
18	1	2	16 (с *ESC* 47)	Таблица А.12
19	1	2	16 (с *ESC* 63)	Таблица А.12
20	1	2	16 (с *ESC* 95)	Таблица А.12
21	1	2	16 (с *ESC* 127)	Таблица А.12
22	1	2	16 (с *ESC* 159)	Таблица А.12
23	1	2	16 (с *ESC* 191)	Таблица А.12
24	1	2	16 (с *ESC* 223)	Таблица А.12
25	1	2	16 (с *ESC* 255)	Таблица А.12
26	1	2	16 (с *ESC* 319)	Таблица А.12
27	1	2	16 (с *ESC* 383)	Таблица А.12
28	1	2	16 (с *ESC* 511)	Таблица А.12
29	1	2	16 (с *ESC* 767)	Таблица А.12
30	1	2	16 (с *ESC* 1023)	Таблица А.12
31	1	2	16 (с *ESC* 2047)	Таблица А.12

6.4 Бесшумное кодирование для мелкоструктурной масштабируемости

6.4.1 Описание инструмента

BSAC подерживает разрядно-модульное арифметическое кодирование и является именем бесшумного кодера и средством форматирования полезной нагрузки потока битов, которое обеспечивает мелкоструктурную масштабируемость и устойчивость к ошибкам в кодере *MPEG-4 General Audio (GA) coder.* Модуль бесшумного кодирования *BSAC* является альтернативой модуля кодирования *AAC*, в отношении всех других модулей кодер на базе *AAC* остается неизменным. Бесшумное кодирование *BSAC* используется, чтобы сделать полезную нагрузку потока битов масштабируемой и устойчивой к ошибкам, а также уменьшить избыточность масштабных коэффициентов и квантованного спектра. Процесс бесшумного декодирования *BSAC* разделяется на 4 подпункта. В 6.4.2—6.4.6 описывают подробный процесс декодирования спектральных данных, относящихся к стерео или *pns* данным, масштабные коэффициенты и дополнительную информацию о полосе кодирования.

6.4.2 Декодирование разрядно-модульных спектральных данных (*bsac_spectral_data*)

6.4.2.1 Описание

BSAC использует разрядно-модульную схему квантованных спектральных коэффициентов, чтобы обеспечить мелкоструктурную масштабируемость. Он также кодирует разрядно-модульные данные, используя схему двоичного арифметического кодирования, чтобы уменьшить средние биты, переданные не претерпевая потери точности.

В масштабируемой схеме кодирования *BSAC* квантованная последовательность делится на полосы кодирования. Квантованная последовательность отображается в разрядно-модульную последовательность в пределах полосы кодирования. Бесшумное кодирование разрядно-модульных битов опирается на таблицу вероятности полосы кодирования, значения и другие контексты.

Значение разрядно-модульных данных является позицией разрядно-модульного бита, который будет кодирован.

Флаги *sign_is_coded []* обновляются с кодированием векторов из *MCB* в *LSB*. Они инициализируются в 0, когда знак квантованного спектра кодируется, они устанавливаются в 1.

Таблица вероятности для кодирования разрядно-модульных данных в пределах каждой полосы кодирования включается в элемент потока битов *cband_si_type* и передается начиная с самой низкой полосы кодирования и продолжая до самой высокой полосы кодирования, выделенной каждому уровню.

6.4.2.2 Определения

6.4.2.2.1 Элементы данных

acod_sliced_bit [ch] [g] [i] Арифметическая кодовая комбинация, необходимая для арифметического декодирования разрядно-модульного бита. Используя этот декодируемый бит, мы можем восстановить значение каждого бита квантованного спектрального значения. Фактически восстановленное битовое значение является зависимым от значения разрядно-модульного бита.

acod_sign [ch] [g] [i] Арифметическая кодовая комбинация из двоичного арифметического кодирования *sign_bit*. Вероятность символа "0" определяется в 0,5, что использует 8192 в качестве 14-битового числа с фиксированной точкой. *sign_bit* указывает бит знака для ненулевого коэффициента. "1" указывает отрицательный коэффициент, а "0" — положительный. Когда значению бита квантованного сигнала впервые присваивается 1, бит знака арифметически кодируется и отправляется.

6.4.2.2.2 Элементы справки

layer Индекс уровня масштабируемости.

snf Значение вектора, который будет декодироваться.

ch Индекс канала.

nch Номер канала.

cur_snf [i] Текущего значение *i*-го вектора. *cur_snf[]* инициализируется в *Abit[cband]*.

maxsnf Максимум текущего значения векторов, которые будут декодироваться.

snf Индекс значения.

layer_data_available () Функция, которая возвращает "1" пока доступна полезная нагрузка потока битов каждого уровня, иначе — "0". Она указывает, доступна ли остающаяся полезная нагрузка потока битов каждого уровня.

layer_group [layer] Указывает групповой индекс спектральных данных, которые будут вновь добавлены в уровень масштабируемости.

layer_start_index [layer] Указывает индекс самого низкого спектрального компонента, который будет вновь добавлен в уровне масштабируемости.

layer_end_index [layer] Указывает индекс самого высокого спектрального компонента, который будет вновь добавлен в уровне масштабируемости.

start_index [g] Указывает индекс самого низкого спектрального компонента, который будет кодирован в группе *g*.

end_index [g] Указывает индекс самого высокого спектрального компонента, который будет кодирован в группе *g*.

sliced_bit Декодируемое значение разрядно-модульных битов квантованного спектра.

sample [ch] [g] [i] Квантованные спектральные коэффициенты, восстановленные из декодированных разрядно-модульных данных линии спектра *i* в канале *ch* и индекс группы *g*.

sign_is_coded [ch] [g] [i] Флаг, который указывает, кодирован ли в канале *ch* и группе с индексом *g* знак *i*-го квантованного спектра (1), или нет (0).

sign_bit [ch] [g] [i] Знаковый бит для ненулевого коэффициента. "1" указывает отрицательный коэффициент, а "0" — положительный. Когда значению бита квантованного сигнала впервые присваивается 1, знаковый бит арифметически кодируется и отправляется.

6.4.2.3 Процесс декодирования

В кодере *BSAC* абсолютные значения квантованных спектральных коэффициентов отображаются в разрядно-модульную последовательность. Эти разрядно-модульные биты являются символами арифметического кодирования. Все разрядно-модульные биты являются двоично-арифметически кодированными от коэффициента самой низкой частоты до коэффициента самой высокой частоты уровня масштабируемости, начиная с плоскости *Most Significant Bit* (старший значащий бит) (*MCB*) и продвигаясь до плоскости *Least Significant Bit* (младший значащий бит) (*LSB*). Арифметическое кодирование битов знака, связанных с ненулевым коэффициентом, следует за кодированием разрядно-модульного бита, когда разрядно-модульный бит спектрального коэффициента впервые равен 1.

Для случая нескольких окон на блок связанный и, возможно, сгруппированный и чередующийся набор спектральных коэффициентов обрабатывается как единственный набор коэффициентов, которые следуют от низких до высоких. Этот набор спектральных коэффициентов, возможно, должен быть дечередован после того, как они декодируются. Спектральная информация для всех полос масштабного коэффициента, равных или больше чем *max_sfb*, обнуляется.

После того, как все данные *MCB* кодируются от самой низкой линии частоты до самой высокой, тот же самый процесс кодирования повторяется, пока не кодируются данные *LSB*, или данные уровня недоступны.

Длина доступной полезной нагрузки потока битов (*available_len []*) инициализируется в начале каждого уровня. Предполагаемая длина кодовой комбинации (*est_cw_len*), которая будет декодироваться, вычисляется исходя из процесса арифметического декодирования. После арифметического декодирования символа длина доступной полезной нагрузки потока битов должна быть обновлена, вычитая из этого предполагаемую длину кодовой комбинации. Мы можем определить, доступна ли оставшаяся полезная нагрузка потока битов каждого уровня или нет, проверяя *available_len*.

Разрядно-модульные данные декодируются с вероятностью, которая выбирается среди значений, перечисленных в таблицах А.56—А.77.

Чтобы арифметически кодировать символы (разрядно-модульные биты) значение вероятности должно быть определено. Двоичная таблица вероятности составляется из значений вероятности (*p*0) символа '0'. Прежде всего, таблица вероятности выбирается используя *cband_si* как показано в таблице А.31. Затем в таблице вероятности выбирается подтаблица согласно контексту, такому как текущее значение спектрального коэффициента, и более высокие разрядно-модульности, которые декодировались. Все векторы более высоких разрядно-модульностей *higher_bit_vector* инициализируются в 0 перед запуском кодирования разрядно-модульных данных. Всякий раз, когда кодируется разрядно-модульность, вектор *higher_bit_vector* обновляется следующим образом:

higher_bit_vector[ch][g][i] = (higher_bit_vector[ch][g][i]<<1) + decoded_bitslice;
if(higher_bit_vector[ch][g][i]) {
if (higher_bit_vector[ch][g][i] > 15*)*
*p*0*_index* = 15*;*
else
*p*0*_index* = *higher_bit_vector[ch][g][i]* -1*;*
}.

Среди нескольких значений в подтаблице выбирается вероятность (*p*0). Чтобы выбрать одно из нескольких значений вероятности в подтаблице, должен быть решен индекс вероятности. Если вектор более высокой разрядно-модульности является ненулевым, индекс вероятности (*p*0) будет (*higher_bit_vector [ch] [g] [i]* - 1). Иначе выбор полагается на разрядно-модульные биты последовательных неналоженных 4 спектральных данных, как показано в таблице А.34.

Однако, если доступный размер кодовой комбинации меньше чем 14, есть ограничения на выбранное значение вероятности следующим образом:

if (available_len <14*) {*
*if (p*0 < *min_p*0*[available_len])*
*p*0 = *min_p*0*[available_len];*
*else if (p*0 > *max_p*0*[available_len])*
*p*0 = *max_p*0*[available_len];*
}

Минимальная вероятность *min_p*0*[]* и максимальная вероятность *max_p*0*[]* перечисляются в таблицах А.35 и А.36.

Есть 23 таблицы вероятности, которые могут использоваться для кодирования/декодирования разрядно-модульных данных. 23 таблицы вероятности обеспечивают покрытие различной статистики разрядно-модульности. Чтобы передать таблицу вероятности, используемую в процессе кодирования, таблица вероятности включается в элемент синтаксиса *cband_si.* После декодирования *cband_si* таблица вероятности отображается из *cband_si*, используя таблицу А.33, и декодирование разрядно-модульных данных должно быть запущено.

Текущее значение спектрального коэффициента представляет разрядную матрицу разрядно-модульности, которая будет декодироваться. Таблица А.33 показывает плоскость *MCB* декодируемой выборки согласно *cband_si.* Текущие значения *cur_snf []* всех спектральных коэффициентов в пределах полосы кодирования инициализируются в плоскость *MCB.*

Арифметическое декодирование бита знака, связанного с ненулевым коэффициентом, следует за арифметическим декодированием разрядно-модульного бита, когда битовое значение квантованного спектрального коэффициента впервые равно 1, с 1, указывающей отрицательный коэффициент, и 0, указывающим положительный. Флаг *sign_is_coded []* представляет декодировался ли бит знака квантованного спектра или нет. Прежде чем запускается декодирование разрядно-модульных данных, все флаги *sign_is_coded* устанавливаются в 0. После того, как бит знака декодируется, флаг *sign_is_coded* устанавливается в 1. Процесс декодирования бита знака может быть обобщен следующим образом:

i = the spectral line index
if (sample[ch][g][i] && !sign_is_coded[ch][g][i]) {
arithmetic decoding of the sign bit;
sign_is_coded[ch][g][i] = 1*;*
}

Декодируемый символ необходимо восстановить в выборке.

6.4.3 Декодирование *stereo_info, ms_used* и *noise_flag*

6.4.3.1 Описания

Схема масштабируемого кодирования *BSAC* включает бесшумное кодирование, которое отличается от кодирования *MPEG*-4 *AAC* и дополнительно уменьшает избыточность связанных со стерео данных.

Декодирование связанных со стерео данными и данными перцепционной шумовой замены (*pns*) зависит от *pns_data_present* и *stereo_info*, которые указывает маску стерео. Так как декодируемые данные являются тем же самым значением с *MPEG*-4 *AAC*, обработка связанных со стерео *MPEG*-4 *AAC* и *pns* следует за декодированием связанных со стерео данных и данных *pns*.

6.4.3.2 Определения

6.4.3.2.1 Элементы данных

acode_ms_used[g][sfb] — Арифметическое кодовое слово из арифметического кодирования *ms_used*, которое является однобитовым флагом на полосу масштабного коэффициента, указывающим, что кодирование *M/S* используется в группе окон *g* и полосе масштабного коэффициента *sfb*, следующим образом:
0 — независимый;
1 — *ms_used*.

acode_stereo_info[g][sfb] — Арифметическое кодовое слово из арифметического кодирования *stereo_info*, которое является двухбитовым флагом на полосу масштабного коэффициента, указывающим, что кодирование *M/S* или кодирование интенсивности используются в группе окон *g* и полосе масштабного коэффициента *sfb*, следующим образом:
1 — 00 независимый;
2 — 01 *ms_used*;
10 — 10 *Intensity_in_phase*;
11 — *Intensity_out_of_phase* или *noise_flag_is_used*.

Примечание — Если *ms_mask_present* равно 3, *noise_flag_l* и *noise_flag_r* равны 0, то *stereo_info* интерпретируется как несовпадающая по фазе стерео интенсивность независимо от величины *pns_data_present.*

acode_noise_flag[g][sfb] — Арифметическое кодовое слово из арифметического кодирования *noise_flag*, которое является 1-битовым флагом на полосу масштабного коэффициента, указывающим, используется ли перцепционная шумовая замена (1) или нет (0) в группе окон *g* и полосе масштабного коэффициента *sfb*.

acode_noise_flag_l[g][sfb] — Арифметическое кодовое слово из арифметического кодирования *noise_flag_l*, которое является 1-битовым флагом на полосу масштабного коэффициента, указывающим, используется ли перцепционная шумовая замена (1) или нет (0) в левом канале, группе окон *g* и полосе масштабного коэффициента *sfb*.

acode_noise_flag_r[g][sfb] — Арифметическое кодовое слово из арифметического кодирования *noise_flag*, которое является 1-битовым флагом на полосу масштабного коэфициента, указывающим, используется ли перцепционная шумовая замена (1) или нет (0) в правом канале, группе окон *g* и полосе масштабного коэффициента *sfb*.

acode_noise_mode[g][sfb] Арифметическое кодовое слово из арифметического кодирования *noise_mode*, которое является двухбитовым флагом на полосу масштабного коэффициента, указывающим, какая шумовая замена используется в группе окон *g* и полосе масштабного коэффициента *sfb*, следующим образом:
00 — шумовая замена *L+R* (независимая);
01 — шумовая замена *L+R* (коррелированная);
10 — шумовая замена *L+R* (коррелированная, несовпадающая по фазе);
11 — зарезервировано.

6.4.3.2.2 Элементы справки

ch Индекс канала.
g Индекс группы.
sfb Индекс полосы масштабного коэффициента в пределах группы.
layer Индекс уровня масштабируемости.
nch Номер канала
ms_mask_present Это двухбитовое поле указывает, какая маска стерео:
00 — независимая;
01 — однобитовая маска *ms_used* расположена в части дополнительной информации слоя *sfb*;
10 — все *ms_used* являются единицами;
11 — двухбитовая маска *stereo_info*, расположеная в части дополнительной информации слоя *sfb*.
layer_group[layer] Указывает индекс группы спектральных данных, которые будут вновь добавлены в слое масштабируемости.
layer_start_sfb[layer] Указывает индекс самой низкой полосы масштабного коэффициента, который будет вновь добавлен в слое масштабируемости.
layer_end_sfb[layer] Указывает индекс самой высокой полосы масштабного коэффициента, который будет вновь добавлен в слое масштабируемости.

6.4.3.3 Процесс декодирования

Процесс декодирования *ms_mask_present*, *noise_flag* или *ms_used* зависит от *pns_data_present*, номера канала и *ms_mask_present*. Флаг *pns_data_present* передается как элемент в синтаксисе *general_header* (). *Pns_data_present* указывает, используется ли инструмент *pns* или нет в каждом фрейме. *Stereo_info indeicates* указывает маску стерео следующим образом:

00 — независимая;
01 — однобитовая маска *ms_used* располагается в части дополнительной информации *sfb* уровня;
10 — все *ms_used* являются единицами;
11 — двубитовая маска *stereo_info* располагается в части дополнительной информации *sfb* уровня;

Процесс декодирования классифицируется следующим образом:

- 1 канал, данные *pns* отсутствуют:

если число каналов равно 1 и данные *pns* отсутствуют, никаких элементов данных, связанных со стерео или *pns* нет;

- 1 канал, данные *pns*:

если число каналов равно 1 и данные *pns* присутствуют, шумовой флаг полос масштабного коэффициента между *pns_start_sfb* и *max_sfb* арифметически декодируется, используя модель, показанную в таблице А.54. Замена перцепционного шума производится согласно флагу декодируемого шума;

- 2 канала, *ms_mask_present*=0 (независимый), данные *pns* отсутствуют:

если *ms_mask_present* равен 0 и данные *pns* отсутствуют, арифметическое декодирование *stereo_info* или *ms_used* не требуется;

- 2 канала, *ms_mask_present*=0 (независимый), данные *pns* присутствуют:

если *ms_mask_present* равен 0 и данные *pns* присутствуют, шумовой флаг для *pns* арифметически декодируется используя модель, показанную в таблице А.54. Перцепционная шумовая замена независимого режима производится согласно флагу декодируемого шума;

- 2 канала, *ms_mask_present*=2 (весь *ms_used*), данные *pns* присутствуют или данные *pns* отсутствуют:

все значения *ms_used* в этом случае являются единицами. Так, обработка стерео *M/S* для *AAC* производится во всей полосе масштабного коэффициента и не может быть никакой обработки *pns* независимо от флага *pns_data_present*;

- 2 канала, *ms_mask_present*=1 (опционно *ms_used*), данные *pns* присутствуют или данные *pns* отсутствуют:

в этом случае передается однобитовая маска полос *max_sfb* для *ms_used*. *ms_used* является арифметически декодируемой с использованием модели *ms_used*, данной в таблице А.52. Обработка *M/S* стерео для *AAC* производится или нет согласно декодируемому *ms_used*. Если *ms_used* равен 1, обработка *pns* отсутствует;

- 2 канала, *ms_mask_present*=3 (опционно *ms_used/intensity/pns*), данные *pns* отсутствуют.

Сначала арифметически декодируется *stereo_info* с использованием модели *stereo_info*, данной в таблице А.53.

stereo_info является двухбитовым флагом на полосу масштабного коэффициента указывающим, что кодирование *M/S* или кодирование интенсивности в группе окон *g* и полосе масштабного коэффициента *sfb* используется следующим образом:

00 — независимо;

01 — *ms_used;*

10 — *Intensity_in_phase;*

11 — *Intensity_out_of_phase*.

Если *stereo_info* не равен 0, стерео *M/S* или стерео интенсивности *AAC* производится с этими декодируемыми данными. Если же данные *pns* отсутствуют *pns* не обрабатывается;

- 2 канала, *ms_mask_present*=3 (опционно *ms_used/intensity/pns*), данные *pns stereo_info* арифметически декодируются, используя модель *stereo_info*, данную в таблице А.53:

если *stereo_info* равен 1 или 2, обработка стерео *M/S* или стерео интенсивности *AAC* производится с этими декодируемыми данными и какая-либо обработка *pns* отсутствует. Если *stereo_info* равен 3 и полоса масштабного коэффициента больше или равна *pns_start_sfb*, шумовой флаг для *pns* арифметически декодируется, используя модель, данную в таблице А.54;

если оба шумовых флага обоих каналов равны 1, режим шумовой замены арифметически декодируется, используя модель, данную в таблице А.55. Перцепционный шум заменяется или обработка стерео интенсивности *out_of_phase* производится согласно режиму замены. В противном случае перцепционный шум заменяется, только если шумовой флаг равен 1;

если *stereo_info* равен 3 и полоса масштабного коэффициента меньше *pns_start_sfb*, производится обработка стерео интенсивности *out_of_phase*.

6.4.4 Декодирование масштабных коэффициентов, шумовой энергии и позиции стерео интенсивности

6.4.4.1 Описание

Схема масштабируемого кодирования *BSAC* включает бесшумное кодирование, которое отличается от *AAC* и дополнительно уменьшает избыточность масштабных коэффициентов.

max_scalefactor кодируется как 8-битовое целое число без знака. Масштабные коэффициенты дифференцированно кодируются относительно значения *max_scalefactor* и затем арифметически кодируются с использованием модели дифференциального масштабного коэффициента.

6.4.4.2 Определения

6.4.4.2.1 Элементы данных

acode_scf_index [ch] [g] [sfb]	Арифметическая кодовая комбинация из кодирования дифференциальных масштабных коэффициентов.
acode_max_noise_energy [ch]	Арифметическая кодовая комбинация из кодирования максимума шумовых энергий.
acode_dpcm_noise_energy_index [ch] [g] [sfb]	Арифметическая кодовая комбинация из кодирования индекса дифференциальной шумой энергии.
acode_is_position_index [g] [sfb]	Арифметическая кодовая комбинация из кодирования индекса *poistion* стерео интенсивности.

6.4.4.2.1 Элементы справки

ch	Индекс канала.
g	Индекс группы.
sfb	Индекс полосы масштабного коэффициента в пределах группы.
layer	Индекс уровня масштабируемости.
nch	Номер канала.
layer_group[layer]	Указывает индекс группы спектральных данных, которые будут вновь добавлены в уровень масштабируемости.

layer_start_sfb [layer]	Указывает индекс самой низкой полосы масштабного коэффициента, который будет вновь добавлен в уровне масштабируемости.
layer_end_sfb [layer]	Указывает самый высокий индекс полосы масштабного коэффициента, который будет вновь добавлен в уровень масштабируемости.
scf [ch] [g] [sfb]	Указывает масштабные коэффициенты.
max _noise_energy [ch]	Указывает максимум шумовой энергии.
dpcm_noise_energy_index [ch] [g] [sfb]	Указывает дифференциальный индекс шумовой энергии.
is_position_index [g] [sfb]	Указывает индекс *poistion* стерео интенсивности.

6.4.4.3 Процесс декодирования

Спектральные коэффициенты делятся на полосы масштабного коэффициента, которые содержат кратное 4 число квантованных спектральных коэффициентов. У каждой полосы масштабного коэффициента имеется масштабный коэффициент.

Индекс дифференциального масштабного коэффициента арифметически декодируется, используя арифметическую модель данную в таблице А.32. Арифметическая модель масштабного коэффициента для базового уровня дается как 3-битовый целочисленный элемент данных без знака *base_scf_model.* Арифметическая модель масштабного коэффициента для уровней расширения дается как 3-битовый целочисленный элемент данных без знака, *enh_scf_model.*

Для всех масштабных коэффициентов различие величины смещения арифметически декодируются. Все масштабные коэффициенты вычисляются исходя из различия и величины смещения. Величина смещения дается явно как 8-битовое *PCM* в элементе данных *max_scalefactor [ch].*

Следующий псевдокод описывает, как декодировать масштабные коэффициенты *scf [ch] [g] [sfb]* в базовом уровне и каждом уровне расширения:

```
for (ch = 0; ch <nch; ch ++) {
g = layer_group [layer];
for (sfb = layer_start_sfb [layer]; sfb <layer_end_sfb [layer]; sfb ++) {
diff_scf = arithmetic_decoding ();
scf [ch] [g] [sfb] = max_scalefactor [ch] - diff_scf;
}
}
```

Если кодирование замены шума активно для определенной группы и полосы масштабного коэффициента, величина шумовой энергии передается вместо масштабного коэффициента соответствующего канала.

Шумовые энергии являются арифметическим кодированием различных значений. Для всех шумовых энергий отличие в величине смещения арифметически декодируется. Все шумовые энергии вычисляются исходя из различия и значения смещения. Значение смещения *max_noise_energy [ch]* арифметически декодируется прежде, чем декодируется первая дифференциальная шумовая энергия.

Декодирование шумовой энергии в каждом уровне определяется следующим псевдокодом:

```
for (ch =0; ch <nch; ch ++) {
g = layer_group [layer];
for (sfb = layer_start_sfb [layer]; sfb <layer_end_sfb [layer]; sfb ++) {
if (noise_flag [ch] [g] [sfb]) {
dpcm_noise_energy_index [ch] [g] [sfb] = arithmetic_decoding ();
noise_nrg [ch] [g] [sfb] = max_noise_energy [ch] - dpcm_noise_energy [ch] [g] [sfb];
}
}
}
```

Информация о направлении для декодирования стерео интенсивности представляется значением позиции стерео интенсивности, указывающим соотношение между масштабированием левого и правого каналов. Если интенсивность стерео активна для определенной группы и полосы масштабного коэффициента, значение позиции интенсивности стерео передается вместо масштабного коэффициента правого канала.

Декодирование позиции стерео интенсивности в каждом уровне определяется следующим псевдокодом:

```
g = layer_group [layer]
for (sfb = layer_start_sfb [layer]; sfb <layer_end_sfb [layer]; sfb ++) {
```

if (stereo_info [g] [sfb] && ch == 1) {
is_position_index [g] [sfb] = arithmetic_decoding ();
is (is_position_sign [g] [sfb] +1)/2)
is_position [g] [sfb] = - (int) ((is_position_index [g] [sfb] +1)/2);
else
is_position [g] [sfb] = (int) (is_position_index [g] [sfb]/2);
}
}

6.4.5 Декодирование дополнительной информации о кодировании полосы

6.4.5.1 Описание

В масштабируемой схеме кодирования *BSAC* спектральные коэффициенты делятся на полосы кодирования, которые содержат 32 квантованных спектральных коэффициента для бесшумного кодирования. Полосы кодирования являются основными единицами, используемыми для бесшумного кодирования. Набор разрядно-модульной последовательности делится на полосы кодирования. Плоскость *MCB* и таблица вероятности каждой полосы кодирования включаются в дополнительную информацию о полосе кодирования этого уровня *cband_si,* как показано в таблице А.33. Дополнительная информация о полосе кодирования каждого уровня передается, начиная с самой низкой полосы кодирования (*layer_start_cband* [*layer*]) и продолжается до самой высокой полосы кодирования (*layer_end_cband* [*layer*]). Для всех *cband_si* это арифметическое кодирование с использованием арифметической модели, как показано в таблице А.31.

6.4.5.2 Определения

Элемент данных:

acode_cband_si [ch] [g] [cband] — Арифметическая кодовая комбинация из арифметического кодирования *cband_si* для каждой полосы кодирования.

Элементы справки:

g — Индекс группы.
cband — Индекс полосы кодирования в пределах группы.
ch — Индекс канала.
nch — Номер канала.
layer_group [layer] — Указывает на индекс группы спектральных данных, которые будут вновь добавлены в уровень масштабируемости.
layer_start_cband [layer] — Указывает индекс самой низкой полосы кодирования, который будет вновь добавлен в уровень масштабируемости.
layer_end_cband [layer] — Указывает индекс самой высокой полосы кодирования, который будет вновь добавлен в уровень масштабируемости.

6.4.5.3 Процесс декодирования

cband_si арифметически кодируется, используя арифметическую модель данную в таблице А.31. Арифметическая модель используется для кодирования *cband_si* и зависит от 5-разрядного целого числа без знака в элементе данных *cband_si_type*, как показано в таблице А.31. Самое большое значение *decodable cband_si* дается в таблице А.31. Если декодируемое *cband_si* больше этого значения, можно считать, что в потоке битов была битовая ошибка.

Следующий псевдокод описывает, как декодировать *cband_si cband_si [ch] [g] [cband]* в базовом уровне и каждом уровне расширения:

g = layer_group [layer];
for (ch = 0; ch <nch; ch ++) {
for (cband = layer_start_cband [layer]; cband <layer_cband [layer]; cband ++) {
cband_si [ch] [g] [cband] = arithmetic_decoding ();
if (cband_si [ch] [g] [cband]> largest_cband_si)
bit_error_is_generated;
}
},

где *layer_cband [layer]* — стартовая полоса кодирования и *layer_cband [layer]* является конечной полосой кодирования для декодирования индекса арифметической модели в каждом уровне.

6.4.6 Сегментированное двоичное арифметическое кодирование (*SBA*)

6.4.6.1 Описание инструмента

Сегментированное двоичное арифметическое кодирование (*SBA*) основано на том факте, что арифметические кодовые комбинации могут быть разделены в известных позициях так, чтобы эти кодовые комби-

нации могли декодироваться независимо от любой ошибки в пределах других разделов. Поэтому этот инструмент исключает распространение ошибок на эти разделы. Арифметическое кодирование должно инициализироваться в начале этих сегментов и завершаться в конце этих сегментов, чтобы локализовать арифметические кодовые комбинации. Этот инструмент активируется, если элемент синтаксиса *sba_mode* равен 1. Флаг должен быть установлен в 1, если *BSAC* используется в подверженной ошибкам среде.

6.4.6.2 Определения

Определения отсутствуют, потому что добавляются только процессы инициализации и завершения в начале и в конце сегментов, чтобы локализовать арифметические кодовые комбинации.

6.4.6.3 Процесс декодирования

Арифметическое кодирование завершается в конце сегментов, и повторно инициализируется в начале следующего сегмента. Сегмент составляется из уровней масштабируемости. *terminal_layer [layer]* указывает, является ли каждый уровень последним уровнем сегмента, который устанавливается следующим образом:

for (layer = 0*; layer <(top_layer+slayer_size-1); layer ++) {*
if (layer_start_cband [layer]! = layer_start_cband [layer+1*])*
terminal_layer [layer] = 1*;*
else terminal_layer [layer] = 0*;*
}
},

где *toplayer* является верхним уровнем, который будет закодирован, *layer_max_cband []* являются максимальным пределом полосы кодирования, который будет закодирован и *slayer_size* является размером подуровня базового уровня.

В декодере полезная нагрузка потока битов каждого уровня выделяется из полной полезной нагрузки потока битов. Если предыдущий уровень является последним в сегменте, полезная нагрузка потока битов выделения сохраняется в независимом буфере и процесс арифметического декодирования повторно инициализируется. В противном случае полезная нагрузка потока битов выделения связывается с нагрузкой из предыдущего уровня и используется для арифметического декодирования последовательно.

Чтобы сделать арифметическое декодирование в полной мере, если уровень является последним из сегмента, к полезной нагрузке потока битов выделения должно быть привязано 32-разрядное нулевое значение.

6.5 Квантование чередующегося вектора

6.5.1 Описание инструмента

Этот процесс генерирует сглаженный спектр *MDCT*, используя векторное квантование. Данный инструмент квантования обеспечивает высокое усиление кодирования даже при более низких скоростях передачи. Полезная нагрузка потока битов для этого квантователя имеет простую структуру фиксированной длины, таким образом он устойчив к ошибкам канала передачи.

Процесс декодирования состоит из части векторного квантования и части реконструкции. В части векторного квантования подвекторы определяются индексом кодового вектора. Затем подвекторы чередуются и объединяются в один выходной вектор.

6.5.2 Определения

Вводы:

fb_shift [] []	Элемент синтаксиса, указывающий активную полосу частот адаптивного управления шириной полосы.
index0 []	Элемент данных, указывающий на номер кодового вектора сборника кодов 0.
index1 []	Элемент данных, указывающий на номер кодового вектора сборника кодов 1.
window_sequence	Элемент данных, указывающий тип последовательности окон.
side_info_bits	Число битов для дополнительной информации.
bitrate	Системный параметр указывающий битовую скорость.
used_bits	Число битов, используемых инструментом с переменной битовой скоростью, таким как инструмент долгосрочного прогноза.
lyr	Указывает номер уровня расширения. Номер 0 присваивается базовому уровню.

Выводы:

x_flat []	Восстановленные коэффициенты *MDCT*.

Параметры:

FRAME_SIZE	Длина фрейма.
MAXBIT	Максимальное число битов для представления индекса сборника кодов формы.
N_CH	Номера каналов.
N_DIV	Номера подвекторов.
N_SF	Номера подкадров во фрейме.
sp_cv0 [] []	Сборник кодов формы сопряженного канала 0 (Элементы даются в таблицах А.21, А.23, А.25, А.27).
sp_cv1 [] []	Сборник кодов формы сопряженного канала 1 (элементы даются в таблицах А.22, А.24, А.26, А.28).
SP_CB_SIZE	Размер сборника кодов формы.
*shape_index*0	Указывает выбранный кодовый вектор формы *MDCT* кодовой книги 0.
*shape_index*1	Указывает выбранный кодовый вектор формы *MDCT* кодовой книги 1.
*pol*0	Отрицает выбранный кодовый вектор формы *MDCT* кодовой книги 0.
*pol*1	Отрицает выбранный кодовый вектор формы *MDCT* кодовой книги 1.

6.5.3 Установки параметров

Список векторов сборника кодов формы сглаженных коэффициентов *MDCT*, *sp_cv*0 *[] []* и *sp_cv*1 *[] []* приводится в приложении А.

Параметры первоначально устанавливаются, как перечислено ниже:

MAXBIT_SHAPE = 5;

MAXBIT= MAXBIT_SHAPE + 1;

SP_CB_SIZE= 1.

6.5.4 Процесс декодирования

6.5.4.1 Инициализации

На основе *bits_available_vq*, определенного в 5.2.5.3.2, вычисляются *N_DIV* и длина каждого подвектора *length []*,

*N_DIV =((int)((bits_available_vq +MAXBIT*2-1)/(MAXBIT*2))) для idiv=0; idiv <ntt_N_DIV; idiv ++*

*length [idiv] = ((int) (N_FR*qsample) *N_SF*N_CH+N_DIV-1-idiv) / N_DIV,*

где *N_FR* является числом выборок в подкадре, и *qsample* определяется в 6.5.4.4.

6.5.4.2 Распаковка индекса

Индекс квантования состоит из информации о коде полярности и формы. Так, в первой стадии инверсного квантования входные индексы распаковываются и извлекаются полярности и формы.

Извлечение полярностей описывается следующим образом:

for (idiv = 0; idiv <N_DIV; idiv ++) {

*pol*0 *[idiv] = 2 * (index*0 *[idiv] / SP_CB_SIZE) - 1;*

*pol*1 *[idiv] = 2 * (index*1 *[idiv] / SP_CB_SIZE) - 1;*

},

где:

*pol*0 *[]* — полярность сопряженного канала 0;

*pol*1 *[]* — полярность сопряженного канала 1.

Извлечение кода формы описывается следующим образом:

for (idiv = 0; idiv <N_DIV; idiv ++) {

*index_shape*0 *[idiv] = index*0 *[idiv] % SP_CB_SIZE;*

*index_shape*1 *[idiv] = index*1 *[idiv] % SP_CB_SIZE;*

}.

6.5.4.3 Реконструкции

Выходные коэффициенты восстанавливаются следующим образом:

for (idiv = 0; idiv <N_DIV; idiv ++) {

for (icv = 0; icv <длина [idiv]; icv ++) {

if ((icv <длина [0]-1) &&

*((% N_DIV (N_SF*N_CH) == 0 && (N_SF*N_CH)> 1) || ((N_SF*N_CH) &0x1) == 0)))*

*itmp = ((idiv+icv) %N_DIV) +icv*N_DIV;*

else

*itmp = idiv + icv * N_DIV;*

*ismp = itmp / (N_SF*N_CH) + ((itmp % (N_SF*N_CH)) * (FRAME_SIZE / (N_SF*N_CH)));*

*x_flat_tmp [ismp] = (pol0 [idiv] *sp_cv0 [index_shape0 [idiv]] [icv] + pol1 [idiv] *sp_cv1 [index_shape1 [idiv]] [icv]) / 2;*

}

}

где:

icv — указывает число выборок в векторе кода формы;

idiv — указывает подвектор чередующегося подразделения;

ismp — указывает число выборок в подфрейме;

itmp — целое число.

6.5.4.4 Адаптивный выбор активной полосы

Эта процедура выбирает активную полосу в зависимости от параметра *fb_shift*.

Если *lyr*=0, активная полоса фиксируется, как перечислено ниже:

*bandUpper_i = 95 * BPS / ISAMPF;*

bandUpper_i = min (100000, bandUpper_i);

*bandUpper_i * = 16384;*

bandUpper_i + = 1562;

bandUpper_i / = 3125;

qsample = (double) (bandupper_i)/524288;

AC_TOP [lyr] [i_ch] [0] = qsample;

AC_BTM [lyr] [i_ch] [0] = 0,0,

где *ISAMP* является целочисленной частотой дискретизации полученной усечением значений стандартной частоты, перечисленных в правой графе таблицы 82 и *BPS* является битовой скоростью основанной на байт-синхронизирующих битах для фрейма и это равняется:

*(int) (((FRAME_SIZE * bitrate/sampling_frequency)/8+0,5) *8) *sampling_frequency/FRAME_SIZE/N_CH.*

Если *lyr* больше или равно '1', верхний предел ширины полосы квантования определенного уровня определяется следующим образом:

totalbps=bpsbase;

for (iscl0 = 1; iscl0 <= lyr; iscl0 ++) {

totalbps + = BPS_SCL [iscl0];

}

upperlimit_i = (totalbps 100) / ISAMPF;*

upperlimit_i = min (100000, upperlimit_i);

upperlimit_i = 16384;*

upperlimit_i + = 1562;

upperlimit_i / = 3125;

upperlimit = (double) (upperlimit_i)/524288.

UPPER_BOUNDARY и *LOWER_BOUNDARY* ширины полосы квантования, в зависимости от выбора полосы, код и значение битовой скорости на канал *BPS_SCL [lyr]* для каждого уровня расширения определяются следующим образом:

BPS_SCL [lyr] должно быть основано на байт-синхронизирующих битах для фрейма, и оно равняется:

*(int) (((FRAME_SIZE * bitrate/sampling_frequency)/8+0,5) *8) *sampling_frequency/FRAME_SIZE/N_CH;*

*qsample_i = (BPS_SCL [lyr] * 130) / ISAMPF;*

qsample_i = min (100000, qsample_i);

*qsample_i * = 16384;*

qsample_i + = 1562;

qsample_i / = 3125;

bias_i = (upperlimit_i-qsample_i)/4;

if (qsample_i <bias_i) {

bias_i = upperlimit_i/4;

qsample_i = bias_i;

};

bias = (double) bias_i;

qsample = (double) (qsample_i)/524288./ 16384*32 */,*

if (bias <= 0.0) bias = 0,0;

```
for (i_ch = 0; i_ch <N_CH; i_ch ++) {
AC_TOP [lyr] [i_ch] [0] = qsample;
AC_BTM [lyr] [i_ch] [0] = 0,0;
AC_TOP [lyr] [i_ch] [1] = qsample+bias;
AC_BTM [lyr] [i_ch] [1] = bias;
AC_TOP [lyr] [i_ch] [2] = qsample + bias*2;
AC_BTM [lyr] [i_ch] [2] = bias*2;
AC_TOP [lyr] [i_ch] [3] = qsample + bias*3;
AC_BTM [lyr] [i_ch] [3] = bias*3;
},
```

где *AC_BTM* и *AC_TOP* являются нижней и верхней частотой активной полосы, соответственно. Значения колеблются в диапазоне от 0 до 1,0.

Более низкие и верхние границы в домене *MDCT* вычисляются следующим образом:

```
for (i_ch = 0; i_ch <N_CH; i_ch ++) {
LO WER_BOUNDARY [lyr] [i_ch] = AC_BTM [lyr] [i_ch] [fb_shft] * N_FR;
UPPER_BOUNDARY [lyr] [i_ch] = AC_TOP [lyr] [i_ch] [fb_shift] * N_FR;
}.
```

Затем вывод *x_flat []* копируется из *x_flat_tmp []* следующим образом:

```
for (i_ch = 0; i_ch <N_CH; i_ch ++) {
for (isf = 0; isf <N_SF; isf ++) {
for (ismp = 0; ismp <LOWER_BOUNDARY [lyr] [i_ch]; ismp ++) {
x_flat [ismp + (isf+i_ch*N_SF) *N_FR] = 0;
}
for (ismp = LOWER_BOUNDARY [lyr] [i_ch]; ismp <UPPER_BOUNDARY [lyr] [i_ch]; ismp ++) {
ismp2 = ismp - LOWER_BOUNDARY [lyr] [i_ch];
x_flat [ismp + (isf+i_ch*N_SF) *N_FR] = x_flat_tmp [ismp2 + (isf+i_ch*N_SF) *N_FR];
}
for (ismp = UPPER_BOUNDARY [lyr] [i_ch]; ismp <N_FR; ismp ++) {
x_flat [ismp + (isf+i_ch*N_SF) *N_FR] = 0.;
}
}
}.
```

6.6 Прогноз частотной области

Использование инструмента прогноза является зависимым от типа объекта/профиля. Инструмент прогноза частотной области может использоваться только для *AudioObjectType* 1 (*AAC Main*).

6.7 Долгосрочный прогноз (*LTP*)

6.7.1 Описания инструмента

Долгосрочный прогноз (*LTP*) является эффективным инструментом для уменьшения избыточности сигнала между последовательными фреймами кодирования. Этот инструмент особенно эффективен для частей сигнала, у которых имеется четкое свойство шага. Сложность реализации *LTP* значительно ниже, чем сложность обратного адаптивного прогноза. Поскольку долгосрочное прогнозирующее устройство является прямым адаптивным прогнозирующим устройством (коэффициенты прогноза отправляются как дополнительная информация), оно по сути менее чувствительно к числовым ошибкам округления в декодере или битовым ошибкам в переданных спектральных коэффициентах.

LTP ограничивается только длинными окнами.

6.7.2 Определения

ltp_data_present	1 бит, указывающий, используется ли прогноз в текущем фрейме (1) или нет (0) (всегда существует).
ltp_lag	11-разрядное число, определяющее оптимальную задержку от 0 до 2047.
ltp_coef	3-разрядный индекс, указывающий коэффициент *LTP* в таблице 152.

Таблица 152 — коэффициент *LTP*

Значение *ltp_coef*	Значение коэффициента *LTP*
000	0,570829
001	0,696616
010	0,813004
011	0,911304
100	0,984900
101	1,067894
110	1,194601
111	1,369533

ltp_long_used 1 бит для каждой полосы масштабного коэффициента (*sfb*), где *LTP* может использоваться для указания включен ли *LTP* (1) в этом *sfb* или выключен (0).

Для аудиообъектного типа *ER AAC LD* доступны следующие элементы данных:

ltp_lag_update 1 бит, указывающий, передается ли новое значение для задержки *ltp*. В силу высокой повторяемости запаздывания *LTP* для многих сигналов один дополнительный бит сигнализирует, что запаздывание предыдущего фрейма повторяется (*ltp_lag_update* == 0). В ином случае передается новое значение для запаздывания *ltp* (*ltp_lag_update* == 1).

ltp_lag 10-разрядное число, определяющее оптимальную задержку от 0 до 1023.

Максимальное количество полос масштабного коэффициента, используемых в прогнозе, ограничивается константой:

MAX_LTP_LONG_SFB = 40 (для длинных фреймов).

6.7.3 Процесс декодирования

Процесс декодирования для *LTP* выполняется в каждом окне текущего фрейма путем применения фильтрации 1-*tap IIR* во временном интервале, чтобы предсказать выборки в текущем фрейме по выборкам в предыдущих фреймах. Процессом управляет переданная дополнительная информация методом двух шагов. Первый шаг управления определяет, используется ли *LTP* вообще для текущего фрейма. В случае длинного окна второй шаг управления определяет, на каких полосах масштабного коэффициента используется *LTP*. В начале процесса декодирования восстановленные выборки временного интервала инициализируются нулями.

Для каждого фрейма дополнительная информация *LTP* извлекается из полезной нагрузки потока битов, чтобы управлять дальнейшей работой прогнозирующего устройства в декодере. В случае *single_channel_element* () управляющая информация допустима для канала с этим элементом. В случае *channel_pair_element* () имеются два набора управляющей информации. Порядок появления наборов управляющей информации *LTP* в случае *common_window* == 1 таков, что дополнительная информация *LTP* сначала извлекается для левого канала, а затем для правого канала соответствующего *channel_pair_element* ().

Вначале читается бит *ltp_data_present*. Если этот бит не установлен в (0), тогда *LTP* выключается для текущего фрейма и какая-либо дальнейшая дополнительная информация прогнозирующего устройства отсутствует. В этом случае флаг *ltp_long_used* для каждой полосы масштабного коэффициента, сохраняемый в декодере, должен быть обнулен. Если бит *ltp_data_present* устанавливается в (1), тогда *LTP* используется для текущего фрейма и параметры *LTP* читаются.

Для длинных окон параметры *LTP* используются, чтобы вычислить предсказанные сигналы временного интервала, используя следующую формулу:

*x_est (i) = ltp_coef*x_rec (i - M. - ltp_lag)*

i = 0, ..., *N*-1,

где: *x_est (i)* являются предсказанными выборками;

x_rec (i) являются восстановленными выборками временного интервала

N — длина окна преобразования;

M = *N/2*, если *aot* == *ER AAC LD*, иначе *M* = 0.

Другое значение для *M*, используемое в комбинации с кодеком низкой задержки, служит для достижения подобного диапазона возможных значений запаздывания в абсолютное время, несмотря на более короткий фрейм.

Контрольная точка для индекса *i* и контента буфера *x_rec* располагается так, чтобы *x_rec* (0... *N*/2 - 1) содержал последнюю искаженную половину окна из *IMDCT* (многооконная работа с перекрывающей

добавкой также применяемая), и *x_rec* (*N*/2... *N*-1) всегда нули. Остальная часть *x_rec* (*i* <0) содержит предыдущие полностью восстановленные выборки временного интервала, то есть, вывод декодера.

Используя *MDCT* для длинных окон, получаются предсказанные спектральные компоненты для текущего фрейма из предсказанного сигнала временного интервала. Затем читаются биты *ltp_long_used* из полезной нагрузки потока битов, которые управляют использованием прогноза в каждой полосе масштабного коэффициента индивидуально, то есть, если бит устанавливается для определенной полосы масштабного коэффициента, все предсказанные спектральные компоненты этой полосы масштабного коэффициента используются. Иначе предсказанные спектральные компоненты устанавливаются в нули. Таким образом, если *ltp_long_used* бит устанавливается, то квантованная ошибка прогноза, восстановленная из переданных данных, добавляется к предсказанному спектральному компоненту. Если бит не устанавливается (0), то квантованное значение спектрального компонента восстанавливается непосредственно из переданных данных.

Часть реконструкции сигнала процесса декодирования для одного канала может быть описана как следующий псевдокод. Здесь *x_est* является предсказанным сигналом временного интервала, *X_est* является соответствующим вектором частотной области, *Y_rec* является вектором декодируемых спектральных коэффициентов и *X_rec* является вектором восстановленных спектральных коэффициентов.

```
if (ONLY_LONG_SEQUENCE || LONG_START_SEQUENCE || LONG_STOP_SEQUENCE) {
x_est = predict ();
X_est = MDCT (x_est);
for (sfb = 0; sfb <min (max_sfb, MAX_LTP_LONG_SFB); sfb ++) {
if (ltp_data_present && ltp_long_used [sfb])
X_rec = X_est + Y_rec;
else
X_rec = Y_rec;
}
}.
```

6.7.4 Интеграция *LTP* с другими инструментами *GA*

6.7.4.1 *LTP* с *TNS*

Поскольку *TNS* должен быть применен к восстановленному спектру, фильтрация *TNS* должна произойти после *LTP* в цепочке декодирования. Это делает необходимым дополнительный аналитический фильтр *TNS* в цикле *LTP*.

6.7.4.2 *LTP* с *PNS*

Одновременное использование *LTP* и *PNS* не предотвращается в синтаксисе. Если и *LTP*, и *PNS* задействуются на той же самой полосе масштабного коэффициента, *PNS* имеет приоритет и никакой прогноз не применяется к этой полосе.

6.7.4.3 *LTP* с зависимо коммутируемой связью

Никакая зависимо коммутируемая связь и следовательно никакой зависимо коммутируемый *CCE* не разрешаются ни в каком аудиообъектном типе, который использует *LTP*.

6.7.5 *LTP* в масштабируемом декодере *GA*

В масштабируемом кодере *LTP* используется только на самом низком уровне кодирования *GA*, и обновление прогнозирующего устройства основывается на выводе временного интервала первого уровня *GA*. Могут использоваться или *LTP*, или базовый кодер, но не оба одновременно.

Процесс декодирования *LTP* подобен процессу, используемому, когда самый низкий уровень является базовым кодером. Сама часть *LTP* декодируется точно таким же образом, как в немасштабируемом декодере *GA*.

В масштабируемой конфигурации одновременное использование *LTP* и *Intensity Stereo* (стерео интенсивности) допускается в синтаксисе. Однако, если и *LTP*, и *Intensity Stereo* задействованы в той же самой полосе масштабного коэффициента в первом уровне *GA*, *Intensity Stereo* имеет приоритет и никакой прогноз не применяется к этой полосе.

6.8 Объединенное кодирование

6.8.1 Стерео *M/C*

6.8.1.1 Описание инструмента

Объединенное кодирование каналов *M/S* действует на парах каналов. Каналы чаще всего спариваются так, что у них есть симметричное представление относительно слушателя, такое как левый/правый или левое окружение/правое окружение. Первый канал в паре обозначается как "левый", а второй как

"правый". На основе спектрального коэффициента вектор, сформированный сигналами левого и правого каналов, восстанавливается или де-матрицируется либо матрицей идентичности

$$\begin{bmatrix} l \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} l \\ r \end{bmatrix},$$

или инверсной матрицей M/S

$$\begin{bmatrix} l \\ r \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} m \\ s \end{bmatrix}.$$

Решение, какую матрицу использовать, принимается по полосе масштабного коэффициента на основе полосы масштабного коэффициента, как обозначено флагами *ms_used*. Для аудиообъектных типов *AAC Main, LC, SSR*, и *LTP* кодирование соединения каналов *M/S* может использоваться только, если *common_window* равен '1'. Для масштабируемого аудио объектного типа *AAC* всегда используются общие окна между этими двумя звуковыми каналами, так что кодирование *M/S* всегда бывает возможно.

6.8.1.2 Определения

ms_mask_present Это двухбитовое поле указывает, что маска *MC* имеет вид:
00 — все нули;
01 — маска *max_sfb* полос *ms_used* следует за этим полем;
10 — все единицы;
11 — зарезервировано.

ms_used [g] [sfb] Однобитовый флаг на полосу масштабного коэффициента, указывающий, что кодирование *M/S* используется в группе окон *g* и полосе масштабного коэффициента *sfb*.

l_spec [] Массив, содержащий спектр левого канала соответствующей пары каналов.

r_spec [] Массив, содержащий спектр правого канала соответствующей пары каналов.

is_intensity (g, sfb) Функция, возвращающая состояние интенсивности.

is_noise (g, sfb) Функция, возвращающая состояние шумовой замены.

6.8.1.3 Процесс декодирования

Восстанавливает спектральные коэффициенты первого ("левого") и второго ("правого") каналов, как определено флагами *ms_mask_present* и *ms_used [] []* следующим образом:

```
if (ms_mask_present> = 1) {
for (g = 0; g <num_window_groups; g ++) {
for  (b = 0; b <window_group_length [g]; b ++) {
for (sfb = 0; sfb <max_sfb; sfb ++) {
if ((ms_used [g] [sfb] || ms_mask_present == 2) &&
! is_intensity (g, sfb) &&! is_noise (g, sfb)) {
for (i= 0; i <swb_offset [sfb+1]-swb_offset [sfb]; i ++) {
tmp = l_spec [g] [b] [sfb] [i] - r_spec [g] [b] [sfb] [i];
l_spec [g] [b] [sfb] [i] = l_spec [g] [b] [sfb] [i] +
r_spec [g] [b] [sfb] [i];
r_spec [g] [b] [sfb] [i] = tmp;}
}
}
}
}
}.
```

ms_used [] [] также используется в контексте кодирования стерео интенсивности и перцепционной шумовой замены. Если кодирование стерео интенсивности или шумовая замена задействованы для определенной полосы масштабного коэффициента, никакое декодирование стерео *M/S* не выполняется.

6.8.1.4 Интеграция инструмента стерео *M/S* для аудиообъектного типа масштабируемого *AAC*

Та же самая маска *MC* применяется ко всем уровням. Если последующие уровни определяют увеличение *max_sfb*, *ms_mask_present* и *ms_used [] []* передается только для дополнительных полос масштабного коэффициента и групп.

6.8.2 Стерео интенсивности (*IS*)

6.8.2.1 Описание инструмента

Этот инструмент используется для реализации кодирующий объединенной стерео интенсивности между обоими каналами пары каналов. Таким образом, выводы обоих каналов получаются из единственного набора спектральных коэффициентов после процесса инверсного квантования. Это делается выборочно на основе полосы масштабного коэффициента, когда стерео интенсивности отмечается как активное.

6.8.2.2 Определения

hcod_sf [] Кодовая комбинация Хаффмана из таблицы кода Хаффмана, используемая для кодирования масштабных коэффициентов.

dpcm_is_position [] [] Дифференцированно закодированная позиция стерео интенсивности.

Is_position[group] [sfb] Позиция стерео интенсивности для каждой группы и полосы масштабного коэффициента.

l_spec [] Массив, содержащий спектр левого канала соответствующей пары каналов.

r_spec [] Массив, содержащий спектр правого канала соответствующей пары каналов.

6.8.2.3 Процесс декодирования

Об использовании кодирования стерео интенсивности сообщается при помощи сборников псевдокодов *INTENSITY_HCB* и *INTENSITY_HCB2* (15 и 14) только в правом канале элемента *channel_pair_element ()*, имея общее *ics_info ()* (*common_window* == 1). *INTENSITY_HCB* и *INTENSITY_HCB2* сигнализируют о синфазном и с несовпадением по фазе кодировании интенсивности стерео, соответственно.

В случае немасштабируемого декодера *GA* фазовое соотношение кодирования стерео интенсивности может быть инвертировано посредством поля *ms_used*. Поскольку кодирование стерео *M/S* и кодирование стерео интенсивности являются взаимоисключающими для определенной полосы масштабного коэффициента и группы, первичное фазовое соотношение, задаваемое таблицами кода Хаффмана, изменяется от синфазного до несовпадающего по фазе или наоборот, если соответствующий бит *ms_used* устанавливается для соответствующей полосы.

Направленная информация для декодирования стерео интенсивности представляется значением "позиция стерео интенсивности", указывающим соотношение между масштабированием левого и правого каналов. Если кодирование стерео интенсивности является активным для определенной группы и полосы масштабного коэффициента, значение позиции стерео интенсивности передается вместо масштабного коэффициента правого канала. Позиции интенсивности кодируются точно так же как масштабные коэффициенты, то есть кодированием дифференциальных значений методом Хаффмана с двумя различиями:

- нет никакого первого значения, которое отправляется как *PCM*. Вместо этого запускается дифференциальное декодирование, принимая последнее значение позиции стерео интенсивности равным нулю;

- дифференциальное декодирование выполняется отдельно между масштабными коэффициентами и позициями стерео интенсивности. Декодер масштабного коэффициента игнорирует вставленные значения позиции стерео интенсивности и наоборот.

Тот же самый сборник кодов используется для того, чтобы кодировать позиции стерео интенсивности, как и для масштабных коэффициентов.

Для использования в декодировании стерео интенсивности определяются две псевдофункции:

```
function is_intensity(group,sfb) {
+1   for window groups / scalefactor bands with right channel
codebook sfb_cb[group][sfb] == INTENSITY_HCB
-1   for window groups / scalefactor bands with right channel
codebook sfb_cb[group][sfb] == INTENSITY_HCB2
0        otherwise
}
function invert_intensity(group,sfb) {
1-2*ms_used[group][sfb]     if (ms_mask_present == 1) && aot != AAC scalable
+1                                  otherwise
}.
```

Декодирование стерео интенсивности для одной пары каналов определяется следующим псевдокодом:

```
p = 0;
for (g = 0; g < num_window_groups; g++) {
/* Decode intensity positions for this group */
for (sfb = 0; sfb < max_sfb; sfb++)
```

```
if (is_intensity(g,sfb))
is_position[g][sfb] = p += dpcm_is_position[g][sfb];
/* Do intensity stereo decoding */
for (b = 0; b < window_group_length[g]; b++) {
for (sfb = 0; sfb < max_sfb; sfb++) {
if (is_intensity(g,sfb)) {
scale = is_intensity(g,sfb) * invert_intensity(g,sfb) *
0,5^(0,25*is_position[g][sfb]);
/* Scale from left to right channel,
do not touch left channel */
for (i = 0; i < swb_offset[sfb+1]-swb_offset[sfb]; i++)
r_spec[g][b][sfb][i] = scale * l_spec[g][b][sfb][i];
}
}
}
}.
```

В случае кодирования длины реверсивной переменной (*RVLC*) нет никакого последнего значения, которое передается как *PCM*. Вместо этого запускается обратное дифференциальное декодирование, предполагая, что последнее значение позиции стерео интенсивности — нулевое. Декодирование позиций стерео интенсивности определяется следующим псевдокодом:

```
p = dpcm_is_last_position;
for (g = win-1; g> = 0; g- -) {
for (sfb = sfbmax-1; sfb> = 0; sfb- -) {
is_pos [g] [sfb] =p;
p - = dpcm_is_pos [g] [sfb];
}
}.
```

6.8.2.4 Интеграция с инструментом внутриканального прогноза

Для полос масштабного коэффициента, кодированных в стерео интенсивности, соответствующие прогнозирующие устройства в правом канале выключаются, таким образом эффективно переопределяя состояния, назначенные маской *prediction_used*. Обновление этих прогнозирующих устройств производится, подавая декодируемые спектральные значения стерео интенсивности правого канала как последнее квантованное значение x_{rec} (n-1). Эти значения следуют из процесса масштабирования из левого канала в правый, как описано в псевдокоде.

6.8.2.5 Интеграция с инструментом долгосрочного прогноза

В случае немасштабируемой конфигурации функция долгосрочного прогноза не зависит от стерео интенсивности. Инструмент *LTP* должен быть применен к обоим каналам элемента пары каналов после того, как было выполнено кодирование стерео интенсивности.

6.8.2.6 Интеграция инструмента интенсивности для аудио объектного типа масштабируемого *AAC*

Если определенная полоса масштабного коэффициента и группа кодируются стерео интенсивностью, то этот вклад в спектральные компоненты выходного сигнала опускается, если спектральные коэффициенты передаются для этой полосы масштабного коэффициента и группы в каком-либо из более высоких уровней (чему способствует выходной сигнал) посредством номера сборника кодов неинтенсивности.

Если определенная *scalefactor* полоса и группа кодируются стерео интенсивностью и если та же самая полоса масштабного коэффициента в следующем уровне расширения также кодируется стерео интенсивностью, объединяются только спектральные компоненты левого канала. Спектральные компоненты правого канала повторно вычисляются для уровня расширения, используя фактор интенсивности этого уровня расширения.

В масштабируемой конфигурации одновременное использование стерео интенсивности и *LTP* не препятствует в синтаксисе. Однако, если и стерео интенсивности, и *LTP* задействованы в той же самой полосе масштабного коэффициента в первом уровне *GA*, стерео интенсивности имеет приоритет и никакой прогноз не применяется к этой полосе.

В случае масштабируемой конфигурации *AAC* поле *ms_used* игнорируется в декодируемых полосах масштабного коэффициента стерео интенсивности, но может все еще сигнализировать об использовании декодирующего стерео *M/S* в более высоких уровнях (расширения).

6.8.3 Спаривание канала

(Аналогично ГОСТ Р 54713—2011)

6.8.3.1 Описание инструмента

coupling_channel element()'s обеспечивает две функциональности: во-первых, может использоваться спаривание каналов, чтобы реализовать обобщенное кодирование стерео интенсивности, где спектры канала могут быть совместно использованы через границы канала, во-вторых, спаривание каналов может использоваться, чтобы динамически выполнить включение одного звукового объекта в образ стерео.

Этот инструмент включает параметры, зависимые от определенного типа объекта.

6.8.3.2 Определения

ind_sw_cce_flag	Один бит, указывающий, является ли связанный целевой элемент синтаксиса независимо коммутируемым (1) или зависимо коммутируемым (0).
num_coupled_elements	Число связанных целевых каналов равно *num_coupled_elements*+1.
cc_target_is_cpe	Один бит, указывающий, является ли связанный целевой элемент синтаксиса *CPE* (1) или *SCE* (0).
cc_target_tag_select	Четырехбитовое поле, определяющее *element_instance_tag* связанного целевого элемента синтаксиса.
cc_l	Один бит, указывающий, что список значений *gain_element* применяется к левому каналу пары каналов.
cc_r	Один бит, указывающий, что список значений *gain_element* применяется к правому каналу пары каналов.
cc_domain	Один бит, указывающий, выполняется ли связь до (0) или после (1) декодирования *TNS* связанных целевых каналов.
gain_element_sign	Один бит, указывающий, содержат ли переданные значения *gain_element* информацию о синфазной / несовпадающей по фазе связи (1) или нет (0).
gain_element_scale	Определяет амплитудное разрешение *cc_scale* операции масштабирования.
common_gain_element_present [c]	Один бит, указывающий, передаются ли кодированные по Хаффману значения *common_gain_element* (1) или передаются кодированные по Хаффману значения дифференциального *gain_elements* (0).
dpcm_gain_element [] []	Дифференцированно кодированный элемент усиления.
gain_element [group] [sfb]	Элемент усиления для каждой группы и полосы масштабного коэффициента.
common_gain_element []	Элемент усиления, который используется для всех групп окон и полос масштабного коэффициента одного связанного целевого канала.
spectrum_m (idx, domain)	Указатель на спектральные данные, связанные с *single_channel_element* () с индексом *idx*. В зависимости от значения домена указываются спектральные коэффициенты до (0) или после (1) декодирования *TNS*.
spectrum_l (idx, domain)	Указатель на спектральные данные, связанные с левым каналом *channel_pair_element ()* с индексом *idx*. В зависимости от значения домена указываются спектральные коэффициенты до (0) или после (1) декодирования *TNS*.
spectrum_r (idx, domain)	Указатель на спектральные данные, связанных с правым каналом *channel_pair_element ()* с индексом *idx*. В зависимости от значения домена указываются спектральные коэффициенты декодирования *TNS* до (0) или после (1).

6.8.3.3 Процесс декодирования

Связывающийся канал базируется на встроенном *single_channel_element ()*, который объединяется с некоторыми выделенными полями, чтобы разместить его особое назначение.

Связанные целевые элементы синтаксиса (*SCEs* или *CPEs*) адресуются, используя два элемента синтаксиса. Во-первых, поле *cc_target_is_cpe* выбирает, адресуются ли *SCE* или *CPE*. Во-вторых, поле *cc_target_tag_select* выбирает *instance_tag* для *SCE/CPE*.

Операция масштабирования, включенная в связывание каналов, определяется значениями *gain_element*, которые описывают применимый коэффициент усиления и знак. В соответствии с процедура-

ми кодирования для масштабных коэффициентов и позиций стерео интенсивности, значения *gain_element* дифференцированно кодируются, используя таблицу Хаффмана для масштабных коэффициентов. Точно так же декодируемые коэффициенты усиления для связывания относятся к группам окон спектральных коэффициентов.

Независимо коммутируемые *CCEs* по сравнению с зависимо коммутируемыми *CCEs*.

Существуют два вида *CCEs*. Это независимо коммутируемые, и зависимо коммутируемые *CCEs*. Независимо коммутируемый *CCE* является *CCE*, в котором состояние окна (то есть *window_sequence* и *window_shape*) *CCE* не должно соответствовать состоянию окна любого канала *SCE* или *CPE*, с которым связывается *CCE* (целевые каналы). У этого варианта есть несколько важных последствий:

- во-первых, требуется, чтобы независимо коммутируемый *CCE* использовал только элемент *common_gain*, но не список из *gain_elements*;

- во-вторых, независимо коммутируемый *CCE* должен декодироваться полностью во временной домен (то есть включая гребенку фильтров синтеза) прежде, чем он будет масштабироваться и добавляться в различные каналы *SCE* и *CPE*, с которыми он связывается в случае, когда состояние окна не соответствует.

У зависимо коммутируемого *CCE*, с другой стороны, должно быть состояние окна, которое соответствует всем каналам целевого *SCE* и *CPE*, с которыми он связывается как определено списком элементов *cc_l и cc_r*. В этом случае *CCE* должен только декодироваться в частотную область и затем масштабироваться, как назначено списком усиления прежде, чем он будет добавлен к целевым каналам *SCE* или *CPE*.

Следующий псевдокод в функции *decode_coupling_channel ()* определяет работу декодирования для зависимого коммутируемого элемента канала связывания. Сначала спектральные коэффициенты встраиваемого *single_channel_element ()* декодируются во внутренний буфер. Так как элементы усиления для первой связываемой цели (*list_index* == 0) не передаются, все значения *gain_element*, связанные с этой целью, предполагаются равными 0, то есть связывающийся канал добавляется к связываемому целевому каналу в его естественном масштабировании. Иначе спектральные коэффициенты масштабируются и добавляются к коэффициентам связываемых целевых каналов, используя соответствующий список значений *gain_element*.

Независимо коммутируемый *CCE* декодируется, как и зависимо коммутируемый *CCE*, имеющий только *common_gain_elements*. Однако получающийся масштабируемый спектр преобразовывается обратно в его временное представление и затем связывается во временном домене.

Списки *gain_element* могут быть совместно использованы левым и правым каналами элемента пары целевых каналов. Об этом сообщается как *cc_l*, так и *cc_r*, являющимися нулем, как обозначено в таблице ниже:

Таблица 153 — Списки совместно используемых *gain_element*

cc_l, cc_r	Совместно используемый список усиления присутствует	Левый список усиления присутствует	Правый список усиления присутствует
0, 0	Да	Нет	Нет
0, 1	Нет	Нет	Да
1, 0	Нет	Да	Нет
1, 1	Нет	Да	Да

```
decode_coupling_channel()
{
/*
first:
decode spectral coefficients of embedded single_channel_element
into buffer «cc_spectrum[]»
(no pseudo code is given for this task)
second:
Couple spectral coefficients onto target channels
(according to the following pseudo code)
*/
```

```
list_index = 0;
for (c = 0; c < num_coupled_elements+1; c++)  {
if (!cc_target_is_cpe[c])  {
couple_channel( cc_spectrum,
spectrum_m( cc_target_tag_select[c], cc_domain ),
list_index++ );
}
if (cc_target_is_cpe[c])  {
if (!cc_l[c]  &&  !cc_r[c])  {
couple_channel( cc_spectrum,
spectrum_l( cc_target_tag_select[c], cc_domain ),
list_index );
couple_channel( cc_spectrum,
spectrum_r( cc_target_tag_select[c], cc_domain ),
list_index++ );
}
if (cc_l[c])  {
couple_channel( cc_spectrum,
spectrum_l( cc_target_tag_select[c], cc_domain ),
list_index++ );
}
if (cc_r[c])  {
couple_channel( cc_spectrum,
spectrum_r( cc_target_tag_select[c], cc_domain ),
list_index++ );
}
}
}
}
couple_channel(source_spectrum[], dest_spectrum[], gain_list_index)
{
idx = gain_list_index;
a = 0;
cc_scale = cc_scale_table[gain_element_scale];
for (g = 0; g < num_window_groups; g++) {
/* Decode coupling gain elements for this group */
if (common_gain_element_present[idx]) {
for (sfb = 0; sfb < max_sfb; sfb++) {
gain_element[idx][g][sfb] = common_gain_element[idx];
}
}
else {
for (sfb = 0; sfb < max_sfb; sfb++) {
if (sfb_cb[g][sfb] != ZERO_HCB)
gain_element[idx][g][sfb] =
a += dpcm_gain_element[idx][g][sfb];
}
}
/* Do coupling onto target channels */
for (b = 0; b < window_group_length[b]; b++) {
for (sfb = 0; sfb < max_sfb; sfb++) {
if (gain_element_sign) {
cc_sign = 1 - 2*(gain_element[idx][g][sfb] & 0x1);
gain = gain_element[idx][g][sfb] >> 1;
```

```
}
else {
cc_sign = 1;
gain = gain_element[idx][g][sfb]; }
cc_gain = cc_sign * cc_scale^gain;
for (i = 0; i<swb_offset[sfb+1]-swb_offset[sfb]; i++)
dest_spectrum[g][b][sfb][i] +=
cc_gain * source_spectrum[g][b][sfb][i];
}
}
}
}
}
/* Do coupling onto target channels */
for (b = 0; b < window_group_length[b]; b++) {
for (sfb = 0; sfb < max_sfb; sfb++) {
if (sfb_cb[g][sfb] != ZERO_HCB) {
cc_gain[idx][g][sfb] =
cc_sign[idx][g][sfb] * cc_scale^gain_element[idx][g][sfb];
for (i = 0; i < swb_offset[sfb+1]-swb_offset[sfb]; i++)
dest_spectrum[g][b][sfb][i] +=
cc_gain[idx][g][sfb] * source_spectrum[g][b][sfb][i];
}
}
}
}
}.
```

Примечание — Массив *sfb_cb* представляет отношение данных сборника кодов к встроенному *single_channel_element () CCE* (не связанный целевой канал).

6.8.3.4 Таблица

Таблица 154 — Разрешение масштабирования для связывания канала (*cc_scale_table*)

Значение "*gain_element_scale*"	Амплитудное разрешение "*cc_scale*"	Размер шага [дБ]
0	2^(1/8)	0,75
1	2^(1/4)	1,50
2	2^(1/2)	3,00
3	2^1	6,00

6.9 Временное формирование шума (*TNS*)

(Аналогично ГОСТ Р 54713—2011)

6.9.1 Описание инструмента

Временное формирование шума используется, чтобы управлять временной формой шума квантования в пределах каждого окна преобразования. Это делается, применяя процесс фильтрации к частям спектральных данных каждого канала.

6.9.2 Определения

n_filt [w] — Число фильтров формирования шума, используемое для окна *w*.

coef_res [w] — Маркер, указывающий на разрешение переданных коэффициентов фильтра для окна *w*, переключающееся между разрешением 3 бита (0) и 4 бита (1).

length [w] [filt] — Длина области, к которой применяется один фильтр в окне *w* (в единицах полос масштабного коэффициента).

order [w] [filt] — Порядок одного фильтра формирования шума примененного к окну *w*.

direction [w] [filt] Один бит, указывающий, применяется ли фильтр в восходящем (0) или нисходящем (1) направлении.

coef_compress [w] [filt] Один бит, указывающий, опускаются ли при передаче (1) или нет (0) старшие значащие биты коэффициентов фильтра формирования шума в окне *w*.

coef [w] [filt] [i] Коэффициенты одного фильтра формирования шума применяемые к окну *w*.

spec [w] [k] Массив, содержащий спектр для окна *w* обрабатываемого канала.

В зависимости от *window_sequence* размер следующих элементов данных переключается для каждого окна преобразовывания согласно размеру этого окна:

Таблица 155 — Размер элементов данных

Имя	Окно с 128 спектральными линиями	Другой размер окна
'n filt'	1	2
Длина	4	6
Порядок	3	5

6.9.3 Процесс декодирования

Процесс декодирования для временного формирования шума выполняется отдельно в каждом окне текущего фрейма, применяя полюсную фильтрацию к выбранным областям спектральных коэффициентов. Число фильтров формирования шума, применяемых к каждому окну, определяется "*n_filt*". Целевой диапазон спектральных коэффициентов определяется в единицах полос масштабного коэффициента, считающих в обратном порядке полосы длины от верхней полосы (или нижней предыдущей полосы формирования шума).

Сначала переданные коэффициенты фильтра должны декодироваться, то есть преобразовываться в числа со знаком, инверсно квантоваться, преобразовываться в коэффициенты *LPC*, как описано в функции *tns_decode_coef*(). Затем к целевым частотным областям спектральных коэффициентов канала применяются полюсные фильтры. Чтобы определить направление, в котором фильтр продвигают через коэффициенты, используется маркер "направление" (0=*upward*, 1=*downward*). Константа *TNS_MAX_BANDS* определяет максимальное количество полос масштабного коэффициента, к которым применяется временное формирование шума. Максимальный возможный порядок фильтра определяется константой *TNS_MAX_ORDER*. Обе константы являются зависимыми параметрами объектного типа.

Процесс декодирования для одного канала может быть описан следующим псевдокодом:

```
/* TNS decoding for one channel and frame */
tns_decode_frame()
{
for (w = 0; w < num_windows; w++) {
bottom = num_swb;
for (f = 0; f < n_filt[w]; f++)  {
top = bottom;
bottom = max( top - length[w][f], 0 );
tns_order = min( order[w][f], TNS_MAX_ORDER );
if (!tns_order)  continue;
tns_decode_coef( tns_order, coef_res[w]+3, coef_compress[w][f],
coef[w][f], lpc[] );
start  = swb_offset[min(bottom,TNS_MAX_BANDS,max_sfb)];
end =  swb_offset[min(top,TNS_MAX_BANDS,max_sfb)];
if ((size = end - start) <= 0)  continue;
if (direction[w][f])  {
inc = -1;  start = end - 1;
}  else  {
inc =  1; }
tns_ar_filter( &spec[w][start], size, inc, lpc[], tns_order );
}
}
}
```

```
/* Decoder transmitted coefficients for one TNS filter */
tns_decode_coef( order, coef_res_bits, coef_compress, coef[], a[] )
{
/* Some internal tables */
sgn_mask[]  = { 0x2,  0x4,  0x8 };
neg_mask[]  = { ~0x3, ~0x7, ~0xf };
/* size used for transmission */
coef_res2 = coef_res_bits - coef_compress;
s_mask = sgn_mask[ coef_res2 - 2 ];  /* mask for sign bit */
n_mask = neg_mask[ coef_res2 - 2 ];  /* mask for padding neg. values */
/* Conversion to signed integer */
for (i = 0; i < order; i++)
tmp[i] = (coef[i] & s_mask) ? (coef[i] | n_mask) : coef[i];
/* Inverse quantization */
iqfac = ((1 << (coef_res_bits-1)) – 0,5) / (p/2,0);
iqfac_m = ((1 << (coef_res_bits-1)) + 0,5) / (p/2,0);
for (i = 0; i < order; i++)  {
tmp2[i] = sin( tmp[i] / ((tmp[i] >= 0) ? iqfac : iqfac_m) );
}
}
/* Conversion to LPC coefficients */
a[0] = 1;
for (m = 1; m <= order; m++) {
for (i = 1; i < m; i++) {                    /* loop only while i<m */
b[i] = a[i] + tmp2[m-1] * a[m-i];
}
for (i = 1; i < m; i++) {                    /* loop only while i<m */
a[i] = b[i];
}
a[m] = tmp2[m-1];                                /* changed */
}
tns_ar_filter( spectrum[], size, inc, lpc[], order )
{
- Simple all-pole filter of order "order" defined by
y(n) =  x(n) - lpc[1]*y(n-1) - ... - lpc[order]*y(n-order)
- The state variables of the filter are initialized to zero every time
- The output data is written over the input data ("in-place operation")
- An input vector of "size" samples is processed and the index increment to the next data sample is given
by "inc"
}.
```

Этот псевдокод использует интерпретацию „*C*"-стиля массивов и векторов, то есть если *coef[w] [filt] [i]* описывает коэффициенты для всех окон и фильтров, *coef[w] [filt]* является указателем на коэффициенты одного определенного окна и фильтра. Коэффициент идентификатора используется в качестве формального параметра в функции *tns_decode_coef* ().

6.9.4 Максимальный порядок *TNS* и ширина полосы

Значение для константы *MAX_TNS_ODER* зависит от аудио объектного типа и работы с окнами, таблица 156 определяет *MAX_TNS_ORDER* в зависимости от этих параметров.

Таблица 156 — Определение *TNS_MAX_ORDER* в зависимости от *AOT* и работы с окнами

	Работа с окнами	
	Короткие окна	Длинные окна
AOT 1 (AAC Main)	7	20
Другое *AOT* используя *TNS*	7	12

Согласно частоте дискретизации и используемому аудио объектному типу значение для константы *TNS_MAX_BANDS* устанавливается в соответствии с таблицей 157.

Таблица 157 — Определение *TNS_MAX_BANDS* в зависимости от *AOT*, работы с окнами и частоты дискретизации

Частота дискретизации, Гц	Аудиообъектные типы без гребенки фильтров *PQF* (длинные окна)	Аудиообъектные типы без гребенки фильтров *PQF* (короткие окна)	Аудиообъектные типы с гребенкой фильтров *PQF* (длинные окна)	Аудиообъектные типы с гребенкой фильтров *PQF* (короткие окна)
96000	31	9	28	7
88200	31	9	28	7
64000	34	10	27	7
48000	40	14	26	6
44100	42	14	26	6
32000	51	14	26	6
24000	46	14	29	7
22050	46	14	29	7
16000	42	14	23	8
12000	42	14	23	8
11025	42	14	23	8
8000	39	14	19	7

6.9.5 TNS в масштабируемом кодере

Для первого моно уровня и для обоих каналов первого уровня стерео доступен бит *tns_data_present,* который активирует использование *TNS* для определенного канала. Информация о фильтре *TNS* не обязательно передается наряду с битом включения. Таблица 158 перечисляет исходный канал, из которого должна быть взята информация о фильтре *TNS* для определенного канала вывода.

Во всех конфигурациях *TwinVQ-mono/AAC-mono* и *TwinVQ-stereo/AAC-stereo* любой первый уровень *TwinVQ* использует *TNS* (*tns_data_present* == 1 в *tvq_main_header (),* бит в *ASME tns_data_present* отсутствует) или *ASME* (*tns_data_present* == 0 в *tvq_main_header (),* бит *tns_data_present* в *ASME*), но не оба одновременно.

Во всех конфигурациях *TwinVQ-mono/AAC-stereo* первый уровень *TwinVQ* переносит бит *tns_data_present* и два бита *tns_data_present* имеют место в первом уровне стерео *AAC* (один для каждого канала аудиовыхода).

Во всех конфигурациях *TwinVQ-mono/AAC-mono/AAC-stereo* любой первый уровень *TwinVQ* использует *TNS* и поэтому биты *tns_data_present* и *tns_data* () имеют место в *tvq_main_header* (), или первый моноуровень *AAC* и в обоих случаях два бита *tns_data_present* в первом уровне стерео *AAC.*

Таблица 158 — Информация канала источника в информации фильтра TNS в зависимости от *tns_data_present* в масштабируемом кодере моно/стерео

tns_data_present M-Канал	*tns_data_present L*-Канал	*tns_data_present R*-Канал	Информация *TNS M* Исходный Канал.	Информация *TNS L* Исходный Канал.	Информация *TNS R* Исходный Канал.
0	0	0	—	—	—
1	0	0	*M*	*M*	*M*
0	1	1	—	*L*	*R*
0	1	0	—	*L*	—

Окончание таблицы 158

tns_data_present *M*-Канал	*tns_data_present* *L*-Канал	*tns_data_present* *R*-Канал	Информация *TNS M* Исходный Канал.	Информация *TNS L* Исходный Канал..	Информация *TNS R* Исходный Канал.
0	0	1	—	—	*R*
1	0	1	*M*	*M*	*R/M*
1	1	0	*M*	*L/M*	*M*
1	1	1	*M*	*L/M*	*R/M*

Записи *TNS L/M* и *R/M* описывают последовательное расположение двух фильтров *TNS*. Дополнительно в этом случае применяются следующие правила:

- выполнение *M-Filter* должно остановиться в полосе масштабного коэффициента, которая обозначается параметром *max_sfb* самого высокого моноуровня;

- *M=filter* не вычисляется, если более низкая граница *L-* или *R-Filter* ниже, чем *max_sfb* самого высокого моноуровня. Это позволяет при желании переопределять *M-Filter* для более низких полос частот.

Если *TNS* используется в масштабируемом кодере с базовым кодером, к выводу *MDCT* дожен быть примен фильтр кодера *TNS* указывающего уровня моно *AAC*, который используется, чтобы генерировать спектр базового кодера. Эти фильтры кодера используют коэффициенты *LPC*, уже декодированные для соответствующих фильтров декодера *TNS*. Фильтры продвигаются через указанный целевой частотный диапазон точно по пути, описанному для фильтра декодера. Различие между фильтрацией декодера и кодера состоит в том, что каждый полюсный (авторегрессивный) фильтр декодера, используемый для декодирования *TNS*, заменяется его инверсным (бесполюсным, скользящего среднего значения) фильтром.

Если *TNS* используется в масштабируемом кодере с кодером *TwinVQ* и если уровень *TwinVQ* не использует *TNS*, к выводу декодера *TwinVQ* должен быть применен фильтр кодера *TNS* указывающего моно уровня *AAC*. Эти фильтры кодера используют коэффициенты *LPC*, уже декодированные для соответствующих фильтров декодера *TNS*. Фильтры продвигаются через указанный целевой частотный диапазон точно по пути, описанному для фильтра декодера. Различие между фильтрацией декодера и кодера состоит в том, что каждый полюсный (авторегрессивный) фильтр декодера, используемый для декодирования *TNS*, заменяется его инверсным (бесполюсным, скользящего среднего значения) фильтром.

Уравнение фильтра имеет вид:

$$y[n] = x[n] + lpc[1] * x[n-1] + \ldots + lpc[order] * x[n-order].$$

Число фильтров, направление фильтрации и т. д. управляются точно так же, как в процессе декодирования.

6.9.5.1 База моно + *AAC* стерео без какого-либо уровня моно *AAC*

Если *TNS* используется в масштабируемом кодере с базовым кодером, к выводу из базового кодера должен быть применен фильтр кодера *TNS* или левого или правого канала (в зависимости от флага *tns_channel_mono_layer*) первого уровня стерео *AAC*.

Если *TNS* используется в масштабируемом кодере с кодером *TwinVQ*, и если уровень *TwinVQ* не использует *TNS*, к выводу декодера *TwinVQ* должен быть применен фильтр кодера *TNS* или левого или правого канала (в зависимости от флага *tns_channel_mono_layer*) первого уровня стерео *AAC*.

6.10 Нормализация спектра

6.10.1 Описание инструмента

В декодере *TwinVQ* спектральная денормализация используется в комбинации с инверсным векторным квантованием коэффициентов *MDCT*, репродукция которых имеет глобально плоскую форму. Используя этот инструмент, регенерируется спектральная огибающая с применением декодирования усиления, огибающей *Bark-scale* и огибающей, определенной параметрами *LPC*. Огибающая *Bark-scale* восстанавливается, используя декодер векторного квантования. Коэффициенты *LPC* квантуются в домене *LSP* посредством двухэтапного квантования вектора разделения со скользящим средним межфреймовым прогнозом. Декодированные коэффициенты *LSP* непосредственно используются для генерации амплитудного спектра (квадратный корень спектральной огибающей энергии).

В режиме длинного блока *MDCT* к сглаженным коэффициентам *MDCT* для кодера с низкой скоростью дополнительно добавляются периодические пиковые компоненты.

6.10.2 Определения

a	Коэффициенты прогноза *MA*, используемые для квантования *LSP*.
alfq [] []	Коэффициенты прогноза для *Bark-envelope*.
AMP_MAX	Максимальное значение квантователя mu-закона для глобального усиления.
AMP_NM	Коэффициент нормализации для глобального усиления.
BASF_STEP	Размер шага квантователя базовой частоты для кодирования периодических пиковых компонентов.
bfreq []	Базовая частота периодических пиковых компонентов.
blim_h []	Фактор управления шириной полосы (верхняя часть).
blim_l []	Фактор управления шириной полосы (нижняя часть).
BLIM_STEP_H	Число шагов квантования управления шириной полосы (верхняя часть).
BLIM_STEP_L	Число шагов квантования управления шириной полосы (нижняя часть).
CUT_M_H	Минимальное отношение ширины полосы (верхняя часть).
CUT_M_L	Максимальное отношение ширины полосы (нижняя часть).
cv_env [] []	Кодовые векторы сборника кодов огибающей
env[] [] []	Огибающая *Bark-scale*, спроектированная на ось частот *Bark-scale*.
fb_shift [] []	Элемент синтаксиса, указывающий активную полосу частот адаптивного управления полосой пропускания.
FW_ALF_STEP	Коэффициент прогноза *MA* для квантования огибающей *Bark-scale*.
FW_CB_LEN	Длина кодового вектора сборника кодов огибающей *Bark-scale*.
FW_N_DIV	Число подразделений чередования квантования вектора огибающей *Bark-scale*.
gain_p [] []	Коэффициенты усиления периодических пиковых компонентов.
gain [] []	Коэффициенты усиления коэффициентов *MDCT*.
global_gain []	Глобальное усиление коэффициентов *MDCT*, нормализованных *AMP_NM*.
index_blim_h []	Элемент синтаксиса, указывающий управление верхней частью полосы пропускания
index_blim_l []	Элемент синтаксиса, указывающий управление нижней частью полосы пропускания.
index_env [] [] []	Элементы синтаксиса, указывающие элементы огибающей *Bark-scale*.
index_fw_alf []	Элемент синтаксиса, указывающий переключатель прогноза *MA* квантования огибающей *Bark-scale*.
index_gain [] []	Элементы синтаксиса, указывающие глобальное усиление коэффициентов *MDCT*.
index_gain_sb [] [] []	Элементы синтаксиса, указывающие усиление подблока коэффициентов *MDCT*.
*index_lsp*0 *[] []*	Элементы синтаксиса, указывающие коэффициенты прогноза *MA*, используемые для квантования *LSP*.
*index_lsp*1 *[]*	Элемент синтаксиса, указывающий квантование *LSP* первой стадии.
index_lsp2[] []	Элемент синтаксиса, указывающий квантование *LSP* второй стадии.
index_pgain []	Элемент синтаксиса, указывающий усиление периодических пиковых компонентов.
index_pit [] []	Элементы синтаксиса, указывающие базовую частоту периодических пиковых компонентов.
index_shape0_p []	Элемент синтаксиса, указывающий индекс квантования пиковых элементов для вектора формы спаренного канала 0.
*index_shape*1*_p []*	Индекс квантования периодических пиковых элементов для вектора формы спаренного канала 1.
isp []	Таблица точки разделения для квантования *LSP* второго этапа.
lengthp []	Длины векторов кода для квантования периодических пиковых компонентов.
lnenv [] []	Огибающая *Bark-scale*, спроектированная на ось частот линейной шкалы.
LOWER_BOUNDARY [] []	Нижняя граница активной полосы частот, используемая в уровнях масштабируемости.
lpenv [] []	Спектральная огибающая *LPC*.
lsp [] []	Коэффициенты *LPC*, диапазон которых устанавливается от нуля до π.

lyr	Указывает номер уровня расширения. Номер 0 присваивается для базового уровня.
LSP_SPLIT	Число разделений векторного квантования 2-ого этапа для кодирования *LSP*.
MU	Фактор mu для квантования по mu-закону для усиления.
N_CRB	Числа поддиапазонов для кодирования огибающей *Bark-scale*.
N_DIV_P	Число подразделений чередования для кодирования периодических пиковых компонентов.
N_FR	Число выборок в подфрейме.
N_FR_P	Число элементов периодических пиковых компонентов.
N_SF	Число подфреймов во фрейме.
p_cv_env [] []	Вектор огибающей *Bark-scale*, восстановленный в предыдущем фрейме.
pit []	Периодические пиковые компоненты.
pit_seq [] [] []	Периодические пиковые компоненты, спроектированные в линейную шкалу.
PIT_CB_SIZE	Размер сборника кодов для квантования периодических пиковых компонентов.
PGAIN_MAX	Максимальная величина квантователя по mu-закону для усиления периодических пиковых компонентов.
PGAIN_MU	Фактор mu для квантования по mu-закону для периодических пиковых компонентов.
PGAIN_STEP	Размер шага квантователя по mu-закону для усиления периодических пиковых компонентов.
pol0_p	Полярность спаренного канала 0 для квантования периодических пиковых компонентов.
pol1_p	Полярность спаренного канала 1 для квантования периодических пиковых компонентов.
pit_cv0 []	Форма *econstructed* сопряженного канала 0 для периодических пиков компонентов квантования.
pit_cv1 []	Восстановленная форма спаренного канала 1 для периодических пиков компонентов квантования.
STEP	Размер шага квантователя по mu-закону для глобального усиления.
SUB_AMP_MAX	Максимальная амплитуда квантователя по mu-закону для коэффициента усиления подфрейма.
SUB_AMP_NM	Коэффициент нормализации для коэффициента усиления подфрейма.
SUB_STEP	Размер шага квантователя по mu-закону для коэффициента усиления подфрейма.
subg_ratio []	Коэффициент усиления подфрейма.
UPPER_BOUNDARY [] []	Верхняя граница активной полосы частот, используемая в масштабируемых уровнях.
v []	Коэффициенты *LSP*.
v1 []	Восстановленный вектор из *VQ* 1-го этапа для кодирования *LSP*.
v2 []	Восстановленный вектор из *VQ* 2-го этапа для кодирования *LSP*.
x_flat []	Нормализованные коэффициенты *MDCT* (ввод).
spec [] [] []	Денормализованные коэффициенты *MDCT* (вывод).

6.10.3 Процесс декодирования

Процесс декодирования состоит из пяти частей: декодирование усиления, декодирование огибающей *Bark-scale*, декодирование периодических пиковых компонент, декодирование спектра *LPC* и инверсная нормализация.

6.10.3.1 Инициализации

Прежде, чем запустить любой процесс, очищаются памяти прогноза *p_cv_env [] []* и $\alpha_i^{(j)}$.

6.10.3.2 Декодирование усиления

В первом шаге декодирования усиления декодируется глобальное усиление, используя инверсный квантователь по mu-закону, описанный следующим образом:

for (i_ch = 0; i_ch <N_CH; i_ch ++) {
*g_temp = index_gain * STEP + STEP / 2;*
global_gain =
*(AMP_MAX * (exp10 (g_temp * log10 (1. + MU) / AMP_MAX)-1) / MU) / AMP_NM;*
}.

Затем декодируются коэффициенты усиления подполосы, используя инверсный квантователь по mu-закону, описанный следующим образом:

```
for (i_ch = 0; i_ch <N_CH; i_ch ++) {
if (N_SF> 1) {
for (isf = 0; isf <N_SF; isf ++) {
g_temp = index_gain_sb [i_ch] [isf+1] * SUB_STEP +SUB_STEP/2.;
subg_ratio [isf] =
(SUB_AMP_MAX * (exp10 (g_temp*log10 (1. + MU)/SUB_AMP_MAX)-1)/MU)
/ SUB_AMP_NM;
}
}
else {
subg_ratio [i_ch] [0] = 1;
}
}.
```

Восстанавливаются коэффициенты усиления следующим образом:

```
for (i_ch = 0; i_ch <N_CH; i_ch ++) {
for (isf = 0; isf <N_SF; isf ++) {
gain [i_ch] [isf] = global_gain [i_ch] * subg_ratio [i_ch] [isf] / SUB_AMP_NM;
}
}.
```

6.10.3.3 Декодирование периодических пиковых компонентов

Периодические пиковые компоненты опционно добавляются к входным коэффициентам. Периодические пиковые компоненты кодируются, используя векторное квантование. Этот процесс является активным, когда параметры *ppc_present* устанавливаются в *TRUE*. Иначе все элементы выходного массива, *pit_seq []* обнуляются и процесс пропускается.

```
MAXBIT_P = 6
PIT_CB_SIZE = (1 <<MAXBIT_P).
```

6.10.3.3.1 Декодирование полярности

```
for (idiv = 0; idiv <N_DIV_P; idiv ++) {
pol0 [idiv] = 2 * (index_shape0_p [idiv] / PIT_CB_SIZE) - 1;
pol1 [idiv] = 2 * (index_shape1_p [idiv] / PIT_CB_SIZE) - 1;
}.
```

6.10.3.3.2 Декодирование кода формы

```
for (idiv = 0; idiv <N_DIV_P; idiv ++) {
index0 [idiv] = index_shape0_p [idiv] % PIT_CB_SIZE;
index1 [idiv] = index_shape1_p [idiv] % PIT_CB_SIZE;
}.
```

Декодирование усиления периодических пиковых компонентов:

```
for (i_ch = 0; i_ch <N_CH; i_ch ++) {
temp = index_pgain [i_ch] * PGAIN_STEP + PGAIN_STEP / 2;
gain_p [i_ch] =
(PGAIN_MAX * (exp10 (temp*log10 (1. + PGAIN_MU)/PGAIN_MAX)-1)
/PGAIN_MU)/AMP_NM;
}.
```

6.10.3.3.3 Реконструкция периодических пиковых компонентов

Имеется два шага процедур. Сначала вычисляются длины векторов кода для периодических пиковых компонентов *lengthp []*, затем вычисляются периодические пиковые компоненты *pit[]*.

```
for (idiv = 0; idiv <N_DIV_P; idiv ++) {
lengthp [idiv] = (N_FR_P*N_CH+N_DIV_P-1-idiv) / N_DIV_P;
}
for (idiv = 0; idiv <N_DIV_P; idiv ++) {
if (N_CH == 1) {
for (icv = 0; icv <lengthp [idiv]; icv ++) {
ismp = idiv + icv * N_DIV_P;
```

```
pit [ismp] = (pol0 [idiv] *pit_cv0 [index0 [idiv] [icv]] +
pol1 [idiv] *pit_cv1 [index1 [idiv]] [icv]) / 2;
}
}
else {
for (icv = 0; icv <lengthp [idiv]-1; icv ++) {
ismp = ((icv+idiv) %N_DIV_P) + icv * N_DIV_P;
ismp = ismp/2 + (ismp%2) *20;
pit [ismp] = (pol0 [idiv] *pit_cv0 [index0 [idiv] [icv]] +
pol1 [idiv] *pit_cv1 [index1 [idiv]] [icv]) / 2;
}
icv = lengthp [idiv]-1;
ismp = idiv + icv* N_DIV_P;
ismp = ismp/2 + (ismp%2) *20;
pit [ismp] = (pol0 [idiv] *pit_cv0 [index0 [idiv] [icv]] +
pol1 [idiv] *pit_cv1 [index1 [idiv]] [icv]) / 2;
}
}.
```

6.10.3.3.4 Проектирование периодических пиковых компонентов в линейной шкале

Сначала параметры вычисляются следующим образом:

```
fcmin = log2 ((N_FR / (double) ISAMPF) *0,2);
fcmax = log2 ((N_FR / (double) ISAMPF) *2,4);
if (ISAMPF == 8) bandwidth = 1,5;
else, if (ISAMPF> = 11) bandwidth = 2,0;
else, if (ISAMPF> = 22) bandwidth = 4,0;
if (bandwidth <1./UPPER_BOUNDARY [0] [0]) bandwidth = 1./UPPER_BOUNDARY [0] [0],
```

где *fcmin* является минимальной частотой при квантовании, *fcmax* является максимальной частотой и *ISAMPF* является целочисленной частотой дискретизации, усеченной из значений стандартной частоты, перечисленных в правой графе таблицы 82.

Затем базовая частота периодических пиковых компонентов декодируется согласно следующей процедуре:

```
for (i_ch = 0; i_ch <N_CH; i_ch ++) {
pow_i = (int) (pow (1,009792f, (float) index_pit [i_sup]) *4096. + 0,5);
bl_i = (int) ((float) block_size_samples / (float) isampf * 0,2*1024+0,5);
pitch_i = pow_i *bl_i/256.;
bfreq [l_ch] = pitch_i/16384.;
}.
```

Прежде, чем спроектировать периодические пиковые компоненты в линейную шкалу, все элементы целевого массива *pit_seq [] []* обнуляются:

```
for (i_ch = 0; i_ch <N_CH; i_ch ++) {
for (ismp = 0; ismp <N_FR; ismp ++) {
pit_seq [i_ch] [ismp] = 0.;
}
}.
```

Затем восстановленные периодические пиковые компоненты проектируются в линейную шкалу следующим образом:

```
for (i_ch = 0; i_ch <N_CH; i_ch ++) {
if (bandwidth * upperlimit_i <16384) {
tmpnp0_i = pitch_i*16384. / (upperlimit_i);
tmpnp1_i = tmpnp0_i *N_FR_P;
tmpnp0_i = tmpnp1_I / N_FR;
npcount = tmpnp0_i/16384.;
} else {
tmpnp0_i = pitch_i * bandwidth*2;
tmpnp1_i = tmpnp0_i *N_FR_P;
```

```
tmpnp0_i = tmpnp1_i/N_FR;
npcount = tmpnp0_i / 32768;
}
iscount=0;
for (jj = 0; jj <npcount/2; jj ++) {
pit_seq [i_ch] [jj] = pit [jj+i_ch*N_FR_P];
iscount ++;
}
for (ii = 0; ii <(ntt_N_FR_P) && (iscount <ntt_N_FR_P); ii ++) {
tmpnp0_i = pitch_i * (ii+1);
tmpnp0_i + = 8192;
i_smp = tmpnp0_i / 16384;
for (jj = - npcount/2; jj <(npcount-1)/2+1; jj ++) {
pit_seq [i_ch] [i_smp+jj] = pit [iscount+i_ch*N_FR_P];
iscount ++;
if (iscount> = N_FR_P) break;
}
}
}.
```

В случае пропуска процесса декодирования периодических пиковых компонентов все элементы массива компонентов шага *pit_seq [] []* обнуляются.

6.10.3.4 Декодирование огибающей *bark-scale*

Огибающая *bark-scale* декодируется в каждом подфрейме. Существуют два этапа процедуры: инверсное квантование векторов огибающей *env [] []* и проектирование огибающих *bark-scale env [] []* в огибающие линейной шкалы *lnenv [] []*.

6.10.3.4.1 Инверсное квантование вектора огибающих

```
for (i_ch = 0; i_ch <N_CH; i_ch ++) {
for (isf = 0; isf <N_SF; isf ++) {
alfq [i_ch] [isf] = index_fw_alf [i_ch] [isf] * FW_ALF_STEP;
for (ifdiv = 0; ifdiv <FW_N_DIV; ifdiv ++) {
for (icv = 0; icv <FW_CB_LEN; icv ++) {
ienv = FW_N_DIV * icv + ifdiv;
dtmp = cv_env [index_env [ich] [isf] [ifdiv]] [icv];
env [i_ch] [isf] [ienv] = dtmp + alfq [i_ch] [isf] * p_cv_env [i_ch] [icv] + 1;
p_cv_env [i_ch] [icv] = dtmp;
}
}
}
}.
```

cv_env [] [] является сборником кодов огибающей *bark-scale,* перечисленных в таблице А.29.

6.10.3.4.2 Проектирование огибающей *bark-scale* в линейную шкалу

Огибающие *env [] [] []* выражаются, используя шкалу *Bark* на оси частот. Процедура денормализации требует огибающих линейной шкалы.

Перед процессом проектирования определяется граничная таблица подполосы *bark-scale*, *crb_tbl []*. В случае базового уровня (*lyr* = 0),

```
if (sampling_rate <= 16000)
use the scalefactor band table of AAC for 16 kHz up to 41st value
else
use the scalefactor band table of AAC for 24 kHz up to 41st value.
```

Число полос масштабного коэффициента равно 42 для размера длинного фрейма.

42-ое значение является длиной фрейма длинного фрейма (1024 или 960).

Оценка *Barkscale* не используется для коротких фреймов.

Если *lyr*> = 1, значения таблицы подполос *Bark-scale* вычисляются следующим образом:

```
for (i_ch = 0; i_ch <N_CH; i_ch ++) {
lower_band_i = (int) (AC_BTM [lyr] [i_ch] [fb_shift] *16384.);
```

*upper_band_i = (int) (AC_TOP [lyr] [i_ch] [fb_shift] *16384.);*
average_number_of_lines =
*(int) (frame_length * (upper_band_i-lower_band_i))/N_CRB;*
for (i = 0; i <N_CRB-1; i ++) {
*crb_tbl [i_ch] [i] = (int) ((i+1) * (i+1) * average_number_of_lines/N_CRB/2,0)*
*crb_tbl [i_ch] [i] + = (i+1) * average_number_of_lines/2,0 +8192;*
crb_tbl [i_ch] [i] / = 16384.
*crb_tbl {i_ch] [i] + = (int) (frame_length*lower_band_i)/16384.;*
}
*crb_tbl [i_ch] [N_CRB-1] = (int) (frame_length*lower_band_i)/16384. + (int) (frame_length * (upper_band_i-lower_band_i))/16384.;*
}.

После того, как определяется *crb_tbl [] []*, процесс проектирования выполняется следующим образом:

for (i_ch = 0; i_ch <N_CH; i_ch ++) {
for (isf = 0; isf <N_SF; isf ++) {
ismp = 0
for (ienv = 0; ienv <N_CRB; ienv ++) {
while (ismp <crb_tbl [i_ch] [ienv]) {
lnenv [i_ch] [isf] [ismp] = env [i_ch] [isf] [ienv];
ismp ++;
}
}
}
}.

Значения *UPPER_BOUNDARY [] []* и *LOWER_BOUNDARY [] []* определяются в 6.5.4.4.

Если *postprocess_present* является активным, *lnenv [i_ch] [isf] [ismp]* изменяется следующим образом:

*If (lnenv [i_ch] [isf] [ismp] <1,0) lnenv [i_ch] [isf] [ismp] = lnenv [i_ch] [isf] [ismp] * lnenv [i_ch] [isf] [ismp].*

6.10.3.5 Декодирование спектра *LPC*

Спектр *LPC* представляется коэффициентами *LSP*. В процессе декодирования коэффициенты *LSP* сначала восстанавливаются, затем они преобразовываются в спектр *LPC*, который представляет квадратный корень из спектра энергии.

6.10.3.5.1 Декодирование коэффициентов *LSP*, используя прогноз *MA*

Коэффициенты прогноза *MA* определяются путем обращения к таблице коэффициентов $\alpha t_i^{(j)}$. Правило имеет вид:

$$\alpha_i^{(j)}[i_ch] = \alpha t_i^{(j)}[i_ch]\ (index_lsp0[i_ch]),$$

где *i* — порядок *LPC*, и *j* является порядком прогноза *MA*. Коэффициенты $\alpha t_i^{(j)}$ приведены в таблицах А.19 и А.20.

6.10.3.5.2 Инверсное квантование первой стадии декодирования *LSP*

$$v1_i[i_ch] = lspcode1_i(index_lsp1[i_ch]),$$

где *lspcode1* является сборником кодов *LSP* первого этапа, перечисленных таблицах А.19 и А.20.

6.10.3.5.3 Инверсное квантование второго этапа декодирования *LSP*

$$v2_i[i_ch] = lspcode2_i(index_lsp2[i_ch]\ [k]),$$

где *lspcode2* является сборником кодов *LSP* второго этапа. Значения *isp* (*k*) перечисляются в таблице 161.

6.10.3.5.4 Реконструкция коэффициентов *LSP*

Коэффициенты *LSP lsp [] []* вычисляется следующим образом:

$$\overline{v[i_ch]} = \overline{v1[i_ch]} + \overline{v2[i_ch]}, \qquad \text{для } i_ch \text{ от 0 до } N_CH\text{-}1$$

$$\alpha_i^{(0)}[i_ch] = 1 - \sum_{j=1}^{MA_NP} \alpha_i^{(j)}[i_ch], \qquad \text{для } i \text{ от 1 до } N_PR,\ i_ch \text{ от 0 до } N_CH\text{-}1$$

$$lsp[i_ch][i] = \sum_{j=1}^{MA_NP} \alpha_i^{(j)}[i_ch] v_i^{(-j)}[i_ch], \quad \text{для } i \text{ от 1 до } N_PR, \ i_ch \text{ от 0 до } N_CH\text{-}1$$

$$\vec{v}^{(j-1)}[i_ch] = \vec{v}^{(j)}[i_ch], \quad \text{для } j \text{ от } -MA_NP\text{-}1 \text{ до } 0, \ i_ch \text{ от 0 до } N_CH\text{-}1.$$

6.10.3.5.5 Преобразование параметров *LSP* в спектр *LPC*

Огибающая амплитудного спектра *LPC*, соответствующая *ii*-му коэффициенту *MDCT*, *lpenv [] [ii]*, определяется следующим образом:

lpenv [] [] представляет огибающую амплитуды, начиная с получения огибающей исходного спектра *LPC* из квадратного корня энергетического спектра в кодере.

```
for (i_ch = 0; i_ch <N_CH; i_ch ++) {
for (ii = 1; ii <= N_FR-1; ii ++) {
for (i = 2, P [i_ch] = 1,0; i <= N_PR; i + = 2)
P [i_ch] * = (cos (PI*ii/N_FR) - cos (lsp [i])) ^2;
for (i = 1, Q [i_ch] = 1,0; i <= N_PR; i + = 2)
Q [i_ch] * = (cos (PI*ii/N_FR) - cos (lsp [i])) ^2;
lpenv [i_ch] [ii] = 1 / ((1 - cos (PI*ii/N_FR)) *P [i_ch] + (1+cos (PI*ii/N_FR)) *Q [i_ch]);
}
}.
```

В случае длинных фреймов (*N_FR* == 1024, или 960), *lpenv [] [ii]* должен быть вычислен только в точках частоты *ii*.

Для остающихся частотных точек значения *lpenv [ii]* вычисляются линейной интерполяцией из уже вычисленных значений в ближайших двух частотных точках. Если частотные точки больше чем *N_FR-8*, *lpenv [] [ii]* должен быть равен *lpenv [] [N_FR-8]*.

Если этот инструмент используется в качестве элемента масштабируемого кодера, спектр *LPC* сжимается в активную полосу частот:

```
for (i_ch = 0; i_ch <N_CH; i_ch ++) {
nfr_lu = UPPER_BOUNDARY [lyr] [i_ch] - LOWER_BOUNDARY [lyr] [i_ch];
for (ismp = 0; ismp <LOWER_BOUNDARY [lyr] [i_ch]; ismp ++) {
lpenv_tmp [i_ch] [ismp] = 0;
}
upperband_i = (int) (AC_TOP [lyr] [i_ch] [fb_shift] * 16384.);
lowerband_i = (int) (AC_BTM [lyr] [i_ch] [fb_shift] * 16384.);
ftmp = (16384*16384) / (upperband_i - lowerband_i);
for (ismp = 0; ismp <nfr_lu; ismp ++) {
ftmp = (интервал) (ismp*ftmp)/16384;
lpenv_tmp [i_ch] [ismp+LOWER_BOUNDARY [lyr] [i_ch]] = lpenv [i_ch] [ftmp];
}
for (ismp = UPPER_BOUNDARY [lyr] [i_ch]; ismp <N_FR; ismp ++) {
lpenv_tmp [i_ch] [ismp] = 0;
}
for (ismp = 0; ismp <N_FR; ismp ++) {
lpenv [i_ch] [ismp] = lpenv_tmp [i_ch] [ismp];
}
}.
```

Значения *UPPER_BOUNDARY*, *LOWER_BOUNDARY*, *AC_TOP* и *AC_BTM* определяются в 6.5.4.4.

6.10.3.5 Инверсная нормализация

Входные коэффициенты *x_flat []* применяются к инверсной нормализации согласно следующей процедуре и создаются выходные коэффициенты *spec [] [] []*.

```
for (i_ch = 0; i_ch <N_CH; i_ch ++) {
for (isf = 0; isf <N_SF; isf ++) {
for (ismp = 0; ismp <N_FR; ismp ++) {
spec [isf] [ismp] =
(x_flat [ismp + (isf+i_ch*N_SF) *N_FR]
```

**lnenv [i_ch] [isf] [ismp] * gain [i_ch] [isf] + p_gain [i_ch] *pit_seq [i_ch] [ismp])*
**lpenv [i_ch] [ismp];*
}
}
}.

6.10.3.5 Управление шириной полосы

Эта функциональность допустима, только если флаг *bandlimit_present* находится в состоянии *ON*.

После инверсной нормализации обнуляются верхние и нижние полосы выходных коэффициентов *spec [] [] []*. В модулях декодирования полосы пропускания более высокое отношение сигнальной полосы пропускания *blim_h []* декодируется следующим образом:

for (i_ch = 0; i_ch <N_CH; i_ch ++) {
blim_h [i_ch] =
*(1. - (1.-CUT_M_H) * (double) index_blim_h [i_ch] / (double) BLIM_STEP_H))*
** UPPER_BOUNDARY [0] [i_ch];*
}.

blim_l [] декодируется следующим образом:

for (i_ch = 0; i_ch <N_CH; i_ch ++) {
if (index_blim_l [i_ch] == 1)
blim_l [i_ch] = LOWER_BOUNDARY [0] [i_ch] +CUT_M_L;
else
blim_l [i_ch] = 0;
}.

В модуле ограничения полосы пропускания верхняя и нижняя части коэффициентов *MDCT* обнуляются следующим образом:

for (i_ch = 0; i_ch <N_CH; i_ch ++) {
*NbaseH = blim_h [i_ch] * N_FR;*
*NbaseL = blim_l [i_ch] * N_FR;*
for (isf = 0; isf <N_SF; isf ++) {
for (ismp = NbaseH; ismp <N_FR; ismp ++) {
spec [i_ch] [isf] [ismp] = 0;
}
for (ismp = 0; ismp <NbaseL; ismp ++) {
spec [i_ch] [isf] [ismp] = 0;
}
}
}.

6.10.4 Таблицы

Параметры, используемые в инверсном процессе нормализации спектра, устанавливаются в таблицах 159, 160, 161. Другие параметры, связанные с синтаксисом, определяются в 5.2.5.3.

Таблица 159 — Таблица параметров для ядра и режима core_960

Параметр	Величина	Значение
Общее:		
AMP_MAX	16000	Максимальная амплитуда в кодировании по mu-закону усиления фрейма
AMP_NM	1024	Нормализованная амплитуда сглаженных коэффициентов
BLIM_BITS_H	2	Биты для управления верхней полосой пропускания
BLIM_BITS_L	*1*	Биты для управления нижней полосой пропускания
BLIM_STEP_H	4	Число шагов квантования ограничения полосы
CUT_M_H	0,7	Минимальное соотношение полосы пропускания (верхняя часть)

Окончание таблицы 159

Параметр	Величина	Значение
CUT_M_L	0,0025	Максимальное соотношение полосы пропускания (нижняя часть)
FW_ALF_STEP	0,5	Коэффициент прогноза *MA* для кодирования огибающей
MA_NP	1	Порядок прогноза *MA* (кодирование *LSP*)
MU	100	Параметр mu в кодировании по mu-закону энергии фрейма
N_PR	20	Порядок *LPC*
NUM_STEP	512	Количество шагов квантования усиления
STEP	*AMP_MAX/N UM_STEP*-1)=31,31	Ширина шага квантования усиления
SUB_AMP_MAX	4700	Максимум отношения усиления подфрейма к усилению фрейма
SUB_AMP_NM	1024	Нормализованная амплитуда подфрейма сглаженных коэффициентов
SUB_GAIN_BIT	4	Число битов для кодирования усиления подфрейма
SUB_NUM_STEP	16	Число шагов *sub-gain-quantization*
SUB_STEP	*SUB_AMP_MAX/(SUB_ NUM_STEP*-1) = 313,3	Ширина шага *sub-gain-quantization*
PGAIN_MAX	20000	Максимальное значение квантователя по mu-закону для усиления *ppc*
PGAIN_MU	200	Фактор mu для квантования по mu-закону для *ppc*
PGAIN_STEP	*PGAIN_MAX/* (1<<*PGAIN_BIT*)	Размер шага квантователя по mu-закону для усиления *ppc*
Длинный блок:		
FW_CB_LEN	6	Длина вектора кода огибающей
FW_N_BIT	6	Биты кода огибающей
FW_N_DIV	7	Число подразделений в кодировании огибающей
N_CRB	42	Число подполос *bark-scale*
Короткий блок:		
N_FR	128/120	Размер блока *MDCT*

Таблица 160 — Таблица параметров для режима расширения и *enhance*_960

Параметр	Величина	Значение
Общее		
AMP_MAX	8000	Максимальная амплитуда в кодировании усиления фрейма по mu-закону
AMP NM	1024	Нормализованная амплитуда сглаженных коэффициентов
FW ALF STEP	0,5	Коэффициент прогноза *MA* для кодирования огибающей
MA_NP	1	Порядок прогноза *MA* (кодирование *LSP*)
MU	100	Параметр mu в кодировании энергии фрейма по mu-закону
N_PR	20	Порядок *LPC*

Окончание таблицы 160

Параметр	Величина	Значение
NUM_STEP	256	Номера шагов квантования усиления
STEP	*AMP_MAX/ (NUM_STEP*-1) = 31,37	Ширина шага квантования усиления
SUB AMP MAX	6000	Максимум отношения усиления подфрейма к усилению фрейма
SUB AMP NM	1024	Нормализованная амплитуда сглаженных коэффициентов в подфрейме
SUB GAIN BIT	4	Число битов для кодирования усиления подфрейма
SUB NUM STEP	16	Число шагов квантования усиления *sub*
SUB_STEP	*SUB_AMP_MAX/ (SUB NUM STEP*-1)= *400,0*	Ширина шага квантования усиления *sub*
Длинный блок:		
FW CB LEN	7	Длина вектора кода огибающей
FW N BIT	6	Биты кода огибающей
FW N DIV	7	Число подразделений в кодировании огибающей
N CRB	42	Число подполос *bark-scale*
N_FR	1024/960	Размер блока *MDCT*
Короткий блок		
N_FR	128/120	Размер блока *MDCT*

Таблица 161 — Величины *isp []*

split_num	*isp[split_num]*
0	0
1	5
2	14
3	20

6.11 Гребенка фильтров и переключение блока

6.11.1 Описание инструмента

Представление времени/частоты сигнала отображается во временную область путем подачи его в модуль гребенки фильтров. Этот модуль состоит из инверсного модифицированного дискретного косинусоидального преобразования (*IMDCT*), окна и функции перекрытия-добавления. Чтобы адаптировать разрешающую способность времени/частоты гребенки фильтров к характеристикам входного сигнала, также принимается инструмент переключения блока. *N* представляет длину окна, где *N* является функцией *window_sequence*. Для каждого канала *N*/2 время-частотные величины $X_{i,k}$ преобразовываются в *N* величин временного интервала $x_{i,n}$, посредством *IMDCT*. После применения функции окна для каждого канала, первая половина последовательности $z_{i,n}$ добавляется ко второй половине предыдущего блока оконной последовательности $z_{(i-1),n}$, чтобы восстановить выходные выборки для каждого канала $out_{i,n}$.

6.11.2 Определения

window_sequence 2 бита, указывающие, какая последовательность окна (то есть размер блока) используется.

window_shape 1 бит, указывающий, какая функция окна выбирается.

6.11.3 Процесс декодирования

6.11.3.1 *IMDCT*

Аналитическое выражение *IMDCT* имеет вид:

$$x_{i,n} = \frac{2}{N}\sum_{k=0}^{\frac{2}{N}-1} spec[i][k]\cos\left(\frac{2\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right) \text{ для } \quad 0 \le n < N,$$

где:

n — индекс выборки

i — индекс окна

k — индекс спектрального коэффициента

N — длина окна, основанная на значении *window_sequence*

$n_0 = (N/2+1)/2$

Длина окна синтеза N для инверсного преобразования является функцией элемента синтаксиса *window_sequence* и алгоритмического контекста. Она определяется следующим образом:

- длина окна 2048:

$N =$ 2048, если *ONLY_LONG_SEQUENCE* (0x0)
2048, если *LONG_START_SEQUENCE* (0x1)
256, если *EIGHT_SHORT_SEQUENCE* (0x2), (8 раз)
2048, если *LONG_STOP_SEQUENCE* (0x3);

- длина окна 1920:

$N =$ 1920, если *ONLY_LONG_SEQUENCE* (0x0)
1920, если *LONG_START_SEQUENCE* (0x1)
240, если *EIGHT_SHORT_SEQUENCE* (0x2), (8 раз)
1920, если *LONG_STOP_SEQUENCE* (0x3).

Значимые блочные переходы следующие:

- для *ONLY_LONG_SEQUENCE* до { *ONLY_LONG_SEQUENCE*; *LONG_START_SEQUENCE;*

- для *LONG_START_SEQUENCE* до { *EIGHT_SHORT_SEQUENCE*; *LONG_STOP_SEQUENCE;*

- для *LONG_STOP_SEQUENCE* до { *ONLY_LONG_SEQUENCE*; *LONG_START_SEQUENCE;*

- для *EIGHT_SHORT_SEQUENCE* до { *EIGHT_SHORT_SEQUENCE*; *LONG_STOP_SEQUENCE.*

В дополнение к значимым блочным переходам возможны следующие переходы:

- для *ONLY_LONG_SEQUENCE* до { *EIGHT_SHORT_SEQUENCE*; *LONG_STOP_SEQUENCE;*

- для *LONG_START_SEQUENCE* до { *ONLY_LONG_SEQUENCE*; *LONG_START_SEQUENCE;*

- для *LONG_STOP_SEQUENCE* до { *EIGHT_SHORT_SEQUENCE*; *LONG_STOP_SEQUENCE;*

- для *EIGHT_SHORT_SEQUENCE* до { *ONLY_LONG_SEQUENCE*; *LONG_START_SEQUENCE.*

Это приведет к разумно гладкому переходу от одного блока к следующему.

6.11.3.2 Работа с окнами и переключение блока

В зависимости от элементов *window_sequence* и *window_shape* используются различные окна преобразования. А комбинация половин окна, описанная следующим образом, предлагает все возможные *window_sequences.*

Для *window_shape* == 1 коэффициенты окна даются окном, полученным по Кайзеру-Бесселю (*KBD*), следующим образом:

$$W_{KBD_LEFT,N}(n) = \sqrt{\frac{\sum_{p=0}^{n}[W'(p,\alpha)]}{\sum_{p=0}^{N/2}[W'(p,\alpha)]}} \quad \text{для} \quad 0 \le n < \frac{N}{2}$$

$$W_{KBD_RIGHT,N}(n) = \sqrt{\frac{\sum_{p=0}^{N-n-1}[W'(p,\alpha)]}{\sum_{p=0}^{N/2}[W'(p,\alpha)]}} \quad \text{для} \quad \frac{N}{2} \le n < N$$

где

W' (функция ядра окна Кайзера-Бесселя) определяется следующим образом:

$$W'(n,\alpha) = \frac{I_0\left[\pi\alpha\sqrt{1{,}0-\left(\frac{n-N/4}{n/4}\right)^2}\right]}{I_0[\pi\alpha]} \quad \text{для} \quad 0 \le n \le \frac{N}{2}$$

$$I_0[x] = \sum_{k=0}^{\infty}\left[\frac{\left(\frac{x}{2}\right)^k}{k!}\right]^2$$

α = альфа-фактор окна ядра,

$$\alpha = \begin{cases} 4 \text{ для } N=2048\ (1920) \\ 6 \text{ для } N=256\ (240) \end{cases}$$

Иначе для *window_shape* == 0 синусное окно используется следующим образом:

$$W_{SIN_LEFT,N}(n) = \sin\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)\right) \quad \text{для} \quad 0 \le n < \frac{N}{2}$$

$$W_{SIN_RIGHT,N}(n) = \sin\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)\right) \quad \text{для} \quad \frac{N}{2} \le n < N\,.$$

Длина окна *N* может быть 2048 (1920) или 256 (240) для *KBD* и синусоидального окна. Для всех видов *window_sequences window_shape* левой половины первого окна преобразования определяется формой окна предыдущего блока. Следующая формула выражает этот факт:

$$W_{LEFT,N}(n) = \begin{cases} W_{KBD_LEFT,N}(n), \text{ если } window_shape_previous_blok == 1 \\ W_{SIN_LEFT,N}(n), \text{ если } window_shape_previous_blok == 0, \end{cases}$$

где:

window_shape_previous_block: window_shape предыдущего блока (*i*-1).

Чтобы декодировать первый *raw_data_block* (), *window_shape* левой и правой половины окна идентичны.

а) *ONLY_LONG_SEQUENCE*:

window_sequence == *ONLY_LONG_SEQUENCE* равно одному *LONG_WINDOW* с полной длиной окна *N_l* из 2048 (1920).

Для *window_shape* == 1 окно для *ONLY_LONG_SEQUENCE* дается следующим образом:

$$W(n) = \begin{cases} W_{LEFT,N_l}(n), \text{ для } 0 \le n < N_l/2 \\ W_{KBD_RIGFT,N_l}(n), \text{ для } N_l/2 \le n < N_l. \end{cases}$$

Если *window_shape* == 0 окно для *ONLY_LONG_SEQUENCE* может быть описано следующим образом:

$$W(n) = \begin{cases} W_{LEFT,N_l}(n), \text{ для } 0 \le n < N_l/2 \\ W_{SIN_RIGFT,N_l}(n), \text{ для } N_l/2 \le n < N_l. \end{cases}$$

После работы с окнами значения временной области ($z_{i,n}$) могут быть выражены как:

$$z_{i,n} = w(n) \cdot x_{i,n};$$

б) *LONG_START_SEQUENCE:*

LONG_START_SEQUENCE необходима, чтобы получить корректное перекрытие и добавление для блочного перехода от *ONLY_LONG_SEQUENCE* к *EIGHT_SHORT_SEQUENCE*.

Длина окон *N_l* и *N_s* устанавливается в 2048 (1920) и 256 (240), соответственно.

Если *window_shape* == 1 окно для *LONG_START_SEQUENCE* дается следующим образом:

$$W(n) = \begin{cases} W_{LEFT,N_l}(n), \text{ для} & 0 \le n < N_l/2 \\ 1{,}0, \text{ для} & N_l/2 \le n < \dfrac{3N_l - N_s}{4} \\ W_{KBD_RIGFT,N_s}\left(n + \dfrac{N_s}{2} - \dfrac{3N_l - N_s}{4}\right), \text{ для} & \dfrac{3N_l - N_s}{4} \le n < \dfrac{3N_l + N_s}{4} \\ 0{,}0, \text{ для} & \dfrac{3N_l + N_s}{4} \le n < N_l. \end{cases}$$

Если *window_shape* == 0 окно для *LONG_START_SEQUENCE* выглядит как:

$$W(n) = \begin{cases} W_{LEFT,N_l}(n), \text{ для} & 0 \le n < N_l/2 \\ 1{,}0, \text{ для} & N_l/2 \le n < \dfrac{3N_l - N_s}{4} \\ W_{SIN_RIGFT,N_s}\left(n + \dfrac{N_s}{2} - \dfrac{3N_l - N_s}{4}\right), \text{ для} & \dfrac{3N_l - N_s}{4} \le n < \dfrac{3N_l + N_s}{4} \\ 0{,}0, \text{ для} & \dfrac{3N_l + N_s}{4} \le n < N_l. \end{cases}$$

Оконные значения временной области могут быть вычислены по формуле, объясняемой в а);

в) *EIGHT_SHORT_SEQUENCE:*

window_sequence == *EIGHT_SHORT_SEQUENCE* включает восемь перекрытых и добавленных *SHORT_WINDOWs* с длиной *N_s* 256 (240) каждое. Полная длина *window_sequence* вместе с ведущим и последующими нулями равна 2048 (1920). Каждый из восьми коротких блоков является сначала отдельно оконным. Номер короткого блока индексируется переменной *j* = 0, …., *M*-1 (*M*=*N_l*/*N_s*).

window_shape предыдущего блока влияет только на первый из восьми коротких блоков *(W_0 (n))*. Если *window_shape* == 1 функции окна могут быть даны следующим образом:

$$W_0(n) = \begin{cases} W_{LEFT,N_s}(n), \text{ для } 0 \le n < N_s/2 \\ W_{KBD_RIGFT,N_s}(n), \text{ для } N_s/2 \le n < N_s \end{cases}$$

$$W_j(n) = \begin{cases} W_{KBD_LEFT,N_s}(n), \text{ для } 0 \le n < N_s/2 \\ W_{KBD_RIGFT,N_s}(n), \text{ для } N_s/2 \le n < N_s \end{cases}, \; 0 < j < M.$$

В ином случае, если *window_shape* == 0, функции окна могут быть описаны как:

$$W_0(n) = \begin{cases} W_{LEFT,N_s}(n), & \text{для } 0 \le n < N_s/2 \\ W_{SIN_RIGHT,N_s}(n), & \text{для } N_s/2 \le n < N_s \end{cases}$$

$$W_j(n) = \begin{cases} W_{SIN_LEFT,N_s}(n), & \text{для } 0 \le n < N_s/2 \\ W_{SIN_RIGHT,N_s}(n), & \text{для } N_s/2 \le n < N_s \end{cases}, \; 0 < j < M$$

$$W_0(n) = \begin{cases} W_{LEFT,N_s}(n), & \text{для } 0 \le n < N_s/2 \\ W_{SIN_RIGHT,N_s}(n), & \text{для } N_s/2 \le n < N_s \end{cases}$$

$$W_j(n) = \begin{cases} W_{SIN_LEFT,N_s}(n), & \text{для } 0 \le n < N_s/2 \\ W_{SIN_RIGHT,N_s}(n), & \text{для } N_s/2 \le n < N_s \end{cases}, \; 0 < j < M.$$

Перекрытие и добавление между *EIGHT_SHORT_SEQUENCE*, приводящие к оконным значениям временной области $z_{i,n}$, описывается следующим образом:

$$W(n) = \begin{cases} 0, \text{ для} & 0 \le n < N_l - N - s/4 \\ x_{0,n-\frac{N_l-N_s}{4}} W_0\left(n - \frac{N_l-N_s}{4}\right), \text{ для} & \frac{N_l-N_s}{4} \le n < \frac{N_l-N_s}{4} \\ x_{j-1,n-\frac{N_l-(2j-3)N_s}{4}} W_{j-1}\left(n - \frac{N_l-(2j-3)N_s}{4}\right) + & \\ x_{j,n-\frac{N_l-(2j-3)N_s}{4}} W_j\left(n - \frac{N_l-(2j-3)N_s}{4}\right), \text{ для} & 1 \le j < M, \\ & \frac{N_l+(2j-1)N_s}{4} \le n < \frac{N_l+(2j+1)N_s}{4} \\ x_{M-1,n-\frac{N_l-(2M-3)N_s}{4}} W_{M-1}\left(n - \frac{N_l-(2M-3)N_s}{4}\right) & \frac{N_l+(2M-1)N_s}{4} \le n < \frac{N_l+(2M+1)N_s}{4} \\ 0, \text{ для} & \frac{N_l+(2M+1)N_s}{4} \le n < N_l; \end{cases}$$

г) *LONG_STOP_SEQUENCE:*

window_sequence необходима для переключения из *EIGHT_SHORT_SEQUENCE* обратно в *ONLY_LONG_SEQUENCE.*

Если *window_shape* == 1, окно для *LONG_STOP_SEQUENCE* дается следующим образом:

$$W(n) = \begin{cases} 0,0, \text{ для} & 0 \le n < \frac{N_l-N_s}{4} \\ W_{LEFT,N_s}\left(n - \frac{N_l-N_s}{4}\right), \text{ для} & \frac{N_l-N_s}{4} \le n < \frac{N_l+N_s}{4} \\ 1,0, \text{ для} & \frac{N_l+N_s}{4} \le n < \frac{N_l}{2} \\ W_{KBD_RIGFT,N_l}(n), \text{ для} & N_l/2 \le n < N_l. \end{cases}$$

Если *window_shape* == 0 окно для *LONG_START_SEQUENCE* определяется с помощью:

$$W(n) = \begin{cases} 0,0, \text{ для} & 0 \le n < \frac{N_l-N_s}{4} \\ W_{LEFT,N_s}\left(n - \frac{N_l-N_s}{4}\right), \text{ для} & \frac{N_l-N_s}{4} \le n < \frac{N_l+N_s}{4} \\ 1,0, \text{ для} & \frac{N_l+N_s}{4} \le n < \frac{N_l}{2} \\ W_{SIN_RIGFT,N_l}(n), \text{ для} & N_l/2 \le n < N_l. \end{cases}$$

Оконные значения временной области могут быть вычислены с помощью формулы, которая объясняется в а).

6.11.3.3 Наложение и добавление к предыдущей последовательностью окон

Помимо наложения и добавления в пределах *EIGHT_SHORT_SEQUENCE*, первая (левая) половина каждой *window_sequence* перекрывается и добавляется к второй (правой) половине предыдущей *window_sequence*, получая в заключительном временном интервале значения $out_{i,n}$. Математическое выражение для этой работы может быть описано следующим образом:

$$out_{i,n} = z_{i,n} + z_{i-1,\, n+N/2} \quad \text{для} \quad 0 \le n < N/2,\ N = 2048\ (1920).$$

Это допустимо для всех четырех возможных *window_sequences.*

6.12 Управление усилением

6.12.1 Описание инструмента

Инструмент управления усилением составляется из нескольких компенсаторов усиления, этапов обработки наложения/добавления и этапа *IPQF* (*Inverse Polyphase Quadrature Filter* (инверсный полифазный квадратурный фильтр)). Этот инструмент получает неперекрывающиеся последовательности сигналов, обеспеченные этапами *IMDCT*, *window_sequence* и *gain_control_data*, и затем воспроизводит выходные данные *PCM.*

Благодаря характеристикам блока фильтров *PQF* порядок коэффициентов *MDCT* в каждой нечетной полосе *PQF* должен быть инвертирован. Это делается инвертируя спектральный порядок коэффициентов *MDCT*, то есть обменивая коэффициенты более высокой частоты *MDCT* на коэффициенты более низкой частоты *MDCT.*

Если используется инструмент управления усилением, конфигурация инструмента блока фильтров изменяется следующим образом. В случае последовательности окон *window_sequence* типа *EIGHT_SHORT_SEQUENCE* число коэффициентов для *IMDCT* равно 32 вместо 128 и выполняются восемь *IMDCTs.* В случае других значений *window_sequence* число коэффициентов для *IMDCT* равно 256 вместо 1024 и выполняется один *IMDCT.* Во всех случаях инструмент блока фильтров выводит в общей сложности 2048 неперекрывающихся значений на фрейм. Эти значения предоставляются инструменту управления усилением как $U_{W,B}(j)$, определенные в 6.12.3.3.

IPQF комбинирует четыре универсальных полосы частот и производит декодированный выходной сигнал временной области. Искаженные компоненты, вносимые *PQF* в кодере, отменяются *IPQF.*

Значениями усиления для каждой полосы можно управлять независимо за исключением самой низкой полосы частот. Размер шага управления усилением равен 2^n, где *n* является целым числом.

Инструмент регулировки усиления выводит временную последовательность сигнала, которая является *AS (n)*, определенной в 6.12.3.4.

6.12.2 Определения

Данные управления усилением	Дополнительная информация, указывающая значения усиления и позиции, используемые для изменения усиления.
полоса *IPQF*	Каждая полоса разделения *IPQF*
adjust_num	3-битовое поле, указывающее число изменений усиления для каждой полосы *IPQF.* Максимальное количество изменений усиления равно семи.
max_band	2-битовое поле, указывающее число полос *IPQF*, в которых происходило управление усилением их сигнала. Значения этих величин показаны ниже: 0 — никакие полосы не имеют активированного управления усилением; 1 — управление усилением производилось во 2-ой полосе *IPQF*; 2 — управление усилением производилось во 2-ой и 3-ей полосах *IPQF*; 3 — управление усилением производилось во 2-ой, 3-ей и 4-ой полосах *IPQF*.
Alevcod	4-битовое поле, указывающее величину усиления для одного изменения усиления.
aloccode	2-, 4-, или 5-разрядное поле, указывающее позицию для одного изменения усиления. Длина этих данных изменяется в зависимости от последовательности окон.

6.12.3 Процесс декодирования

Для декодирования требуются следующие четыре процесса.

(1) Декодирование данных управления усилением.

(2) Установка финкции управления усилением.

(3) Работа с окнами и наложение управления усилением.

(4) Фильтр синтеза.

6.12.3.1 Декодирование данных управления усилением

Данные управления усилением восстанавливаются следующим образом.

(1)

$NAD_{W,B}$ = *adjust_num*[*B*][*W*]

(2)

$ALOC_{W,B}(m)$ = *AdjLoc*(*aloccode*[*B*][*W*][*m* – 1], $1 \le m < NAD_{W,B}$

$ALEV_{W,B}(m) = 2^{AdjLEV(alevcode[B][W][m-1]}$, $1 \le m < NAD_{W,B}$

(3)

$ALOC_{W,B}(0) = 0$

$$ALEV_{W,B}(0) = \begin{cases} 1, \text{ если } NAD_{W,B} = 0 \\ ALEV_{W,B}(1), \text{ в других случаях} \end{cases}$$

(4)

$$ALEV_{W,B}(0) = \begin{cases} 256, W = 0 \text{ если } ONLY_LONG_SEQUENCE \\ 112, W = 0 \text{ если } ONLY_START_SEQUENCE \\ 32, W = 1 \text{ если } ONLY_START_SEQUENCE \\ 32, 0 \le W \le 7 \text{ если } EIGHT_SHORT_SEQUENCE \\ 112, W = 0 \text{ если } LONG_STOP_SEQUENCE \\ 256, W = 1 \text{ если } LONG_STOP_SEQUENCE, \end{cases}$$

где

$NAD_{W,B}$ — номер информации об управлении усилением, целое число;

$ALOC_{W,B}(m)$ — расположение управления усилением, целое число;

$ALEV_{W,B}(m)$ — уровень управления усилением, целочисленное действительное число;

B — ID полосы, целое число от 1 до 3;

W — ID окна, целое число от 0 до 7;

m — целое число.

aloccode [B] [W] [m] должен быть установлен так, чтобы $\{ALOC_{W,B}(m)\}$ удовлетворил следующим условиям:

$$ALOC_{W,B}(m_1) < ALOC_{W,B}(m_2), \quad 1 \le m_1 < m_2 \le NAD_{W,B} + 1.$$

В случаях *LONG_START_SEQUENCE* и *LONG_STOP_SEQUENCE* значения 14 и 15 для *aloccode [B] [0] [m]* недопустимы. *AdjLoc* () определяется в таблице 162. *AdjLev* () определяется в таблице 163.

6.12.3.2 Установка функции управления усилением

Функция управления усилением получается следующим образом.

(1)

$M_{W,B,j}$ = Max{*m*: $ALOC_{W,B}(m) < j$},

$0 \le j \le 255$, *W*==0 если *ONLY_LONG_SEQUENCE*

$0 \le j \le 111$, *W*==0 если *ONLY_START_SEQUENCE*

$0 \le j \le 31$, *W*==1 если *ONLY_START_SEQUENCE*

$0 \le j \le 31$ $0 \le W \le 7$ если *EIGHT_SHORT_SEQUENCE*

$0 \le j \le 111$, *W*==0 если *LONG_STOP_SEQUENCE*

$0 \le j \le 255$, *W*==1 если *LONG_STOP_SEQUENCE*

(2)

$$FMD_{W,B}(j) = \begin{cases} Inter \begin{pmatrix} ALEV_{W,B}(M_{W,B,j}), \\ ALEV_{W,B}(M_{W,B,j} + 1), \\ j - ALOC_{W,B}(M_{W,B,j}) \end{pmatrix}, \text{ если } ALOC_{W,B}(M_{W,B,j}) \le j \le ALOC_{W,B}(M_{W,B,j}) + 7 \\ ALEV_{W,B}(M_{W,B,j} + 1), \text{ в других случаях} \end{cases}$$

(3)
если *ONLY_LONG_SEQUENCE*

$$GMF_{0,B}(j) = \begin{cases} ALEV_{0,B}(0) \text{x} PFMD_B(j), \text{ если } 0 \le j \le 255 \\ FMD_{0,B}(j-256), \text{ если } 256 \le j \le 511 \end{cases}$$

$PFMD_B(j) = FMD_{0,B}(j)$, если $256 \le j \le 511$
если *LONG_STAR_SEQUENCE*

$$GMF_{0,B}(j) = \begin{cases} ALEV_{0,B}(0) \text{x} ALEV_{1,B}(0) \text{x} PFMD_B(j), \text{ если } 0 \le j \le 255 \\ ALEV_{1,B}(0) \text{x} FMD_{0,B}(j-256), \text{ если } 256 \le j \le 367 \\ FMD_{1,B}(j-368), \text{ если } 368 \le j \le 399 \\ 1, \text{ если } 400 \le j \le 511 \end{cases}$$

$PFMD_B(j) = FMD_{1,B}(j)$, если $0 \le j \le 31$
если *EIGHT_SHORT_SEQUENCE*

$$GMF_{W,B}(j) = \begin{cases} ALEV_{W,B}(0) \text{x} PFMD_B(j), \text{ если } W = 0,\ 0 \le j \le 31 \\ ALEV_{W,B}(0) \text{x} FMD_{W-1,B}(j), \text{ если } 1 \le W \le 7,\ 256 \le j \le 367 \\ FMD_{W,B}(j-32), \text{ если } 1 \le W \le 7,\ 32 \le j \le 63 \end{cases}$$

$PFMD_B(j) = FMD_{7,B}(j)$, если $0 \le j \le 31$
если *EIGHT_SHORT_SEQUENCE*

$$GMF_{0,B}(j) = \begin{cases} 1, \text{ если } 0 \le j \le 111 \\ ALEV_{0,B}(0) \text{x} ALEV_{1,B}(0) \text{x} PFMD_B(j-112), \text{ если } 112 \le j \le 143 \\ ALEV_{1,B}(0) \text{x} FMD_{0,B}(j-114), \text{ если } 114 \le j \le 255 \\ FMD_{1,B}(j-256), \text{ если } 256 \le j \le 511 \end{cases}$$

$PFMD_B(j) = FMD_{1,B}(j)$, если $0 \le j \le 255$
(4)

$$AD_{W,B}(j) = \frac{1}{GMF_{W,B}(j)},$$

$0 \le j \le 511$, W==0 если *ONLY_LONG_SEQUENCE*
$0 \le j \le 511$, W==0 если *LONG_START_SEQUENCE*
$0 \le j \le 63$ $0 \le W \le 7$ если *EIGHT_SHORT_SEQUENCE*
$0 \le j \le 511$, W==0 если *LONG_STOP_SEQUENCE*,
где:

$FMD_{W,B}(j)$ — функция модификации фрагмента, действительное число;
$PFMD_B(j)$ — функция модификации фрагмента предыдущего фрейма, действительное число;
$GMF_{W,B}(j)$ — функция модификации усиления, действительное число;
$AD_{W,B}(j)$ — функция управления усилением, действительное число;
$ALOC_{W,B}(m)$ — расположение управления усилением, определенное в 6.12.3.1, целое число;
$ALEV_{W,B}(m)$ — уровень управления усилением, определенный в 6.12.3.1, целочисленное действительное число;
B — ID полосы, целое число от 1 до 3;
W — ID окна, целое число от 0 до 7;
$M_{W,B,j}$ — целое число;
m — целое число.

и

$$Inter(a,b,j) = 2^{\frac{(8-j)\log_2(b)}{8}}.$$

Начальное значение $PFMD_B(j)$ должно быть установлено в 1,0.

6.12.3.3 Работа с окнами управления усилением и перекрытие

Данные выборки полосы получаются посредством процессов (1) и (2), показанных ниже.

(1) Работа с окнами управления усилением:

если B=0

$T_{W,B}(j) = U_{W,B}(j)$,

$0 \le j \le 511$, W==0 если *ONLY_LONG_SEQUENCE*

$0 \le j \le 511$, W==0 если *LONG_START_SEQUENCE*

$0 \le j \le 63$ $0 \le W \le 7$ если *EIGHT_SHORT_SEQUENCE*

$0 \le j \le 511$, W==0 если *LONG_STOP_SEQUENCE*,

тогда

$T_{W,B}(j) = AD_{W,B}(j) \times U_{W,B}(j)$,

$0 \le j \le 511$, W==0 если *ONLY_LONG_SEQUENCE*

$0 \le j \le 511$, W==0 если *LONG_START_SEQUENCE*

$0 \le j \le 63$ $0 \le W \le 7$ если *EIGHT_SHORT_SEQUENCE*

$0 \le j \le 511$, W==0 если *LONG_STOP_SEQUENCE*,

(2) Наложение

если *ONLY_ LONG_SEQUENCE*

$V_B(j) = PT_B(j) + T_{0,B}(j)$, если $0 \le j \le 255$

$PT_B(j) = T_{0,B}(j + 256)$, если $0 \le j \le 255$;

если *LONG_START_SEQUENCE*

$V_B(j) = PT_B(j) + T_{0,B}(j)$, если $0 \le j \le 255$

$V_B(j + 255) = T_{0,B}(j + 256)$, если $0 \le j \le 111$

$PT_B(j) = T_{0,B}(j + 368)$, если $0 \le j \le 31$;

если *EIGHT_SHORT_SEQUENCE*

$V_B(j) = PT_B(j) + T_{W,B}(j)$, W=0, $0 \le j \le 31$

$V_B(32W + j) = T_{W-1,B}(j + 32) + T_{W,B}(j)$, $1 \le W \le 7$, $0 \le j \le 31$

$PT_B(j) = T_{0,B}(j + 32)$, W=7, $0 \le j \le 31$;

если *LONG_STOP_SEQUENCE*

$V_B(j) = PT_B(j) + T_{0,B}(j+112)$, если $0 \le j \le 31$

$V_B(j + 32) = T_{0,B}(j + 114)$, если $0 \le j \le 111$

$PT_B(j) = T_{0,B}(j + 256)$, если $0 \le j \le 255$,

где:

$U_{W,B}(j)$ — данные спектра полосы, действительное число;

$T_{W,B}(j)$ — выборочные данные управляемого блока усиления, действительное число;

$PT_B(j)$ — выборочные данные управляемого блока усиления предыдущего фрейма, действительное число;

$V_B(j)$ — выборочные данные полосы, действительное число;

$AD_{W,B}(j)$ — функция управления усилением определенная в 6.12.3.2, действительное число B;

B — ID полосы, целое число от 0 до 3;

W — ID окна, целое число от 0 до 7;

J — целое число.

Начальное значение $PT_B(j)$ должно быть установлено в 0,0.

6.12.3.4 Фильтр синтеза

Данные аудиосэмпла получаются из следующих уравнений.

(1)

$$\tilde{V}_B(j) = \begin{cases} V_B(k), \text{если } j == 4k \\ 0 \end{cases}, \quad 0 \le B \le 3$$

(2)

$$Q_B(j) = Q(j) \times \cos\left(\frac{(2B+1)(2j-3)\pi}{16}\right), \quad 0 \le j \le 95, \quad 0 \le B \le 3$$

(3)

$$AS(n) = \sum_{B=0}^{3} \sum_{j=0}^{95} Q_B(j) \times \tilde{V}_B(n-j),$$

где:

$AS(n)$ — данные аудиосэмпла;

$V_B(n)$ — демонстрационные данные полосы, определенные в 6.12.3.3, действительное число;

$\tilde{V}_B(j)$ — интерполированные демонстрационные данные полосы, действительное число;

$Q_B(j)$ — коэффициенты фильтра синтеза, действительное число;

$Q(j)$ — коэффициенты прототипа, данные ниже, действительное число;

B — ID полосы, целое число от 0 до 3;

W — ID окна, целое число от 0 до 7;

n — целое число;

j — целое число;

k — целое число.

Значения от Q (0) до Q (47) показаны в таблице 164. Значения от Q (48) до Q (95) получаются из следующего уравнения.

$$Q(j) = Q(95-j),\ 48 \le j \le 95$$

6.12.4 Таблицы

Таблица 162 — *AdjLoc ()*

AC	*AdjLoc(AC)*	*AC*	*AdjLoc(AC)*
0	0	16	128
1	8	17	136
2	16	18	144
3	24	19	152
4	32	20	160
5	40	21	168
6	48	22	176
7	56	23	184
8	64	24	192
9	72	25	200
10	80	26	208
11	88	27	216
12	96	28	224
13	104	29	232
14	112	30	240
15	120	31	248

Таблица 163 — *AdjLew ()*

AV	*AdjLev(AV)*	*AV*	*AdjLev(AV)*
0	-4	8	4
1	-3	9	5
2	-2	10	6
3	-1	11	7
4	0	12	8
5	1	13	9
6	2	14	10
7	3	15	11

Таблица 164 — *Q ()*

j	*Q(j)*	*j*	*Q(j)*
0	9,7655291007575512E–05	24	–2,2656858741499447E–02
1	1,3809589379038567E–04	25	–6,8031113858963354E–03
2	9,8400749256623534E–05	26	1,5085400948280744E–02
3	–8,6671544782335723E–05	27	3,9750993388272739E–02
4	–4,6217998911921346E–04	28	6,2445363629436743E–02
5	–1,0211814095158174E–03	29	7,7622327748721326E–02
6	–1,6772149340010668E–03	30	7,9968338496132926E–02
7	–2,2533338951411081E–03	31	6,5615493068475583E–02
8	–2,4987888343213967E–03	32	3,3313658300882690E–02
9	–2,1390815966761882E–03	33	–1,4691563058190206E–02
10	–9,5595397454597772E–04	34	–7,2307890475334147E–02
11	1,1172111530118943E–03	35	–1,2993222541703875E–01
12	3,9091309127348584E–03	36	–1,7551641029040532E–01
13	6,9635703420118673E–03	37	–1,9626543957670528E–01
14	9,5595442159478339E–03	38	–1,8073330670215029E–01
15	1,0815766540021360E–02	39	–1,2097653136035738E–01
16	9,8770514991715300E–03	40	–1,4377370758549035E–02
17	6,1562567291327357E–03	41	1,3522730742860303E–01
18	–4,1793946063629710E–04	42	3,1737852699301633E–01
19	–9,2128743097707640E–03	43	5,1590021798482233E–01
20	–1,8830775873369020E–02	44	7,1080020379761377E–01
21	–2,7226498457701823E–02	45	8,8090632488444798E–01
22	–3,2022840857588906E–02	46	1,0068321641150089E+00
23	–3,0996332527754609E–02	47	1,0737914947736096E+00

6.13 Перцепционная шумовая замена (*PNS*)

6.13.1 Описание инструмента

Этот инструмент используется, чтобы реализовать кодирование замены перцепционного шума в пределах *ICS*. Таким образом, определенные наборы спектральных коэффициентов получают из случайных векторов, а не из кодированных по Хаффману символов и процесса инверсного квантования. Это делается выборочно на базе полосы масштабного коэффициента и группы, когда перцепционная шумовая замена отмечается как активная.

PNS также может быть использован в сочетании с *bit sliced* арифметическим кодированием. В то время как синтаксис полезной нагрузки потока битов там отличается, процесс декодирования является тем же самым.

6.13.2 Определения

hcod_sf []	Кодовая комбинация Хаффмана из таблицы кодов Хаффмана, используемая для кодирования масштабных коэффициентов.
dpcm_noise_nrg [] []	Дифференцированно закодированная шумовая энергия.
noise_nrg [группа] [sfb]	Шумовая энергия для каждой группы и полосы масштабного коэффициента.
spec []	Массив, содержащий спектр соответствующего канала.

6.13.3 Процесс декодирования

Об использовании инструмента замены перцепционного шума сообщается при помощи использования псевдо сборника кодов *NOISE_HCB* (13).

Если та же самая полоса масштабного коэффициента и группа кодируются перцепционной шумовой заменой в обоих каналах из пары каналов, корреляцией шумового сигнала можно управлять посредством поля *ms_used*. В то время как процесс генерации шума по умолчанию работает независимо для каждого канала (отдельная генерация случайных векторов), тот же самый случайный вектор используется для обоих каналов, если *ms_used* [] устанавливается для определенной полосы масштабного коэффициента и группы или *ms_mask_present* устанавливается в '10'. В этом случае никакое кодирование стерео *M/S* не выполняется (потому что кодирование стерео *M/S* и кодирование замены шума являются взаимоисключающими). Если та же самая полоса масштабного коэффициента и группа кодируются перцепционной шумовой заменой только в одном канале пары каналов, установка *ms_used []* не учитывается.

Информация об энергии для декодирования перцепционной шумовой замены представляется величиной шумовой энергии, указывающей полную энергию заменяющих спектральных коэффициентов с шагом 1,5 дБ. Если кодирование шумовой замены является активным для определенной группы и полосы масштабного коэффициента, вместо масштабного коэффициента соответствующего канала передается величина шумовой энергии.

Шумовые энергии кодируются точно так же как масштабные коэффициенты, то есть методом кодирования Хаффмана дифференциальных значений:

- стартовое значение для декодирования *DPCM* дается *global_gain*;
- дифференциальное декодирование производится отдельно между масштабными коэффициентами, позициями интенсивности стерео и энергиями шума. Иначе говоря, декодер шумовой энергии игнорирует значения вставленных масштабных коэффициентов и позиции интенсивности стерео и наоборот.

Для кодирования шумовых энергий используется тот же самый сборник кодов, что и для кодирования масштабных коэффициентов.

Для использования в декодировании замены перцепционного шума определяется одна псевдофункция:

```
function is_noise(group,sfb) {
1   for window groups / scalefactor bands with
      codebook sfb_cb[group][sfb] == NOISE_HCB
0       otherwise
}.
```

Процесс декодирования шумовой замены для одного канала определяется следующим псевдокодом:

```
nrg = global_gain - NOISE_OFFSET - 256;
for (g=0; g<num_window_groups; g++) {
/* Decode noise energies for this group */
for (sfb=0; sfb<max_sfb; sfb++) {
if (is_noise(g,sfb)) {
```

```
nrg += dpcm_noise_nrg[g][sfb];
noise_nrg[g][sfb] = nrg;
}
}
/* Do perceptual noise substitution decoding */
for (b=0; b<window_group_length[g]; b++) {
for (sfb=0; sfb<max_sfb; sfb++)  {
if (is_noise(g,sfb)) {
size = swb_offset[sfb+1] - swb_offset[sfb];
/* Generate random vector */
gen_rand_vector( &spec[g][b][sfb][0], size );
nrg=0;
for (i=0; i<size; i++) {
nrg+= spec[g][b][sfb][i] * spec[g][b][sfb][i];
}
sqrt_nrg = sqrt (nrg);
scale *= 2,0^(0,25*noise_nrg [g][sfb]) / sqrt_nrg;
/* scale random vector to desired target energy */
for (i=0; i<size; i++) {
spec[g][b][sfb][i] *= scale;
}
}
}
}
}.
```

Чтобы адаптировать диапазон величин среднего значения энергии шумов к обычному диапазону масштабных коэффициентов, используется постоянная *NOISE_OFFSET* и она имеет значение 90.

Функция *gen_rand_vector* (*addr, size*) генерирует вектор длины <*size*> со случайными значениями со знаком, тогда как их сумма квадратов не равна нулю. Генератор подходящих случайных чисел может быть реализован, используя одно умножение/накопление на случайное значение.

В случае обратимого кодирования переменной длины (*RVLC*) стартовое значение для обратного декодирования *DPCM* задается *reversible_global_gain*. Декодирование шумовых энергий определяется следующим псевдокодом:

```
nrg = rev_global_gain-NOISE_OFFSET-256+dpcm_noise_last_position;
for (g = win-1; g >= 0; g--) {
for (sfb = sfbmax-1; sfb >= 0; sfb--) {
noise_nrg[g][sfb]=nrg;
nrg -= dpcm_noise_nrg[g][sfb];
}
}.
```

6.13.4 Интеграция с инструментами внутриканального прогноза

Для полос масштабного коэффициента, кодированных с использованием *PNS*, соответствующие прогнозирующие устройства переключаются в "выключено", таким образом эффективно переопределяя состояние, заданное маской *prediction_used*. Для полос масштабного коэффициента, кодированных перцепционной шумовой заменой, прогнозирующие устройства, принадлежащие соответствующим спектральным коэффициентам, сбрасываются. Обновление этих прогнозирующих устройств выполняется подачей значения нуля как "последнее квантованное значение" $x_{rec}(n-1)$.

Если и долгосрочный прогноз и *PNS* являются активными для определенной полосы масштабного коэффициента и группы, *PNS* получает приоритет, то есть спектральные коэффициенты в этой полосе масштабного коэффициента вырабатываются только инструментом *PNS*.

6.13.5 Интеграция с другими инструментами *AAC*

Имеют место следующие взаимодействия между инструментом перцепционной шумовой замены и другими инструментами *AAC*:

- определение нового номера сборника псевдокодов Хаффмана *NOISE_HCB* = 13;
- во время декодирования квантованных спектральных коэффициентов по Хаффману таблица сборника кодов Хаффмана *NOISE_HCB* обрабатывается точно так же, как нулевой сборник кодов *ZERO_HCB*,

то есть никакие кодовые комбинации Хаффмана не читаются для соответствующей полосы масштабного коэффициента и группы;

- если та же самая полоса масштабного коэффициента и группа кодируются перцепционной шумовой заменой в обоих каналах канальной пары, никакое декодирование стерео *M/S* для этой полосы масштабного коэффициента и группы не выполняется;

- псевдошумовые компоненты, сгенерированные инструментом перцепционно шумовой замены, вводятся в выходной спектр до шага обработки временного формирования шума (*TNS*).

6.13.6 Интеграция в масштабируемый кодер на базе *AAC* (масштабируемый *AAC* типа *AudioObjectType*)

Для использование инструмента перцепционной шумовой замены в масштабируемом кодере на основе *AAC* применяют следующие правила:

- если определенная полоса масштабного коэффициента кодируется перцепционной шумовой заменой в уровне *N*, это вносит вклад в выходной спектр объединенных уровней *N* и *N*+1, только если выполняются все следующие требования:

а) оба уровня *N* и *N*+1 являются уровнями или моно, или стерео;

б) уровень *N*+1 не использует стерео интенсивности в этой полосе масштабного коэффициента;

в) уровень *N*+1 не использует *PNS* в этой полосе масштабного коэффициента;

г) все спектральные коэффициенты уровня *N*+1 в этой полосе масштабного коэффициента декодируются в ноль. В случае *M/S* кодирования это требуется для обоих каналов элемента пары каналов;

- если определенная полоса масштабного коэффициента и группа кодируются перцепционной шумовой заменой в обоих каналах пары каналов, верхние уровни (расширения) все еще могут использовать флаг стерео *M/S ms_used [] []*, чтобы сигнализировать об использовании декодирования стерео *M/S*.

6.14 Модуль частотно-избирательного переключателя (*FSS*)

Блок частотно-избирательного коммутатора (*FSS*) используется в различных конфигурациях масштабируемого кодера. Он состоит из банка коммутаторов, который имеет функцию выбора одного из двух входных сигналов, независимо для каждой полосы масштабного коэффициента (*sfb*). Для каждой *sfb* доступен бит управления, который управляет выбором. Форма передачи этих битов управления отличается в зависимости от конфигурации, в которой используется модуль *FSS*.

6.14.1 *FSS* в объединенных *TwinVQ/CELP-AAC* системах

6.14.1.1 Определения

dc_group — Четыре последовательных полосы масштабного коэффициента, если тип окна не *SHORT_WINDOW*. Одна полоса *diff_short_lines*, если тип окна является *SHORT_WINDOW*.

no_of_dc_groups — Если тип окна не является *SHORT_WINDOW*, число групп в зависимости от частоты дискретизации дается в таблице 131. Если тип окна является *SHORT_WINDOW*, *no_of_dc_groups* равен '1'.

diff_short_lines — Используется только в случае, если тип окна является *SHORT_WINDOW*. Число линий спектра в единственном *dc_group* на окно в зависимости от частоты дискретизации, дается в таблице 131.

diff_control_sfb [w] [sfb] — Применяется только в случае, если тип окна не является *SHORT_WINDOW*. Они являются декодируемыми величинами *diff_control [w] [dc_group]*. Для каждой коммутируемой полосы масштабного коэффициента доступен один бит управления.

6.14.1.2 Декодирование для комбинации *CELP - AAC*

В полезной нагрузке потока битов *diff_control [w] [dc_group]* используется для того, чтобы передать закодированные по Хаффману значения *diff_control_sfb [w] [sfb]* согласно следующей таблице:

Таблица 165 — Таблица кода Хаффмана для *diff_control [w] [dc_group]*

Индекс	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Код	0	20	21	22	23	24	25	8	9	26	27	28	29	30	31	1
Длина	2	5	5	5	5	5	5	4	4	5	5	5	5	5	5	2

6.14.1.3 *FSS* для комбинации *TwinVQ - AAC*

То же самое кодирование методом Хаффмана значений *diff_control_sfb [w] [sfb]*, которое определяется для комбинации *CELP - AAC*, также используется для объединенных систем *TwinVQ - AAC*. Однако значение *no_of_dc_groups* вычисляется исходя из значения *max_sfb* последнего уровня *TwinVQ* следующим образом:

no_of_dc_groups = int ((max_sfb + 3) / 4).

Значение для *diff_short_lines* берется из таблицы полосы масштабного коэффициента для *SHORT_WINDOW* соответствующей частоты дискретизации:

diff_short_lines = swb_offset [max_sfb].

6.14.1.4 Общие процессы декодирования

Декодирование, если тип окна не является *SHORT_WINDOW:*

после хаффмановского декодирования *diff_control [w] [dc_group]* из полезной нагрузки потока битов, массив *diff_control_sfb [w] [sfb]* генерируется согласно:

```
if ( ! SHORT_WINDOW ) {
dc_group = 0;
while (dc_group < no_of_ dc_groups) {
for (i = 0; i < 4; i++) {
diff_control_sfb[0][dc_group*4+i] = diff_control[0][dc_group]  & 0x8;
diff_control[0][dc_group] <<= 1;
}
dc_group++;
}
}.
```

Для всех полос масштабного коэффициента, которые не получили значение, назначенное в *diff_control_sfb [w] [sfb]* в вышеупомянутой процедуре, *diff_control_sfb [w] [sfb]* устанавливается в '1';

Переключение для всех полос масштабного коэффициента выполняется согласно:

```
if (diff_control_sfb[w][sfb] == 0) {
spectrum_out[w][sfb] = spectrum_AAC[w][sfb] + spectrum_Celp/TwinVQ[w][sfb];
} else {
spectrum_out[w][sfb] = spectrum_AAC[w][sfb];
}.
```

Декодирование, если тип окна является *SHORT_WINDOW*:

если тип окна является *SHORT_WINDOW*, есть только одна полоса *diff_short_lines* на окно где применяется механизм управления разностью:

для линий спектра от 0 до *diff_short_lines*-1:

```
if (diff_control_sfb[w][0] == 0) {
spectrum_out[w] = spectrum_AAC [w] + spectrum_Celp/TwinVQ[w];
} else {
spectrum_out[w] = spectrum_AAC[w];
};
```

для остающихся линий вывод переключателя идентичен вводу:

spectrum_out[w] = spectrum_AAC [w].

6.14.2 *FSS* в объединенной, масштабируемой конфигурации моно/стерео

6.14.2.1 Процесс декодирования

В объединенном кодере моностерео, где сигнал моно, который получается из ввода стерео, кодируется с одним или более уровнями моно и позже кодируется с одним или более уровнями стерео, инструмент *FSS* также используется, чтобы управлять добавлением выходного сигнала объединенных этапов *M*-кодирования в сигналы левого (*L*) или правого (*R*) каналов этапов кодирования стерео. В этом случае число обработанных полос в текущем уровне равно *max_sfb* текущего уровня. Однако, если *last_max_sfb* больше чем *max_sfb* текущего уровня, неиспользованные биты *diff_control_lr [w] [sfb]* сохраняются для последующего уровня. Биты управления для этих модулей *FSS* непосредственно доступны из элементов синтаксиса *diff_control_lr [w] [sfb].* Поскольку сигналы объединенного *L+M* или *R+M* в одной полосе масштабного коэффициента (*sfb*) необходимы только на последующих этапах кодирования текущего уровня, и если *MS* кодирование не выбрано для определенного *sfb*, *diff_control_lr [w] [sfb]* передается только для *sfb*, для которого *MS* кодирование не выбирается.

Декодирование, если тип окна не является *SHORT_WINDOW*:

для всех полос масштабного коэффициента, для которых не передается никакое значение, *diff_control_lr [w] [sfb]* устанавливается в '1';

переключение для всех полос масштабного коэффициента производится согласно:

if (diff_control_lr[w][sfb] == 0) {

*spectrum_L/R_out[w][sfb] = spectrum_L/R[w][sfb] + 2 * spectrum_M[w][sfb];*

} else {

spectrum_L/R_out[w][sfb] = spectrum_L/R[w][sfb];

};

декодирование, если тип окна является *SHORT_WINDOW:*

если тип окна является *SHORT_WINDOW*, значение *diff_control_lr [win] [0]* используется для всех полос масштабного коэффициента от 0 до *last_max_sfb*-1. Для всех других полос используется значение '1'.

6.15 Инструмент фильтра повышенной дискретизации

6.15.1 Описание инструмента

Инструмент фильтра повышения дискретизации используется, чтобы адаптировать частоту дискретизации базового кодера (*CELP*) к частоте дискретизации кодера времени/частоты. Фильтр повышения дискретизации использует блок фильтров *MDCT* кодера *AAC*. Этот блок фильтров очень похож на блок фильтров *IMDCT*, который используется в декодере. Они оба используют те же самые функции окна.

Блок фильтров берет блок временных выборок вывода базового кодера и вставляет соответствующее число нулей между этими выборками, чтобы генерировать сигнал на требующейся более высокой частоте дискретизации. Эти значения повышенной дискретизации затем задерживаются на число выборок, заданных элементом данных *coreCoderDelay* в *GASpecificConfig ()* первого уровня расширения, и затем модулируются той же функцией окна, которая используется для *IMDCT* блока. Используются тип окна и форма окна из *IMDCT*. Чтобы сохранить *RAM* в декодере, также возможно задержать полезную нагрузку базового потока битов на соответствующее число базовых фреймов вместо того, чтобы задерживать повышенно дискретизированный базовый сигнал. Каждый блок входных выборок перекрывается на 50 % непосредственно предшествующим блоком. Длина преобразованного входного блока N устанавливается или в 2048 (1920) или в 256 (240) выборок в зависимости от значения *frameLengthFlag*.

Вывод блока фильтров *MDCT* соединяется с модулем *FSS*, который использует только выходные значения в полосах *FSS*. Так как верхняя полоса *FSS* не превышает половины нижней частоты дискретизации, искажающие эффекты пропускаются.

6.15.2 Определения

up-sampling-factor — Отношение частоты дискретизации кодера *T/F* и частоты дискретизации базового кодера.

$x_{in\text{-}mdct\text{-}core}\ [i]$ — Временное поле данных, которое используется, чтобы хранить ввод в дискретизированный блок фильтров *MDCT*.

$x_{out\text{-}core}\ [i]$ — Выходные выборки базового декодера.

6.15.3 Процесс декодирования

Длина аналитического окна *N* для преобразования является функцией элемента синтаксиса *window_sequence* и алгоритмического контекста. Она получается идентичным способом описанным для инструмента *Filterbank* и *Blockswitching*.

6.15.3.1 Повышенная дискретизация вставкой нулей

Ввод в блок фильтров сгенерирован:

$x_{in\text{-}mdct\text{-}core}\ [k] = 0$ для $k = [0: N/2\text{-}1]$

$x_{in\text{-}mdct\text{-}core}\ [up\text{-}sampling\text{-}factor*i] = x_{out\text{-}core}\ [i]$ * " up-sampling-factor ", для $i = [0: N/2/up\text{-}sampling\text{-}factor\text{-}1]$.

6.15.3.2 Работа с окнами и переключение блока

Адаптация разрешающей способности частото-временной части блока фильтров выполняется путем смещения между преобразованиями, чьи входные длины являются 2048 (1920) или 256 (240) выборками, синхронно к декодеру блока фильтров *IMDCT*. Выбор между парами 2048/256 или 1920/240 делается в зависимости от значения *frameLengthFlag*.

Оконные значения временного интервала могут быть вычислены при использовании тех же самых окон *w (n)* как определено для блока фильтров *IMDCT*.

Оконные коэффициенты вычисляются

$$z_{i,n} = w(n)\, x'_{in_mdct_core}(n).$$

6.15.3.3 *MDCT*

Оконные коэффициенты преобразовываются в частотную область с помощью *MDCT*. Спектральный коэффициент *MDCT* $X_{i,k}$ определяется следующим образом:

$$x_{i,k} = 2\sum_{n=0}^{N-1} z_{i,n} \cos\left(\frac{2\pi}{N}\left(n + n_0\right)\left(k + \frac{1}{2}\right)\right), \text{ для } 0 \le k \le N/2,$$

где:

Z_{in} — оконная входная последовательность;

n — индекс выборки;

k — индекс спектрального коэффициента;

i — индекс блока;

N — длина окна одного преобразования, основанная на значении *window_sequence;*

$n_0 = (N/2+1) / 2$.

Только выходные значения от 0 до *N*/2/*up-sampling-factor*-1 могут использоваться без искажения от помех дискретизации. Это обеспечивается следующим модулем *FSS*.

6.16 Инструменты для устойчивости к ошибкам *AAC*

6.16.1 Виртуальные сборники кодов для данных раздела *AAC*

6.16.1.1 Описание инструмента

Виртуальные сборники кодов используются, чтобы ограничить наибольшее абсолютное значение, разрешенное в пределах определенной полосы масштабного коэффициента, где разрешены значения *escape,* то есть, где сборник кодов 11 используется первоначально. Этот инструмент позволяет 17 различных индексов сборника кодов (11, 16... 31) для сборника кодов *escape*. Все эти индексы сборника кодов относятся к сборнику кодов 11. Их поэтому называют виртуальными сборниками кодов. Различием между этими индексами сборника кодов является позволенный максимум спектральных значений, принадлежащих соответствующему разделу. Из-за этого в пределах спектральных данных могут быть расположены ошибки, приводящие к слишком большим спектральным значениям, и соответствующие линии спектра могут быть скрыты.

6.16.1.2 Процесс декодирования

Смотри 5.2.3.2.

6.16.2 *RVLC* для масштабных коэффициентов *AAC*

6.16.2.1 Описание инструмента

RVLC (обратимое кодирование переменной длины) используется вместо кодирования методом Хаффмана, чтобы достигнуть кодирования энтропии масштабных коэффициентов, из-за его лучшей производительности с точки зрения устойчивости к ошибкам. Это можно рассматривать как плагин инструмента бесшумного кодирования, который позволяет декодировать данные кодированного устойчивого к ошибкам масштабного коэффициента.

RVLC включает дополнительное обратное декодирование. Дополнительно возможно некоторое обнаружение ошибок, потому что не все узлы дерева кодирования используются в качестве кодовых комбинаций. Устойчивая к ошибкам работа *RVLC* тем лучше, чем меньше число кодовых комбинаций. Поэтому таблица *RVLC* содержит только значения от – 7 до + 7, тогда как исходный сборник кодов Хаффмана содержит значения от – 60 до + 60. Декодируемое значение ± 7 используется в качестве *ESC_FLAG*. Это сигнализирует о том, что существует значение *escape,* которое должно быть добавлено к + 7 или вычтено из – 7, чтобы найти фактическое значение масштабного коэффициента. Это значение *escape* является закодированным по Хаффману.

Необходимо передать дополнительное значение, чтобы иметь начальную точку для обратного декодирования для закодированных масштабных коэффициентов *DPCM*. Это значение называют реверсивным глобальным усилением. Если используются кодирование стерео интенсивности или *PNS*, для них также необходимы дополнительные значения. Чтобы позволить обратное декодирование, должна быть передана длина части полезной нагрузки потока битов *RVLC*. Длина части полезной нагрузки потока битов, содержащей кодовые комбинации *escape,* должна быть передана, чтобы сохранить синхронизацию в случае битовых ошибок.

6.16.2.2 Определения

Следующие элементы данных доступны в пределах полезной нагрузки потока битов, если *GASpecificConfig ()* отпирает инструмент *RVLC*.

sf_concealment — Поле данных, которое указывает на подобие между масштабными коэффициентами последнего фрейма и таковыми из текущего. Оно должно быть установлено в '0', если масштабные коэффициенты последнего фрейма будут несходными с коэффициентами из текущего фрейма. Оно должно устанавливаться в '1', если они будут подобны.

Примечание — Это поле данных не требуется, чтобы декодировать свободную от ошибок полезную нагрузку, но может использоваться, чтобы применить соответствующие стратегии маскировки в случае поврежденных данных масштабного коэффициента. Никакой критерий подобия не определяется, так как маскировка выходит за область этого стандарта.

rev_global_gain — Содержит последнее значение масштабного коэффициента как стартовое значение для обратного декодирования. Длина этого поля данных составляет 8 битов.

length_of_rvlc_sf — Поле данных, которое содержит длину текущей части данных *RVLC* в битах, включая стартовое значение *DPCM* для *PNS*. Длина этого поля данных зависит от *window_sequence*. Если *window_sequence == EIGHT_SHORT_SEQUENCE*, поле состоит из 11 битов, иначе оно состоит из 9 битов.

rvlc_cod_sf — Слово *RVLC* из таблицы *RVLC*, используемой для кодирования масштабных коэффициентов, позиций интенсивности или шумовой энергии.

sf_escapes_present — Поле данных, которое сигнализирует, есть ли кодированные в полезной нагрузке потока битов *escape* или нет. Длина этих данных составляет 1 бит.

length_of_rvlc_escapes — Поле данных, которое содержит длину части данных *escape* текущего *RVLC* в битах. Длина этих данных составляет 8 битов.

rvlc_esc_sf — Кодовая комбинация Хаффмана из таблицы Хаффмана для значений *RVLC-ESC-values* используется для того, чтобы кодировать значения, большие чем ± 6.

dpcm_is_last_position — Значение *DPCM*, позволяющее обратное декодирование части данных стерео интенсивности. Это симметричное *dpcm_is_position* значение.

dpcm_noise_last_position — Значение *DPCM*, позволяющее обратное декодирование части данных *PNS*. Длина из этих данных 9 битов. Это симметричное *dpcm_noise_n* значение *rg*.

6.16.2.3 Процесс декодирования

Смотри 6.2.3.2.

6.16.2.4 Таблицы

Таблица 166 — Сборник кодов *RVLC*

Индекс	Длина	Кодовая комбинация
–7	7	65
–6	9	257
–5	8	129
–4	6	33
–3	5	17
–2	4	9
–1	3	5
0	1	0
1	3	7
2	5	27
3	6	51
4	7	107
5	8	195
6	9	427
7	7	99

Таблица 167 — Асимметричные (запрещенные) кодовые комбинации

Длина	Кодовая комбинация
6	50
7	96
9	256
8	194
7	98
6	52
9	426
8	212

Таблица 168 — Сборник кодов Хаффмана для значений *RVLC escape*

Индекс	Длина	Кодовая комбинация	Индекс	Длина	Кодовая комбинация
0	2	2	27	20	473482
1	2	0	28	20	473483
2	3	6	29	20	473484
3	3	2	30	20	473485
4	4	14	31	20	473486
5	5	31	32	20	473487
6	5	15	33	20	473488
7	5	13	34	20	473489
8	6	61	35	20	473490
9	6	29	36	20	473491
10	6	25	37	20	473492
11	6	24	38	20	473493
12	7	120	39	20	473494
13	7	56	40	20	473495
14	8	242	41	20	473496
15	8	114	42	20	473497
16	9	486	43	20	473498
17	9	230	44	20	473499
18	10	974	45	20	473500
19	10	463	46	20	473501
20	11	1950	47	20	473502
21	11	1951	48	20	473503
22	11	925	49	19	236736
23	12	1848	50	19	236737
24	14	7399	51	19	236738
25	13	3698	52	19	236739
26	15	14797	53	19	236740

6.16.3 Переупорядочивание кодовой комбинации Хаффмана (*HCR*) для спектральных данных *AAC*

6.16.3.1 Описание инструмента

Алгоритм переупорядочивания кодовой комбинации Хаффмана (*HCR*) для спектральных данных *AAC* основан на том факте, что часть кодовых комбинаций может быть помещена в известные позиции так, чтобы эти кодовые комбинации могли декодироваться независимо от любой ошибки в пределах других кодовых комбинаций. Поэтому этот алгоритм избегает распространения ошибок на эти кодовые комбинации, так называемые приоритетные кодовые комбинации (*PCW*). Чтобы достигнуть этого, определяются сегменты известной длины и эти кодовые комбинации помещаются в начале этих сегментов.

Остающиеся кодовые комбинации (неприоритетные кодовые комбинации, *non-PCW*) заполнены в промежутки, оставленные *PCWs*, использующим специальный алгоритм, который минимизирует распространение ошибок в *non-PCWs* кодовые комбинации.

Этот алгоритм переупорядочения не увеличивает размер спектральных данных.

Прежде, чем применить сам алгоритм переупорядочения, к кодовым комбинациям применяется процесс предварительной сортировки. Он сортирует все кодовые комбинации в зависимости от их важности, то есть он определяет *PCWs*.

6.16.3.2 Определения

Следующие элементы данных доступны в пределах полезной нагрузки потока битов, если *GASpecificConfig ()* включает инструмент *HCR*.

length_of_reordered_spectral_data — 14-разрядное поле данных, которое содержит длину спектральных данных в битах. Максимальное значение равно 6144 в случаях *single_channel_element* (), *coupling_channel_element* () и *lfe_channel_element* () и 12288 в случае *channel_pair_element* (). Большие значения резервируются для будущего использования. Если появляются эти значения, текущие декодеры должны заменить их допустимым максимальным значением.

length_of_longest_codeword — 6-разрядное поле данных, которое содержит длину самой длинной кодовой комбинации, доступной в пределах текущих спектральных данных в битах. Это поле используется, чтобы уменьшить расстояние между защищенными кодовыми комбинациями. Допустимые значения между 0 и 49. Значения между 50 и 63 резервируются для будущего использования. Если появляются эти значения, текущие декодеры должны заменить их на 49.

6.16.3.3 Структура полезной нагрузки потока битов

6.16.3.3.1 Предварительная сортировка

Подпункт 5.2.3.5 не допустим, если используется предварительная сортировка. Вместо этого должна быть применена процедура, описанная в следующих абзацах.

Для объяснения шагов предварительной сортировки вводится термин "модуль". Модуль покрывает четыре линии спектра, то есть две двухмерные кодовые комбинации или одну четырехмерную кодовую комбинацию. В случае двух двухмерных кодовых комбинаций сохраняется их естественный порядок, то есть кодовая комбинация более высокой частоты следует за кодовой комбинацией более низкой частоты.

В случае одного длинного окна (1024 линии спектра на длинный блок, один длинный блок на фрейм) каждое окно содержит 256 модулей. В случае восьми коротких окон (128 линий спектра на короткий блок, восемь коротких блоков на фрейм) каждое окно содержит 32 модуля.

Первый шаг предварительной сортировки.

Модули, представляющие ту же самую часть спектра, собираются вместе во временном порядке и обозначаются как группа модулей. В случае одного длинного окна каждая группа модулей содержит один модуль. В случае восьми коротких окон каждая группа модулей содержит восемь модулей.

Группы модуля собираются в возрастающем спектральном направлении. Для одного длинного окна, которое дает исходный порядок кодового слова, и для восьми коротких окон было применено чередование окон на базе модуля.

При использовании этой схемы кодовые комбинации, представляющие самые низкие частоты, являются первыми кодовыми комбинациями в пределах спектральных данных как для длинных, так и для коротких блоков.

Таблица 169 показывает в качестве примера вывод первого шага предварительной сортировки для коротких блоков, принимая двухмерные сборники кодов для окна 0, 1, 6 и 7 и четырехмерные сборники кодов для окна 2, 3, 4 и 5.

Второй шаг предварительной сортировки.

Чем больше энергии, которую содержит линия спектра, тем более слышимо ее искажение. Энергия в пределах спектральной строки связывается с используемым сборником кодов. Сборники кодов с низкими числами могут представить только низкие значения и позволяют только небольшие ошибки, в то время как сборники кодов с высокими числами могут представлять высокие значения и позволяют большие ошибки.

Поэтому кодовые комбинации предварительно сортируются в зависимости от используемого сборника кодов. Если используются устойчивый к ошибкам раздел данных — порядок 11, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 9/10, 7/8, 5/6, 3/4, 1/2. Если используются нормальный раздел данных– порядок 11, 9/10, 7/8, 5/6, 3/4, 1/2. Этот порядок основан на наибольшем абсолютном значении таблиц. Этот второй шаг предварительной сортировки выполняется на описанном модуле базой модуля, используемой в первом шаге предварительной сортировки. Вывод первого шага предварительной сортировки сканируется последовательным способом для каждого сборника кодов.

Присвоение номеров модулям согласно следующей метрике может выполнить эти два шага предварительной сортировки:

codbookPriority [32] =

{x,21,21,20,20,19,19,18,18,17,17,0,*x,x,x,x*,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1*}*

*assignedUnitNr = (codebookPriority [cb] * maxNrOfLinesInWindow*

*+ nrOfFirstLineInUnit) * MaxNrOfWindows + window*

с:

codebookPriority [cb]	Приоритет сборника кодов согласно второму шагу предварительной сортировки.
maxNrOfLinesInWindow	Постоянное число: *f* — в случае одного длинного окна и 128 — в случае восьми коротких окон.
nrOfFirstLineInUnit	Число между 0 и 1020 — в случае одного длинного окна и между 0 и 124 — в случае восьми коротких окон (это число всегда является кратным числа четыре).
maxNrOfWindows	Постоянное число: 1 — в случае одного длинного окна и 8 — в случае восьми коротких окон.
window	Всегда 0 — в случае одного длинного окна, число между 0 и 7— в случае восьми коротких окон и вида модулей в порядке возрастания с использованием этих присвоенных номеров модуля.

Примечание для кодера — Чтобы уменьшить слышимые артефакты в случае ошибок в пределах спектральных данных, настоятельно рекомендуется использовать сборник кодов 11 только в случае необходимости.

6.16.3.3.2 Ширина сегмента и инстанцирование (реализация) сегмента

Ширины сегментов зависят от используемого сборника кодов Хаффмана. Они получаются как минимумы (в зависимости от сборника кодов) максимальной длины кодовой комбинации и длины самой долинной переданной кодовой комбинации:

segmentWidth = минута (maxCwLen, length_of_longest_codeword).

Таблица 170 показывает значения *maxCwLen* в зависимости от сборника кодов Хаффмана.

Сегменты реализуются, пока не исчерпывается доступный буфер, тогда как размер этого буфера дается элементом данных *length_of_reordered_spectral_data*. Остающиеся биты в конце буфера увеличивают размер последнего сегмента.

6.16.3.3.3 Порядок кодовых комбинаций Хаффмана в спектральных данных

Схема записи для *non-PCWs*.

Предложенная схема вводит термин набор. Набор содержит определенное число кодовых комбинаций. Если *N* является числом сегментов, все наборы, кроме последнего, содержат *N* кодовых комбинаций. *Non-PCWs* пишутся последовательно в эти наборы. В силу алгоритма предварительной сортировки набор один содержит самые важные *non-PCWs*. Важность кодовых комбинаций, сохраненных в рамках набора, является тем меньше, чем выше номер набора.

Наборы пишутся последовательно. Если один набор был записан полностью, запускается запись следующего набора. Чтобы улучшить поведение распространения ошибок между последовательными наборами, направление записи в пределах сегментов изменяется от набора к набору. В то время как *PCWs* пишутся слева направо, кодовые комбинации набора один пишутся справа налево, кодовые комбинации

набора два снова пишутся слева направо и так далее. Запись в остающуюся часть сегмента всегда начинается в наиболее удаленной позиции остающейся части того сегмента (крайняя левая позиция для направления записи слева направо и самая правая позиция для направления записи справа налево).

Записи набора могут потребовать несколько проб.

Первая проба: первая кодовая комбинация текущего набора пишется в остающуюся часть первого сегмента, вторая кодовая комбинация — в остающуюся часть второго сегмента и так далее. Последняя кодовая комбинация текущего набора пишется в остающуюся часть последнего сегмента.

Вторая проба: остающаяся часть первой кодовой комбинации (если есть) пишется в остающуюся часть второго сегмента, остающаяся часть второй кодовой комбинации (если есть) — в остающуюся часть третьего сегмента и так далее. Остающаяся часть последней кодовой комбинации (если есть) пишется в остающуюся часть первого сегмента (сдвиг по модулю).

Если кодовая комбинация не вписывается в остающуюся часть сегмента, она пишется только частично, а ее остающаяся часть сохраняется. После максимум через *N* проб все кодовые комбинации полностью записываются в сегменты.

6.16.3.3.4 Процесс кодирования

Структура переупорядоченных спектральных данных не может быть описана в пределах синтаксиса в стиле языка *C*, который обычно используется.

```
/* helper functions */
void InitReordering(void);
/* Initializes variables used by the reordering functions like the segment
 widths and the used offsets in segments and codewords. */
void InitRemainingBitsInCodeword(void);
/* Initializes remainingBitsInCodeword[] array for each codeword with
the total size of the codeword. */
int  WriteCodewordToSegment(codewordNr, segmentNr, direction);
/* Writes a codeword or only a part of a codeword indexed by codewordNr
to the segment indexed by segmentNr with a given direction.
Write offsets for each segment are handled internally.
The function returns the number of bits written to the segment.
This number may be lower than the codeword length.
WriteCodewordToSegment handles already written parts of the codeword
internally. */
void ToggleWriteDirection(void);
/* Toggles the write direction in the segments between forward and backward. */
/* (input) variables */
numberOfCodewords; /* 15 in the example */
numberOfSegments; /* 6 in the example */
numberOfSets;      /*  3 in the example */
ReorderSpectralData()
{
InitReordering();
InitRemainingBitsInCodeword();
/* first step: write PCWs (set 0) */
writeDirection = forward;
for (codeword = 0; codeword < numberOfSegments; codeword ++) {
WriteCodewordToSegment(codeword, codeword, writeDirection);
}
/* second step: write nonPCWs */
for (set = 1; set < numberOfSets; set++) {
ToggleWriteDirection(); for (trial = 0; trial < numberOfSegments; trial++) {
for (codewordBase = 0; codewordBase < numberOfSegments; codewordBase++) {
segment  = (trial + codewordBase) % numberOfSegments;
codeword = codewordBase + set*numberOfSegments;
```

```
if (remainingBitsInCodeword[codeword] > 0) {
remainingBitsInCodeword[codeword] -= WriteCodewordToSegment(codeword,
                                    segment,
                                    writeDirection);
}
}
}
}
}.
```

6.16.3.4 Процесс декодирования

См. 5.2.3.2 и 6.3.3.

6.16.3.5 Таблицы

Таблица 169 — Пример вывода первого шага предварительной сортировки для коротких блоков, принимая двухмерные сборники кодов для окон 0, 1, 6 и 7 и четырехмерные сборники кодов для окон 2, 3, 4 и 5

Индекс	Запись кодовой комбинации		Индекс	Запись кодовой комбинации	
	Окно	Индекс окна		Окно	Индекс окна
0	0	0	13	0	13
1	0	1	14	1	14
2	1	2	15	1	15
3	1	3	16	2	16
4	2	4	17	3	17
5	3	5	18	4	18
6	4	6	19	5	19
7	5	7	20	6	20
8	6	8	21	6	21
9	6	9	22	7	22
10	7	10	23	7	23
11	7	11	…	…	…
12	0	12			

Таблица 170 — Значения *maxCwLen* в зависимости от сборника кодов Хаффмана

Сборник кодов	Максимальная длина кодовой комбинации (*maxCwLen*)
0	0
1	11
2	9
3	20
4	16
5	13
6	11
7	14
8	12
9	17

Окончание таблицы 170

Сборник кодов	Максимальная длина кодовой комбинации (*maxCwLen*)
10	14
11	49
16	14
17	17
18	21
19	21
20	25
21	25
22	29
23	29
24	29
25	29
26	33
27	33
28	33
29	37
30	37
31	41

6.17 Кодек с низкой задержкой

6.17.1 Введение

Функциональность низкой задержки кодирования обеспечивает возможность расширить использование универсального аудиокодирования на низкой скорости передачи приложениям, требующим очень низкой задержки цепочки кодирования/декодирования (например, полнодуплексная (двухсторонняя) связь в реальном времени).

Этот подпункт определяет кодер аудио с низкой задержкой, предоставляющий режим с алгоритмической задержкой не превышающей 20 мс.

Полная алгоритмическая задержка общего аудио кодера определяется следующими факторами:

- длина фрейма.

Для основанной на блоке обработки должно пройти определенное количество времени, чтобы собрать выборки, принадлежащие одному блоку;

- задержка *Filterbank*:

Использование пары блока фильтров анализа-синтеза вызывает определенной величины задержку;

- предвидение для решения о переключении блока:

Из-за лежащих в основе принципов схемы переключения блока, обнаружение переходных процессов должно использовать определенную степень предвидения, чтобы гарантировать, что все части переходного сигнала покрываются должным образом короткими окнами;

- использование разрядного резервуара.

В то время как разрядный резервуар облегчает использование локально варьируемой скорости передачи, это накладывает дополнительную задержку в зависимости от размера разрядного резервуара относительно средней скорости передачи на блок.

Полная алгоритмическая задержка может быть вычислена как

$$t_{delay} = \frac{N_{Frame} + N_{FB} + N_{lock_ahead} + N_{bitres}}{F_s},$$

где *Fs* является частотой дискретизации кодера, N_{Frame} является размером фрейма, N_{FB} является

задержкой из-за блока фильтров (*s*), N_{look_ahead} соответствует задержке прогноза переключения блока и N_{bitres} является задержкой из-за использования разрядного резервуара.

Кодек с низкой задержкой получается из аудио объектного типа *AAC LTP*, то есть кодер, состоящий из кодека *AAC* низкого уровня сложности плюс инструменты *PNS* (перцепционная шумовая замена) и *LTP* (долгосрочное прогнозирующее устройство).

6.17.2 Описание кодера

Кодер с низкой задержкой определяется следующими модификациями относительно стандартного алгоритма (то есть аудио объектный тип *AAC LTP*), чтобы достигнуть низкой задержки работы.

6.17.2.1 Размер фрейма/длина окна

Длина аналитического окна уменьшается до 1024 или 960 выборок временного интервала, соответствующих 512 и 480 спектральным значениям, соответственно. Последний выбор позволяет кодеру иметь размер фрейма, который соразмерен с широко используемыми кодеками для разговорных сигналов (20 мс). Соответствующие таблицы полосы масштабного коэффициента даются в 5.4.

6.17.2.2 Переключение блока

Из-за влияния времени предварительного прогноза на полную задержку никакое переключение блока не используется.

6.17.2.3 Форма окна

Как показано в предыдущей главе, переключение блока не используется в кодере с низкой задержкой, чтобы сохранить задержку столь малой насколько возможно. Как альтернативный инструмент для улучшения кодирования переходных сигналов кодер с низкой задержкой использует функцию переключения формы окна с модификацией по сравнению с *AAC* нормальной задержки. Кодер с низкой задержкой все еще использует синусоидальную форму окна, но полученное Кайзер-Бесселевое окно заменяется окном с низким перекрытием. Как обозначено его именем, у этого окна довольно низкое перекрытие со следующим окном, таким образом оптимизируемым для использования инструмента *TNS*, чтобы предотвратить артефакты упреждающего эха в случае переходных сигналов. Для нормального кодирования не переходных сигналов синусоидальное окно используется из-за его выгодной частотной характеристики.

В соответствии с *AAC* с нормальной задержкой *window_shape* указывает форму хвостовой части (то есть второй половины) аналитического окна. Форма ведущей части (то есть первой половины) аналитического окна идентична *window_shape* последнего блока.

Таблица 171 — Окно в зависимости от *window_shape*

window_shape	Окно
0x0	Синусоидное
0x1	Низкое перекрытие

Окно низкого перекрытия определяется выражением:

$$W(i) = \begin{cases} 0 & i = 0..3 \cdot N/16 - 1 \\ \sin\left[\frac{\pi(i - 3 \cdot N/16 + 0{,}5)}{N/4}\right] & i = 3 \cdot N/16..5 \cdot N/16 - 1 \\ 1 & i = 5 \cdot N/16..11 \cdot N/16 - 1 \\ \sin\left[\frac{\pi(i - 9 \cdot N/16 + 0{,}5)}{N/4}\right] & i = 11 \cdot N/16..13 \cdot N/16 - 1 \\ 0 & i = 13 \cdot N/16..N - 1 \end{cases},$$

где $N = 1024$ или $N = 960$.

6.17.2.4 Использование разрядного резервуара

Использование разрядного резервуара минимизируется, чтобы достигнуть требуемой целевой задержки. Как крайний случай никакой разрядный резервуар вообще не используется.

6.17.2.5 Таблицы для временного формирования шума (*TNS*)

Следующие таблицы определяют значение *TNS_MAX_BANDS* для кодера с низкой задержкой.

Таблица 172 — *TNS_MAX_BANDS* в случае 480 выборок на фрейм

Частота дискретизаци	*TNS_MAX_BANDS*
48000	31
44100	32
32000	37
24000	30
22050	30

Таблица 173 — *TNS_MAX_BANDS* в случае 512 выборок на фрейм

Частота дискретизаци	*TNS_MAX_BANDS*
48000	31
44100	32
32000	37
24000	31
22050	31

6.17.2.6 Долгосрочный прогноз

Размер буфера задержки *LTP* уменьшается пропорционально размеру фрейма. Таким образом, размер равен 2048 и 1920 выборкам для величины фрейма *N*=512 и *N*=480, соответственно.

6.17.2.7 Адаптация к системам, использующим более низкие частоты дискретизации

В определенных приложениях может понадобиться интегрировать декодер с низкой задержкой в аудиосистему, работающую с более низкими частотами дискретизации (например, 16 кГц), в то время как номинальная частота дискретизации полезной нагрузки потока битов намного выше (например, 48 кГц, соответствуя задержке алгоритмического кодека, приблизительно 20 мс). В таких случаях целесообразно декодировать вывод кодека с низкой задержкой непосредственно на целевой частоте дискретизации вместо того, чтобы использовать дополнительное преобразование частоты дискретизации после декодирования.

Это может быть аппроксимировано соответствующим уменьшением как размера фрейма, так и частоты дискретизации на некоторый целочисленный коэффициент (например, 2, 3), приводящим к той же самой разрешающей способности кодека по времени/частоте. Например, вывод кодека может быть сгенерирован на частоте дискретизации 16 кГц вместо номинальных 48 кГц, сохраняя только нижнюю треть (то есть 480/3 = 160) спектральных коэффициентов до блока фильтров синтеза и уменьшая размер инверсного преобразования до одной трети (то есть, размер окна 960/3 = 320).

Как следствие, декодирование для более низких частот дискретизации уменьшает требования как к памяти, так и вычислительным ресурсам, но, возможно, не вырабатывает точно такой же выход как декодирование полной пропускной способности, сопровождаемое ограничением полосы и преобразованием частоты дискретизации.

Декодирование на более низкой частоте дискретизации не влияет на интерпретацию уровней, которая обращается к номинальной частоте дискретизации полезной нагрузки потока битов *AAC* с низкой задержкой.

6.18 Инструмент *SBR*

6.18.1 Описание инструмента

Человеческая речь и музыкальные инструменты генерируют любые квазистационарные сигналы возбуждения, которые появляются из колеблющихся систем, или сигналы возникают из различных источников

шумов. Широкополосный спектр возбуждения может быть инициализирован одним источником или набором нескольких источников, например, голосовыми связками, струнами, язычками инструментов и т. д. У них различные частотные компоненты в зависимости от источника. Сигналы возбуждения впоследствии фильтруются резонаторами, таким как голосовой трактат, корпус скрипки и т. д., придавая речи или музыкальному инструменту их характерный оттенок или тембр. Ограничение пропускной способности такого сигнала эквивалентно усечению последовательности гармоник. Такое усечение изменяет воспринимаемый тембр, и аудиосигнал звучит «приглушенным» или «тусклым», что может привести к уменьшению разборчивость речи.

Инструмент *SBR* (репликация полосы спектра) расширяет полосу частот декодируемого аудиосигнала с ограниченной пропускной способностью. Процесс основывается на репликации последовательностей гармоник, ранее усеченных, чтобы уменьшить скорость передачи данных, на основе доступного сигнала с ограниченной полосой и управляющей информацией, полученной из кодера. Отношение между тональными и шумоподобными компонентами сохраняется адаптивной инверсной фильтрацией так же как дополнительным добавлением шума и синусоид.

6.18.2 Определения

6.18.2.1 Определения специфики *SBR*

Полоса	Группа последовательных поддиапазонов *QMF* (как в полосе ограничителя, полосе минимального уровня шума, и т.д.).
chirp factor	Фактор расширения пропускной способности формант описываемый полиномиалом *LPC*.
Down Sampled SBR	Инструмент *SBR* с измененным блоком фильтров синтеза, приводящий к той же самой частоте дискретизации выходного сигнала с пониженной частотой дискретизации, как входной сигнал, на инструменте *SBR*. Может использоваться, когда требуется вывод с более низкой частотой дискретизации.
Масштабный коэффициент огибающей	Элемент, представляющий усредненную энергию сигнала по области, описанной полосой частот и сегментом времени.
Полоса частот	Интервал в частоте, группе последовательных поддиапазонов *QMF*.
Граница частоты	Разграничитель полосы частот, выраженный как определенный поддиапазон *QMF*.
NA	Неприменимый.
Минимальный уровень шума	Вектор масштабных коэффициентов минимального уровня шума.
Масштабный коэффициент минимального уровня шума	Элемент, связанный с областью, описанной полосой частот и сегментом времени, представляющий соотношение между энергией шума, который будет добавлен к сгенерированному сигналу *HF* с регулируемой огибающей, и энергией его самого.
Патч	Ряд смежных *QMF* поддиапазонов, перемещенных в различное частотное расположение.
QMF	Квадратурный фильтр зеркала.
SBR	Репликация полосы спектра.
Огибающая *SBR*	Вектор масштабных коэффициентов огибающей.
Фрейм *SBR*	Сегмент времени, связанный с одним элементом данных расширения *SBR*.
Диапазон *SBR*	Частотный диапазон сигнала, сгенерированного алгоритмом *SBR*.
Поддиапазон	Частотный диапазон, представленный одной строкой в матрице *QMF*, переносящий подвыбранный сигнал.
Граница времени	Разграничитель сегмента времени, выраженный как определенный временной интервал.
Сегмент времени	Интервал во времени, группа последовательных временных интервалов.
Сетка времени/частоты	Описание временных сегментов огибающей *SBR* и связанных таблиц разрешающей способности по частоте, а также описание сегментов времени минимального уровня шума.
Временной интервал	Наилучшее разрешение во времени для огибающих *SBR* и минимальных уровней шума. Один временной интервал равняется двум подвыборкам в домене *QMF*.

6.18.2.2 Нотация особенностей *SBR*

Описание инструмента *SBR* использует следующую нотацию:

- векторы обозначаются полужирными строчными именами;

- матрицы (и векторы векторов) обозначаются полужирными прописными однобуквенными именами;
- переменные обозначаются курсивом.
- функции обозначаются как *func (x);*
- элементы данных обозначаются, как многословные имена с префиксным *"bs _"*, например, *bs_bitstream_element.*

Выражение $\sum_{k=a}^{b} f(k)$ оценивается обнуленным, если $b < a$.

Для блок-схем принимается нормальная интерпретация псевдокода, без округления или усечения, если явно не утверждено.

6.18.2.3 Скалярные операции

X^* комплексно сопряженные для X.

6.18.2.4 Векторные операции

y = sort(x). y равен сортированному вектору *x*, где элементы x сортируются в порядке возрастания.

y = length(x). y является числом элементов вектора *x*.

Y = ceil(x) представляет округление до ближайшего целого в направлении к бесконечности.

6.18.2.5 Константы

$\varepsilon = 1$	Константа, чтобы избежать деления на ноль, например, на 96 дБ ниже максимального входного сигнала.
HI = 1	Индекс используется для разрешения высокочастотной огибающей *SBR*.
LO = 0	Индекс используется для разрешения низкочастотной огибающей *SBR*.
NOISE_FLOOR_OFFSET = 6	Смещение минимального уровня шума.
RATE = 2	Константа, указывающая число выборок поддиапазона *QMF* на временной интервал.

6.18.2.6 Переменные

ch	Текущий канал.
bsco	Смещение кодека с масштабируемой полосой пропускания, указывает для системы с масштабируемой полосой пропускания число поддиапазонов *QMF* выше k_x в диапазоне *SBR*, которые должны быть заменены данными *AAC* из уровня расширения с масштабируемой полосой пропускания. Если базовый кодер с масштабируемой полосой пропускания не используется, переменная является нулем.
E_{Orig}	Имеет L_E столбцов, где каждый столбец имеет длину N_{Low} или N_{High} в зависимости от разрешающей способности по частоте для каждой огибающей *SBR*. Элементы в E_{Orig} содержат масштабные коэффициенты огибающей исходного сигнала.
$F=[f_{TableLow}, f_{TableHigh}]$	Имеет векторы на два столбца, содержащие таблицы границы частоты для низкой и высокой разрешающей способности по частоте.
Fs_{SBR}	Внутренняя частота дискретизации инструмента *SBR Tool*, двойная частота дискретизации базового кодера (после отображения частоты дискретизации, таблица 82). Частота дискретизации расширенного выходного сигнала *SBR* равна внутренней частоте дискретизации инструмента *SBR Tool*, если *SBR Tool* не работает в субдискретизированном режиме. Если *SBR Tool* работает в субдискретизированном режиме, выходная частота дискретизации равна частоте дискретизации базового кодера.
f_{Master}	Имеет длину $N_{Master}+1$ и содержит информацию о группировке задающей частоты *QMF*.
$f_{TableHigh}$	Имеет длину $N_{High}+1$ и содержит границы частоты для огибающих *SBR* с высокочастотным разрешением
$f_{TableLim}$	Имеет длину N_L+1 и содержит границы частоты, используемые ограничителем.
$f_{TableLow}$	Имеет длину $N_{Low}+1$ и содержит границы частоты для огибающих *SBR* с низкочастотным разрешением.
$f_{TableNoise}$	Имеет длину N_Q+1 и содержит границы частоты, используемые минимальными уровнями шума.
k_x	Первый поддиапазон *QMF* в диапазоне *SBR*.
k_0	Первый поддиапазон *QMF* в таблице f_{Master}.
L_E	Число огибающих *SBR*.
L_Q	Число минимальных уровней шума.

M	Число поддиапазонов *QMF* в диапазоне *SBR*.
middleBorder	Точки определенной границы времени.
N_L	Число полос ограничителя.
N_{Master}	Число полос частот в таблице разрешения задающей частоты.
N_Q	Число полос минимального уровня шума.
n = [N_{Low}, N_{High}]	Число полос частот для разрешения низкой и высокой частоты.
numPatches	Переменная, указывающая число патчей в диапазоне *SBR*.
numTimeSlots	Число временных интервалов огибающей *SBR*, которые существуют в пределах фрейма *AAC*, 16 — для 1024 фрейма *AAC* и 15 — для 960 фреймов *AAC*.
panOffset = [24,12]	Значения смещения для огибающей *SBR* и данных минимального уровня шума, когда используются связанные каналы.
patchBorders	Вектор, содержащий границы частоты патчей.
patchNumsubbands	Вектор, содержащий число поддиапазонов в каждом патче.
Q_{Orig}	Имеет столбцы L_Q, где каждый столбец имеет длину N_Q и содержит минимальный уровень шума *scalefactor*.
$r = [r_0...,r_{L-1}]$	Разрешающая способность по частоте для всех огибающих *SBR* в текущем фрейме *SBR*, ноль для низкого разрешения, единица для высокого разрешения.
reset	Переменная в кодере и декодере, которая устанавливается в единицу, если определенные элементы данных изменились в сравнении с предыдущим фреймом *SBR*, иначе устанавливается в ноль.
t_E	Имеет длину L_E+1 и содержит временные границы пуска и остановки для всех огибающих *SBR* в текущем фрейме *SBR*.
t_{HFAdj}	Смещение для модуля регулятора огибающей.
t_{HFGen}	Смещение для модуля *HF*-генерации.
t_Q	Имеет длину L_Q+1 и содержит временные границы пуска и остановки для всех минимальных уровней шума в текущем фрейме *SBR*.
W	Матрица поддиапазона, где хранятся фильтруемые выборки поддиапазона *QMF*.
X_{High}	Матрица поддиапазона входного банка *QMF* комплекса в регулятор *HF*.
X_{Low}	Матрица поддиапазона входного банка *QMF* комплекса в генератор *HF*.
Y	Матрица поддиапазона выходного банка *QMF* комплекса из регулятора *HF*.

6.18.3 Процесс декодирования

6.18.3.1 Введение

SBR включает адаптивную разрешающую способность по времени и по частоте для кодирования огибающей и корректировки. Адаптация получается гибкой группировкой выборок поддиапазона *QMF* во времени и частоте. Для каждой такой группы вычисляется и передается соответствующий масштабный коэффициент. В 6.18.3 описывается, как воссоздать группировку по времени и частоте, выбранную кодером. Кроме того, это показывает, как декодируются дельта кодированные огибающие *SBR* и минимальные уровни шума. Процесс декодирования обрисовывается в общих чертах для одного единственного элемента канала так же, как для одного элемента пары каналов. Для элемента единственного канала номер канала обозначается нулем. Для элементов пары каналов два канала индексируются нулем и единицей, где ноль представляет данные декодируемого первого канала в элементе пары каналов, а единица представляет декодируемый вторым каналом в элементе пары каналов. Система сбрасывается (*reset* = 1), если значение какого-либо из следующих элементов данных в заголовке *SBR* отличается от значения из предыдущего фрейма *SBR*:

- *bs_start_freq;*
- *bs_stop_freq;*
- *bs_freq_scale;*
- *bs_alter_scale;*
- *bs_xover_band;*
- *bs_noise_bands.*

6.18.3.2 Таблицы полосы частот

Группировка выборок поддиапазона *QMF* по частоте описывается таблицами полосы частот. Таблицы определяются функциями, большинство параметров которых передается в заголовке *SBR*. Для каждой огибающей *SBR* доступны две таблицы полосы частот: таблица разрешения высокой частоты $f_{TableHigh}$

и таблица разрешения низкой частоты, $f_{TableLow}$. У минимального уровня шума и ограничителя также есть соответствующие таблицы полосы частот $f_{TableNoise}$ и $f_{TableLim}$. Все вышеупомянутые таблицы получаются из одной таблицы полосы задающей частоты f_{Master}. Таблицы полосы частот содержат границы частоты для каждой полосы частот, представленных как поддиапазоны *QMF*. Каждая полоса частот определяется начальной границей частоты и конечной границей частоты. Поддиапазон *QMF*, обозначенный начальной границей частоты, включается в полосу частот, а поддиапазон *QMF*, обозначенный конечной границей частоты, исключается из полосы частот. Для всех таблиц, полученных из f_{Master}, конечная граница полосы частот n равняется начальной границе полосы частот n+1, где n является произвольной полосой частот в таблице.

6.18.3.2.1 Таблица полосы задающей частоты

Чтобы создать таблицу полосы задающей частоты, сначала должны быть вычислены поддиапазоны *QMF*, представляющие границы таблицы. Поддиапазон, представляющий нижнюю границу частоты таблицы, обозначается k_0, и определяется так:

$k_0 = startMin = offset(bs_start_freg)$,

где

$$offset = \begin{cases} [-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7], & Fs_{SBR} = 16000 \\ [-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7,9,11,13], & Fs_{SBR} = 22050 \\ [-5,-3,-2,-1,0,1,2,3,4,5,6,7,9,11,13,16], & Fs_{SBR} = 24000 \\ [-6,-4,-2,-1,0,1,2,3,4,5,6,7,9,11,13,16], & Fs_{SBR} = 32000 \\ [-4,-2,-1,0,1,2,3,4,5,6,7,9,11,13,16,20], & 44100 \le Fs_{SBR} \le 64000 \\ [-2,-1,0,1,2,3,4,5,6,7,9,11,13,16,20,24], & Fs_{SBR} > 64000 \end{cases}$$

$$startMin \begin{cases} NINT\left(3000\frac{128}{Fs_{SBR}}\right), & Fs_{SBR} \le 32000 \\ NINT\left(4000\frac{128}{Fs_{SBR}}\right), & 32000 \le Fs_{SBR} \le 64000 \\ NINT\left(3000\frac{128}{Fs_{SBR}}\right), & 64000 \le Fs_{SBR}. \end{cases}$$

Верхняя граница частоты, обозначенная k_2, определяется:

$$k_2 \begin{cases} min\left(64, stopMin + \sum_{i=0}^{bs_stop_freg-1} stopDkSort(i)\right), & 0 \le bs_stop_freg \le 14 \\ min(64, 2k_0), & bs_stop_freg = 14 \\ min(64, 3k_0), & bs_stop_freg = 15, \end{cases}$$

где

$$startMin \begin{cases} NINT\left(6000\frac{128}{Fs_{SBR}}\right), & Fs_{SBR} \le 32000 \\ NINT\left(8000\frac{128}{Fs_{SBR}}\right), & 32000 \le Fs_{SBR} \le 64000 \\ NINT\left(10000\frac{128}{Fs_{SBR}}\right), & 64000 \le Fs_{SBR} \end{cases}$$

stopDkSort=sory(stopDk)

$$stopDk(p) = NINT\left(stopMin\left(\frac{64}{stopMin}\right)^{\frac{p-1}{13}}\right) - \left(stopMin\left(\frac{64}{stopMin}\right)^{\frac{p}{13}}\right), 0 \le p \le 12.$$

Входными переменными являются k_0 и k_2, как вычислено выше, и элементы данных *bs_freq_scale* и *bs_alter_scale*. Таблица f_{Master} определяется только для $k_2 > k_0$.

6.18.3.2.2 Таблицы полосы частот

Таблица полосы частот $f_{TableHigh}$, используемая для огибающей *SBR* с высокочастотным разрешением, получается извлечением подмножества границ из f_{Master} согласно:

$$N_{Higt} = N_{Master} - bs_xover_band$$

$$N_{Higt} = INT\left(\frac{N_{Higt}}{2}\right) + \left(N_{Higt} - 2INT\left(\frac{N_{Higt}}{2}\right)\right)$$

$$n = [N_{LOW}, N_{Higt}]$$

$$f_{TableHigh}(k) = f_{Master}(k + bs_xover_band), 0 \le k \le N_{Higt}$$

$$M = f_{TableHigh}(N_{Higt}) - f_{TableHigh}(0)$$

$$k_x = f_{TableHigh}(0).$$

Таблица полосы частот $f_{TableLow}$, используемая для огибающих *SBR* с низкочастотным разрешением, получается путем извлечения подмножества границ из $f_{TableHigh}$ согласно:

$$f_{TableHigh}(k) = f_{TableHigh}(i(k)), 0 \le k \le N_{Low},$$

где *i(k)* определяется выражением

$$i(k) = \begin{cases} 0, \text{ если } k = 0 \\ 2k - \frac{1-(-1)^{N_{High}}}{2}, \text{ если } k = 0. \end{cases}$$

Таблица полосы частот минимального уровня шума $f_{TableNoise}$, извлекается из $f_{TableLow}$ согласно:

$$f_{TableNoise}(k) = f_{TableLow}(i(k)), 0 \le k \le N_0,$$

где *i(k)* и N_Q определяются выражением

$$i(k) = \begin{cases} 0, \text{ если } k = 0 \\ i(k-1) + INT\left(\frac{N_{Low} - i(k-1)}{N_0 + 1 - k}\right), \text{ если } k \ne 0 \end{cases}$$

$$N_Q = \max\left(1, NINT\left(bs_noise_bands \frac{\log\left(\frac{k_2}{k_x}\right)}{\log(2)}\right)\right), 0 \le bs_noise_bands \le 3.$$

6.18.3.2.3 Таблица полосы частот ограничителя

Таблица полосы частот ограничителя $f_{TableLim}$ создается, чтобы иметь или только одну полосу ограничителя по всему диапазону SBR, или приблизительно 1, 2 или 3 полосы на октаву, согласно сообщению *bs_limiter_bands* из полезной нагрузки потока битов. Таблица содержит индексы поддиапазонов блока фильтров синтеза, где число элементов равняется числу полос плюс один. Первый элемент всегда k_x, $f_{TableLim}$ является подмножеством объединения $f_{TableLow}$ и границ патча, определенных в 6.18.6.

Если *bs_limiter_bands* является нулем, используется только одна полоса ограничителя и $f_{TableLim}$ создается как

$$f_{TableLim} = [f_{TableLow}(0), f_{TableLow}(N_{Low})]$$

$$N_i = 1.$$

Переменные *numPatches*, *patchBorders* и *patchNumsubbands* вычисляются в 6.18.6.

6.18.3.3 Сетка время/частота

Часть сетки времени/частоты полезной нагрузки потока битов описывает число огибающих *SBR* и минимальных уровней шума, а также сегмент времени, связанный с каждой огибающей *SBR* и минимальным уровнем шума. Кроме того, она описывает какую таблицу полосы частот использовать для каждой огибающей *SBR*. Используются четыре различных класса *SBR* фрейма — *FIXFIX, FIXVAR, VARFIX* и *VARVA*. У каждого из них имеются различные возможности относительно выбора сетки времени/частоты. Имена указывают являются ли расположения ведущей и хвостовой границ фрейма *SBR* (то есть рамки фрейма *SBR*) переменными или нет с синтаксической точки зрения. Временные сегменты огибающей *SBR* и минимального уровня шума описываются векторами $t_E(l)$ и $t_Q(l)$, соответственно, которые содержат границы для каждого временного сегмента, выраженные во временных интервалах. Каждый временной сегмент определяется границей времени начала и границей времени конца. Временной интервал, обозначенный границей времени начала, включается в сегмент времени, временной интервал, обозначенный границей времени конца, исключается из сегмента времени. Для обоих векторов граница времени конца сегмента времени l равняется границе времени начала сегмента времени $l+1$, где l является произвольным сегментом времени в векторе. Вычисление $t_E(l)$ описывается ниже.

Сначала из полезной нагрузки потока битов получают ведущую границу *SBR* фрейма *absBordLead* и конечную границу *SBR* фрейма *absBordTrail* согласно:

$$absBordLead = \begin{cases} 0 & ,bs_frame_class = FIXFIX \text{ or } FIXVAR \\ bs_var_bord_0 & ,bs_frame_class = VARVAR \text{ or } VARFIX \end{cases}$$

$$absBordTrail = \begin{cases} numTimeSlots & ,bs_frame_class = FIXFIX \text{ or } VARFIX \\ bs_var_bord_1 + numTimeSlots & ,bs_frame_class = VARVAR \text{ or } FIXVAR \end{cases}.$$

Чтобы декодировать временные границы всех огибающих *SBR* в пределах фрейма *SBR*, вычисляется число относительных границ, связанных с ведущими и конечными границами времени соответственно согласно:

$$n_{RelLead} = \begin{cases} L_E - 1 & ,bs_frame_class = FIXFIX \\ 0 & ,bs_frame_class = FIXVAR \\ bs_num_rel_0 & ,bs_frame_class = VARVAR \text{ or } VARFIX \end{cases}$$

$$n_{RelTrail} = \begin{cases} 0 & ,bs_frame_class = FIXFIX \text{ or } VARFIX \\ bs_num_rel_1 & ,bs_frame_class = VARVAR \text{ or } FIXVAR, \end{cases}$$

где

$$L_E = bs_num_env.$$

Вектор границы времени огибающей *SBR* текущего фрейма *SBR*, $t_E(l)$ вычисляется согласно:

$$t_E(l) = \begin{cases} absBordLead \text{ если } l = 0 \\ absBordTrail \text{ если } l = L_E \\ absBordLead + \sum_{i=0}^{l-1} relBordLead(i) \text{ если } 1 \le l \le n_{Realeand} \\ absBordLead + \sum_{i=0}^{L_E - l - 1} relBordTrail(i) \text{ если } n_{RelLeand} \le l \le L_E, \end{cases}$$

где $0 \le l \le L_E$ и *relBordLead*(l) и *relBordTrail*(l) являются векторами, содержащими относительные границы, связанные с ведущими и конечными границами, соответственно. Оба вектора (если применяются) определены ниже.

$$relBordLead(l) = \begin{cases} NINT\left(\frac{numTimeSlots}{L_E}\right), bs_frame_class = FIXFIX \\ NA, bs_frame_class = FIXVAR \\ bs_rel_bord_0(l), bs_frame_class = VARVAR \text{ для } VARFIX, \end{cases}$$

где $0 \le l \le n_{RelLeadl}$

$$relBordLead(l) = \begin{cases} NA, bs_frame_class = FIXFIX \text{ или } VARFIX \\ bs_rel_bord_1(l), bs_frame_class = VARVAR \text{ для } FIXVAR, \end{cases}$$

где $0 \le l \le n_{RelTrail}$.

В пределах одного фрейма *SBR* могут быть один или два минимальных уровня шума. Временные границы минимального уровня шума получаются из вектора временной границы огибающей *SBR* согласно:

$$L_Q = bs_num_noise$$

$$t_Q = \begin{cases} [t_E(0), t_E(1)], L_E = 1 \\ [t_E(0), t_E(middleBorder), t_E(L_E)], L_E > 1, \end{cases}$$

где *middleBorder* = *func (bs_frame_class, bs_pointer, L_E)* вычисляется согласно таблице 174.

Таблица 174 — Функция *middleBorder*

bs_pointer	*bs_frame_class*		
	FIXFIX	*VARFIX*	*FIXVAR, VARVAR*
= 0	$\frac{L_E}{2}$	1	$L_E - 1$
= 1	$\frac{L_E}{2}$	$L_E - 1$	$L_E - 1$
> 1	$\frac{L_E}{2}$	*bs_pointer* – 1	L_E + 1– *bs_pointer*

Каждая огибающая *SBR* может иметь разрешение высокой или низкой частоты. Это описывается вектором разрешающей способности *r*(l) по частоте огибающей *SBR*, который вычисляется согласно:

$$r(l) = bs_freq_res(l), \quad 0 \le l < L_E,$$

где

$$r(l) = 0 \text{ для } f_{TableLow}$$

$$r(l) = 0 \text{ для } f_{TableHigh}.$$

6.18.3.4 Декодирование огибающей *SBR* и минимального уровеня шума

Дельта-кодирование масштабных коэффициентов огибающей и масштабных коэффициентов минимального уровня шума, выполняется или во временном, или в частотном направлении для каждой огибающей *SBR* и минимального уровня шума. Когда применяется дельта-кодирование во временном направлении по фрейму *SBR*, первая огибающая *SBR* в текущем фрейме *SBR* является дельта-кодированной относительно последней огибающей *SBR* предыдущего фрейма *SBR*. То же самое справедливо для минимальных уровней шума.

Если инструмент *SBR Tool* используется с масштабируемым кодеком *AAC*, масштабные коэффициенты огибающей уровня расширения стерео могут декодироваться, только если уровень расширения доступен и переменная *enhanceLayDecE* равна единице. Переменная *enhanceLayDecE* определяется как:

$$enhanceLayDecE = \begin{cases} 1, \text{ если } enhanceLayDecE' = 1 \text{ и } bs_df_env(0) = 0, \\ 0, \text{ в других случаях} \end{cases}$$

где *enhanceLayDecE'* представляет значение предыдущего *SBR* фрейма. Для первого *SBR* фрейма *enhanceLayDecE'* устанавливается в единицу. Если масштабируемая система не используется, то переменная *enhanceLayDecE* должна быть константой, установленной в единицу.

Вывод масштабных коэффициентов огибающей E из дельта-кодированных масштабных коэффициентов огибающей E_{Delta} определяется с помощью:

$$E(k,l) = \begin{cases} \sum_{i=0}^{k} \delta E_{Delta}(i,l), \begin{cases} 0 \le l < L_E \\ 0 \le k < n(r(l)) \\ bs_df_env(l) = 0 \end{cases} \\ g_E(k,l) + \delta E_{Delta}(k,l), \begin{cases} 0 \le l < L_E \\ 0 \le k < n(r(l)) \\ bs_df_env(l) = 0 \\ r(l) = g(l) \end{cases} \\ g_E(i(k),l) + \delta E_{Delta}(k,l), \begin{cases} 0 \le l < L_E \\ 0 \le k < n(r(l)) \\ bs_df_env(l) = 0 \\ r(l) = 0 \\ r(l) = 1 \\ i(k) \text{ зависит от } f_{TableHigh}(i(k)) = f_{TableLow}(k) \end{cases} \\ g_E(i(k),l) + \delta E_{Delta}(k,l), \begin{cases} 0 \le l < L_E \\ 0 \le k < n(r(l)) \\ bs_df_env(l) = 0 \\ r(l) = 1 \\ q(l) = 0 \\ i(k) \text{ зависит от } f_{TableLow}(i(k)) \le f_{TableHigh}(k) < f_{TableLow}(i(k)+1, \end{cases} \end{cases}$$

где

$$\delta = \begin{cases} 2, \text{ если } ch = 1 \text{ и } bs_coupling = 1 \\ 0, \text{ в других случаях} \end{cases}$$

и где $g_E(k,l)$ и $g(l)$ определяются ниже, а $E_{Delta}(k,l)$ считывается из элемента данных *bs_data_env*, как показанно ниже. Масштабные коэффициенты огибающей из предыдущего *SBR* фрейма, E' необходимы, когда дельта-кодирование во временном направлении по границам *SBR* фрейма. Число огибающих *SBR* предыдущего фрейма *SBR* обозначено L_E' и также необходимо в этом случае так же, как вектор разрешающей способности по частоте предыдущего фрейма *SBR*,обозначенный r'.

$$g_E(k,l)=\begin{cases}E(k,l-1),\begin{cases}1\le l<L_E\\0\le k<n(r(l))\end{cases}\\E'(k,L'_E-1),\begin{cases}l=0\\0\le k<n(r(l))\end{cases}\end{cases} \quad \text{и } q(l)=\begin{cases}r(l-1) & ,1\le l<L_E\\r'(L'_E-1), & l=0\end{cases}$$

$$E_{Delta}(k,l)=bs_data_env[ch][l][k],\begin{cases}0\le l<L_E\\0\le k<n(r(l).\end{cases}$$

Если используется инструмент *SBR Tool* с масштабируемым кодеком *AAC*, данные минимального уровня шума уровня расширения стерео могут декодироваться, только если уровень расширения доступен и переменная *enhanceLayDecQ* является единицей. Переменная *ienhanceLayDecQ* определяется как:

$$enhanceLayDecQ = \begin{cases}1\text{ ,если } enhanceLayDecQ'=1 \text{ и } bs_df_noise(0)=0,\\0,\text{ в других случаях}\end{cases}$$

где *enhanceLayDecQ'* представляет значение предыдущего фрейма *SBR*. Для первого фрейма *SBRienhanceLayDecQ'* устанавливается в единицу. Если масштабируемая система не используется, то переменная *enhanceLayDecQ* должна быть константой, установленной в единицу.

Получение данных минимального уровня шума *Q* из данных дельта-кодированного минимального уровня шума Q_{Delta}, определяется:

$$Q(k,l)=\begin{cases}\sum_{i=0}^{k}\delta Q_{Delta}(i,l),\begin{cases}1\le l<L_E\\0\le k<n(r(l))\\bs_df_noise(l)=0\end{cases}\\Q(k,l-1)+\delta Q_{Delta}(k,l),\begin{cases}1\le l<L_Q\\0\le k<N_Q\\bs_df_noise(l)=1\end{cases}\\Q'(k,L'_Q-1)+\delta Q_{Delta}(k,0),\begin{cases}l=1\\0\le k<N_Q\\bs_df_noise(0)=1,\end{cases}\end{cases}$$

где

$$\delta=\begin{cases}2\text{ ,если } ch=1 \text{ и } bs_coupling=1\\0,\text{ в других случаях}\end{cases}$$

и где *Q'* является масштабными коэффициентами минимального уровня шума из предыдущего фрейма *SBR*, а L_Q' является числом минимальных уровней шума из предыдущего фрейма *SBR*. Q_{Delta} *(k, l)* читается из элемента данных *bs_data_noise*, как показано ниже.

$$Q_{Delta}(k,l)=bs_data_noise[ch][l][k],\begin{cases}1\le l<L_Q\\0\le k<N_Q.\end{cases}$$

6.18.3.5 Деквантизация и декодирование стерео

Для квантования масштабных коэффициентов огибающей доступны два шага квантования, *bs_amp_res* =0 соответствует шагу квантования 1.5 дБ, а *bs_amp_res* = 1, соответствует шагу квантования 3.0 дБ. Для одноканального элемента масштабные коэффициенты огибающей деквантуются согласно:

$$E_{Orig}(k,l)=64\ \text{x}\ 2^{\frac{E(k,l)}{a}},\begin{cases}1\le l<L_E\\0\le k<n(\text{r(l)'}\end{cases}$$

где

$$a = \begin{cases} 2, bs_amp_res = 0 \\ 1, bs_amp_res = 1 \end{cases}.$$

Масштабные коэффициенты минимального уровня деквантуются согласно:

$$Q_{Orig}(k,l) = 2^{NOISE_FLOOR_OFFSET - Q(k,l)}, \begin{cases} 0 \le l < L_Q \\ 0 \le k < N_Q \end{cases}.$$

Для элемента пары каналов, где режим сцепления не используется, отдельные каналы обрабатываются, как в случае элемента единственного канала выше.

Если используется режим сцепления *bs_coupling* = 1, временные сетки t_E и t_Q одинаковые для обоих каналов. Пусть E_0, E_1, и Q_0, Q_1 представляют декодируемые масштабные коэффициенты огибающей и масштабные коэффициенты минимального уровня шума в соответствии с обрисованным выше процессом декодирования. Нижний индекс ноль представляет декодируемый первый канал (средняя энергия и средний минимальный уровень шума исходного левого и правого каналов), а нижний индекс единица представляет декодируемый второй канал (отношение энергии и отношение минимального уровня шума исходного левого и правого каналов).

Ниже показано как деквантизируются масштабные коэффициенты огибающейи минимального уровня шума в режиме сцепления (*bs_coupling* = 1).

$$E_{LeftOrig}(k,l) = 64x\frac{2^{\frac{E_0(k,l)}{a}+1}}{1+2^{\frac{panOffset(bs_amp_res) - E_1(k,l)}{a}}}, \begin{cases} 1 \le l < L_E \\ 0 \le k < n(r(l)) \end{cases}$$

$$E_{RightOrig}(k,l) = 64x\frac{2^{\frac{E_0(k,l)}{a}+1}}{1+2^{\frac{E_1(k,l) - panOffset(bs_amp_res)}{a}}}, \begin{cases} 1 \le l < L_E \\ 0 \le k < n(r(l)) \end{cases}$$

$$Q_{OrigLeft}(k,l) = \frac{2^{NOISE_FLOOR_OFFSET - Q_0(k,l)+1}}{1+2^{panOffset(1) - Q_1(k,l)}}, \begin{cases} 1 \le l < L_Q \\ 0 \le k < N_Q \end{cases}$$

$$Q_{OrigRight}(k,l) = \frac{2^{NOISE_FLOOR_OFFSET - Q_0(k,l)+1}}{1+2^{Q_1(k,l) - panOffset(1)}}, \begin{cases} 1 \le l < L_Q \\ 0 \le k < N_Q \end{cases}$$

Выше огибающие *SBR* и минимальные уровни шума обозначены $E_{OrigLeft}$, $E_{OrigRight}$, $Q_{OrigLeft}$ и $Q_{OrigRight}$, чтобы отличать относящиеся к двум каналам в элементе пары каналов. Так как никакая зависимость каналов после вышеупомянутого декодирования и деквантования не существует, огибающие и минимальные уровни шума будут с этого момента упоминаться как E_{Orig} и Q_{Orig}.

Если инструмент *SBR Tool* используется с масштабируемым кодеком *AAC* и уровень расширения стерео не присутствует, данные в элементе пары каналов, доступные в базовом уровне моно, должны быть деквантизированы как одноканальные элементы, то есть $E_{LeftOrig}(k, l)$ вычисляется как $E_{Orig}(k, l)$.

Если уровень расширения стерео будет доступен, то масштабные коэффициенты огибающей в элементе пары каналов должны быть деквантизированыи стерео декодированы как элемент пары каналов, при условии, что переменная *enhanceLayDecE* является единицей. Если переменная *enhanceLayDecE* является нулем, масштабные коэффициенты огибающей первого канала деквантизируютсякак элемент единственного канала, то есть $E_{LeftOrig}(k, l)$вычисляется как $E_{Orig}(k, l)$,и масштабные коэффициенты огибающей второго канала должны быть установлены в те же значения как масштабные коэффициенты огибающей первого канала, то есть $E_{RightOrig}(k, l) = E_{LeftOrig}(k, l)$.

То же касается масштабных коэффициентов минимального уровня шума, то есть, если уровень расширения стерео отсутствует, данные в элементе пары каналов, доступные в базовом уровне моно, будут деквантизированы как одноканальный элемент, то есть $Q_{LeftOrig}(k, l)$ вычисляется как $Q_{Orig}(k, l)$.

Если уровень расширения стерео доступен, то масштабные коэффициенты минимального уровня шума в элементе пары каналов должны быть деквантизированы и стереодекодированы как элемент пары каналов, при условии, что переменная *enhanceLayDecQ* является единицей. Если переменная *enhanceLayDecQ* является нулем, масштабные коэффициенты минимального уровня шума первого канала деквантизируются как одноканальный элемент, то есть $Q_{LeftOrig}(k,l)$ вычисляется как $Q_{Orig}(k,l)$, и масштабные коэффициенты минимального уровня шума второго канала будут установлены в те же значения, как масштабные коэффициенты минимального уровня шума первого канала, то есть $Q_{RightOrig}(k, l) = Q_{LeftOrig}(k, l)$.

6.18.3.6 Требования

К данным *SBR* применяются следующие требования:

- число поддиапазонов *QMF*, покрытых *SBR*, то есть $k_2 - k_0$, должно удовлетворять:

$$k2 - k0 \le \begin{cases} 48, Fs_{SBR} \le 32 \text{ кГц} \\ 35, Fs_{SBR} = 44{,}1 \text{ кГц} \\ 32, Fs_{SBR} \ge 48 \text{ кГц}; \end{cases}$$

- граница конечной частоты диапазона *SBR* должна быть в пределах $\frac{Fs_{SBR}}{2}$, то есть $k_x + M \le 64$;

- граница начальной частоты диапазона *SBR* должна быть в пределах $\frac{Fs_{SBR}}{4}$, то есть $k_x \le 32$;

- число масштабных коэффициентов *SBR* во фрейме *SBR*, L_E должно удовлетворять:

$$L_E \le \begin{cases} 4, bs_frame_class = FIXFIX \\ 5, bs_frame_class = VARVAR; \end{cases}$$

- число масштабных коэффициентов минимального уровня шума, N_Q, должно удовлетворять $N_Q \le 5$;

- число патчей *numPatches*, должно удовлетворять *numPatches* ≤ 5;

- для одноканальных элементов и для элементов пары каналов, где связь не используется (*bs_coupling* = 0), квантованные масштабные коэффициенты минимального уровня шума *Q* должны удовлетворять: $Q \in [0,30]$;

- для элементов пары каналов, где связь используется (*bs_coupling* = 1), квантованные масштабные коэффициенты минимального уровеня шума должны удовлетворить:

$$Q_0 \in [0,30]$$

$$Q_1 \in [0,2panOffset(1)];$$

- для элементов пары каналов, где связь используется (*bs_coupling* = 1), квантованные масштабные коэффициеты минимального уровня шума для второго декодируемого канала и квантованные масштабные коэффициенты огибающей для второго декодируемого канала должны быть четным целым числом;

- дельта-кодированные масштабные коэффициеты огибающей и масштабные коэффициенты минимального уровня шума, должны быть в пределах диапазона таблиц Хаффмана в А.6.1

- если в системе, которая работает с моно/стерео масштабируемостью, будет использоваться инструмент *SBR*, то бит *bs_coupling* должен быть установлен в единицу;

- если в системе, которая работает с полосой пропускания или масштабируемостью моно/стерео, используется инструмент *SBR*, все полезные нагрузки потока битов *SBR*, кроме части расширения стерео данных *SBR*, должны быть помещены в самый нижний уровень потока данных. Часть расширения стерео данных *SBR*, которая должна быть помещена в самый низкий уровень потока данных, переносящего данные стерео. Все данные *SBR* должны покрывать самый большой диапазон *SBR*, который может иметь место для различных уровней в потоке.

6.18.4 Блоки фильтров *SBR*

6.18.4.1 Блок фильтров анализа

Банк *QMF* используется, чтобы разделить выходной сигнал временного интервала из базового декодера на 32 сигнала поддиапазонов. Вывод из блока фильтров, то есть выборки поддиапазонов, оценивается комплексно и таким образом сверхдискретизированы с коэффициентом два по сравнению с регуляр-

ным банком *QMF*. Массив *x* принимается состоящим из 320 входных выборок временного интервала. Более высокий индекс в массив соответствует более старым выборкам. Фильтрация включает следующие шаги:

- сместить выборки в массиве *x* на 32 позициями. Самые старые 32 выборки отбрасываются, а 32 новых выборки сохраняются в позициях от 0 до 31;

- умножить выборки массива x на любой коэффициент окна *c*. Коэффициенты окна можно найти в таблице А.89;

- суммировать выборки чтобы создать массив с 64 элементами *u*.

- вычислить 32 новые выборки поддиапазона матричной операцией *Mu*,

где

$$M(k,n) = 2exp\left(\frac{i\pi(k+0,5)(2n-0,5)}{64}\right), \begin{cases} 0 \le k < 32 \\ 0 \le n < 64 \end{cases}.$$

exp () в уравнении обозначает комплексную экспоненциальную функцию, а *i* является мнимым модулем.

Каждый цикл производит 32 комплексных выборки поддиапазона, представляющих вывод от одного поддиапазона блока фильтров. Для каждого фрейма *SBR* блок фильтров будет вырабатывать выборки поддиапазона *numTimeSlots RATE* для каждого поддиапазона, соответствующие сигналу временного интервала длиной *numTimeSlots RATE* 32 выборок. *W [k] [l]* соответствует выборке поддиапазона 1 в поддиапазоне *QMF k*.

6.18.4.2 Блок фильтров синтеза

Фильтрация обрабатываемых *SBR* сигналов поддиапазона достигается использованием банка *QMF* с 64 поддиапазонами. Вывод из блока фильтров является вещественным числом выборок временного интервала. Фильтрация синтеза включает следующие шаги, где принимается массив *v*, состоящий из 1280 выборок:

- сместить выборки в массиве *v* на 128 позиций. Самые старые 128 выборок отбрасываются;

- 64 новые оцененные комплексные выборки поддиапазона умножаются на матрицу *N*, где

$$N(k,n) = \frac{1}{64} exp\left(\frac{i\pi(k+0,5)(2n-255)}{128}\right), \begin{cases} 0 \le k < 64 \\ 0 \le n < 128 \end{cases}.$$

В уравнении *exp ()* обозначает комплексную экспоненциальную функцию и *i* является мнимой единицей. Реальная часть вывода сохраняется в позициях от 0 до 127 массива *v*;

- извлечь выборки из *v*, чтобы создать массив *g* с 640 элементами;

- умножить выборки массива *g* на окно *c* чтобы создать массив *w*. Коэффициенты окна *c* смогут быть найдены в таблице А.89 и являются такими же, как для блока фильтров анализа;

- вычислить 64 новых выходных выборки суммированием выборок из массива *w*.

Каждый фрейм *SBR* производит вывод 64 выборок временного интервала *numTimeSlots · RATE*. *X [k] [l]* соответствует выборке 1 поддиапазона в поддиапазоне *QMF k*, и каждый новый цикл производит 64 выборки временного интервала как выход.

6.18.4.3 Блок фильтров субдискретизируемого синтеза

Фильтрация субдискретизируемого синтеза *SBR*-обработанных сигналов поддиапазона достигается, используя 32-канальный банк *QMF*. Вывод из блока фильтров является вещественными выборками временного интервала. Фильтрация синтеза включает следующие шаги, где массив vсостоящий из 640 выборок, принимается:

- сместить выборки в массиве *v* на 64 позиции. Самые старые 64 выборки отбрасываются;

- 32 новые комплексные выборки поддиапазона умножаются на матрицу *N*, где

$$N(k,n) = \frac{1}{64} exp\left(\frac{i\pi(k+0,5)(2n-127,5)}{64}\right), \begin{cases} 0 \le k < 32 \\ 0 \le n < 64 \end{cases}.$$

В уравнении *exp ()* обозначает комплексную экспоненциальную функцию, и *i* — мнимая единица. Реальная часть вывода из этой работы сохраняется в позициях от 0 до 63 массива *v*;

- извлечь выборки из *v* чтобы создать массив *g* с 320 элементами;

- умножить выборки массива *g* на любой другой коэффициент окна *w*. Коэффициенты окна *c* могут быть найдены в таблице А.89, и они являются такими же, как для блока фильтров анализа;

- вычислить 32 новых выходных выборки суммированием выборок из массива *w*.

Каждый фрейм *SBR* производит вывод · 32 выборок временного интервала *numTimeSlots RATE*. *X[k][l]* соответствует выборке 1 поддиапазона в поддиапазоне *QMF k*, и каждый новый цикл производит 32 выборки временного интервала как выход.

6.18.4.4 Комплексно-экспоненциальный сдвиг по фазе в комбинации с банками *SBR QMF*

Следующий подпункт разъясняет разрешенное использование фазовых сдвигов в комбинации с анализом *QMF*и блоком фильтров синтеза. Фазовые сдвиги не влияют на качество звука и допускаются, чтобы облегчить эффективные реализации. Данная реализации блока фильтров, работающая в определенном режиме частоты дискретизации, то есть при нормальной работе с двойной частотой или в режиме субдискретизируемого *SBR*, не должна иметь фазовых сдвигов, которые зависят от потока битов.

Теория для банков *QMF*, используемых в *SBR*, является комплексно-экспоненциальным расширением теории косинусно модулируемых банков фильтров. В косинусо модулируемых банках фильтров фильтры анализа $h_k(n)$ и синтеза $f_k(n)$ даются следующим образом:

$$h_k(n) = p_0(n)\cos\left(\frac{\pi}{M}(k+0{,}5)\left(n-\frac{N}{2}\right)+\theta_k\right), \begin{cases} 0 \le k < M \\ 0 \le n < N, \end{cases}$$

где

$$f_k(n) = p_0(n)\cos\left(\frac{\pi}{M}(k+0{,}5)\left(n-\frac{N}{2}\right)-\theta_k\right), \begin{cases} 0 \le k < M \\ 0 \le n < N, \end{cases}$$

где $p_0(n)$ является реальным симметричным прототипным фильтром низких частот, *M* обозначает число каналов и *N* порядок прототипного фильтра. θ_k являются зависимыми от канала факторами, необходимыми для отмены альтернативных основным терминов. Можно показать, что ограничения отмены альтернативы становятся устаревшими при расширении косинусно модулированного банка фильтров с комплексно-экспоненциальной модуляцией. Таким образом, для банков *SBR QMF* коэффициенты фильтра как анализа, так и синтеза будут

$$h_k(n) = f_k(n) = p_0(n)\exp\left(i\frac{\pi}{M}(k+0{,}5)\left(n-\frac{N}{2}\right)\right), \begin{cases} 0 \le k < M \\ 0 \le n < N. \end{cases}$$

Так как выборки поддиапазона из банка фильтров являются комплексными, банку фильтров анализа может быть добавлен дополнительно возможный зависимый от канала шаг фазового сдвига. Эти дополнительные фазовые сдвиги нужно компенсировать перед банком фильтров синтеза.

В то время как фазосдвигающие термины могут иметь произвольные значения без нарушения анализа *QMF* цепочки синтеза, они ограничиваются до определенных значений критериями соответствия. Это имеет место, потому что сигнал *SBR* будет влиять на выбор фазовых факторов, в то время как сигнала низких частот, приходящего из декодера *AAC*, не будет. На качество звука выходного сигнала это не влияет. Зависимые от канала фазовые сдвиги $\alpha(k)$ должны быть

$$\alpha(k) = \frac{\pi}{M}\left\{(k+0{,}5)\left(\varphi+\frac{N}{2}\right)+\beta\right\},$$

где φ должно быть ограничено значением являющимся целочисленным кратным числом 1/M, а β ограничено целым числом. Далее возможно идентифицировать значения $\alpha(k)$ для комплексных банков *QMF*, обрисованных в предыдущих подпунктах. Учитывая матрицу модуляции *M* или *N*, для любого из банков *QMF* выражение для коэффициентов фильтра, фазосдвигающие факторы следующие

$$\alpha(k) = -\frac{\pi}{M}(k+0{,}5)31{,}75,$$

где $\varphi = 0{,}25$ и $\beta = 32$ для $M = 32$.

6.18.5 Обзор инструмента *SBR*

Чтобы синхронизировать данные огибающей *SBR* и вывод базового декодера *AAC*, полезная нагрузка потока битов *SBR* должна быть задержан на по времени относительно полезной нагрузки базового потока битов *AAC*, то есть части *SBR* в кодере работают с вовремя задержанными относительно базового

кодера *AAC* аудиосэмплами. Чтобы достигнуть синхронизированного выходного сигнала, в декодере должны быть подтверждены следующие шаги:

- устройство деформатирования полезной нагрузки потока битов делит полезную нагрузку потока битов на две части: часть базового кодера *AAC* и часть *SBR*;

- часть полезной нагрузки потока битов *SBR* подается на синтаксический анализатор полезной нагрузки потока битов, сопровождаемый деквантизацией. Необработанные данные декодируются по Хаффману;

- часть полезной нагрузки потока битов *AAC* подается на базовый декодер *AAC*, где полезная нагрузка потока битов текущего фрейма *SBR* декодируется, приводя к блоку аудиосигнала временного интервала в 1024 выборки или 960 выборок в зависимости от размера фрейма;

- аудиоблок базового кодера подается на банк *QMF* анализа. Если используется масштабируемый базовый кодер, то должен использоваться аудиоблок, представляющий самый высокий доступный уровень;

- банк *QMF* анализа выполняет фильтрацию аудиосигнала базового кодера. В 6.18.4.1 описывается аналитический набор фильтров. Фильтруемая низкая полоса поддиапазона определяется с помощью X_{Low} согласно:

$$X_{Low}(k,l) = \begin{cases} W(k,l-t_{HFGen}), & 0 \le k < k_x,\ t_{HFGen} \le l < l_f + t_{HFGen} \\ 0, & k_x \le k < 32,\ t_{HFGen} \le l < l_f + t_{HFGen} \\ W'(k,l+l_f-t_{HFGen}), & 0 \le k < k'_x,\ 0 \le l < t_{HFGen} \\ 0, & k'_x \le k < 32,\ 0 \le l < t_{HFGen}, \end{cases}$$

где W' является матрицей W из предыдущего фрейма и k'_x является значением k_x из предыдущего фрейма и где l_f = *numTimeSlots RATE*. Если используется масштабируемый *SBR* или если используется инструмент *SBR* для чистой повышающей дискретизации без обработки *SBR*, вместо уравнения выше применяттся следующее:

$$X_{Low}(k,l) = \begin{cases} W(k,l-t_{HFGen}), & 0 \le k < 32,\ t_{HFGen} \le l < l_f + t_{HFGen} \\ W'(k,l+l_f-t_{HFGen}), & 0 \le k < 32,\ 0 \le l < t_{HFGen}; \end{cases}$$

- выход из банка анализа *QMF* задерживается на t_{HFGen} выбора поддиапазона, прежде, чем поступить в банк синтеза *QMF*. Чтобы достигнуть синхронизации $t_{HFGen} = 8$;

- генератор *HF* вычисляет X_{High}, и матрицу X_{Low}. Процесс руководствуется данными *SBR*, содержавшимися в текущем фрейме *SBR*;

- регулятор огибающей вычисляет матрицу *Y* при данной матрице X_{High} и данных огибающей *SBR*, извлеченных из полезной нагрузки потока битов *SBR*. Чтобы достигнуть синхронизации t_{HFAdj} должен быть установлен в $t_{HFAdj} = 2$, то есть регулятор огибающей работает с выборками поддиапазона $t_{HFGen} - t_{HFAdj}$ с задержанными данными;

- банк *QMF* синтеза работает с выводом из банка *QMF* анализа и выводом из регулятора огибающей. Он сначала создает матрицу *X* из этих выводов согласно:

$$X_{Low}(k,l) = \begin{cases} X_{LOW}(k,l+t_{HFAdj}), & 0 \le k < k'_x + bsco',\ 0 \le l < l_{Temp} \\ Y'(k,l+t_{HFAdj}+l_f), & k'_x + bsco' \le k < k'_x + M',\ 0 \le l < l_{Temp} \\ 0, & \max(k'_x + bsco', k'_x + M') \le k < 64,\ 0 \le l < l_{Temp} \\ X_{LOW}(k,l+t_{HFAdj}), & 0 \le k < k_x + bsco,\ l_{Temp} \le l < l_f \\ Y(k,l+t_{HFAdj}), & k_x + bsco \le k < k_x + M,\ l_{Temp} \le l < l_f \\ 0, & \max(k_x + bsco, k_x + M) \le k < 64,\ l_{Temp} \le l < l_f, \end{cases}$$

где l_f = *numTimeSlots RATE* и где l_{Temp} = *RATE* t_E' (L_E') — *numTimeSlots RATE* и ' указывает на значение предыдущего фрейма *SBR*. При запуске l_{Temp}, k_x' и *bsco'* обнуляются. Где

bsco' = 0, если не используется масштабируемый базовый кодер, для которой

bsco = max (*INT(maxAACLine* 32/*frameLength)* – k_x,0) и где

bsco = max (*INT(maxAACLine* 32/*frameLength) – lsb*,0) и где

$$maxAACLine = \begin{cases} swb_offset_long_window[\max(max_sfb - 1,0)], \text{ для длинных блоков} \\ 8\, swb_offset_short_window[\max(max_sfb - 1,0)], \text{ для коротких блоков.} \end{cases}$$

Если инструмент *SBR* используется для чисто повышающей дискретизации без обработки *SBR*, матрица *X* создается согласно:

$$x(k,l) = \begin{cases} X_{Low}(k, l + t_{HFAdj}), & 0 \le k < 32,\ 0 \le l < numTimeSlotsRATE \\ 0, & 32 \le k < 64,\ 0 \le l < numTimeSlotsRATE. \end{cases}$$

Соответственно матрица выборок $X(k,l)$, $0 \le k < 64$, $0 \le l < numTimeRATE$ синтезируется в банке *QMF* синтеза в соответствии с 6.18.4.2.

6.18.6 Генерация *HF*

6.18.6.1 Введение

Цель генератора *HF* это вставка или копирование ряда сигналов поддиапазона, полученных из анализа банка фильтров от последовательных поддиапазонов матрицы X_{Low} до последовательных поддиапазонов матрицы X_{High}. Определение вставки, то есть ряда патчей и исходных диапазонов для отдельных патчей, описывается векторами *patchNumsubbands* и *patchStartSubband*, и переменной *numPatches*. Сигналы поддиапазона X_{High} инверсно фильтруются согласно уровням инверсной фильтрации, сообщаемым из кодера.

6.18.6.2 Инверсная фильтрация

Инверсная фильтрация выполняется в два шага. Сначала выполняется линейное предсказание на сигналах поддиапазона X_{Low}. Затем производится фактическая инверсная фильтрация независимо для каждого из сигналов поддиапазона, вставленных в X_{High} генератором *HF*. Сигналы поддиапазона комплексные, что приводит к комплексным коэффициентам фильтра для линейного предсказания так же как для инверсной фильтрации. Коэффициенты фильтра прогноза получаются методом ковариации. Вычисленные элементы матрицы ковариации таковы:

$$\phi_k(i,j) = \sum_{n=0}^{numTimeSlotsRATE + 6 - l} X_{Low}(k, n - i + t_{HFAdj}) X'_{Low}(k, n - j + t_{HFAdj}), \begin{cases} 0 \le i < 3 \\ 1 \le j < 3 \\ 0 \le k < k_0. \end{cases}$$

Коэффициенты $\alpha_0(k)$ и $\alpha_1(k)$, используемые для фильтрации сигнала поддиапазона, вычисляются так:

$$d(k) = \phi_k(2,2)\phi_k(1,1) - \frac{1}{1 + \varepsilon_{Inv}} \left| \phi_k(1,2) \right|^2,$$

$$\alpha_1(k) = \begin{cases} \dfrac{\phi_k(0,1)\phi_k(1,2) - \phi_k(0,2)\phi_k(1,1)}{d(k)}, & d(k) \ne 0 \\ 0, & d(k) = 0 \end{cases}$$

$$\alpha_0(k) = \begin{cases} \dfrac{\phi_k(0,1) + \alpha_1(k)\, \phi'_k(1,2)}{\phi_k(1,1)}, & \phi_k(1,1) \ne 0 \\ 0, & \phi_k(1,1) = 0, \end{cases}$$

в первой формуле ε_{Inv} является параметром релаксации. Кроме того, если какая-либо из величин $\alpha_0(k)$ и $\alpha_1(k)$ больше или равна 4, оба коэффициента обнуляются.

Вычисление факторов *chirp, bwArray,* показано ниже. Каждый фактор *chirp* используется в рамках определенного частотного диапазона, определенного таблицей полосы частот минимального уровня шума $f_{TableNoise}$.

$$bwArray(i) = \begin{cases} 0, \text{если } tempBw(i) < 0{,}015625 \\ tempBw(i), \text{ если } tempBw(i) \ge 0{,}015625 \end{cases}, \ 0 \le i < N_Q,$$

где *tempBw(i)* вычисляется как

$$tempBw(i) = \begin{cases} 0{,}75000 newBw + 0{,}25000 bwArray'(i), \text{ если } newBw < bwArray'(i) \\ 0{,}90625 newBw + 0{,}09375 bwArray'(i), \text{ если } newBw < bwArray'(i) \end{cases}, \; 0 \le i < N_Q$$

bwArray' являются значениями *bwArray*, вычисленными в предыдущем фрейме *SBR* и являются нулем для первого фрейма *SBR*. *newBw* является функцией *bs_invf_mode* (*i*) и *bs_invf_moder'* (*i*), данной таблицей 175, где *bs_invf_moder'* являются значениями *bs_invf_mode* из предыдущего фрейма *SBR* и являются нулем для первого фрейма.

Таблица 175 — Функция *newBw*

bs_invf_mode(i)'	*bs_invf_mode(i)*			
	Выключено	Нижний	Промежуточный	Сильный
Выключено	0,0	0,6	0,9	0,98
Нижний	0,6	0,75	0,9	0,98
Промежуточный	0,0	0,75	0,9	0,98
Сильный	0,0	0,75	0,9	0,98

6.18.6.3 Генератор *HF*

Выходная переменная *numPatches* является целочисленным значением, определяющим число патчей. *patchStartSubband* и *patchNumsubbands* являются векторами, содержащими вывод данных из алгоритма решения патча.

Генерация *HF* получается согласно:

$$X_{High}(k,l + t_{HFAdj}) = X_{Low}(p,l + t_{HFAdj}) + bwArray(g(k))\alpha_0(p)X_{Low}(p,l - 1 + t_{HFAdj}) + [bwArray(g(k))]^2 \alpha_1(p)X_{Low}(p,l - 2 + t_{HFAdj}),$$

где *g(k)* определяется $g(k) f_{TableNoise}(g(k)) \le k < f_{TableNoise}(g(k) + 1)$ и

$$\begin{cases} k = k_x + x + \sum_{g=0}^{i-1} patchNumSubbands(g) \\ p = patchStartSubbands(i) + x \end{cases}$$

для

$$0 \le x < patchNumSubbands(i),\; 0 \le i < numPatches,\; RATEt_E(0) \le l < RATEt_E(L_E)$$

6.18.7 Корректировка *HF*

6.18.7.1 Введение

Регулятор огибающей принимает входную *QMF*-матрицу X_{High} и производит выходную *QMF*-матрицу *Y*. Корректировка огибающей выполняется на весь диапазон *SBR*, покрывающий *M* поддиапазонов *QMF*, начиная с поддиапазона k_x, в течение периода времени, охваченного текущим фреймом *SBR* (обозначенный вектором t_E). Все временные матрицы и векторы индексируются от нуля, удаляя смещение k_x. Корректировки огибающей является независимым от канала, и обрисовано только для одного канала, и только для одного фрейма *SBR*. Переменные получены в результате обработки предыдущего фрейма *SBR*, предполагаются нулевыми для первого фрейма *SBR*.

6.18.7.2 Отображение

Данные извлеченные из полезной нагрузки потока битов являются векторами (или матрицами), содержащими элементы данных, представляющими частотный диапазон нескольких поддиапазонов *QMF*. Эти сгруппированные данные отображаются на самую высокую доступную разрешающую способность по частоте для корректировки огибающей, то есть в отдельные поддиапазоны *QMF* в пределах диапазона *SBR*. Это означает, что у нескольких смежных поддиапазонов в отображенных векторах (или матрицах) будет то же самое значение.

Отображение масштабных коэффициентов огибающей и масштабных коэффициентов минимального уровня шума обрисовывается ниже. Огибающая *SBR* отображается на разрешение банка *QMF* с сохраненным разрешением по времени. Масштабные коэффициенты мнимального уровня шума отображается на разрешающую способность по частоте блока фильтров, но с разрешением по времени масштабных коэффициентов огибающей.

$$E_{OrigMapped}(m - k_x l) = E_{Orig}(i,l),\ F(i,r(l)) \le m < F(i+1,r(l)),\ 0 \le i < n(r(l)),\ 0 \le l < L_E$$

$$Q_{Mapped}(m - k_x l) = Q_{Orig}(i,\ k(l)),\ f_{TableNoise}(i) \le m < f_{TableNoise}(i+1),\ 0 \le i < N_Q,\ 0 \le l < L_E,$$

где $k(l)$ определяется

$$RATE \cdot t_E(l) \ge RATE \cdot t_Q(k(l)),\ RATE \cdot t_E(l+1) \le RATE \cdot t_Q(k(l)+1),\ \text{и}\ F(i,r(l))$$

индексируется как строка, столбец $F(i,r(l))$ дает $f_{TableLow}(i)$ для $r(l) = LO$ и

$$f_{TableHigh}(i) \quad \text{для } r(l) = HI.$$

Чтобы упростить, вводятся две матрицы, $S_{IndexMapped}$ и S_{Mapped}. Первая — это двоичная матрица, указывающая в каких поддиапазонах *QMF* должны быть добавлены синусоиды, последняя является матрицей используемой чтобы компенсировать значения энергии для полос частоты, где добавляется синусоида. Если полезная нагрузка потока битов указывает на синусоиду в поддиапазоне *QMF*, где не было ни одной в предыдущем фрейме *SBR*, сгенерированная синусоида должна начинаться в позиции, обозначенной l_A (таблица 176) в текущем фрейме *SBR*. Сгенерированная синусоида помещается в середине полосы с высоким разрешением по частоте, согласно следующему:

Пусть,

$$S_{Index}(i) = \begin{cases} bs_add_harmonic(i), & bs_add_harmonic_flag = 1 \\ 0, & bs_add_harmonic_flag = 0 \end{cases}, 0 \le i < N_{Higt}$$

$$S_{IndexMapped}(m - k_x, l) = \begin{cases} 0\ bs_add_harmonic(i) & \text{если } m \ne \left(\dfrac{f_{TableHigt}(i+1) + f_{TableHigt}(i)}{2}\right) \\ S_{Index}(i)\delta_{Stop}(m - k_x, l) & \text{если } m = \left(\dfrac{f_{TableHigt}(i+1) + f_{TableHigt}(i)}{2}\right) \end{cases}$$

для

$$f_{TableHigt}(i) \le m < f_{TableHigt}(i+1),\ 0 \le i < N_{High},\ 0 \le l < L_E,$$

где

$$\delta_{Step}(m_x, l) = \begin{cases} 1 \text{ если } (l \ge l_A) \text{ или } (S'_{IndexMapped}(m, L'_E - 1) = 1) \\ 0 \text{ в других случаях} \end{cases}$$

и где l_A определяется согласно таблице 176.

Таблица 176 — Таблица для вычисления l_A

bs_pointer	*bs_frame_class*		
	FIXFIX	*FIXVAR,VARVAR*	*VARFIX*
= 0	–1	–1	–1
=1	–1	L_E+1-*bs_pointer*	–1
>1	–1	L_E+1-*bs_pointer*	*bs_pointer*-1

и $S'_{IndexMapped}$ является $S_{IndexMapped}$ предыдущего фрейма *SBR* для того же самого частотного диапазона. Если частотный диапазон больше для текущего фрейма записи для поддиапазонов QMF, не охваченные предыдущим $S_{IndexMapped}$ равны нулю.

Разрешающая способность по частоте переданной информации о дополнительных синусоидах является постоянной, поэтому нужно рассмотреть переменную разрешающую способность по частоте масштабных коэффициентов огибающей. Поскольку разрешающая способность по частоте масштабных коэффициентов огибающей всегда грубее или столь же точна, как разрешающая способность дополнительных данных синусоиды, переменная разрешающая способность по частоте обрабатывается согласно нижеследующему:

$$S_{Mapped}(m - k_x, l) = \delta_S(i,l),\ l_i \le m < u_i,\ \begin{cases} u_i = F(i+1, r(l)) \\ l_i = F(i, r(l)) \end{cases}$$

для $0 \le I < n(r(l))$, $0 \le l < L_E$,

где

$$\delta_S(i,l) = \begin{cases} 1,\ 1 \in \{S_{IndexMapped}(j - k_x, l) : l_i = F(i, r(l)) \le j < l_i = F(i+1, r(l))\} \\ 0 \text{ в других случаях} \end{cases}$$

Функция $\delta_S(i,l)$ возвращает единицу, если какая-либо запись в матрице $S_{IndexMapped}$ равна единице в пределах данных границ, то есть если дополнительная синусоида присутствует в пределах текущей полосы частот. Матрица S_{Mapped} одна для всех поддиапазонов *QMF* в полосах масштабных коэффициентов где должна быть добавлена дополнительная синусоида.

6.18.7.3 Оценка текущей огибающей

Чтобы корректировать огибающую, текущий фрейм *SBR* и огибающая текущего сигнала *SBR* должны быть оценены. Это делается в зависимости от элемента данных *bs_interpol_freq*. Огибающая *SBR* оценивается путем усреднения возведенных в квадрат комплексных выборок поддиапазона за различное время и частотные области, данные сеткой времени/частоты, представленной t_E и r.

Если используется интерполяция (*bs_interpol_freq* = 1):

$$E_{Curr}(m,l) = \frac{\sum_{i = RATEt_E(l) + t_{HFAdj}}^{RATEt_E(l+1) - 1 + t_{HFAdj}} \left|X_{High}(m + k_x, l)\right|^2}{(RATEt_E(l+1) - RATEt_E(l))},\ 0 \le m < M,\ 0 \le l < L_E$$

иначе, никакая интерполяция не используется (*bs_interpol_freq* = 0):

$$E_{Curr}(k - k_x, l) = \frac{\sum_{i = RATEt_E(l) + t_{HFAdj}}^{RATEt_E(l+1) - 1 + t_{HFAdj}} \sum_{j = k_l}^{k_h} \left|X_{High}(j,l)\right|^2}{(RATEt_E(l+1) - RATEt_E(l))(k_h - k_l + 1)},$$

$$k_l \le k < k_h,\ \begin{cases} k_l = F(p, r(l)) \\ k_h = F(p+1, r(l)) - 1 \end{cases},\ 0 \le p < n(r(l)),\ 0 \le l < L_E$$

Если интерполяция используется, энергии усредняются по каждому поддиапазону банка фильтров *QMF filterbank*, иначе энергии усредняются по каждой полосе частот. В любом случае энергии сохраняются с разрешающей способностью по частоте банка фильтров *QMF*. Следовательно матрица E_{Curr} имеет L_E столбцов (один для каждой огибающей *SBR*) и M строк (число поддиапазонов *QMF*, охваченных диапазоном *SBR*).

6.18.7.4 Вычисление уровней дополнительных компонентов сигнала *HF*

Масштабный коэффициент минимального уровня шума является отношением между энергией шума, который будет добавлен к сгенерированному *HF*-сигналу X_{High} с регулируемой огибающей, и энергией его самого. Следовательно, чтобы добавить корректный уровень шума, масштабный коэффициент

минимального уровня шума должен быть преобразован в надлежащее амплитудное значение, согласно следующему.

$$Q_M(m,l) = \sqrt{E_{OrigMapped}(m,l)\,\frac{Q_{Mapped}(m,l)}{1+Q_{Mapped}(m,l)}},\ 0 \le m < M,\ 0 \le l < L_E$$

Уровни синусоид получаются из масштабных коэффициентов огибающей *SBR* согласно следующему.

$$S_M(m,l) = \sqrt{E_{OrigMapped}(m,l)\,\frac{S_{Mapped}(m,l)}{1+Q_{Mapped}(m,l)}},\ 0 \le m < M,\ 0 \le l < L_E.$$

6.18.7.5 Вычисление усиления

Усиление, которое будет применено для выборок поддиапазона, чтобы сохранить корректную огибающую, вычисляется соответственно выражению ниже. Уровень дополнительных синусоид, так же как уровень дополнительного добавленного шума, принимаются во внимание.

$$G_M(m,l) = \begin{cases} \sqrt{\dfrac{E_{OrigMapped}(m,l)}{(\varepsilon + E_{Curr}(m,l))(1+\delta(l)Q_{Mapped}(m,l))}} & \text{если } S_{Mapped}(m,l) = 0 \\ \sqrt{\dfrac{E_{OrigMapped}(m,l)}{(\varepsilon + E_{Curr}(m,l))}\,\dfrac{Q_{Mapped}(m,l)}{(1+Q_{Mapped}(m,l))}} & \text{если } S_{Mapped}(m,l) \ne 0 \end{cases}, \quad 0 \le m < M,\ 0 \le l < L_E,$$

где

$$\delta(l) = \begin{cases} 0 \text{ если } l = l_A \text{ или } l = l_{APref} \\ -1 \text{ в других случаях} \end{cases}$$

и где

$$l_{APref} = \begin{cases} 0 \text{ если } l'_A = L'_E \\ -1 \text{ в других случаях} \end{cases}$$

вводится путем извлечения из l'_A и L'_E, которые являются значениями l_A и L_E предыдущего фрейма *SBR*.

Чтобы избежать подстановки нежелательного шума, значения усиления ограничиваются согласно следующему. Полный уровень определенной полосы ограничителя корректируется, чтобы компенсировать потерю энергии, наложенную ограничителем.

$$G_{Max_{Temp}}(k,l) = \sqrt{\frac{\varepsilon_0 + \sum_{i=f_{TableLim}(k)-K_x}^{f_{TableLim}(k+1)-1-K_x} E_{OrigMapped}(i,l)}{\varepsilon_0 + \sum_{i=f_{TableLim}(k)-K_x}^{f_{TableLim}(k+1)-1-K_x} E_{Curr}(i,l)}}\ limGain(bs_limiter_gains),\ 0 \le k < N_E,\ 0 \le l < L_E$$

$$G_{Max}(m,l) = \min\left(G_{Max_{Temp}}(k(m),l), 10^5\right)\ 0 \le m < M,\ 0 \le l < L_E,$$

где $k(m)$ определяется как

$$f_{TableLim}(k(m)) \le m + k_x < f_{TableLim}(k(m)+1),$$

и где

$$limGain = [0{,}70795,\ 1{,}0,\ 1{,}41254,\ 10^{10}] \text{ и } \varepsilon = 10^{-12}.$$

Дополнительный шум, добавленный к сгенерированному сигналу *HF*, ограничивается в пропорции к потере энергии из-за ограничения значений усиления согласно следующему:

$$Q_{M_{Lim}}(m,l) = \min\left(Q_M(m,l), Q_M(m,l)\frac{G_{Max}(m,l)}{G(m,l)}\right),\ 0 \le m < M,\ 0 \le l < L_E.$$

Значения усиления ограничиваются согласно следующему:

$$G_{Lim}(m,l) = \min(G(m,l), G_{\mathrm{Max}}(m,l)),\ 0 \le m < M,\ 0 \le l < L_E.$$

Ограничитель компенсируют корректируя общее усиление для полосы ограничителя, в пропорции к потерянной из-за ограничения энергии. Это вычисляется согласно следующему:

$$G_{Boost_{Temp}}(k,l) = \sqrt{\frac{\varepsilon_0 + \sum_{i=f_{TableLim}(k)-k_x}^{f_{TableLim}(k+1)-1-k_x} E_{OrigMapped}(i,l)}{\varepsilon_0 + \sum_{i=f_{TableLim}(k)-k_x}^{f_{TableLim}(k+1)-1-k_x} \left(E_{Curr}(i,l)G^2_{Lim}(i,l) + S^2_M(i,l) + \delta(S_M(i,l),l)Q^2_{M_{Lim}}(i,l)\right)}}$$

для $0 \le k < N_L$, $0 \le l < L_E$,

где

$$\delta(S_M(i,l),l) = \begin{cases} 0 \text{ если } S_M(i,l) \ne 0 \text{ или } l = l_A \text{ или } l = l_{APrev} \\ -1 \text{ в других случаях} \end{cases}.$$

Компенсация, или коэффициент усиления, ограничивается, чтобы не получить слишком высокие значения энергии, согласно:

$$G_{Boost}(m,l) = \min\left(G_{Boost_{Temp}}(k(m),l),\ 1{,}584893192\right),\ 0 \le m < M,\ 0 \le l < L_E,$$

где $k(m)$ определяется

$$f_{TableLim}(k(m)) \le m + k_x < f_{TableLim}(k(m)+1)$$

и где $\varepsilon_0 = 10^{-12}$.

Эта компенсация применяется к усилению, масштабным коэффициентам минимального уровня шума и уровню синусоиды, согласно нижеприведенному.

$$G_{LimBoost}(m,l) = G_{Lim}(m,l), G_{Boost}(m,l),\ 0 \le m < M,\ 0 \le l < L_E,$$

$$Q_{M_{LimBoost}}(m,l) = Q_{M_{Lim}}(m,l), G_{Boost}(m,l),\ 0 \le m < M,\ 0 \le l < L_E,$$

$$S_{MBoost}(m,l) = S_M(m,l), G_{Boost}(m,l),\ 0 \le m < M,\ 0 \le l < L_E.$$

6.18.7.6 Сборка сигналов *HF*

Аналогично отображению данных огибающей *SBR* и данных минимального уровня шума в более высокую разрешающую способность по времени и по частоте значения усиления, представляющие промежуток нескольких подвыборок *QMF*, отображаются в наиболее высокое разрешение по времени, доступное для корректировки огибающей, то есть в отдельные подвыборки *QMF* в рамках текущего фрейма *SBR*.

Значения усиления, которые будут применены к выборкам поддиапазона, сглаживаются, используя фильтр h_{Smooth}. Переменная h_{SL} используется, чтобы выбрать, применяется ли сглаживание или нет, в соответствии с:

$$h_{SL} = \begin{cases} 4,\ bs_smoothing_mode = 0 \\ 0,\ bs_smoothing_mode = 1. \end{cases}$$

Используемый фильтр определяется как следующее:

$$h_{Smooth} = \begin{bmatrix} 0{,}33333333333333 \\ 0{,}30150283239582 \\ 0{,}21816949906249 \\ 0{,}11516383427084 \\ 0{,}03183050093751 \end{bmatrix}.$$

Сглаженные значения усиления G_{Filt} вычисляются согласно следующему уравнению:

$$G_{Temp}(m,i + h_{SL}) = G_{LimBoost}(m,l),\ RATE\ t_E(l) \le i < RATE\ t_E(l + 1),\ 0 \le m < M,\ 0 \le l < L_E$$

$$G_{Filt}(m,i) = \begin{cases} \sum_{j=0}^{h_{SL}} G_{Temp}(m,i - j + h_{SL}) h_{Smooth}(j) \text{ если } l \ne l_A \text{ и } l \ne l_{APrev} \\ G_{Temp}(m,i + h_{SL}) \text{ в других случаях} \end{cases}$$

для $RATE\ t_E(l) \le i < RATE\ t_E(l + 1),\ 0 \le m < M,\ 0 \le l < L_E$.

Первые h_{SL} столбцов матрицы G_{Temp} являются последними h_{SL} столбцами матрицы G_{Temp} предыдущего фрейма *SBR*, если флаг сброса устанавливается (*reset* = 1), в этом случае первые h_{SL} столбцов матрицы G_{Temp} равны $G_{LimBoost}(m,0)$ для всех поддиапазонов *QMF* в пределах диапазона *SBR*.

Сглаженные значения усиления применяются к входной матрице поддиапазонов X_{High}, для всех огибающих *SBR* текущего фрейма *SBR*, согласно:

$$W_1(m,i) = G_{Filt}(m,l) X_{High}(m + k_x, i + t_{HfAdj}),\ \ RATE\ t_E(0) \le i < RATE\ t_E(L_E),\ 0 \le m < M.$$

Уровень шума сглаживается аналогично сглаживанию значений усиления, используя фильтр h_{Smooth} длины h_{SL}.

$$G_{Temp}(m,i + h_{SL}) = G_{M_{LimBoost}}(m,l),\ RATE\ t_E(l) \le i < RATE\ t_E(l + 1),\ 0 \le m < \mathrm{M},\ 0 \le l < L_E$$

$$G_{Filt}(m,i) = \begin{cases} Q_{Temp}(m,i) \text{ если } l \ne l_A \text{ и } l \ne l_{APrev} \text{ и } S_{MBoost}(m,l) = 0 \text{ и } h_{SL} = 0 \\ \sum_{j=0}^{h_{SL}} Q_{Temp}(m,i - j + h_{SL}) h_{Smooth}(j) \text{ если } l \ne l_A \text{ и } l \ne l_{APrev} \text{ и } S_{MBoost}(m,l) = 0 \text{ и } h_{SL} \ne 0 \\ 0 \text{ в других случаях} \end{cases}$$

для $RATE\ t_E(l) \le i < RATE\ t_E(l + 1),\ 0 \le m < M,\ 0 \le l < L_E$.

Первые h_{SL} столбцов матрицы Q_{Temp} являются последними h_{SL} столбцами матрицы Q_{Temp} предыдущего фрейма *SBR*, если флаг сброса устанавливается (*reset* = 1), в этом случае первые h_{SL} столбцов матрицы Q_{Temp} равны $Q_{MLimBoost}(m,0)$ для всех поддиапазонов *QMF* в пределах диапазона *SBR*.

Шум, основанный на шумовой таблице *V* (таблица А.91), добавляется к выводу согласно:

$$\begin{cases} Re\{w_2(m,i)\} = Re\{w_2(m,i)\} + Q_{Filt}(m,i) V(0, f_{IndexNoise}(i)) \\ Im\{w_2(m,i)\} = Im\{w_2(m,i)\} + Q_{Filt}(m,i) V(1, f_{IndexNoise}(i)) \end{cases}, \begin{cases} RATE\ t_E(l) \le i < RATE\ t_E(l+1) \\ 0 < m < M,\ 0 \le l < L_E, \end{cases}$$

где $f_{IndexNoise}(i) = (index_{Noise} + (i - RATE\ t_E(0))M + m + 1) mod (512)$,

и $index_{Noise}$ является последним $f_{IndexNoise}$ из предыдущего фрейма *SBR*, если флаг сброса устанавливается (*reset*=1) для этого случая $index_{Noise}$ =0.

В уравнении выше, $V(0, f_{IndexNoise(i)}) = \varphi_{ReNoise}$, $(f_{IndexNoise(i)})$ и $V(1, f_{IndexNoise(i)}) = \varphi_{InNoise}(f_{IndexNoise(i)})$, где $\varphi_{ReNoise}(i)$ и $\varphi_{InNoise}(i)$ определяются в таблице А.91.

Синусоиды добавляются на уровне, данном $S_{MBoost}(m,l)$ для поддиапазонов *QMF*, указанных $S_{IndexMapped}(m, 1)$. Это дает окончательную матрицу выхода *QMF Y*, согласно:

$$\begin{cases} Re\{w_2(m,i)\} = Re\{Y(m+k_x, i+t_{HFAdj})\} = Re\{w_2(m,i)\} + \psi_{Re}(m,l,i) \\ Im\{w_2(m,i)\} = Im\{Y(m+k_x, i+t_{HFAdj})\} = Im\{w_2(m,i)\} + \psi_{Im}(m,l,i) \end{cases}, \begin{cases} RATE\,t_E(l) \le i < RATE\,t_E(l+1) \\ 0 \le m < M,\ 0 \le l < L_E, \end{cases}$$

где

$$\psi_{Re}(m,l,i) = S_{MBoost}(m,l)\varphi_{Re,sin}(f_{IndexSine}(i))$$

$$\psi_{Im}(m,l,i) = S_{MBoost}(m,l)(-1)^{m+k_x}\varphi_{Im,sin}(f_{IndexSine}(i))$$

и где φ и $f_{IndexSine}$ определяются ниже как:

$$\begin{cases} \varphi_{Re,sin} = [1,0,-1,0] \\ \varphi_{Im,sin} = [0,1,0,-1] \end{cases} \text{и } f_{IndexSine}(i) = (index_{Sine} + i - RATE \cdot t_E(0))mod(4),$$

$$\begin{cases} \varphi_{Re,sin} = [1,0,\ -1,0] \\ \varphi_{Im,sin} = [1,0,\ -1,0] \end{cases} \text{и } f_{IndexSine}(i) = (index_{Sine} + i - RATE\ t_E(0))mod(4),$$

$index_{Sine}$ = (последний $f_{IndexSine}$ из предыдущего фрейма SBR +1)$mod(4)$, или $index_{Sine} = 0$ для первого фрейма.

6.18.8 Инструмент *SBR* малой мощности

6.18.8.1 Введение

Инструмент *SBR* малой мощности работает на вещественных сигналах, и, следовательно, используется вещественный блок фильтров. Инструмент *SBR* малой мощности включает дополнительные модули, чтобы уменьшить искажение, вносимое из-за обработки вещественной части.

6.18.8.2 Блоки фильтров инструмента *SBR* малой мощности

6.18.8.2.1 Введение

Для инструмента *SBR* малой мощности используются действительные блоки фильтров. Следовательно, блоки фильтров, описанные в подпункте 6.18.4, должны быть заменены следующими блоками фильтров анализа и синтеза.

6.18.8.2.2 Блок фильтров вещественнозначного анализа

Вещественнозначный банк *QMF* используется, чтобы разделить выходной сигнал временного интервала из базового декодера на 32 сигнала поддиапазонов. Выход из банка фильтров, то есть выборки поддиапазона, являются вещественночисленными и критично выбранными. Массив *x* принимается состоящий из 320 входных выборок временного интервала. Более высокий индекс в массиве соответствует более старым выборкам. Фильтрация включает следующие шаги:

- сместить выборки в массиве *x* на 32 позициями. Самые старые 32 выборки отбрасываются и 32 новых выборки сохраняются в позициях от 0 до 31;
- умножить выборки массива *x* на другой коэффициент окна *c*. Коэффициенты окна можно найти в таблице А.89;
- суммировать выборки согласно формуле в блок-схеме, чтобы создать массив с 64 элементами *u*;
- вычислить новые 32 выборки поддиапазона с помощью матрицы $M_r u$, где

$$M_r(k,n) = 2\cos\left(\frac{\pi(k+0{,}5)(2n-96)}{64}\right), \begin{cases} 0 \le k < 32 \\ 0 \le n < 64 \end{cases}.$$

Каждый цикл производит 32 выборки поддиапазона, каждый из которых представляет вывод из одного поддиапазона блока фильтров. Для каждого фрейма *SBR* блок фильтров вырабатывает выборки поддиапазона *numTimeSlots RATE* для каждого поддиапазона, соответственно сигналу временного интервала длины 32 выборки *numTimeSlots RATE. W [k] [l]* соответствует выборке 1 поддиапазона *QMF k*.

6.18.8.2.3 Блок фильтров вещественнозначного синтеза

Фильтрация синтеза сигналов поддиапазона *SBR*-обработки достигается использованием банка *QMF* с 64 поддиапазонами. Выход из банка фильтров является вещественнозначными выборками вре-

менного интервала. Массив *v* принимается состоящий из 1280 выборок Фильтрация синтеза включает следующие шаги:

- сместить выборки в массиве *v* на 128 позиций. Самые старые 128 выборок отбрасываются;
- 64 новые выборки поддиапазона умножаются на матрицу N_r, где

$$N_r(k,n) = \frac{1}{32}\cos\left(\frac{\pi(k + 0{,}5)(2n - 64)}{128}\right), \begin{cases} 0 \le k < 64 \\ 0 \le n < 128 \end{cases};$$

- вывод этой операции сохраняется в позициях от 0 до 127 массива *v*;
- извлечь выборки из *v*, чтобы создать 640-элементный массив *e*;
- умножить выборки массива *g* на окно *c*, чтобы создать массив *w*. Коэффициенты окна *c* могут быть найдены в таблице А.89, и являются теми же, как для блока фильтров анализа.
- вычислить 64 новых выходных выборки суммированием выборок из массива *w*.

Каждый фрейм *SBR* дает выход из 64 выборок временного интервала *numTimeSlots · RATE. X [k] [l]* соответствует выборке l поддиапазона *QMF k*, и каждый новый цикл вырабатывает 64 выборки временного интервала в качестве выхода.

6.18.8.2.4 Субдискретизированный блок фильтров вещественнозначного синтеза

Фильтрация синтеза *SBR*-обработанных сигналов поддиапазона достигается, используя банк *QMF* с 32 каналами. Вывод из блока фильтров является вещественнозначными выборками временного интервала. Массив *v* принимается состоящий из 640 выборок Фильтрация синтеза включает следующие шаги:

- сместить выборки в массиве *v* на 64 позициями. Самые старые 64 выборки отбрасываются;
- 32 новых выборки поддиапазона умножаются на матрицу N_r, где

$$N_r(k,n) = \frac{1}{32}\cos\left(\frac{\pi(k + 0{,}5)(2n - 32)}{64}\right), \begin{cases} 0 \le k < 32 \\ 0 \le n < 64 \end{cases};$$

- выход из этой операции сохраняется в позициях от 0 до 61 массива *v*;
- извлечь выборки из v чтобы создать массив *g* с 320 элементами;
- умножить выборки массива *g* на коэффициенты окна *c* через один, чтобы получить массив *w*. Коэффициенты окна *c* могут быть найдены в таблице А.89, и такие же как для блока фильтров анализа;
- вычислить 32 новых выходных выборки суммированием выборок из массива *w*.

Каждый фрейм *SBR* производит вывод 32 выборок временного интервала *umTimeSlots · RATE. X [k] [l]* соответствует выборке 1 поддиапазона *QMF k*, и каждый новый цикл производит 32 выборки временного интервала в качестве выхода.

6.18.8.3 Обнаружение искажений

Чтобы минимизировать внесение искажений регулятором огибающей, идентифицируются поддиапазоны *QMF*, где потенциально будет внесено сильное искажение. Модуль обнаружения использует данные из модуля генерации *HF* и из модуля корректировки *HF*.

Алгоритм обнаружения искажения вычисляет коэффициент отражения для каждого поддиапазона в нижней полосе.

$$ref(k) = \begin{cases} \min\left(\max\left(-\frac{f_k(0,1)}{f_k(1,1)}, -1\right)\right) \text{ если } f_k(1,1) \neq 0 \\ 0 \text{ в других случаях} \end{cases}, 0 \le k < k_0$$

Учитывая коэффициенты отражения *ref*, вычисляется степень искажения *deg* для нижней полосы.

Степень искажения в верхней полосе получается при использовании информации о патче, доступной в 6.18.6, согласно:

$$degPatched(k) = \begin{cases} 0 \text{ если } x = 0 \\ deg(p) \text{ в других случаях} \end{cases},$$

где

$$\begin{cases} k = k_x + x + \sum_{g=0}^{i-1} patchNumSubbands(g) \\ p = patchStartSubbands(i) + x \end{cases}, 0 \le x < patchNumSubbands(i),\ 0 \le i < numPatches.$$

Так как информация о патче, возможно, не охватывает целый диапазон *SBR*, степень искажения в частотной области от места, где патч заканчивается, до места, где заканчивается диапазон *SBR*, определяются так:

$$degPatched(k) = 0,\ k_x + \sum_{g+0}^{numPatches-1} patchNumSubbands(g) \le k < k_x + M.$$

Кроме того, алгоритм сокращения искажений нуждается в таблице, чтобы указать группировку значений усиления. Эта таблица F_{Group} имеет L_E векторов длины $2\,n_G(l)$, представляющих желательную группировку усиления для каждой огибающей *SBR* фрейма *SBR*. Таблица отличается от предыдущих таблиц в смысле текста, так как у нее есть отдельные индексы запуска и остановки для каждой группы частот, тогда как для прежних таблиц индекс остановки предыдущей группы берется в качестве индекса запуска текущей группы. Следовательно, вектор, представляющий H_G групп, имеет длину $2H_G$ записей, тогда как таблица ранее используемого стиля была бы длиной H_G+1 записей. Индекс остановки группы является исключенным, то есть поддиапазон индекса остановки не включается в группу.

6.18.8.4 Модификация вычисления энергии

Так как версия инструмента *SBR* малой мощности не использует комплекснозначное представление сигналов, требуется модификация вычисления энергии.

Данные уравнения:

$$E_{Curr}(m,l) = \frac{\sum_{i=RATEt_E(l)+t_{HFAdj}}^{RATEt_E(l+1)-1+t_{HFAdj}} \left|X_{High}(m+k_x,i)\right|^2}{(RATEt_E(l+1) - RATEt_E(l))},\ 0 \le m < M,\ 0 \le l < L_E$$

и

$$E_{Curr}(k-k_x,l) = \frac{\sum_{i=RATEt_E(l)+t_{HFAdj}}^{RATEt_E(l+1)-1+t_{HFAdj}} \sum_{j=k_l}^{k_h} \left|X_{High}(j,i)\right|^2}{(RATEt_E(l+1) - RATEt_E(l))(k_h - k_l + 1)},$$

заменяются

$$E_{Curr}(m,l) = \frac{2\sum_{i=RATEt_E(l)+t_{HFAdj}}^{RATEt_E(l+1)-1+t_{HFAdj}} \left|X_{High}(m+k_x,i)\right|^2}{(RATEt_E(l+1) - RATEt_E(l))},\ 0 \le m < M,\ 0 \le l < L_E$$

и

$$E_{Curr}(k-k_x,l) = \frac{2\sum_{i=RATEt_E(l)+t_{HFAdj}}^{RATEt_E(l+1)-1+t_{HFAdj}} \sum_{j=k_l}^{k_h} \left|X_{High}(j,i)\right|^2}{(RATEt_E(l+1) - RATEt_E(l))(k_h - k_l + 1)}.$$

6.18.8.5 Уменьшение искажения

Модуль уменьшения искажения пересчитывает значения усиления, вычисленные модулем корректировки *HF*. Переменные корректировки *HF*, используются модулем уменьшения искажения. Для реализации с малой мощностью выходная переменная из модуля уменьшения искажения G_A должна использоваться вместо $G_{LimBoost}$.

Энергия сигналов в поддиапазонах, на которые оказано влияние, если использовались расчетные значения усиления $G_{LimBoost}$, была бы:

$$E_{Total}(k,l) = \sum_{i=F_{Group}(2k,i)-k_x}^{F_{Group}(2k+1,i)-1-k_x} G^2_{LimBoost}(i,l)E_{Curr}(i,l),\ 0 \le k < n_G(l),\ 0 \le l < L_E.$$

Учитывая эту целевую энергию E_{Total}, значение целевого усиления вычисляется следующим образом:

$$G^2_{Targer}(k,l) = \frac{E_{Total}(k,l)}{\varepsilon_0 + \sum_{i=F_{Group}(2k,i)-k_x}^{F_{Group}(2k+1,i)-1-k_x} E_{Curr}(i,l)},\ 0 \le k < n_G(l),\ 0 \le l < L_E.$$

С учетом вычисленного выше целевого усиления, вычисляется новое значение усиления как взвешенная сумма исходного значения усиления и вновь вычисленного целевого усиления:

$$G^2_{ARtemp}(m - k_x,l) = \alpha(m)G^2_{Targer}(k,l) + (1 - \alpha(m))G^2_{LimBoost}(m - k_x,l),$$
$$F_{Groop}(2k,l) \le m < F_{Groop}(2k + 1,l)$$
$$0 \le k < n_G(l),\ 0 \le l < L_E,$$

где

$$\alpha(m) = \begin{cases} \max(degPatched(m), degPatched(m+1)) & \text{если } m < M = k_x - 1 \\ degPatched(m) & \text{если } m = M = k_x - 1 \end{cases},$$

где *degPatched(m),* вычисленное в части обнаружения искажения, используется в качестве степени выравнивания усиления между поддиапазоном *m*-1 и поддиапазоном *m*.

Новое значение энергии вычисляется основанная на базе новых значений усиления, согласно:

$$E_{TotalNew}(k,l) = \sum_{i=F_{Group}(2k,i)-k_x}^{F_{Group}(2k+1,i)-1-k_x} G^2_{ARtemp}(i,l)E_{Curr}(i,l),\ 0 \le k < n_G(l),\ 0 \le l < L_E.$$

Чтобы сохранить корректную выходную энергию, в то же время ограничивая регулировку усиления, чтобы не вносить искажения, значение усиления G_A вычисляется согласно:

$$G_A(m - k_x,i) = \begin{cases} G_{ARtemp}(m - k_x,l)\sqrt{\frac{E_{Total}(k(m),l)}{\varepsilon_0 + E_{TotalNew}(k(m),l)}}, & m \in A_G(l) \\ G_{ARtemp}(m - k_x,l), & m \notin A_G(l) \end{cases},\ k_x \le m < M + k_x - 1,\ 0 \le l < L_E,$$

где

$$A_G(l) = \bigcup_{k=0}^{n_G(l)-1} \{m: F_{Group}(2k,l) \le m < F_{Group}(2k+1,l)\},$$

и для $m \notin A_G$ определяется k_m при $F_{Group}(2k(m),l) \le m < F_{Group}(2k(m)+1,l)$.

Значения G_A являются новыми значениями усиления, которые должны использоваться вместо значений $G_{LimBoost}$ в 6.18.7.6.

Для варианта малой мощности инструмента *SBR* процесс сглаживания усиления, описанный в 6.18.7.6, не применяется независимо от значения *bs_smoothing_mode.*

Для добавленных в 6.18.7.6 синусоид, требуются изменения для варианта малой мощности инструмент SBR. Следующие уравнения:

$$\begin{cases} Re\{Y(m+k_x,i+t_{HFAdj})\} = Re\{w_2(m,i)\} + \psi_{Re}(m,l,i) \\ Im\{Y(m+k_x,i+t_{HFAdj})\} = Im\{w_2(m,i)\} + \psi_{Im}(m,l,i) \end{cases},\ \begin{cases} RATEt_E(l) \le i < RATEt_E(l+1) \\ 0 \le m < M,\ 0 \le l < L_E \end{cases},$$

где

$$\psi_{Re}(m,l,i) = S_{MBoost}(m,l)\varphi_{Re,sin}(f_{IndexSine}(i))$$

$$\psi_{Im}(m,l,i) = S_{MBoost}(m,l)(-1)^{m+k_X}\varphi_{Im,sin}(f_{IndexSine}(i))$$

заменяются:

$$Re\{Y(m+k_x,i+t_{HFAdj})\} = \begin{cases} \psi_m(m,l,i), & m=-1 \\ Re\{w_2(m,i)+\psi_{Re}(m,l,i), & 0\le m<M \\ \psi_m(m,l,i), & m=M \text{ и } m+k_X<64 \end{cases},$$

где

$$-1 \le m < M, RATE\ t_E(l) \le i < RATE\ t_E(l+1),\ 0 \le l < L_E$$

$$\psi_m(m,l,i) = \psi_{Re}(m,l,i) - 0{,}00815(-1)^{m+k_X}(\psi_{Re}(m-1,l,i-1) + \psi_{Re}(m+1,l,i+1))$$

$$\psi_{Re}(m,l,i) = S_{MBoost}(m,l)\psi_{Re,sin}(f_{IndexSine}(i))$$

и где $S_{MBoost}(m,l) = 0$ для m < 0 или $m \ge M$.

Вышеупомянутые модификации делаются только для первых 16 (подсчитанных в порядке возрастания частоты) синусоид, для каждого временного сегмента.

Кроме того, так как к $Y(k_X - l, i)$ может быть добавлен сигнал, согласно вышеупомянутому, то есть нижней полосы, или $Y(k_X + M,i)$, то есть один поддиапазон *QMF* выше диапазона *SBR*, следующее уравнение в подпункта 6.18.5 должно быть изменено:

$$X(k,l) = \begin{cases} X_{LOW}(k,l+t_{HFAdj}), & 0 \le k < k'_x + bsco',\ 0 \le l < l_{Temp} \\ Y'(k,l+t_{HFAdj}+l_f), & k'_x + bsco' \le k < k'_x + M',\ 0 \le l < l_{Temp} \\ 0, & \max(k'_x + bsco', k'_x + M') \le k < 64,\ 0 \le l < l_{Temp} \\ X_{LOW}(k,l+t_{HFAdj}), & 0 \le k < k_x + bsco,\ l_{Temp} \le l < l_f \\ Y(k,l+t_{HFAdjf}), & k_x + bsco \le k < k_x + M,\ l_{Temp} \le l < l_f \\ 0, & \max(k_x + bsco, k_x + M) \le k < 64,\ l_{Temp} \le l < l_f \end{cases}$$

Вышеприведенное заменяется на

$$X(k,l) = \begin{cases} X_{LOW}(k,l+t_{HFAdj}), & 0 \le k < k'_x - 1 + bsco',\ 0 \le l < l_{Temp} \\ X_{LOW}(k,l+t_{HFAdj}) + Y'(k,l+t_{HFAdj}+l_f), & k = k'_x - 1 + bsco',\ 0 \le l < l_{Temp} \\ Y'(k,l+t_{HFAdj}+l_f), & k'_x + bsco' \le k \le \min(k'_x + M', 63),\ 0 \le l < l_{Temp} \\ 0, & \begin{cases} \max(k'_x + bsco', k'_x + M') \le k < 64 \\ 0 \le l < l_{Temp} \end{cases} \\ X_{LOW}(k,l+t_{HFAdj}), & 0 \le k < k_x - 1 + bsco,\ l_{Temp} \le l < l_f \\ X_{LOW}(k,l+t_{HFAdj}) + Y(k,l+t_{HFAdjf}), & k < k_x - 1 + bsco,\ l_{Temp} \le l < l_f \\ Y(k,l+t_{HFAdjf}), & \begin{cases} k_x + bsco \le k \le \min(k'_x + M', 63) \\ l_{Temp} \le l < l_f \end{cases} \\ 0, & \begin{cases} \max(k_x + bsco, k_x + M) \le k < 64 \\ l_{Temp} \le l < l_f \end{cases} \end{cases}$$

6.19 *SBR* с низкой задержкой

6.19.1 Введение

SBR с низкой задержкой получается из стандартного инструмента *SBR*, чтобы использоваться как кодер расширения полосы частот в коммуникационных сценариях. Таким образом, алгоритмическая задержка этого инструмента минимизируется, чтобы достигнуть полной задержки достаточно низкой для двухсторонних коммуникационных приложений.

Сводка модификаций:

- длина фрейма, принятая для базового кодека с 512 или 480 выборками на фрейм;
- заблокированная фреймом сетка времени/частоты;
- минимизация задержки в буфере *QMF*;
- использование комплексного банка фильтров с малой задержкой.

Инструмент *SBR* с малой задержкой определяется следующими модификациями относительно стандартного алгоритма (то есть, аудиообъектного типа *SBR*).

6.19.2 Определения, константы и переменные

6.19.2.1 Определения (изменения для подпункта 6.18.2.1.20)

Временной интервал: наименьшее разрешение во времени для огибающих *SBR* и минимального уровня шума. Один временной интервал равняется одной подвыборке в домене *QMF*.

6.19.2.2 Константы (изменения для 6.18.2.5)

RATE: Для базового кодека *ELD AAC* вместо *RATE*= 2 должна использоваться постоянная *RATE*= 1.

6.19.2.3 Переменные (изменения для 6.18.2.6)

numTimeSlots: Число временных интервалов для базового кодека *ELD AAC numTimeSlots* = 16 для фрейма *AAC* на 512, и *numTimeSlots* = 15 для фреймов *AAC* на 480.

t_{HFGen}: Из-за удаленной дополнительной задержки смещение модуля генерации *HF* устанавливается в t_{HFGen} = 2.

6.19.2.4 Инверсная фильтрация (изменения для 6.18.6.2)

Из-за измененного управления буфером вычисление матрицы ковариации должно быть изменено следующим образом:

Верхний предел *n* должен быть изменен с *numTimeSlots* · *RATE* + 6-1 на *numTimeSlots* · RATE – 1.

6.19.3 Заблокированная по времени сетка частот

Сетка времени/частоты для *SBR* с малой задержкой определяется в следующих подпунктах.

6.19.3.1 Классы фрейма

SBR с малой задержкой использует уменьшенный набор классов фрейма, которые перечисляются в таблице 177.

Таблица 177 — *bs_frame_class*

bs_frame_class	Значение
0	*FIXFIX*
1	*LD_TRAN*

6.19.3.2 *sbr_ld_grid ()*

SBR с малой задержкой использует иной синтаксис, чтобы сигнализировать о данных сетки. Синтаксис стандарта *sbr_grid ()* заменяется *sbr_ld_grid ()*, как определено в таблице 178.

Таблица 178 — Синтаксис *sbr_ld_grid*

Синтаксис	Количество битов	Мнемоника
sbr_ld_grid(ch)		
{		
switch (bs_frame_class) {	1	*uimsbf*
case FIXFIX:		
bs_num_env[ch] = 2^ tmp;	2	*uimsbf,* Примечание 1:
if (bs_num_env[ch] == 1)		
bs_amp_res;	1	*uimsbf*

Окончание таблицы 178

Синтаксис	Количество битов	Мнемоника
bs_freq_res[ch][0]; *for (env = 1; env < bs_num_env[ch]; env++)* *bs_freq_res[ch][env] = bs_freq_res[ch][0];* *break;* *case LD_TRAN:* *bs_transient_position* *bs_num_env[ch] =* *LD_Envelope_Table[bs_transient_position][num_envelopes];* *for (env = 0; env < bs_num_env[ch]; env++)* *bs_freq_res[ch][env];* *break;* *}* *if (bs_num_env[ch] > 1)* *bs_num_noise[ch] = 2;* *else* *bs_num_noise[ch] = 1; }*	1 4 1	*uimsbf* Примечание 2
Примечание 1 — *bs_num_env* ограничивается согласно 6.18.3.6. Примечание 2 — Таблица *LD_Envelope_Table* дается в таблице 179		

6.19.3.3 Вычисление $t_E(l)$ (изменения для 6.18.3.3)

В случае, если *bs_frame_class* = *LD_TRAN*:

$t_E(0) = 0$

$t_E(L_E) = numberTimeSlots$

$t_E(l) = LD_EnvelopeTable\ [bs_transient_position]\ [l+1]\ for\ 0 < l < (L_E)$

6.19.3.4 Вычисление l_A (изменения для 6.18.7.2)

В случае, если *bs_frame_class* = *LD_TRAN*:

$l_A = LD_EnvelopeTable\ [bs_transient_position]\ [4]$

6.19.3.5 Таблица поиска огибающей

Таблица 179 — Таблица поиска для *LD_Envelope_Table (bs_transient_position)*

bs_transient_position	*num_envelopes*	*border[1]*	*border[2]*	*transientIdx*
0	2	4	—	0
1	2	5	—	0
2	3	2	6	1
3	3	3	7	1
4	3	4	8	1
5	3	5	9	1
6	3	6	10	1
7	3	7	11	1
8	3	8	12	1
9	3	9	13	1
10*	3/2	10	14/—	1
11	2	11	—	1
12	2	12	—	1
13	2	13	—	1
14	2	14	—	1
15	2	15	—	1

* в случае фрейма *AAC* = 480 используют вторую запись таблицы.

6.19.4 Блок фильтров *SBR* с малой задержкой (изменения для 6.18.4)

Вместо 6.18.4 используйте описание ниже для обработки блока фильтров. Отличаются только работа с окнами и модуляция.

6.19.4.1 Блок фильтров анализа

Сместить выборки в массиве *x* 32 на позиции. Самые старые 32 выборки отбрасываются и 32 новых выборки сохраняются в позициях от 0 до 31.

Умножить выборки массива *x* на коэффициент окна *ci*. Коэффициенты окна *ci* получают линейной интерполяцией коэффициентов *c*, то есть с помощью уравнения

$$ci(i) = \frac{1}{2}[c(2i+1) + c(2i)],\ 0 \le i < 320.$$

Коэффициенты окна *c* можно найти в таблице А.90.

Суммировать выборки согласно формуле в блок-схеме на рисунке 42, чтобы создать массив *u* с 64 элементами.

Вычислить 32 новых выборки поддиапазона, используя матрицу *Mu*, где

$$M(k,n) = 2\exp\left(\frac{i\pi(k+0{,}5)(2n-96)}{64}\right),\ \begin{cases} 0 \le k < 32 \\ 0 \le n < 64 \end{cases}.$$

В этом уравнении exp () обозначает комплексную экспоненциальную функцию, и *i* -мнимый модуль.

6.19.4.2 Блок фильтров синтеза

Сместить выборки в массиве *v* на 128 позиции. Самые старые 128 выборок отбрасываются.

64 новые комплекснозначные выборки поддиапазона умножаются на матрицу *N*, где

$$N(k,n) = \frac{1}{64}\exp\left(\frac{i\pi(k+0{,}5)(2n-63)}{128}\right),\ \begin{cases} 0 \le k < 64 \\ 0 \le n < 128 \end{cases}.$$

В уравнении exp () обозначает комплексную экспоненциальную функцию, и *i* - мнимый модуль. Реальная часть выхода из этой операции сохраняется в позициях от 0 до 127 массива *v*.

Извлечь выборки из *v* чтобы создать *g* массива с 640 элементами.

Умножить выборки массива *g* на окно *c*, чтобы создать массив w. Коэффициенты окна *c* могут быть найдены в таблице А.90.

Вычислить 64 новые выходные выборки суммированием выборок из массива *w*.

6.19.4.3 Блок фильтров субдискретизируемого синтеза

Сместить выборки в массиве *v* на 64 позициями. Самые старые 64 выборки отбрасываются.

32 новый комплекснозначные выборки поддиапазона умножаются на матрицау *N*, где

$$N(k,n) = \frac{1}{64}\exp\left(\frac{i\pi(k+0{,}5)(2n-31)}{128}\right),\ \begin{cases} 0 \le k < 32 \\ 0 \le n < 64 \end{cases}.$$

В уравнении exp () обозначает комплексную экспоненциальную функцию, и *i* – мнимый модуль. Действительная часть выхода этой операции сохраняется в позициях от 0 до 63 массива *v*.

Извлечь выборки из *v* чтобы создать массив *g* с 320 элементами.

Умножить выборки массива *g* на коэффициенты окна *ci*, чтобы создать массив *w*. Коэффициенты окна *ci* получаются линейной интерполяцией коэффициентов *c*, то есть с помощью уравнения

$$ci(i) = \frac{1}{2}[c(2i+1) + c(2i)],\ 0 \le i < 320.$$

Коэффициенты окна *c* могут быть найдены в таблице А.90.

Вычислить 32 новые выходные выборки суммированием выборок из массива *w*.

6.19.4.4 Комплексно-экспоненциальный сдвиг фазы в комбинации с блоком фильтров *SBR* с низкой задержкой

Этот подпункт описывает разрешенное использование фазовых сдвигов в комбинации с комплексно-экспоненциально модулированными блоками фильтров анализа и синтеза с низкой задержкой, описанными в предыдущих подпунктах. Фазовые сдвиги не влияют на качество звука, и допускаются, чтобы облегчить

эффективные реализации. Теория для банков фильтров, используемых в *SBR* с низкой задержкой, является расширением теории косинусно модулированных банков фильтров. В косинусно модулированных банках фильтры анализа $h_k(n)$ и синтеза $f_k(n)$ даются с помощью:

$$h_k(n) = p_0(n)\cos\left(\frac{\pi}{M}(k+0{,}5)\left(n-\frac{O}{2}\right)+\theta_k\right), \begin{cases} 0 \le k < M \\ 0 \le n < N \end{cases}$$

и

$$f_k(n) = p_0(n)\cos\left(\frac{\pi}{M}(k+0{,}5)\left(n-\frac{O}{2}\right)-\theta_k\right), \begin{cases} 0 \le k < M \\ 0 \le n < N \end{cases},$$

где $p_0(n)$ является вещественнозначимым прототипным фильтром низких частот длины N, M обозначает число каналов банка фильтров, и O является порядком прототипного фильтра $p_0(n)$. θ_k являются зависимыми от канала факторами, необходимыми для отмены основных терминов помехи. Можно показать, что отмены помехи становится устаревшей при расширении косинусно модулированного банка фильтров с комплексно-экспоненциальной модуляцией. Таким образом, для комплекснозначных банков *SBR* низкой задержки коэффициенты фильтра и анализа, и синтеза могут быть получены из

$$h_k(n) = f_k(n) = p_0(n)\exp\left(i\frac{\pi}{M}(k+0{,}5)\left(n-\frac{O}{2}\right)\right), \begin{cases} 0 \le k < M \\ 0 \le n < N \end{cases}.$$

Так как выборки поддиапазона из банка фильтров являются комплекснозначимыми, дополнение возможно, к банку фильтров анализа может быть добавлен зависимый от канала шаг фазового сдвига. Эти дополнительные сдвиги фазы должны быть инвертированы перед банком фильтров синтеза.

Хотя фазосдвигающие термины в принципе могут иметь произвольные значения, не нарушая работу цепочки банка фильтров анализа/синтеза, они ограничиваются определенными значениями. Это происходит потому, что на сигнал *SBR* будет влиять выбор фазовых постоянных, в то время как на сигнал нижней полосы, приходящий из декодера *AAC*, не будет. Конкретнее, зависимые от канала фазовые сдвиги $\alpha(k)$ должны быть

$$\alpha(k) = \frac{\pi}{M}\left\{(k+0{,}5)\left(\varphi+\frac{O}{2}\right)+\beta\right\},$$

где φ должно быть ограничено значением, являющимся кратным числом $1/M$ и β должно быть ограничено целым числом. Далее возможно идентифицировать значения $\alpha(k)$ для комплексных банков фильтров, описанных в предыдущих подпунктах. Учитывая матрицу модуляции M или N, для любого из банков фильтров в 6.19.4.1—6.19.4.3, и выражение для коэффициентов фильтра выше, где порядок O фильтра = 639 для M=64 и O = 319 для M=32, фазосдвигающие факторы будут

$$\alpha(k) = \frac{\pi}{2}(k+0{,}5).$$

Это соответствует, например, β = 0 и $\varphi = \frac{M-O}{2}$. Можно отметить что эти значения $\alpha(k)$ являются точными значениями, необходимыми для отмены основных помех косинусно модулированного банка фильтров. В этом причина, почему аргументы в косинусных выражениях в действительнозначимых банках фильтров (6.19.5.1—6.19.5.3) не изменяются по сравнению с аргументами в комплексно-экспоненциальных выражениях для банков фильтров, описанных в 6.19.4.1—6.19.4.3.

6.19.5 Банк фильтро маломощного *SBR* (изменения в 6.18.8.2)

6.19.5.1 Банк фильтров действительнозначного анализа

Сместить выборки в массиве *x* на 32 позиции. Самые старые 32 выборки отбрасываются и 32 новых выборки сохраняются в позициях от 0 до 31.

Умножить выборки массива x на коэффициент окна *ci*. Коэффициенты окна *ci* получаются линейной интерполяцией коэффициентов *c* с помощью уравнения

$$ci(i) = \frac{1}{2}[c(2i+1)+c(2i)],\ 0 \le i < 320.$$

Коэффициенты окна *c* могут быть найдены в таблице А.90.

Суммировать выборки согласно формуле в блок-схеме, чтобы создать массив с 64 элементами *u*.

Вычислите новые 32 выборки поддиапазона с помощью матрицы *M*,*u*, где

$$M(k,n) = 2\cos\left(\frac{\pi(k+0{,}5)(2n-95)}{64}\right), \begin{cases} 0 \le k < 32 \\ 0 \le n < 64 \end{cases}.$$

6.19.5.2 Банк фильтров вещественнозначного синтеза

Сместить выборки в массиве *v* на 128 позиций. Самые старые 128 выборок отбрасываются.

64 новый выборки поддиапазон умножаются на матрицу N_r, где

$$N(k,n) = \frac{1}{32}\cos\left(\frac{\pi(k+0{,}5)(2n-63)}{128}\right), \begin{cases} 0 \le k < 64 \\ 0 \le n < 128 \end{cases}.$$

Выход от этой операции сохраняется в позициях от 0 до 127 массива *v*.

Извлечь выборки из *v* чтобы создать массив *g* с 640 элементами.

Умножить выборки массива *g* на окно *c*, чтобы создать массив *w*. Коэффициенты окна *c* могут быть найдены в таблице А.90.

Вычислить 64 новых выходных выборки суммированием выборок из массива *w*.

6.19.5.3 Банк фильтров субдискретизированного вещественнозначного синтеза

Сместить выборки в массиве *v* на 64 позиции. Самые старые 64 выборки отбрасываются.

32 новый выборки поддиапазон умножаются на матрицу N_r, где

$$N(k,n) = \frac{1}{32}\cos\left(\frac{\pi(k+0{,}5)(2n-31)}{64}\right), \begin{cases} 0 \le k < 32 \\ 0 \le n < 64 \end{cases}.$$

Выход этой операции сохранен в позициях от 0 до 61 массива *v*.

Извлечь выборки из *v* чтобы создать массив *g* с 320 элементами.

Умножить выборки массива *g* на коэффициент окна *ci*, чтобы создать массив *w*. Коэффициенты окна *ci* получаются линейной интерполяцией коэффициентов *c* с помощью уравнения

$$ci(i) = \frac{1}{2}[c(2i+1) + c(2i)],\ 0 \le i < 320.$$

Коэффициенты окна *c* могут быть найдены в таблице А.90.

Вычислить 32 новых выходных выборки суммированием выборок из массива *w*.

6.19.6 Снижение искажение маломощного *SBR* (изменения в подпункт 6.18.8.5)

Для корректной снижения искажения, используя маломощный *AAC-ELD*, добавить следующий вектор

$$\varphi_{Re,cos} = [0{,}74384511, -0{,}66835201, -0{,}74384511, 0{,}668335201].$$

В 6.18.8.5 уменьшение искажения, заменяют следующий набор уравнений

$$-1 \le m < M,RATE\ t_E(l) \le i < RATE\ t_E(l+1),\ 0 \le l < L_E$$

$$\psi_m(m,l,i) = \psi_{Re}(m,l,i) - 0{,}00815(-1)^{m+lsb}(\psi_{Re}(m-1,l,i-1) + \psi_{Re}(m+1,l,i+1))$$

$$\psi_{Re}(m,l,i) = S_{IndexMapped}(m,l)S_{MBoost}(m,l)\psi_{Re,sin}(f_{IndexSine}(i))$$

набором

$$-1 \le m < M,RATE\ t_E(l) \le i < RATE\ t_E(l+1),\ 0 \le l < L_E$$

$$\psi_m(m,l,i) = \psi_{Re}(m,l,i) - 0{,}16733\,((-1)^{(m+lsb)f_{IndexSine}(i)}\psi'_{Re}(m-1,l,i-1) + (-1)^{(m+lsb+1)}\psi'_{Re}(m+1,l,i))$$

$$\psi_{Re}(m,l,i) = S_{IndexMapped}(m,l)S_{MBoost}(m,l)\psi_{Re,\sin}(f_{IndexSine}(i))$$

$$\psi'_{Re}(m,l,i) = S_{IndexMapped}(m,l)S_{MBoost}(m,l)\psi_{Re,\cos}(f_{IndexSine}(i)).$$

6.20 Расширенный кодек с малой задержкой

6.20.1 Архитектура кода

Схема расширенного кодирования с малой задержкой (*AAC-ELD*) обеспечивает расширение функциональности кодирования с малой задержкой, описанной в 6.17. Наименьшая алгоритмическая задержка этого кодека составляет 15 мс. Этот кодек позволяет использовать инструмент *SBR* с малой задержкой, чтобы добиться схемы кодирования с низкой скоростью передачи и низкой задержкой для коммуникационных приложений.

Кодек *ER AAC ELD* получается из кодека *ER AAC LD*, описанного в 6.17 (*ER AAC LD*). Чтобы достигнуть достаточно низкой задержки, особенно в сочетании с низкой задержкой *SBR*, необходимы некоторые модификации, например, использование окна с низкой задержкой (см. 6.20.2).

Для *ER AAC ELD* флаг *ldSbrPresentFlag* в *ELDSpecificConfig* (см. 6.20.3) определяет использование инструмент *SBR* с малой задержкой. Методы сигнализации *SBR* для других *AOTs*, как описано в 1.6.5.1, не разрешены для *ER AAC ELD*. Флаг *ldSbrSamplingRate* определяет поведение инструмент *SBR* с малой задержкой относительно режима двойной скорости и субдискретизированного режима, как описано в 6.20.4.

6.20.2 Окно с малой задержкой

Банк фильтров синтеза изменяется, чтобы принять блок фильтров с малой задержкой. Базовый алгоритм *IMDCT* главным образом неизменен, но с более длинным окном, так что n теперь доходит до $2N$-1 (а не до N-1).

$$x_{i,n} = -\frac{2}{N}\sum_{k=0}^{\frac{N}{2}-1} spec[i][k]\cos\left(\frac{2\pi}{N}\left(n+n_0\right)\left(k+\frac{1}{2}\right)\right), для\ 0 \le n < N,$$

где:

n — индекс выборки;

i — индекс окна;

k — индекс спектрального коэффициента;

N — длина окна;

$n_0 = (-N/2 + 1)/2$ с $N = 960$ или 1024.

Работа с окнами и перекрытие — добавление изменяются по сравнению с 6.11.3.2, 6.11.3.3, 6.17.2.3 следующим образом:

Длина окна N заменяется длиной окна $2N$ с большим перекрытием в прошлом и меньшим перекрытием для будущего (значения $N/8$ являются фактически нулем).

Работа с окнами для Окна с малой задержкой:

$$Z(i) = w_{LD}(n)x_{i,n},$$

где теперь у окна длина $2N$, следовательно $n = 0,\ldots..,\ 2N\text{-}1$, с коэффициентами, перечисленными в таблице А.15 для $N = 1024$ и таблице А.16 для $N = 960$.

Наложить и добавить:

$$out_{i,n} = Z_{i,n} + Z_{i-1,n+\frac{N}{2}} + Z_{i-2,n+N} + Z_{i-3,n+N+\frac{N}{2}}$$

для $0 <= n < N/2$.

6.20.3 *ELDSpecificConfig*

Таблица 180 — Синтаксис *ELDSpecificConfig ()*

Синтаксис	Количество битов	Мнемосхема
ELDSpecificConfig (channelConfiguration)		
{		
frameLengthFlag;	1	*bslbf*
aacSectionDataResilienceFlag;	1	*bslbf*
aacScalefactorDataResilienceFlag;	1	*bslbf*
aacSpectralDataResilienceFlag;	1	*bslbf*
ldSbrPresentFlag;	1	*bslbf*

Окончание таблицы 180

Синтаксис	Количество битов	Мнемосхема
If (ldSbrPresentFlag) { *ldSbrSamplingRate;* *ldSbrCrcFlag;* *ld_sbr_header(channelConfiguration);* *}* *while (eldExtType != ELDEXT_TERM) {* *eldExtLen;* *len = eldExtLen;* *if (eldExtLen == 15) {* *eldExtLenAdd;* *len += eldExtLenAdd;* *}* *if (eldExtLenAdd == 255) {* *eldExtLenAddAdd;* *len += eldExtLenAddAdd;* *}* *switch (eldExtType) {* */* add future eld extension configs here */* *default:* *for(cnt=0; cnt<len; cnt++) {* *other_byte;* *}* *break;* *}* *}* *}*	 1 1 4 4 8 16 8	 *bslbf* *bslbf* *bslbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf*

Таблица 181 — Синтаксис *ld_sbr_header ()*

Синтаксис	Количество битов	Мнемосхема
ld_sbr_header(channelConfiguration) *switch (channelConfiguration){* *case 1:* *case 2:* *numsbrHeader = 1;* *break;* *case 3:* *numsbrHeader = 2;* *break;* *case 4:* *case 5:* *case 6:* *numsbrHeader = 3;* *break;* *case 7:* *numsbrHeader = 4;* *break;* *default:* *numsbrHeader = 0;* *break;* *}* *for (el=0; el<numsbrHeader; el++) {* *sbr_header();* *}* *}*		

6.20.4 Декодирование *ELDSpecificConfig*

Семантика элементов синтаксиса *ELDSpecificConfig* описывается ниже.

frameLengthFlag — См. *GASpecificConfig ()* (5.1.1).

aacSectionDataResilienceFlag — См. *GASpecificConfig ()* (5.1.1).

aacScalefactorDataResilienceFlag — См. *GASpecificConfig ()* (5.1.1).

aacSpectralDataResilienceFlag — См. *GASpecificConfig ()* (5.1.1).

ldSbrPresentFlag — *ldSbrPresentFlag* соответствует переменной *sbrPresentFlag* используемый для *SBR* с не малой задержкой. У декодера то же самое поведение относительно этого флага.

ldSbrCrcFlag — Переменная *ldSbrCrcFlag* сигнализирует о синтаксисе *CRC* для *SBR* с малой задержкой. Вычисление *CRC* включает только полезную нагрузку без каких-либо битов выравнивания байта.

ldSbrSamplingRate — *ldSbrSamplingRate* определяет фактор частоты дискретизации между базовым кодером и *SBR*, где 0 ставится для единичной скорости и 1 для двойной скорости.

eldExtType — Четырехбитовый код, который идентифицирует тип расширения согласно таблице 182.

eldExtLen — Дескриптор длины старой конфигурации расширения в байтах

eldExtLenAdd — Первое дополнительное поле длины старой конфигурации расширения в байтах

eldExtLenAddAdd — Второе дополнительное поле длины старой конфигурации расширения в байтах

other_byte — Переменная помощника для анализа полезной нагрузки неизвестной конфигурации

Таблица 182 — *ELD extension_type*

Символ	Значение *extension_type*	Назначение
ELDEXT TERM	‘0000’	Тег завершения
Зарезервировано	‘0001’	Зарезервировано
…	…	…
Зарезервировано	‘1111’	Зарезервировано

Приложение А
(обязательное)

Нормативные таблицы

Таблицы сборника кодов Хаффмана для бесшумного кодирования *AAC*-типа

Т а б л и ц а А.1 — Сборник кодов Хаффмана для масштабного коэффициента

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
0	18	3ffe8	61	4	a
1	18	3ffe6	62	4	c
2	18	3ffe7	63	5	1b
3	18	3ffe5	64	6	39
4	19	7fff5	65	6	3b
5	19	7fff1	66	7	78
6	19	7ffed	67	7	7a
7	19	7fff6	68	8	f7
8	19	7ffee	69	8	f9
9	19	7ffef	70	9	1f6
10	19	7fff0	71	9	1f9
11	19	7fffc	72	10	3f4
12	19	7fffd	73	10	3f6
13	19	7ffff	74	10	3f8
14	19	7fffe	75	11	7f5
15	19	7fff7	76	11	7f4
16	19	7fff8	77	11	7f6
17	19	7fffb	78	11	7f7
18	19	7fff9	79	12	ff5
19	18	3ffe4	80	12	ff8
20	19	7fffa	81	13	1ff4
21	18	3ffe3	82	13	1ff6
22	17	1ffef	83	13	1ff8
23	17	1fff0	84	14	3ff8
24	16	fff5	85	14	3ff4
25	17	1ffee	86	16	fff0
26	16	fff2	87	15	7ff4
27	16	fff3	88	16	fff6

Окончание таблицы А.1

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
28	16	fff4	89	15	7ff5
29	16	fff1	90	18	3ffe2
30	15	7ff6	91	19	7ffd9
31	15	7ff7	92	19	7ffda
32	14	3ff9	93	19	7ffdb
33	14	3ff5	94	19	7ffdc
34	14	3ff7	95	19	7ffdd
35	14	3ff3	96	19	7ffde
36	14	3ff6	97	19	7ffd8
37	14	3ff2	98	19	7ffd2
38	13	1ff7	99	19	7ffd3
39	13	1ff5	100	19	7ffd4
40	12	ff9	101	19	7ffd5
41	12	ff7	102	19	7ffd6
42	12	Ff6	103	19	7ffd2
43	11	7f9	104	19	7ffdf
44	12	ff4	105	19	7ffe7
45	11	7f8	106	19	7ffe8
46	10	3f9	107	19	7ffe9
47	10	3f7	108	19	7ffea
48	10	3f5	109	19	7ffeb
49	9	1f8	110	19	7ffe6
50	9	1f7	111	19	7ffe0
51	8	fa	112	19	7ffe1
52	8	f8	113	19	7ffe2
53	8	f6	114	19	7ffe3
54	7	79	115	19	7ffe4
55	6	3a	116	19	7ffe5
56	6	38	117	19	7ffd7
57	5	1a	118	19	7ffec
58	4	b	119	19	7fff4
59	3	4	120	19	7fff3
60	1	0			

Таблица А.2 — Сборник кодов Хаффмана 1 для спектра

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
0	11	7f8	41	5	14
1	9	1f1	42	7	65
2	11	7fd	43	5	16
3	10	3f5	44	7	6d
4	7	68	45	9	1e9
5	10	3f0	46	7	63
6	11	7f7	47	9	1e4
7	9	1ec	48	7	6b
8	11	7f5	49	5	13
9	10	3f1	50	7	71
10	7	72	51	9	1e3
11	10	3f4	52	7	70
12	7	74	53	9	1f3
13	5	11	54	11	7fe
14	7	76	55	9	1e7
15	9	1eb	56	11	7f3
16	7	6c	57	9	1ef
17	10	3f6	58	7	60
18	11	7fc	59	9	1ee
19	9	1e1	60	11	7f0
20	11	7f1	61	9	1e2
21	9	1f0	62	11	7fa
22	7	61	63	10	3f3
23	9	1f6	64	7	6a
24	11	7f2	65	9	1e8
25	9	1ea	66	7	75
26	11	7fb	67	5	10
27	9	1f2	68	7	73
28	7	69	69	9	1f4
29	9	1ed	70	7	6e
30	7	77	71	10	3f7
31	5	17	72	11	7f6
32	7	6f	73	9	1e0
33	9	1e6	74	11	7f9
34	7	64	75	10	3f2
35	9	1e5	76	7	66
36	7	67	77	9	1f5
37	5	15	78	11	7ff
38	7	62	79	9	1f7
39	5	12	80	11	7f4
40	1	0			

Таблица А.3 — Сборник кодов Хаффмана 2 для спектра

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
0	9	1f3	41	5	7
1	7	6f	42	6	1d
2	9	1fd	43	5	b
3	8	eb	44	6	30
4	6	23	45	8	ef
5	8	ea	46	6	1c
6	9	1f7	47	7	64
7	8	e8	48	6	1e
8	9	1fa	49	5	c
9	8	f2	50	6	29
10	6	2d	51	8	f3
11	7	70	52	6	2f
12	6	20	53	8	f0
13	5	6	54	9	1fc
14	6	2b	55	7	71
15	7	6e	56	9	1f2
16	6	28	57	8	f4
17	8	e9	58	6	21
18	9	1f9	59	8	e6
19	7	66	60	8	f7
20	8	f8	61	7	68
21	8	e7	62	9	1f8
22	6	1b	63	8	ee
23	8	f1	64	6	22
24	9	1f4	65	7	65
25	7	6b	66	6	31
26	9	1f5	67	4	2
27	8	ec	68	6	26
28	6	2a	69	8	ed
29	7	6c	70	6	25
30	6	2c	71	7	6a
31	5	a	72	9	1fb
32	6	27	73	7	72
33	7	67	74	9	1fe
34	6	1a	75	7	69
35	8	f5	76	6	2e
36	6	24	77	8	f6
37	5	8	78	9	1ff
38	6	1f	79	7	6d
39	5	9	80	9	1f6
40	3	0			

Таблица А.4 — Сборник кодов Хаффмана 3 для спектра

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
0	1	0	41	10	3ef
1	4	9	42	9	1f3
2	8	ef	43	9	1f4
3	4	b	44	11	7f6
4	5	19	45	9	1e8
5	8	f0	46	10	3ea
6	9	1eb	47	13	1ffc
7	9	1e6	48	8	f2
8	10	3f2	49	9	1f1
9	4	a	50	12	ffb
10	6	35	51	10	3f5
11	9	1ef	52	11	7f3
12	6	34	53	12	ffc
13	6	37	54	8	ee
14	9	1e9	55	10	3f7
15	9	1ed	56	15	7ffe
16	9	1e7	57	9	1f0
17	10	3f3	58	11	7f5
18	9	1ee	59	15	7ffd
19	10	3ed	60	13	1ffb
20	13	1ffa	61	14	3ffa
21	9	1ec	62	16	ffff
22	9	1f2	63	8	f1
23	11	7f9	64	10	3f0
24	11	7f8	65	14	3ffc
25	10	3f8	66	9	1ea
26	12	ff8	67	10	3ee
27	4	8	68	14	3ffb
28	6	38	69	12	ff6
29	10	3f6	70	12	ffa
30	6	36	71	15	7ffc
31	7	75	72	11	7f2
32	10	3f1	73	12	ff5
33	10	3eb	74	16	fffe
34	10	3ec	75	10	3f4
35	12	ff4	76	11	7f7
36	5	18	77	15	7ffb
37	7	76	78	12	ff7
38	11	7f4	79	12	ff9
39	6	39	80	15	7ffa
40	7	74			

Таблица А.5 — Сборник кодов Хаффмана 4 для спектра

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
0	4	7	41	7	6b
1	5	16	42	8	e3
2	8	f6	43	7	69
3	5	18	44	9	1f3
4	4	8	45	8	eb
5	8	ef	46	8	e6
6	9	1ef	47	10	3f6
7	8	f3	48	7	6e
8	11	7f8	49	7	6a
9	5	19	50	9	1f4
10	5	17	51	10	3ec
11	8	ed	52	9	1f0
12	5	15	53	10	3f9
13	4	1	54	8	f5
14	8	e2	55	8	ec
15	8	f0	56	11	7fb
16	7	70	57	8	ea
17	10	3f0	58	7	6f
18	9	1ee	59	10	3f7
19	8	f1	60	11	7f9
20	11	7fa	61	10	3f3
21	8	ee	62	12	fff
22	8	e4	63	8	e9
23	10	3f2	64	7	6d
24	11	7f6	65	10	3f8
25	10	3ef	66	7	6c
26	11	7fd	67	7	68
27	4	5	68	9	1f5
28	5	14	69	10	3ee
29	8	f2	70	9	1f2
30	4	9	71	11	7f4
31	4	4	72	11	7f7
32	8	e5	73	10	3f1
33	8	f4	74	12	ffe
34	8	e8	75	10	3ed
35	10	3f4	76	9	1f1
36	4	6	77	11	7f5
37	4	2	78	11	7fe
38	8	e7	79	10	3f5
39	4	3	80	11	7fc
40	4	0			

Таблица А.6 — Сборник кодов Хаффмана 5 для спектра

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
0	13	1fff	41	4	a
1	12	ff7	42	7	71
2	11	7f4	43	8	f3
3	11	7e8	44	11	7e9
4	10	3f1	45	11	7ef
5	11	7ee	46	9	1ee
6	11	7f9	47	8	ef
7	12	ff8	48	5	18
8	13	1ffd	49	4	9
9	12	ffd	50	5	1b
10	11	7f1	51	8	eb
11	10	3e8	52	9	1e9
12	9	1e8	53	11	7ec
13	8	f0	54	11	7f6
14	9	1ec	55	10	3eb
15	10	3ee	56	9	1f3
16	11	7f2	57	8	ed
17	12	ffa	58	7	72
18	12	ff4	59	8	e9
19	10	3ef	60	9	1f1
20	9	1f2	61	10	3ed
21	8	e8	62	11	7f7
22	7	70	63	12	ff6
23	8	ec	64	11	7f0
24	9	1f0	65	10	3e9
25	10	3ea	66	9	1ed
26	11	7f3	67	8	f1
27	11	7eb	68	9	1ea
28	9	1eb	69	10	3ec
29	8	ea	70	11	7f8
30	5	1a	71	12	ff9
31	4	8	72	13	1ffc
32	5	19	73	12	ffc
33	8	ee	74	12	ff5
34	9	1ef	75	11	7ea
35	11	7ed	76	10	3f3
36	10	3f0	77	10	3f2
37	8	f2	78	11	7f5
38	7	73	79	12	ffb
39	4	b	80	13	1ffe
40	1	0			

Таблица А.7 — Сборник кодов Хаффмана 6 для спектра

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
0	11	7fe	41	4	3
1	10	3fd	42	6	2f
2	9	1f1	43	7	73
3	9	1eb	44	9	1fa
4	9	1f4	45	9	1e7
5	9	1ea	46	7	6e
6	9	1f0	47	6	2b
7	10	3fc	48	4	7
8	11	7fd	49	4	1
9	10	3f6	50	4	5
10	9	1e5	51	6	2c
11	8	ea	52	7	6d
12	7	6c	53	9	1ec
13	7	71	54	9	1f9
14	7	68	55	8	ee
15	8	f0	56	6	30
16	9	1e6	57	6	24
17	10	3f7	58	6	2a
18	9	1f3	59	6	25
19	8	ef	60	6	33
20	6	32	61	8	ec
21	6	27	62	9	1f2
22	6	28	63	10	3f8
23	6	26	64	9	1e4
24	6	31	65	8	ed
25	8	eb	66	7	6a
26	9	1f7	67	7	70
27	9	1e8	68	7	69
28	7	6f	69	7	74
29	6	2e	70	8	f1
30	4	8	71	10	3fa
31	4	4	72	11	7ff
32	4	6	73	10	3f9
33	6	29	74	9	1f6
34	7	6b	75	9	1ed
35	9	1ee	76	9	1f8
36	9	1ef	77	9	1e9
37	7	72	78	9	1f5
38	6	2d	79	10	3fb
39	4	2	80	11	7fc
40	4	0			

Таблица А.8 — Сборник кодов Хаффмана 7 для спектра

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
0	1	0	32	8	f3
1	3	5	33	8	ed
2	6	37	34	9	1e8
3	7	74	35	9	1ef
4	8	f2	36	10	3ef
5	9	1eb	37	10	3f1
6	10	3ed	38	10	3f9
7	11	7f7	39	11	7fb
8	3	4	40	9	1ed
9	4	c	41	8	ef
10	6	35	42	9	1ea
11	7	71	43	9	1f2
12	8	ec	44	10	3f3
13	8	ee	45	10	3f8
14	9	1ee	46	11	7f9
15	9	1f5	47	11	7fc
16	6	36	48	10	3ee
17	6	34	49	9	1ec
18	7	72	50	9	1f4
19	8	ea	51	10	3f4
20	8	f1	52	10	3f7
21	9	1e9	53	11	7f8
22	9	1f3	54	12	ffd
23	10	3f5	55	12	ffe
24	7	73	56	11	7f6
25	7	70	57	10	3f0
26	8	eb	58	10	3f2
27	8	f0	59	10	3f6
28	9	1f1	60	11	7fa
29	9	1f0	61	11	7fd
30	10	3ec	62	12	ffc
31	10	3fa	63	12	fff

Таблица А.9 — Сборник кодов Хаффмана 8 для спектра

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
0	5	e	32	7	71
1	4	5	33	6	2b
2	5	10	34	6	2d
3	6	30	35	6	31
4	7	6f	36	7	6d
5	8	f1	37	7	70
6	9	1fa	38	8	f2
7	10	3fe	39	9	1f9
8	4	3	40	8	ef
9	3	0	41	7	68
10	4	4	42	6	33
11	5	12	43	7	6b
12	6	2c	44	7	6e
13	7	6a	45	8	ee
14	7	75	46	8	f9
15	8	f8	47	10	3fc
16	5	f	48	9	1f8
17	4	2	49	7	74
18	4	6	50	7	73
19	5	14	51	8	ed
20	6	2e	52	8	f0
21	7	69	53	8	f6
22	7	72	54	9	1f6
23	8	f5	55	9	1fd
24	6	2f	56	10	3fd
25	5	11	57	8	f3
26	5	13	58	8	f4
27	6	2a	59	8	f7
28	6	32	60	9	1f7
29	7	6c	61	9	1fb
30	8	ec	62	9	1fc
31	8	fa	63	10	3ff

Таблица А.10 — Сборник кодов Хаффмана 9 для спектра

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
0	1	0	85	12	fda
1	3	5	86	12	fe3
2	6	37	87	12	fe9
3	8	e7	88	13	1fe6
4	9	1de	89	13	1ff3
5	10	3ce	90	13	1ff7
6	10	3d9	91	11	7d3
7	11	7c8	92	10	3d8
8	11	7cd	93	10	3e1
9	12	fc8	94	11	7d4
10	12	fdd	95	11	7d9
11	13	1fe4	96	12	fd3
12	13	1fec	97	12	fde
13	3	4	98	13	1fdd
14	4	c	99	13	1fd9
15	6	35	100	13	1fe2
16	7	72	101	13	1fea
17	8	ea	102	13	1ff1
18	8	ed	103	13	1ff6
19	9	1e2	104	11	7d2
20	10	3d1	105	10	3d4
21	10	3d3	106	10	3da
22	10	3e0	107	11	7c7
23	11	7d8	108	11	7d7
24	12	fcf	109	11	7e2
25	12	fd5	110	12	fce
26	6	36	111	12	fdb
27	6	34	112	13	1fd8
28	7	71	113	13	1fee
29	8	e8	114	14	3ff0
30	8	ec	115	13	1ff4
31	9	1e1	116	14	3ff2
32	10	3cf	117	11	7e1
33	10	3dd	118	10	3df
34	10	3db	119	11	7c9
35	11	7d0	120	11	7d6
36	12	fc7	121	12	fca
37	12	fd4	122	12	fd0
38	12	fe4	123	12	fe5
39	8	e6	124	12	fe6
40	7	70	125	13	1feb
41	8	e9	126	13	1fef

Окончание таблицы А.10

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
42	9	1dd	127	14	3ff3
43	9	1e3	128	14	3ff4
44	10	3d2	129	14	3ff5
45	10	3dc	130	12	fe0
46	11	7cc	131	11	7ce
47	11	7ca	132	11	7d5
48	11	7de	133	12	fc6
49	12	fd8	134	12	fd1
50	12	fea	135	12	fe1
51	13	1fdb	136	13	1fe0
52	9	1df	137	13	1fe8
53	8	eb	138	13	1ff0
54	9	1dc	139	14	3ff1
55	9	1e6	140	14	3ff8
56	10	3d5	141	14	3ff6
57	10	3de	142	15	7ffc
58	11	7cb	143	12	fe8
59	11	7dd	144	11	7df
60	11	7dc	145	12	fc9
61	12	fcd	146	12	fd7
62	12	fe2	147	12	fdc
63	12	fe7	148	13	1fdc
64	13	1fe1	149	13	1fdf
65	10	3d0	150	13	1fed
66	9	1e0	151	13	1ff5
67	9	1e4	152	14	3ff9
68	10	3d6	153	14	3ffb
69	11	7c5	154	15	7ffd
70	11	7d1	155	15	7ffe
71	11	7db	156	13	1fe7
72	12	fd2	157	12	fcc
73	11	7e0	158	12	fd6
74	12	fd9	159	12	fdf
75	12	feb	160	13	1fde
76	13	1fe3	161	13	1fda
77	13	1fe9	162	13	1fe5
78	11	7c4	163	13	1ff2
79	9	1e5	164	14	3ffa
80	10	3d7	165	14	3ff7
81	11	7c6	166	14	3ffc
82	11	7cf	167	14	3ffd
83	11	7da	168	15	7fff
84	12	fcb			

Таблица А.11 — Сборник кодов Хаффмана 10 для спектра

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
0	6	22	85	9	1c7
1	5	8	86	9	1ca
2	6	1d	87	9	1e0
3	6	26	88	10	3db
4	7	5f	89	10	3e8
5	8	d3	90	11	7ec
6	9	1cf	91	9	1e3
7	10	3d0	92	8	d2
8	10	3d7	93	8	cb
9	10	3ed	94	8	d0
10	11	7f0	95	8	d7
11	11	7f6	96	8	db
12	12	ffd	97	9	1c6
13	5	7	98	9	1d5
14	4	0	99	9	1d8
15	4	1	100	10	3ca
16	5	9	101	10	3da
17	6	20	102	11	7ea
18	7	54	103	11	7f1
19	7	60	104	9	1e1
20	8	d5	105	8	d4
21	8	dc	106	8	cf
22	9	1d4	107	8	d6
23	10	3cd	108	8	de
24	10	3de	109	8	e1
25	11	7e7	110	9	1d0
26	6	1c	111	9	1d6
27	4	2	112	10	3d1
28	5	6	113	10	3d5
29	5	c	114	10	3f2
30	6	1e	115	11	7ee
31	6	28	116	11	7fb
32	7	5b	117	10	3e9
33	8	cd	118	9	1cd
34	8	d9	119	9	1c8
35	9	1ce	120	9	1cb
36	9	1dc	121	9	1d1
37	10	3d9	122	9	1d7
38	10	3f1	123	9	1df
39	6	25	124	10	3cf
40	5	b	125	10	3e0
41	5	a	126	10	3ef

Окончание таблицы А.11

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
42	5	d	127	11	7e6
43	6	24	128	11	7f8
44	7	57	129	12	ffa
45	7	61	130	10	3eb
46	8	cc	131	9	1dd
47	8	dd	132	9	1d3
48	9	1cc	133	9	1d9
49	9	1de	134	9	1db
50	10	3d3	135	10	3d2
51	10	3e7	136	10	3cc
52	7	5d	137	10	3dc
53	6	21	138	10	3ea
54	6	1f	139	11	7ed
55	6	23	140	11	7f3
56	6	27	141	11	7f9
57	7	59	142	12	ff9
58	7	64	143	11	7f2
59	8	d8	144	10	3ce
60	8	df	145	9	1e4
61	9	1d2	146	10	3cb
62	9	1e2	147	10	3d8
63	10	3dd	148	10	3d6
64	10	3ee	149	10	3e2
65	8	d1	150	10	3e5
66	7	55	151	11	7e8
67	6	29	152	11	7f4
68	7	56	153	11	7f5
69	7	58	154	11	7f7
70	7	62	155	12	ffb
71	8	ce	156	11	7fa
72	8	e0	157	10	3ec
73	8	e2	158	10	3df
74	9	1da	159	10	3e1
75	10	3d4	160	10	3e4
76	10	3e3	161	10	3e6
77	11	7eb	162	10	3f0
78	9	1c9	163	11	7e9
79	7	5e	164	11	7ef
80	7	5a	165	12	ff8
81	7	5c	166	12	ffe
82	7	63	167	12	ffc
83	8	ca	168	12	fff
84	8	da			

Таблица А.12 — Сборник кодов Хаффмана 11 для спектра

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
0	4	0	145	10	38d
1	5	6	146	10	398
2	6	19	147	10	3b7
3	7	3d	148	10	3d3
4	8	9c	149	10	3d1
5	8	c6	150	10	3db
6	9	1a7	151	11	7dd
7	10	390	152	8	b4
8	10	3c2	153	10	3de
9	10	3df	154	9	1a9
10	11	7e6	155	9	19b
11	11	7f3	156	9	19c
12	12	ffb	157	9	1a1
13	11	7ec	158	9	1aa
14	12	ffa	159	9	1ad
15	12	ffe	160	9	1b3
16	10	38e	161	10	38b
17	5	5	162	10	3b2
18	4	1	163	10	3b8
19	5	8	164	10	3ce
20	6	14	165	10	3e1
21	7	37	166	10	3e0
22	7	42	167	11	7d2
23	8	92	168	11	7e5
24	8	af	169	8	b7
25	9	191	170	11	7e3
26	9	1a5	171	9	1bb
27	9	1b5	172	9	1a8
28	10	39e	173	9	1a6
29	10	3c0	174	9	1b0
30	10	3a2	175	9	1b2
31	10	3cd	176	9	1b7
32	11	7d6	177	10	39b
33	8	ae	178	10	39a
34	6	17	179	10	3ba
35	5	7	180	10	3b5
36	5	9	181	10	3d6
37	6	18	182	11	7d7
38	7	39	183	10	3e4
39	7	40	184	11	7d8
40	8	8e	185	11	7ea
41	8	a3	186	8	ba

Продолжение таблицы А.12

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
42	8	b8	187	11	7e8
43	9	199	188	10	3a0
44	9	1ac	189	9	1bd
45	9	1c1	190	9	1b4
46	10	3b1	191	10	38a
47	10	396	192	9	1c4
48	10	3be	193	10	392
49	10	3ca	194	10	3aa
50	8	9d	195	10	3b0
51	7	3c	196	10	3bc
52	6	15	197	10	3d7
53	6	16	198	11	7d4
54	6	1a	199	11	7dc
55	7	3b	200	11	7db
56	7	44	201	11	7d5
57	8	91	202	11	7f0
58	8	a5	203	8	c1
59	8	be	204	11	7fb
60	9	196	205	10	3c8
61	9	1ae	206	10	3a3
62	9	1b9	207	10	395
63	10	3a1	208	10	39d
64	10	391	209	10	3ac
65	10	3a5	210	10	3ae
66	10	3d5	211	10	3c5
67	8	94	212	10	3d8
68	8	9a	213	10	3e2
69	7	36	214	10	3e6
70	7	38	215	11	7e4
71	7	3a	216	11	7e7
72	7	41	217	11	7e0
73	8	8c	218	11	7e9
74	8	9b	219	11	7f7
75	8	b0	220	9	190
76	8	c3	221	11	7f2
77	9	19e	222	10	393
78	9	1ab	223	9	1be
79	9	1bc	224	9	1c0
80	10	39f	225	10	394
81	10	38f	226	10	397
82	10	3a9	227	10	3ad
83	10	3cf	228	10	3c3

Продолжение таблицы А.12

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
84	8	93	229	10	3c1
85	8	bf	230	10	3d2
86	7	3e	231	11	7da
87	7	3f	232	11	7d9
88	7	43	233	11	7df
89	7	45	234	11	7eb
90	8	9e	235	11	7f4
91	8	a7	236	11	7fa
92	8	b9	237	9	195
93	9	194	238	11	7f8
94	9	1a2	239	10	3bd
95	9	1ba	240	10	39c
96	9	1c3	241	10	3ab
97	10	3a6	242	10	3a8
98	10	3a7	243	10	3b3
99	10	3bb	244	10	3b9
100	10	3d4	245	10	3d0
101	8	9f	246	10	3e3
102	9	1a0	247	10	3e5
103	8	8f	248	11	7e2
104	8	8d	249	11	7de
105	8	90	250	11	7ed
106	8	98	251	11	7f1
107	8	a6	252	11	7f9
108	8	b6	253	11	7fc
109	8	c4	254	9	193
110	9	19f	255	12	ffd
111	9	1af	256	10	3dc
112	9	1bf	257	10	3b6
113	10	399	258	10	3c7
114	10	3bf	259	10	3cc
115	10	3b4	260	10	3cb
116	10	3c9	261	10	3d9
117	10	3e7	262	10	3da
118	8	a8	263	11	7d3
119	9	1b6	264	11	7e1
120	8	ab	265	11	7ee
121	8	a4	266	11	7ef
122	8	aa	267	11	7f5
123	8	b2	268	11	7f6
124	8	c2	269	12	ffc
125	8	c5	270	12	fff

Окончание таблицы А.12

Индекс	Длина	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина	Кодовая комбинация (шестнадцатеричная)
126	9	198	271	9	19d
127	9	1a4	272	9	1c2
128	9	1b8	273	8	b5
129	10	38c	274	8	a1
130	10	3a4	275	8	96
131	10	3c4	276	8	97
132	10	3c6	277	8	95
133	10	3dd	278	8	99
134	10	3e8	279	8	a0
135	8	ad	280	8	a2
136	10	3af	281	8	ac
137	9	192	282	8	a9
138	8	bd	283	8	b1
139	8	bc	284	8	b3
140	9	18e	285	8	bb
141	9	197	286	8	c0
142	9	19a	287	9	18f
143	9	1a3	288	5	4
144	9	1b1			

Таблицы окна

Т а б л и ц а А.13 — Кайзер-Бесселево окно для *EIGHT_SHORT_SEQUEN CE* объектного типа *AAC SSR*

i	*w (i)*	*i*	*w (i)*
0	0,0000875914060105	16	0,7446454751465113
1	0,0009321760265333	17	0,8121892962974020
2	0,0032114611466596	18	0,8683559394406505
3	0,0081009893216786	19	0,9125649996381605
4	0,0171240286619181	20	0,9453396205809574
5	0,0320720743527833	21	0,9680864942677585
6	0,0548307856028528	22	0,9827581789763112
7	0,0871361822564870	23	0,9914756203467121
8	0,1302923415174603	24	0,9961964092194694
9	0,1848955425508276	25	0,9984956609571091
10	0,2506163195331889	26	0,9994855586984285
11	0,3260874142923209	27	0,9998533730714648
12	0,4089316830907141	28	0,9999671864476404
13	0,4959414909423747	29	0,9999948432453556
14	0,5833939894958904	30	0,9999995655238333
15	0,6674601983218376	31	0,9999999961638728

Таблица А.14 — Кайзер-Бесселево окно для объектного типа *SSR* для последовательностей других окон

i	*w (i)*	*i*	*w (i)*
0	0,0005851230124487	128	0,7110428359000029
1	0,0009642149851497	129	0,7188474364707993
2	0,0013558207534965	130	0,7265597347077880
3	0,0017771849644394	131	0,7341770687621900
4	0,0022352533849672	132	0,7416968783634273
5	0,0027342299070304	133	0,7491167073477523
6	0,0032773001022195	134	0,7564342060337386
7	0,0038671998069216	135	0,7636471334404891
8	0,0045064443384152	136	0,7707533593446514
9	0,0051974336885144	137	0,7777508661725849
10	0,0059425050016407	138	0,7846377507242818
11	0,0067439602523141	139	0,7914122257259034
12	0,0076040812644888	140	0,7980726212080798
13	0,0085251378135895	141	0,8046173857073919
14	0,0095093917383048	142	0,8110450872887550
15	0,0105590986429280	143	0,8173544143867162
16	0,0116765080854300	144	0,8235441764639875
17	0,0128638627792770	145	0,8296133044858474
18	0,0141233971318631	146	0,8355608512093652
19	0,0154573353235409	147	0,8413859912867303
20	0,0168678890600951	148	0,8470880211822968
21	0,0183572550877256	149	0,8526663589032990
22	0,0199276125319803	150	0,8581205435445334
23	0,0215811201042484	151	0,8634502346476508
24	0,0233199132076965	152	0,8686552113760616
25	0,0251461009666641	153	0,8737353715068081
26	0,0270617631981826	154	0,8786907302411250
27	0,0290689473405856	155	0,8835214188357692
28	0,0311696653515848	156	0,8882276830575707
29	0,0333658905863535	157	0,8928098814640207
30	0,0356595546648444	158	0,8972684835130879
31	0,0380525443366107	159	0,9016040675058185
32	0,0405466983507029	160	0,9058173183656508
33	0,0431438043376910	161	0,9099090252587376
34	0,0458455957104702	162	0,9138800790599416
35	0,0486537485902075	163	0,9177314696695282
36	0,0515698787635492	164	0,9214642831859411
37	0,0545955386770205	165	0,9250796989403991
38	0,0577322144743916	166	0,9285789863994010
39	0,0609813230826460	167	0,9319635019415643
40	0,0643442093520723	168	0,9352346855155568
41	0,0678221432558827	169	0,9383940571861993
42	0,0714163171546603	170	0,9414432135761304

Продолжение таблицы А.14

i	*w (i)*	*i*	*w (i)*
43	0,0751278431308314	171	0,9443838242107182
44	0,0789577503982528	172	0,9472176277741918
45	0,0829069827918993	173	0,9499464282852282
46	0,0869763963425241	174	0,9525720912004834
47	0,0911667569410503	175	0,9550965394547873
48	0,0954787380973307	176	0,9575217494469370
49	0,0999129187977865	177	0,9598497469802043
50	0,1044697814663005	178	0,9620826031668507
51	0,1091497100326053	179	0,9642224303060783
52	0,1139529881122542	180	0,9662713777449607
53	0,1188797973021148	181	0,9682316277319895
54	0,1239302155951605	182	0,9701053912729269
55	0,1291042159181728	183	0,9718949039986892
56	0,1344016647957880	184	0,9736024220549734
57	0,1398223211441467	185	0,9752302180233160
58	0,1453658351972151	186	0,9767805768831932
59	0,1510317475686540	187	0,9782557920246753
60	0,1568194884519144	188	0,9796581613210076
61	0,1627283769610327	189	0,9809899832703159
62	0,1687576206143887	190	0,9822535532154261
63	0,1749063149634756	191	0,9834511596505429
64	0,1811734433685097	192	0,9845850806232530
65	0,1875578769224857	193	0,9856575802399989
66	0,1940583745250518	194	0,9866709052828243
67	0,2006735831073503	195	0,9876272819448033
68	0,2074020380087318	196	0,9885289126911557
69	0,2142421635060113	197	0,9893779732525968
70	0,2211922734956977	198	0,9901766097569984
71	0,2282505723293797	199	0,9909269360049311
72	0,2354151558022098	200	0,9916310308941294
73	0,2426840122941792	201	0,9922909359973702
74	0,2500550240636293	202	0,9929086532976777
75	0,2575259686921987	203	0,9934861430841844
76	0,2650945206801527	204	0,9940253220113651
77	0,2727582531907993	205	0,9945280613237534
78	0,2805146399424422	206	0,9949961852476154
79	0,2883610572460804	207	0,9954314695504363
80	0,2962947861868143	208	0,9958356402684387
81	0,3043130149466800	209	0,9962103726017252
82	0,3124128412663888	210	0,9965572899760172
83	0,3205912750432127	211	0,9968779632693499
84	0,3288452410620226	212	0,9971739102014799
85	0,3371715818562547	213	0,9974465948831872

Окончание таблицы А.14

i	*w (i)*	*i*	*w (i)*
86	0,3455670606953511	214	0,9976974275220812
87	0,3540283646950029	215	0,9979277642809907
88	0,3625521080463003	216	0,9981389072844972
89	0,3711348353596863	217	0,9983321047686901
90	0,3797730251194006	218	0,9985085513687731
91	0,3884630932439016	219	0,9986693885387259
92	0,3972013967475546	220	0,9988157050968516
93	0,4059842374986933	221	0,9989485378906924
94	0,4148078660689724	222	0,9990688725744943
95	0,4236684856687616	223	0,9991776444921379
96	0,4325622561631607	224	0,9992757396582338
97	0,4414852981630577	225	0,9993639958299003
98	0,4504336971855032	226	0,9994432036616085
99	0,4594035078775303	227	0,9995141079353859
100	0,4683907582974173	228	0,9995774088586188
101	0,4773914542472655	229	0,9996337634216871
102	0,4864015836506502	230	0,9996837868076957
103	0,4954171209689973	231	0,9997280538466377
104	0,5044340316502417	232	0,9997671005064359
105	0,5134482766032377	233	0,9998014254134544
106	0,5224558166913167	234	0,9998314913952471
107	0,5314526172383208	235	0,9998577270385304
108	0,5404346525403849	236	0,9998805282555989
109	0,5493979103766972	237	0,9999002598526793
110	0,5583383965124314	238	0,9999172570940037
111	0,5672521391870222	239	0,9999318272557038
112	0,5761351935809411	240	0,9999442511639580
113	0,5849836462541291	241	0,9999547847121726
114	0,5937936195492526	242	0,9999636603523446
115	0,6025612759529649	243	0,9999710885561258
116	0,6112828224083939	244	0,9999772592414866
117	0,6199545145721097	245	0,9999823431612708
118	0,6285726610088878	246	0,9999864932503106
119	0,6371336273176413	247	0,9999898459281599
120	0,6456338401819751	248	0,9999925223548691
121	0,6540697913388968	249	0,9999946296375997
122	0,6624380414593221	250	0,9999962619864214
123	0,6707352239341151	251	0,9999975018180320
124	0,6789580485595255	252	0,9999984208055542
125	0,6871033051160131	253	0,9999990808746198
126	0,6951678668345944	254	0,9999995351446231
127	0,7031486937449871	255	0,9999998288155155

Т а б л и ц а А.15 — Коэффициенты окна w_{LD} для блока фильтров с низкой задержкой для N=1024

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
0	0,00000000	512	1,00028215	1024	0,16098974	1536	–0,01310123
1	0,00000000	513	1,00084319	1025	0,15896561	1537	–0,01306470
2	0,00000000	514	1,00140472	1026	0,15696026	1538	–0,01302556
3	0,00000000	515	1,00196665	1027	0,15497259	1539	–0,01298381
4	0,00000000	516	1,00252889	1028	0,15300151	1540	–0,01293948
5	0,00000000	517	1,00309139	1029	0,15104590	1541	–0,01289255
6	0,00000000	518	1,00365404	1030	0,14910466	1542	–0,01284305
7	0,00000000	519	1,00421679	1031	0,14717666	1543	–0,01279095
8	0,00000000	520	1,00477954	1032	0,14526081	1544	–0,01273625
9	0,00000000	521	1,00534221	1033	0,14335599	1545	–0,01267893
10	0,00000000	522	1,00590474	1034	0,14146111	1546	–0,01261897
11	0,00000000	523	1,00646713	1035	0,13957570	1547	–0,01255632
12	0,00000000	524	1,00702945	1036	0,13769993	1548	–0,01249096
13	0,00000000	525	1,00759179	1037	0,13583399	1549	–0,01242283
14	0,00000000	526	1,00815424	1038	0,13397806	1550	–0,01235190
15	0,00000000	527	1,00871678	1039	0,13213229	1551	–0,01227827
16	0,00000000	528	1,00927930	1040	0,13029682	1552	–0,01220213
17	0,00000000	529	1,00984169	1041	0,12847178	1553	–0,01212366
18	0,00000000	530	1,01040384	1042	0,12665729	1554	–0,01204304
19	0,00000000	531	1,01096575	1043	0,12485353	1555	–0,01196032
20	0,00000000	532	1,01152747	1044	0,12306074	1556	–0,01187543
21	0,00000000	533	1,01208910	1045	0,12127916	1557	–0,01178829
22	0,00000000	534	1,01265070	1046	0,11950900	1558	–0,01169884
23	0,00000000	535	1,01321226	1047	0,11775043	1559	–0,01160718
24	0,00000000	536	1,01433478	1048	0,11600347	1560	–0,01151352
25	0,00000000	537	1,01489551	1049	0,11426820	1561	–0,01141809
26	0,00000000	538	1,01545584	1050	0,11254465	1562	–0,01132111
27	0,00000000	539	1,01601582	1051	0,11083292	1563	–0,01122272
28	0,00000000	540	1,01657553	1052	0,10913318	1564	–0,01112304
29	0,00000000	541	1,01713502	1053	0,10744559	1565	–0,01102217
30	0,00000000	542	1,01769427	1054	0,10577028	1566	–0,01092022
31	0,00000000	543	1,01825316	1055	0,10410733	1567	–0,01081730
32	0,00000000	544	1,01881154	1056	0,10245672	1568	–0,01071355
33	0,00000000	545	1,01433478	1057	0,10081842	1569	–0,01060912
34	0,00000000	546	1,01936929	1058	0,09919240	1570	–0,01050411
35	0,00000000	547	1,01992639	1059	0,09757872	1571	–0,01039854
36	0,00000000	548	1,02048289	1060	0,09597750	1572	–0,01029227
37	0,00000000	549	1,02103888	1061	0,09438884	1573	–0,01018521
38	0,00000000	550	1,02159441	1062	0,09281288	1574	–0,01007727
39	0,00000000	551	1,02214945	1063	0,09124964	1575	–0,00996859
40	0,00000000	552	1,02270387	1064	0,08969907	1576	–0,00985959
41	0,00000000	553	1,02325751	1065	0,08816111	1577	–0,00975063

Продолжение таблицы А.15

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
42	0,00000000	554	1,02381025	1066	0,08663570	1578	–0,00964208
43	0,00000000	555	1,02436204	1067	0,08512288	1579	–0,00953420
44	0,00000000	556	1,02491295	1068	0,08362274	1580	–0,00942723
45	0,00000000	557	1,02546304	1069	0,08213540	1581	–0,00932135
46	0,00000000	558	1,02601238	1070	0,08066096	1582	–0,00921677
47	0,00000000	559	1,02656092	1071	0,07919944	1583	–0,00911364
48	0,00000000	560	1,02710853	1072	0,07775076	1584	–0,00901208
49	0,00000000	561	1,02765508	1073	0,07631484	1585	–0,00891220
50	0,00000000	562	1,02820041	1074	0,07489161	1586	–0,00881412
51	0,00000000	563	1,02874449	1075	0,07348108	1587	–0,00871792
52	0,00000000	564	1,02928737	1076	0,07208335	1588	–0,00862369
53	0,00000000	565	1,02982913	1077	0,07069851	1589	–0,00853153
54	0,00000000	566	1,03036981	1078	0,06932667	1590	–0,00844149
55	0,00000000	567	1,03090937	1079	0,06796781	1591	–0,00835360
56	0,00000000	568	1,03144768	1080	0,06662187	1592	–0,00826785
57	0,00000000	569	1,03198460	1081	0,06528874	1593	–0,00818422
58	0,00000000	570	1,03252000	1082	0,06396833	1594	–0,00810267
59	0,00000000	571	1,03305384	1083	0,06266065	1595	–0,00802312
60	0,00000000	572	1,03358617	1084	0,06136578	1596	–0,00794547
61	0,00000000	573	1,03411707	1085	0,06008380	1597	–0,00786959
62	0,00000000	574	1,03464659	1086	0,05881480	1598	–0,00779533
63	0,00000000	575	1,03517470	1087	0,05755876	1599	–0,00772165
64	0,00000000	576	1,03570128	1088	0,05631557	1600	–0,00764673
65	0,00000000	577	1,03622620	1089	0,05508511	1601	–0,00756886
66	0,00000000	578	1,03674934	1090	0,05386728	1602	–0,00748649
67	0,00000000	579	1,03727066	1091	0,05266206	1603	–0,00739905
68	0,00000000	580	1,03779024	1092	0,05146951	1604	–0,00730681
69	0,00000000	581	1,03830815	1093	0,05028971	1605	–0,00721006
70	0,00000000	582	1,03882446	1094	0,04912272	1606	–0,00710910
71	0,00000000	583	1,03933914	1095	0,04796855	1607	–0,00700419
72	0,00000000	584	1,03985206	1096	0,04682709	1608	–0,00689559
73	0,00000000	585	1,04036312	1097	0,04569825	1609	–0,00678354
74	0,00000000	586	1,04087217	1098	0,04458194	1610	–0,00666829
75	0,00000000	587	1,04137920	1099	0,04347817	1611	–0,00655007
76	0,00000000	588	1,04188428	1100	0,04238704	1612	–0,00642916
77	0,00000000	589	1,04238748	1101	0,04130868	1613	–0,00630579
78	0,00000000	590	1,04288888	1102	0,04024318	1614	–0,00618022
79	0,00000000	591	1,04338845	1103	0,03919056	1615	–0,00605267
80	0,00000000	592	1,04388610	1104	0,03815071	1616	–0,00592333
81	0,00000000	593	1,04438170	1105	0,03712352	1617	–0,00579240
82	0,00000000	594	1,04487515	1106	0,03610890	1618	–0,00566006
83	0,00000000	595	1,04536645	1107	0,03510679	1619	–0,00552651

Продолжение таблицы А.15

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
84	0,00000000	596	1,04585569	1108	0,03411720	1620	–0,00539194
85	0,00000000	597	1,04634297	1109	0,03314013	1621	–0,00525653
86	0,00000000	598	1,04682838	1110	0,03217560	1622	–0,00512047
87	0,00000000	599	1,04731192	1111	0,03122343	1623	–0,00498390
88	0,00000000	600	1,04779350	1112	0,03028332	1624	–0,00484693
89	0,00000000	601	1,04827303	1113	0,02935494	1625	–0,00470969
90	0,00000000	602	1,04875042	1114	0,02843799	1626	–0,00457228
91	0,00000000	603	1,04922568	1115	0,02753230	1627	–0,00443482
92	0,00000000	604	1,04969891	1116	0,02663788	1628	–0,00429746
93	0,00000000	605	1,05017022	1117	0,02575472	1629	–0,00416034
94	0,00000000	606	1,05063974	1118	0,02488283	1630	–0,00402359
95	0,00000000	607	1,05110746	1119	0,02402232	1631	–0,00388738
96	0,00000000	608	1,05157332	1120	0,02317341	1632	–0,00375185
97	0,00000000	609	1,05203721	1121	0,02233631	1633	–0,00361718
98	0,00000000	610	1,05249907	1122	0,02151124	1634	–0,00348350
99	0,00000000	611	1,05295889	1123	0,02069866	1635	–0,00335100
100	0,00000000	612	1,05341676	1124	0,01989922	1636	–0,00321991
101	0,00000000	613	1,05387277	1125	0,01911359	1637	–0,00309043
102	0,00000000	614	1,05432700	1126	0,01834241	1638	–0,00296276
103	0,00000000	615	1,05477948	1127	0,01758563	1639	–0,00283698
104	0,00000000	616	1,05523018	1128	0,01684248	1640	–0,00271307
105	0,00000000	617	1,05567906	1129	0,01611219	1641	–0,00259098
106	0,00000000	618	1,05612608	1130	0,01539398	1642	–0,00247066
107	0,00000000	619	1,05657124	1131	0,01468726	1643	–0,00235210
108	0,00000000	620	1,05701459	1132	0,01399167	1644	–0,00223531
109	0,00000000	621	1,05745616	1133	0,01330687	1645	–0,00212030
110	0,00000000	622	1,05789601	1134	0,01263250	1646	–0,00200709
111	0,00000000	623	1,05833426	1135	0,01196871	1647	–0,00189576
112	0,00000000	624	1,05877109	1136	0,01131609	1648	–0,00178647
113	0,00000000	625	1,05920669	1137	0,01067527	1649	–0,00167936
114	0,00000000	626	1,05964125	1138	0,01004684	1650	–0,00157457
115	0,00000000	627	1,06007444	1139	0,00943077	1651	–0,00147216
116	0,00000000	628	1,06050542	1140	0,00882641	1652	–0,00137205
117	0,00000000	629	1,06093335	1141	0,00823307	1653	–0,00127418
118	0,00000000	630	1,06135746	1142	0,00765011	1654	–0,00117849
119	0,00000000	631	1,06177909	1143	0,00707735	1655	–0,00108498
120	0,00000000	632	1,06220164	1144	0,00651513	1656	–0,00099375
121	0,00000000	633	1,06262858	1145	0,00596377	1657	–0,00090486
122	0,00000000	634	1,06306309	1146	0,00542364	1658	–0,00081840
123	0,00000000	635	1,06350050	1147	0,00489514	1659	–0,00073444
124	0,00000000	636	1,06392837	1148	0,00437884	1660	–0,00065309
125	0,00000000	637	1,06433391	1149	0,00387530	1661	–0,00057445

Продолжение таблицы А.15

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
126	0,00000000	638	1,06470443	1150	0,00338509	1662	–0,00049860
127	0,00000000	639	1,06502996	1151	0,00290795	1663	–0,00042551
128	0,00338834	640	1,06481076	1152	0,00244282	1664	–0,00035503
129	0,00567745	641	1,06469765	1153	0,00198860	1665	–0,00028700
130	0,00847677	642	1,06445004	1154	0,00154417	1666	–0,00022125
131	0,01172641	643	1,06408002	1155	0,00110825	1667	–0,00015761
132	0,01532555	644	1,06361382	1156	0,00067934	1668	–0,00009588
133	0,01917664	645	1,06307719	1157	0,00025589	1669	–0,00003583
134	0,02318809	646	1,06249453	1158	–0,00016357	1670	0,00002272
135	0,02729259	647	1,06188365	1159	–0,00057897	1671	0,00007975
136	0,03144503	648	1,06125612	1160	–0,00098865	1672	0,00013501
137	0,03560261	649	1,06062291	1161	–0,00139089	1673	0,00018828
138	0,03972499	650	1,05999418	1162	–0,00178397	1674	0,00023933
139	0,04379783	651	1,05937132	1163	–0,00216547	1675	0,00028784
140	0,04783094	652	1,05874726	1164	–0,00253230	1676	0,00033342
141	0,05183357	653	1,05811486	1165	–0,00288133	1677	0,00037572
142	0,05581342	654	1,05746728	1166	–0,00320955	1678	0,00041438
143	0,05977723	655	1,05680000	1167	–0,00351626	1679	0,00044939
144	0,06373173	656	1,05611070	1168	–0,00380315	1680	0,00048103
145	0,06768364	657	1,05539715	1169	–0,00407198	1681	0,00050958
146	0,07163937	658	1,05465735	1170	–0,00432457	1682	0,00053533
147	0,07559976	659	1,05389329	1171	–0,00456373	1683	0,00055869
148	0,07956096	660	1,05311083	1172	–0,00479326	1684	0,00058015
149	0,08352024	661	1,05231578	1173	–0,00501699	1685	0,00060022
150	0,08747623	662	1,05151372	1174	–0,00523871	1686	0,00061935
151	0,09143035	663	1,05070811	1175	–0,00546066	1687	0,00063781
152	0,09538618	664	1,04990044	1176	–0,00568360	1688	0,00065568
153	0,09934771	665	1,04909210	1177	–0,00590821	1689	0,00067303
154	0,10331917	666	1,04828434	1178	–0,00613508	1690	0,00068991
155	0,10730456	667	1,04747647	1179	–0,00636311	1691	0,00070619
156	0,11130697	668	1,04666590	1180	–0,00658944	1692	0,00072155
157	0,11532867	669	1,04585003	1181	–0,00681117	1693	0,00073567
158	0,11937133	670	1,04502628	1182	–0,00702540	1694	0,00074826
159	0,12343922	671	1,04419009	1183	–0,00722982	1695	0,00075912
160	0,12753911	672	1,04333499	1184	–0,00742268	1696	0,00076811
161	0,13167705	673	1,04245452	1185	–0,00760226	1697	0,00077509
162	0,13585812	674	1,04154244	1186	–0,00776687	1698	0,00077997
163	0,14008529	675	1,04059452	1187	–0,00791580	1699	0,00078275
164	0,14435986	676	1,03960846	1188	–0,00804933	1700	0,00078351
165	0,14868291	677	1,03858207	1189	–0,00816774	1701	0,00078237
166	0,15305531	678	1,03751326	1190	–0,00827139	1702	0,00077943
167	0,15747594	679	1,03640189	1191	–0,00836122	1703	0,00077484

Продолжение таблицы А.15

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
168	0,16194193	680	1,03524976	1192	–0,00843882	1704	0,00076884
169	0,16645070	681	1,03405868	1193	–0,00850583	1705	0,00076160
170	0,17099991	682	1,03283047	1194	–0,00856383	1706	0,00075335
171	0,17558633	683	1,03156812	1195	–0,00861430	1707	0,00074423
172	0,18020600	684	1,03027574	1196	–0,00865853	1708	0,00073442
173	0,18485548	685	1,02895743	1197	–0,00869781	1709	0,00072404
174	0,18953191	686	1,02761717	1198	–0,00873344	1710	0,00071323
175	0,19423322	687	1,02625804	1199	–0,00876633	1711	0,00070209
176	0,19895800	688	1,02488222	1200	–0,00879707	1712	0,00069068
177	0,20370512	689	1,02349184	1201	–0,00882622	1713	0,00067906
178	0,20847374	690	1,02208892	1202	–0,00885433	1714	0,00066728
179	0,21326312	691	1,02067450	1203	–0,00888132	1715	0,00065534
180	0,21807244	692	1,01924861	1204	–0,00890652	1716	0,00064321
181	0,22290083	693	1,01781123	1205	–0,00892925	1717	0,00063086
182	0,22774742	694	1,01636229	1206	–0,00894881	1718	0,00061824
183	0,23261210	695	1,01490045	1207	–0,00896446	1719	0,00060534
184	0,23749542	696	1,01342315	1208	–0,00897541	1720	0,00059211
185	0,24239767	697	1,01192778	1209	–0,00898088	1721	0,00057855
186	0,24731889	698	1,01041175	1210	–0,00898010	1722	0,00056462
187	0,25225887	699	1,00887284	1211	–0,00897234	1723	0,00055033
188	0,25721719	700	1,00730915	1212	–0,00895696	1724	0,00053566
189	0,26219330	701	1,00571882	1213	–0,00893330	1725	0,00052063
190	0,26718648	702	1,00409996	1214	–0,00890076	1726	0,00050522
191	0,27219630	703	1,00245032	1215	–0,00885914	1727	0,00048949
192	0,27722262	704	1,00076734	1216	–0,00880875	1728	0,00047349
193	0,28226514	705	0,99904842	1217	–0,00874987	1729	0,00045728
194	0,28732336	706	0,99729101	1218	–0,00868282	1730	0,00044092
195	0,29239628	707	0,99549380	1219	–0,00860825	1731	0,00042447
196	0,29748247	708	0,99365664	1220	–0,00852716	1732	0,00040803
197	0,30258055	709	0,99177946	1221	–0,00844055	1733	0,00039166
198	0,30768914	710	0,98986234	1222	–0,00834941	1734	0,00037544
199	0,31280508	711	0,98791024	1223	–0,00825485	1735	0,00035943
200	0,31792385	712	0,98593294	1224	–0,00815807	1736	0,00034371
201	0,32304172	713	0,98394037	1225	–0,00806025	1737	0,00032833
202	0,32815579	714	0,98194226	1226	–0,00796253	1738	0,00031333
203	0,33326397	715	0,97994532	1227	–0,00786519	1739	0,00029874
204	0,33836470	716	0,97795324	1228	–0,00776767	1740	0,00028452
205	0,34345661	717	0,97596955	1229	–0,00766937	1741	0,00027067
206	0,34853868	718	0,97399748	1230	–0,00756971	1742	0,00025715
207	0,35361188	719	0,97203326	1231	–0,00746790	1743	0,00024395
208	0,35867865	720	0,97006624	1232	–0,00736305	1744	0,00023104
209	0,36374072	721	0,96808546	1233	–0,00725422	1745	0,00021842

Продолжение таблицы А.15

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
210	0,36879900	722	0,96608018	1234	–0,00714055	1746	0,00020606
211	0,37385347	723	0,96404416	1235	–0,00702161	1747	0,00019398
212	0,37890349	724	0,96197556	1236	–0,00689746	1748	0,00018218
213	0,38394836	725	0,95987276	1237	–0,00676816	1749	0,00017069
214	0,38898730	726	0,95773420	1238	–0,00663381	1750	0,00015953
215	0,39401912	727	0,95556018	1239	–0,00649489	1751	0,00014871
216	0,39904236	728	0,95335291	1240	–0,00635230	1752	0,00013827
217	0,40405575	729	0,95111462	1241	–0,00620694	1753	0,00012823
218	0,40905820	730	0,94884764	1242	–0,00605969	1754	0,00011861
219	0,41404819	731	0,94655663	1243	–0,00591116	1755	0,00010942
220	0,41902398	732	0,94424858	1244	–0,00576167	1756	0,00010067
221	0,42398423	733	0,94193055	1245	–0,00561155	1757	0,00009236
222	0,42892805	734	0,93960953	1246	–0,00546110	1758	0,00008448
223	0,43385441	735	0,93729154	1247	–0,00531037	1759	0,00007703
224	0,43876210	736	0,93498157	1248	–0,00515917	1760	0,00006999
225	0,44365014	737	0,93268456	1249	–0,00500732	1761	0,00006337
226	0,44851786	738	0,93040503	1250	–0,00485462	1762	0,00005714
227	0,45336632	739	0,92813771	1251	–0,00470075	1763	0,00005129
228	0,45819759	740	0,92586755	1252	–0,00454530	1764	0,00004583
229	0,46301302	741	0,92357910	1253	–0,00438786	1765	0,00004072
230	0,46781309	742	0,92125731	1254	–0,00422805	1766	0,00003597
231	0,47259722	743	0,91889642	1255	–0,00406594	1767	0,00003157
232	0,47736435	744	0,91649998	1256	–0,00390204	1768	0,00002752
233	0,48211365	745	0,91407191	1257	–0,00373686	1769	0,00002380
234	0,48684450	746	0,91161623	1258	–0,00357091	1770	0,00002042
235	0,49155594	747	0,90913975	1259	–0,00340448	1771	0,00001736
236	0,49624679	748	0,90665202	1260	–0,00323770	1772	0,00001461
237	0,50091636	749	0,90416271	1261	–0,00307066	1773	0,00001215
238	0,50556440	750	0,90168115	1262	–0,00290344	1774	0,00000998
239	0,51019132	751	0,89920934	1263	–0,00273610	1775	0,00000807
240	0,51479771	752	0,89674189	1264	–0,00256867	1776	0,00000641
241	0,51938391	753	0,89427312	1265	–0,00240117	1777	0,00000499
242	0,52394998	754	0,89179743	1266	–0,00223365	1778	0,00000378
243	0,52849587	755	0,88931147	1267	–0,00206614	1779	0,00000278
244	0,53302151	756	0,88681415	1268	–0,00189866	1780	0,00000196
245	0,53752680	757	0,88430445	1269	–0,00173123	1781	0,00000132
246	0,54201160	758	0,88178141	1270	–0,00156390	1782	0,00000082
247	0,54647575	759	0,87924528	1271	–0,00139674	1783	0,00000046
248	0,55091916	760	0,87669753	1272	–0,00122989	1784	0,00000020
249	0,55534181	761	0,87413966	1273	–0,00106351	1785	0,00000005
250	0,55974376	762	0,87157318	1274	–0,00089772	1786	–0,00000003
251	0,56412513	763	0,86899958	1275	–0,00073267	1787	–0,00000006

Продолжение таблицы А.15

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
252	0,56848615	764	0,86642037	1276	–0,00056849	1788	–0,00000004
253	0,57282710	765	0,86383703	1277	–0,00040530	1789	–0,00000001
254	0,57714834	766	0,86125106	1278	–0,00024324	1790	0,00000001
255	0,58145030	767	0,85866393	1279	–0,00008241	1791	0,00000001
256	0,58492489	768	0,85604236	1280	0,00008214	1792	0,00000001
257	0,58918511	769	0,85344385	1281	0,00024102	1793	0,00000001
258	0,59342326	770	0,85083093	1282	0,00039922	1794	–0,00000001
259	0,59763936	771	0,84820550	1283	0,00055660	1795	–0,00000004
260	0,60183347	772	0,84556943	1284	0,00071299	1796	–0,00000005
261	0,60600561	773	0,84292458	1285	0,00086826	1797	–0,00000003
262	0,61015581	774	0,84027278	1286	0,00102224	1798	0,00000005
263	0,61428412	775	0,83761586	1287	0,00117480	1799	0,00000020
264	0,61839056	776	0,83495565	1288	0,00132579	1800	0,00000043
265	0,62247517	777	0,83229393	1289	0,00147507	1801	0,00000077
266	0,62653799	778	0,82963243	1290	0,00162252	1802	0,00000123
267	0,63057912	779	0,82697135	1291	0,00176804	1803	0,00000183
268	0,63459872	780	0,82430933	1292	0,00191161	1804	0,00000257
269	0,63859697	781	0,82164496	1293	0,00205319	1805	0,00000348
270	0,64257403	782	0,81897669	1294	0,00219277	1806	0,00000455
271	0,64653001	783	0,81630017	1295	0,00233029	1807	0,00000581
272	0,65046495	784	0,81360822	1296	0,00246567	1808	0,00000727
273	0,65437887	785	0,81089355	1297	0,00259886	1809	0,00000893
274	0,65827181	786	0,80814924	1298	0,00272975	1810	0,00001080
275	0,66214383	787	0,80537741	1299	0,00285832	1811	0,00001290
276	0,66599499	788	0,80258920	1300	0,00298453	1812	0,00001522
277	0,66982535	789	0,79979611	1301	0,00310839	1813	0,00001778
278	0,67363499	790	0,79700954	1302	0,00322990	1814	0,00002057
279	0,67742394	791	0,79423813	1303	0,00334886	1815	0,00002362
280	0,68119219	792	0,79148780	1304	0,00346494	1816	0,00002691
281	0,68493972	793	0,78876432	1305	0,00357778	1817	0,00003044
282	0,68866653	794	0,78607290	1306	0,00368706	1818	0,00003422
283	0,69237258	795	0,78340590	1307	0,00379273	1819	0,00003824
284	0,69605778	796	0,78074288	1308	0,00389501	1820	0,00004250
285	0,69972207	797	0,77806279	1309	0,00399411	1821	0,00004701
286	0,70336537	798	0,77534514	1310	0,00409020	1822	0,00005176
287	0,70698758	799	0,77258187	1311	0,00418350	1823	0,00005676
288	0,71058862	800	0,76977737	1312	0,00427419	1824	0,00006200
289	0,71416837	801	0,76693654	1313	0,00436249	1825	0,00006749
290	0,71772674	802	0,76406441	1314	0,00444858	1826	0,00007322
291	0,72126361	803	0,76116851	1315	0,00453250	1827	0,00007920
292	0,72477889	804	0,75825892	1316	0,00461411	1828	0,00008541
293	0,72827246	805	0,75534582	1317	0,00469328	1829	0,00009186

Продолжение таблицы А.15

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
294	0,73174419	806	0,75243924	1318	0,00476988	1830	0,00009854
295	0,73519392	807	0,74954634	1319	0,00484356	1831	0,00010543
296	0,73862141	808	0,74667135	1320	0,00491375	1832	0,00011251
297	0,74202643	809	0,74381840	1321	0,00497987	1833	0,00011975
298	0,74540874	810	0,74099145	1322	0,00504139	1834	0,00012714
299	0,74876817	811	0,73819147	1323	0,00509806	1835	0,00013465
300	0,75210458	812	0,73541641	1324	0,00514990	1836	0,00014227
301	0,75541785	813	0,73266408	1325	0,00519693	1837	0,00014997
302	0,75870785	814	0,72993193	1326	0,00523920	1838	0,00015775
303	0,76197437	815	0,72720913	1327	0,00527700	1839	0,00016558
304	0,76521709	816	0,72447661	1328	0,00531083	1840	0,00017348
305	0,76843570	817	0,72171494	1329	0,00534122	1841	0,00018144
306	0,77162988	818	0,71890515	1330	0,00536864	1842	0,00018947
307	0,77479939	819	0,71603932	1331	0,00539357	1843	0,00019756
308	0,77794403	820	0,71312056	1332	0,00541649	1844	0,00020573
309	0,78106359	821	0,71015250	1333	0,00543785	1845	0,00021399
310	0,78415789	822	0,70713900	1334	0,00545809	1846	0,00022233
311	0,78722670	823	0,70409084	1335	0,00547713	1847	0,00023076
312	0,79026979	824	0,70102565	1336	0,00549441	1848	0,00023924
313	0,79328694	825	0,69796137	1337	0,00550936	1849	0,00024773
314	0,79627791	826	0,69491556	1338	0,00552146	1850	0,00025621
315	0,79924244	827	0,69189772	1339	0,00553017	1851	0,00026462
316	0,80218027	828	0,68890931	1340	0,00553494	1852	0,00027293
317	0,80509112	829	0,68595141	1341	0,00553524	1853	0,00028108
318	0,80797472	830	0,68302498	1342	0,00553058	1854	0,00028904
319	0,81083081	831	0,68012852	1343	0,00552066	1855	0,00029675
320	0,81365915	832	0,67725801	1344	0,00550536	1856	0,00030419
321	0,81645949	833	0,67440936	1345	0,00548459	1857	0,00031132
322	0,81923160	834	0,67157841	1346	0,00545828	1858	0,00031810
323	0,82197528	835	0,66876081	1347	0,00542662	1859	0,00032453
324	0,82469037	836	0,66595195	1348	0,00539007	1860	0,00033061
325	0,82737673	837	0,66314722	1349	0,00534910	1861	0,00033633
326	0,83003419	838	0,66034194	1350	0,00530415	1862	0,00034169
327	0,83266262	839	0,65753027	1351	0,00525568	1863	0,00034672
328	0,83526186	840	0,65470525	1352	0,00520418	1864	0,00035142
329	0,83783176	841	0,65185984	1353	0,00515009	1865	0,00035580
330	0,84037217	842	0,64898709	1354	0,00509387	1866	0,00035988
331	0,84288297	843	0,64608214	1355	0,00503595	1867	0,00036369
332	0,84536401	844	0,64314221	1356	0,00497674	1868	0,00036723
333	0,84781517	845	0,64016460	1357	0,00491665	1869	0,00037053
334	0,85023632	846	0,63714680	1358	0,00485605	1870	0,00037361
335	0,85262739	847	0,63409034	1359	0,00479503	1871	0,00037647
336	0,85498836	848	0,63100082	1360	0,00473336	1872	0,00037909
337	0,85731921	849	0,62788400	1361	0,00467082	1873	0,00038145

Продолжение таблицы А.15

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
338	0,85961993	850	0,62474577	1362	0,00460721	1874	0,00038352
339	0,86189052	851	0,62159473	1363	0,00454216	1875	0,00038527
340	0,86413101	852	0,61844225	1364	0,00447517	1876	0,00038663
341	0,86634140	853	0,61529977	1365	0,00440575	1877	0,00038757
342	0,86852173	854	0,61217866	1366	0,00433344	1878	0,00038801
343	0,87067211	855	0,60908811	1367	0,00425768	1879	0,00038790
344	0,87279275	856	0,60603510	1368	0,00417786	1880	0,00038717
345	0,87488384	857	0,60302654	1369	0,00409336	1881	0,00038572
346	0,87694559	858	0,60006916	1370	0,00400363	1882	0,00038350
347	0,87897824	859	0,59716588	1371	0,00390837	1883	0,00038044
348	0,88098206	860	0,59431580	1372	0,00380759	1884	0,00037651
349	0,88295729	861	0,59151787	1373	0,00370130	1885	0,00037170
350	0,88490423	862	0,58877068	1374	0,00358952	1886	0,00036597
351	0,88682332	863	0,58606495	1375	0,00347268	1887	0,00035936
352	0,88871519	864	0,58338353	1376	0,00335157	1888	0,00035191
353	0,89058048	865	0,58070891	1377	0,00322699	1889	0,00034370
354	0,89241984	866	0,57802356	1378	0,00309975	1890	0,00033480
355	0,89423391	867	0,57530864	1379	0,00297088	1891	0,00032531
356	0,89602338	868	0,57254404	1380	0,00284164	1892	0,00031537
357	0,89778893	869	0,56970958	1381	0,00271328	1893	0,00030512
358	0,89953126	870	0,56678577	1382	0,00258700	1894	0,00029470
359	0,90125142	871	0,56376860	1383	0,00246328	1895	0,00028417
360	0,90295086	872	0,56066951	1384	0,00234195	1896	0,00027354
361	0,90463104	873	0,55750064	1385	0,00222281	1897	0,00026279
362	0,90629341	874	0,55427451	1386	0,00210562	1898	0,00025191
363	0,90793946	875	0,55101301	1387	0,00198958	1899	0,00024081
364	0,90957067	876	0,54774732	1388	0,00187331	1900	0,00022933
365	0,91118856	877	0,54450907	1389	0,00175546	1901	0,00021731
366	0,91279464	878	0,54132936	1390	0,00163474	1902	0,00020458
367	0,91439073	879	0,53822744	1391	0,00151020	1903	0,00019101
368	0,91597898	880	0,53521072	1392	0,00138130	1904	0,00017654
369	0,91756153	881	0,53228613	1393	0,00124750	1905	0,00016106
370	0,91914049	882	0,52945979	1394	0,00110831	1906	0,00014452
371	0,92071690	883	0,52671997	1395	0,00096411	1907	0,00012694
372	0,92229070	884	0,52403708	1396	0,00081611	1908	0,00010848
373	0,92386182	885	0,52138072	1397	0,00066554	1909	0,00008929
374	0,92542993	886	0,51872085	1398	0,00051363	1910	0,00006953
375	0,92698946	887	0,51603570	1399	0,00036134	1911	0,00004935
376	0,92852960	888	0,51331170	1400	0,00020940	1912	0,00002884
377	0,93003929	889	0,51053560	1401	0,00005853	1913	0,00000813
378	0,93150727	890	0,50769466	1402	–0,00009058	1914	–0,00001268
379	0,93291739	891	0,50478931	1403	–0,00023783	1915	–0,00003357
380	0,93424863	892	0,50183308	1404	–0,00038368	1916	–0,00005457
381	0,93547974	893	0,49884001	1405	–0,00052861	1917	–0,00007574
382	0,93658982	894	0,49582406	1406	–0,00067310	1918	–0,00009714

Продолжение таблицы А.15

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
383	0,93756587	895	0,49279905	1407	–0,00081757	1919	–0,00011882
384	0,93894072	896	0,48985748	1408	–0,00096237	1920	–0,00014082
385	0,93922780	897	0,48679641	1409	–0,00110786	1921	–0,00016318
386	0,93955477	898	0,48379429	1410	–0,00125442	1922	–0,00018595
387	0,93991290	899	0,48085363	1411	–0,00140210	1923	–0,00020912
388	0,94029104	900	0,47796576	1412	–0,00155065	1924	–0,00023265
389	0,94067794	901	0,47512151	1413	–0,00169984	1925	–0,00025650
390	0,94106258	902	0,47231151	1414	–0,00184940	1926	–0,00028060
391	0,94144084	903	0,46952402	1415	–0,00199911	1927	–0,00030492
392	0,94181549	904	0,46674486	1416	–0,00214872	1928	–0,00032941
393	0,94218963	905	0,46395979	1417	–0,00229798	1929	–0,00035400
394	0,94256628	906	0,46115496	1418	–0,00244664	1930	–0,00037865
395	0,94294662	907	0,45832607	1419	–0,00259462	1931	–0,00040333
396	0,94332998	908	0,45547830	1420	–0,00274205	1932	–0,00042804
397	0,94371562	909	0,45261727	1421	–0,00288912	1933	–0,00045279
398	0,94410280	910	0,44974866	1422	–0,00303596	1934	–0,00047759
399	0,94449122	911	0,44688011	1423	–0,00318259	1935	–0,00050243
400	0,94488106	912	0,44402125	1424	–0,00332890	1936	–0,00052728
401	0,94527249	913	0,44118178	1425	–0,00347480	1937	–0,00055209
402	0,94566568	914	0,43837094	1426	–0,00362024	1938	–0,00057685
403	0,94606074	915	0,43558772	1427	–0,00376519	1939	–0,00060153
404	0,94645772	916	0,43282082	1428	–0,00390962	1940	–0,00062611
405	0,94685665	917	0,43005847	1429	–0,00405345	1941	–0,00065056
406	0,94725759	918	0,42728913	1430	–0,00419658	1942	–0,00067485
407	0,94766054	919	0,42450572	1431	–0,00433902	1943	–0,00069895
408	0,94806547	920	0,42170567	1432	–0,00448085	1944	–0,00072287
409	0,94847234	921	0,41888658	1433	–0,00462219	1945	–0,00074660
410	0,94888115	922	0,41604633	1434	–0,00476309	1946	–0,00077013
411	0,94929190	923	0,41318897	1435	–0,00490357	1947	–0,00079345
412	0,94970469	924	0,41032472	1436	–0,00504361	1948	–0,00081653
413	0,95011960	925	0,40746405	1437	–0,00518321	1949	–0,00083936
414	0,95053672	926	0,40461724	1438	–0,00532243	1950	–0,00086192
415	0,95095604	927	0,40178943	1439	–0,00546132	1951	–0,00088421
416	0,95137751	928	0,39898066	1440	–0,00559988	1952	–0,00090619
417	0,95180105	929	0,39619073	1441	–0,00573811	1953	–0,00092786
418	0,95222658	930	0,39341940	1442	–0,00587602	1954	–0,00094919
419	0,95265413	931	0,39066519	1443	–0,00601363	1955	–0,00097017
420	0,95308380	932	0,38792536	1444	–0,00615094	1956	–0,00099077
421	0,95351571	933	0,38519713	1445	–0,00628795	1957	–0,00101098
422	0,95394994	934	0,38247773	1446	–0,00642466	1958	–0,00103077
423	0,95438653	935	0,37976476	1447	–0,00656111	1959	–0,00105012
424	0,95482538	936	0,37705620	1448	–0,00669737	1960	–0,00106904
425	0,95526643	937	0,37435006	1449	–0,00683352	1961	–0,00108750

Продолжение таблицы А.15

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
426	0,95570958	938	0,37164438	1450	–0,00696963	1962	–0,00110549
427	0,95615486	939	0,36893869	1451	–0,00710578	1963	–0,00112301
428	0,95660234	940	0,36623396	1452	–0,00724208	1964	–0,00114005
429	0,95705214	941	0,36353124	1453	–0,00737862	1965	–0,00115660
430	0,95750433	942	0,36083153	1454	–0,00751554	1966	–0,00117265
431	0,95795892	943	0,35813533	1455	–0,00765295	1967	–0,00118821
432	0,95841582	944	0,35544262	1456	–0,00779098	1968	–0,00120325
433	0,95887493	945	0,35275338	1457	–0,00792976	1969	–0,00121779
434	0,95933616	946	0,35006755	1458	–0,00806941	1970	–0,00123180
435	0,95979949	947	0,34738530	1459	–0,00821006	1971	–0,00124528
436	0,96026500	948	0,34470699	1460	–0,00835183	1972	–0,00125822
437	0,96073277	949	0,34203296	1461	–0,00849485	1973	–0,00127061
438	0,96120286	950	0,33936359	1462	–0,00863926	1974	–0,00128243
439	0,96167526	951	0,33669923	1463	–0,00878522	1975	–0,00129368
440	0,96214986	952	0,33404027	1464	–0,00893293	1976	–0,00130435
441	0,96262655	953	0,33138711	1465	–0,00908260	1977	–0,00131445
442	0,96310522	954	0,32874013	1466	–0,00923444	1978	–0,00132395
443	0,96358586	955	0,32609944	1467	–0,00938864	1979	–0,00133285
444	0,96406853	956	0,32346493	1468	–0,00954537	1980	–0,00134113
445	0,96455330	957	0,32083645	1469	–0,00970482	1981	–0,00134878
446	0,96504026	958	0,31821388	1470	–0,00986715	1982	–0,00135578
447	0,96552936	959	0,31559703	1471	–0,01003173	1983	–0,00136215
448	0,96602051	960	0,31298573	1472	–0,01019711	1984	–0,00136797
449	0,96651360	961	0,31037987	1473	–0,01036164	1985	–0,00137333
450	0,96700850	962	0,30777941	1474	–0,01052357	1986	–0,00137834
451	0,96750520	963	0,30518446	1475	–0,01068184	1987	–0,00138305
452	0,96800376	964	0,30259525	1476	–0,01083622	1988	–0,00138748
453	0,96850424	965	0,30001202	1477	–0,01098652	1989	–0,00139163
454	0,96900670	966	0,29743499	1478	–0,01113252	1990	–0,00139551
455	0,96951112	967	0,29486428	1479	–0,01127409	1991	–0,00139913
456	0,97001738	968	0,29229989	1480	–0,01141114	1992	–0,00140249
457	0,97052533	969	0,28974179	1481	–0,01154358	1993	–0,00140559
458	0,97103488	970	0,28718997	1482	–0,01167135	1994	–0,00140844
459	0,97154597	971	0,28464452	1483	–0,01179439	1995	–0,00141102
460	0,97205867	972	0,28210562	1484	–0,01191268	1996	–0,00141334
461	0,97257304	973	0,27957346	1485	–0,01202619	1997	–0,00141538
462	0,97308915	974	0,27704820	1486	–0,01213493	1998	–0,00141714
463	0,97360694	975	0,27452992	1487	–0,01223891	1999	–0,00141861
464	0,97412631	976	0,27201854	1488	–0,01233817	2000	–0,00141978
465	0,97464711	977	0,26951399	1489	–0,01243275	2001	–0,00142064
466	0,97516923	978	0,26701622	1490	–0,01252272	2002	–0,00142117
467	0,97569262	979	0,26452533	1491	–0,01260815	2003	–0,00142138
468	0,97621735	980	0,26204158	1492	–0,01268915	2004	–0,00142125

Окончание таблицы А.15

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
469	0,97674350	981	0,25956526	1493	–0,01276583	2005	–0,00142077
470	0,97727111	982	0,25709662	1494	–0,01283832	2006	–0,00141992
471	0,97780016	983	0,25463583	1495	–0,01290685	2007	–0,00141870
472	0,97833051	984	0,25218294	1496	–0,01297171	2008	–0,00141710
473	0,97886205	985	0,24973798	1497	–0,01303320	2009	–0,00141510
474	0,97939463	986	0,24730100	1498	–0,01309168	2010	–0,00141268
475	0,97992823	987	0,24487207	1499	–0,01314722	2011	–0,00140986
476	0,98046291	988	0,24245133	1500	–0,01319969	2012	–0,00140663
477	0,98099875	989	0,24003893	1501	–0,01324889	2013	–0,00140301
478	0,98153580	990	0,23763500	1502	–0,01329466	2014	–0,00139900
479	0,98207405	991	0,23523959	1503	–0,01333693	2015	–0,00139460
480	0,98261337	992	0,23285262	1504	–0,01337577	2016	–0,00138981
481	0,98315364	993	0,23047401	1505	–0,01341125	2017	–0,00138464
482	0,98369474	994	0,22810369	1506	–0,01344345	2018	–0,00137908
483	0,98423664	995	0,22574170	1507	–0,01347243	2019	–0,00137313
484	0,98477941	996	0,22338818	1508	–0,01349823	2020	–0,00136680
485	0,98532311	997	0,22104329	1509	–0,01352089	2021	–0,00136010
486	0,98586780	998	0,21870719	1510	–0,01354045	2022	–0,00135301
487	0,98641348	999	0,21637986	1511	–0,01355700	2023	–0,00134555
488	0,98696003	1000	0,21406117	1512	–0,01357068	2024	–0,00133772
489	0,98750734	1001	0,21175095	1513	–0,01358164	2025	–0,00132952
490	0,98805530	1002	0,20944904	1514	–0,01359003	2026	–0,00132095
491	0,98860389	1003	0,20715535	1515	–0,01359587	2027	–0,00131201
492	0,98915320	1004	0,20486987	1516	–0,01359901	2028	–0,00130272
493	0,98970328	1005	0,20259261	1517	–0,01359931	2029	–0,00129307
494	0,99025423	1006	0,20032356	1518	–0,01359661	2030	–0,00128309
495	0,99080602	1007	0,19806259	1519	–0,01359087	2031	–0,00127277
496	0,99135855	1008	0,19580944	1520	–0,01358219	2032	–0,00126211
497	0,99191171	1009	0,19356385	1521	–0,01357065	2033	–0,00125113
498	0,99246541	1010	0,19132556	1522	–0,01355637	2034	–0,00123981
499	0,99301962	1011	0,18909442	1523	–0,01353935	2035	–0,00122817
500	0,99357443	1012	0,18687040	1524	–0,01351949	2036	–0,00121622
501	0,99412992	1013	0,18465350	1525	–0,01349670	2037	–0,00120397
502	0,99468617	1014	0,18244372	1526	–0,01347088	2038	–0,00119141
503	0,99524320	1015	0,18024164	1527	–0,01344214	2039	–0,00117859
504	0,99580092	1016	0,17804841	1528	–0,01341078	2040	–0,00116552
505	0,99635926	1017	0,17586521	1529	–0,01337715	2041	–0,00115223
506	0,99691814	1018	0,17369322	1530	–0,01334158	2042	–0,00113877
507	0,99747748	1019	0,17153360	1531	–0,01330442	2043	–0,00112517
508	0,99803721	1020	0,16938755	1532	–0,01326601	2044	–0,00111144
509	0,99859725	1021	0,16725622	1533	–0,01322671	2045	–0,00109764
510	0,99915752	1022	0,16514081	1534	–0,01318689	2046	–0,00108377
511	0,99971793	1023	0,16304247	1535	–0,01314692	2047	–0,00106989

Таблица А.16 — Коэффициенты окна w_{LD} для блока фильтров с низкой задержкой для N=960

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
0	0,00000000	480	1,00058131	960	0,15990780	1440	–0,01308207
1	0,00000000	481	1,00118006	961	0,15776021	1441	–0,01304153
2	0,00000000	482	1,00177930	962	0,15563325	1442	–0,01299802
3	0,00000000	483	1,00237893	963	0,15352557	1443	–0,01295155
4	0,00000000	484	1,00297887	964	0,15143584	1444	–0,01290215
5	0,00000000	485	1,00357902	965	0,14936270	1445	–0,01284980
6	0,00000000	486	1,00417927	966	0,14730481	1446	–0,01279450
7	0,00000000	487	1,00477954	967	0,14526081	1447	–0,01273625
8	0,00000000	488	1,00537972	968	0,14322937	1448	–0,01267501
9	0,00000000	489	1,00597973	969	0,14120918	1449	–0,01261077
10	0,00000000	490	1,00657959	970	0,13919977	1450	–0,01254347
11	0,00000000	491	1,00717940	971	0,13720138	1451	–0,01247306
12	0,00000000	492	1,00777926	972	0,13521422	1452	–0,01239950
13	0,00000000	493	1,00837925	973	0,13323852	1453	–0,01232277
14	0,00000000	494	1,00897929	974	0,13127445	1454	–0,01224304
15	0,00000000	495	1,00957926	975	0,12932216	1455	–0,01216055
16	0,00000000	496	1,01017901	976	0,12738181	1456	–0,01207554
17	0,00000000	497	1,01077847	977	0,12545358	1457	–0,01198813
18	0,00000000	498	1,01137769	978	0,12353773	1458	–0,01189829
19	0,00000000	499	1,01197678	979	0,12163457	1459	–0,01180590
20	0,00000000	500	1,01257582	980	0,11974436	1460	–0,01171090
21	0,00000000	501	1,01317482	981	0,11786730	1461	–0,01161335
22	0,00000000	502	1,01377365	982	0,11600347	1462	–0,01151352
23	0,00000000	503	1,01437217	983	0,11415293	1463	–0,01141167
24	0,00000000	504	1,01497025	984	0,11231573	1464	–0,01130807
25	0,00000000	505	1,01556786	985	0,11049201	1465	–0,01120289
26	0,00000000	506	1,01616510	986	0,10868196	1466	–0,01109626
27	0,00000000	507	1,01676205	987	0,10688578	1467	–0,01098830
28	0,00000000	508	1,01735876	988	0,10510362	1468	–0,01087916
29	0,00000000	509	1,01795514	989	0,10333551	1469	–0,01076898
30	0,00000000	510	1,01855103	990	0,10158143	1470	–0,01065793
31	0,00000000	511	1,01914627	991	0,09984133	1471	–0,01054618
32	0,00000000	512	1,01974076	992	0,09811524	1472	–0,01043380
33	0,00000000	513	1,02033455	993	0,09640327	1473	–0,01032068
34	0,00000000	514	1,02092772	994	0,09470556	1474	–0,01020670
35	0,00000000	515	1,02152037	995	0,09302228	1475	–0,01009171
36	0,00000000	516	1,02211247	996	0,09135347	1476	–0,00997585
37	0,00000000	517	1,02270387	997	0,08969907	1477	–0,00985959
38	0,00000000	518	1,02329439	998	0,08805903	1478	–0,00974338
39	0,00000000	519	1,02388387	999	0,08643326	1479	–0,00962765
40	0,00000000	520	1,02447229	1000	0,08482183	1480	–0,00951273
41	0,00000000	521	1,02505972	1001	0,08322486	1481	–0,00939888
42	0,00000000	522	1,02564624	1002	0,08164249	1482	–0,00928634
43	0,00000000	523	1,02623190	1003	0,08007481	1483	–0,00917534

Продолжение таблицы А.16

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
44	0,00000000	524	1,02681660	1004	0,07852179	1484	–0,00906604
45	0,00000000	525	1,02740017	1005	0,07698335	1485	–0,00895860
46	0,00000000	526	1,02798242	1006	0,07545938	1486	–0,00885313
47	0,00000000	527	1,02856326	1007	0,07394984	1487	–0,00874977
48	0,00000000	528	1,02914272	1008	0,07245482	1488	–0,00864862
49	0,00000000	529	1,02972087	1009	0,07097444	1489	–0,00854979
50	0,00000000	530	1,03029778	1010	0,06950883	1490	–0,00845337
51	0,00000000	531	1,03087344	1011	0,06805800	1491	–0,00835939
52	0,00000000	532	1,03144768	1012	0,06662187	1492	–0,00826785
53	0,00000000	533	1,03202035	1013	0,06520031	1493	–0,00817872
54	0,00000000	534	1,03259127	1014	0,06379324	1494	–0,00809195
55	0,00000000	535	1,03316042	1015	0,06240065	1495	–0,00800745
56	0,00000000	536	1,03372788	1016	0,06102266	1496	–0,00792506
57	0,00000000	537	1,03429373	1017	0,05965936	1497	–0,00784469
58	0,00000000	538	1,03485801	1018	0,05831084	1498	–0,00776588
59	0,00000000	539	1,03542064	1019	0,05697701	1499	–0,00768695
60	0,00000000	540	1,03598146	1020	0,05565775	1500	–0,00760568
61	0,00000000	541	1,03654030	1021	0,05435290	1501	–0,00752004
62	0,00000000	542	1,03709708	1022	0,05306239	1502	–0,00742875
63	0,00000000	543	1,03765185	1023	0,05178628	1503	–0,00733186
64	0,00000000	544	1,03820470	1024	0,05052464	1504	–0,00722976
65	0,00000000	545	1,03875571	1025	0,04927758	1505	–0,00712279
66	0,00000000	546	1,03930488	1026	0,04804510	1506	–0,00701130
67	0,00000000	547	1,03985206	1027	0,04682709	1507	–0,00689559
68	0,00000000	548	1,04039712	1028	0,04562344	1508	–0,00677595
69	0,00000000	549	1,04093989	1029	0,04443405	1509	–0,00665269
70	0,00000000	550	1,04148037	1030	0,04325893	1510	–0,00652610
71	0,00000000	551	1,04201865	1031	0,04209822	1511	–0,00639649
72	0,00000000	552	1,04255481	1032	0,04095208	1512	–0,00626417
73	0,00000000	553	1,04308893	1033	0,03982059	1513	–0,00612943
74	0,00000000	554	1,04362093	1034	0,03870371	1514	–0,00599252
75	0,00000000	555	1,04415068	1035	0,03760131	1515	–0,00585368
76	0,00000000	556	1,04467803	1036	0,03651325	1516	–0,00571315
77	0,00000000	557	1,04520292	1037	0,03543944	1517	–0,00557115
78	0,00000000	558	1,04572542	1038	0,03437987	1518	–0,00542792
79	0,00000000	559	1,04624566	1039	0,03333454	1519	–0,00528367
80	0,00000000	560	1,04676376	1040	0,03230348	1520	–0,00513864
81	0,00000000	561	1,04727974	1041	0,03128653	1521	–0,00499301
82	0,00000000	562	1,04779350	1042	0,03028332	1522	–0,00484693
83	0,00000000	563	1,04830493	1043	0,02929346	1523	–0,00470054
84	0,00000000	564	1,04881391	1044	0,02831658	1524	–0,00455395
85	0,00000000	565	1,04932048	1045	0,02735252	1525	–0,00440733
86	0,00000000	566	1,04982477	1046	0,02640127	1526	–0,00426086
87	0,00000000	567	1,05032693	1047	0,02546283	1527	–0,00411471

Продолжение таблицы А.16

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
88	0,00000000	568	1,05082705	1048	0,02453725	1528	–0,00396904
89	0,00000000	569	1,05132510	1049	0,02362471	1529	–0,00382404
90	0,00000000	570	1,05182098	1050	0,02272547	1530	–0,00367991
91	0,00000000	571	1,05231457	1051	0,02183980	1531	–0,00353684
92	0,00000000	572	1,05280584	1052	0,02096810	1532	–0,00339502
93	0,00000000	573	1,05329485	1053	0,02011108	1533	–0,00325472
94	0,00000000	574	1,05378171	1054	0,01926957	1534	–0,00311618
95	0,00000000	575	1,05426654	1055	0,01844439	1535	–0,00297967
96	0,00000000	576	1,05474937	1056	0,01763565	1536	–0,00284531
97	0,00000000	577	1,05523018	1057	0,01684248	1537	–0,00271307
98	0,00000000	578	1,05570892	1058	0,01606394	1538	–0,00258290
99	0,00000000	579	1,05618554	1059	0,01529909	1539	–0,00245475
100	0,00000000	580	1,05666005	1060	0,01454726	1540	–0,00232860
101	0,00000000	581	1,05713251	1061	0,01380802	1541	–0,00220447
102	0,00000000	582	1,05760297	1062	0,01308092	1542	–0,00208236
103	0,00000000	583	1,05807149	1063	0,01236569	1543	–0,00196233
104	0,00000000	584	1,05853828	1064	0,01166273	1544	–0,00184450
105	0,00000000	585	1,05900355	1065	0,01097281	1545	–0,00172906
106	0,00000000	586	1,05946756	1066	0,01029671	1546	–0,00161620
107	0,00000000	587	1,05993024	1067	0,00963479	1547	–0,00150603
108	0,00000000	588	1,06039075	1068	0,00898646	1548	–0,00139852
109	0,00000000	589	1,06084806	1069	0,00835089	1549	–0,00129358
110	0,00000000	590	1,06130111	1070	0,00772725	1550	–0,00119112
111	0,00000000	591	1,06175099	1071	0,00711521	1551	–0,00109115
112	0,00000000	592	1,06220164	1072	0,00651513	1552	–0,00099375
113	0,00000000	593	1,06265732	1073	0,00592741	1553	–0,00089902
114	0,00000000	594	1,06312146	1074	0,00535249	1554	–0,00080705
115	0,00000000	595	1,06358726	1075	0,00479089	1555	–0,00071796
116	0,00000000	596	1,06403924	1076	0,00424328	1556	–0,00063185
117	0,00000000	597	1,06446186	1077	0,00371041	1557	–0,00054886
118	0,00000000	598	1,06484048	1078	0,00319271	1558	–0,00046904
119	0,00000000	599	1,06516440	1079	0,00268947	1559	–0,00039231
120	0,00101191	600	1,06527864	1080	0,00219928	1560	–0,00031845
121	0,00440397	601	1,06498077	1081	0,00172084	1561	–0,00024728
122	0,00718669	602	1,06470196	1082	0,00125271	1562	–0,00017860
123	0,01072130	603	1,06425743	1083	0,00079311	1563	–0,00011216
124	0,01459757	604	1,06372091	1084	0,00034023	1564	–0,00004772
125	0,01875954	605	1,06311464	1085	–0,00010786	1565	0,00001500
126	0,02308987	606	1,06246622	1086	–0,00055144	1566	0,00007600
127	0,02751541	607	1,06179277	1087	–0,00098865	1567	0,00013501
128	0,03198130	608	1,06110808	1088	–0,00141741	1568	0,00019176
129	0,03643738	609	1,06042455	1089	–0,00183557	1569	0,00024595
130	0,04085290	610	1,05974495	1090	–0,00224010	1570	0,00029720
131	0,04522835	611	1,05906206	1091	–0,00262725	1571	0,00034504

Продолжение таблицы А.16

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
132	0,04957620	612	1,05836706	1092	–0,00299314	1572	0,00038902
133	0,05390454	613	1,05765243	1093	–0,00333475	1573	0,00042881
134	0,05821503	614	1,05691470	1094	–0,00365250	1574	0,00046456
135	0,06251214	615	1,05615178	1095	–0,00394867	1575	0,00049662
136	0,06680463	616	1,05536069	1096	–0,00422533	1576	0,00052534
137	0,07109582	617	1,05454152	1097	–0,00448528	1577	0,00055114
138	0,07538014	618	1,05370030	1098	–0,00473278	1578	0,00057459
139	0,07965207	619	1,05284445	1099	–0,00497252	1579	0,00059629
140	0,08390857	620	1,05198094	1100	–0,00520916	1580	0,00061684
141	0,08815177	621	1,05111433	1101	–0,00544584	1581	0,00063660
142	0,09238785	622	1,05024634	1102	–0,00568360	1582	0,00065568
143	0,09662163	623	1,04937859	1103	–0,00592326	1583	0,00067417
144	0,10085860	624	1,04851245	1104	–0,00616547	1584	0,00069213
145	0,10510892	625	1,04764614	1105	–0,00640861	1585	0,00070935
146	0,10938110	626	1,04677586	1106	–0,00664914	1586	0,00072545
147	0,11367819	627	1,04589855	1107	–0,00688354	1587	0,00074005
148	0,11800355	628	1,04501046	1108	–0,00710845	1588	0,00075283
149	0,12236410	629	1,04410500	1109	–0,00732136	1589	0,00076356
150	0,12676834	630	1,04317417	1110	–0,00752022	1590	0,00077209
151	0,13122384	631	1,04221010	1111	–0,00770289	1591	0,00077828
152	0,13573476	632	1,04120649	1112	–0,00786789	1592	0,00078205
153	0,14030106	633	1,04016012	1113	–0,00801521	1593	0,00078350
154	0,14492340	634	1,03906851	1114	–0,00814526	1594	0,00078275
155	0,14960315	635	1,03792894	1115	–0,00825839	1595	0,00077992
156	0,15433828	636	1,03674090	1116	–0,00835563	1596	0,00077520
157	0,15912396	637	1,03550649	1117	–0,00843882	1597	0,00076884
158	0,16395663	638	1,03422800	1118	–0,00850996	1598	0,00076108
159	0,16883310	639	1,03290769	1119	–0,00857097	1599	0,00075218
160	0,17374837	640	1,03154944	1120	–0,00862360	1600	0,00074232
161	0,17869679	641	1,03015834	1121	–0,00866943	1601	0,00073170
162	0,18367394	642	1,02873938	1122	–0,00871004	1602	0,00072048
163	0,18867661	643	1,02729712	1123	–0,00874688	1603	0,00070881
164	0,19370368	644	1,02583470	1124	–0,00878091	1604	0,00069680
165	0,19875413	645	1,02435463	1125	–0,00881277	1605	0,00068450
166	0,20382641	646	1,02285952	1126	–0,00884320	1606	0,00067201
167	0,20892055	647	1,02135114	1127	–0,00887248	1607	0,00065934
168	0,21403775	648	1,01982974	1128	–0,00890002	1608	0,00064647
169	0,21917761	649	1,01829520	1129	–0,00892494	1609	0,00063335
170	0,22433899	650	1,01674752	1130	–0,00894641	1610	0,00061994
171	0,22952250	651	1,01518534	1131	–0,00896355	1611	0,00060621
172	0,23472991	652	1,01360559	1132	–0,00897541	1612	0,00059211
173	0,23996189	653	1,01200510	1133	–0,00898104	1613	0,00057763
174	0,24521859	654	1,01038076	1134	–0,00897948	1614	0,00056274
175	0,25049930	655	1,00872996	1135	–0,00896990	1615	0,00054743

Продолжение таблицы А.16

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
176	0,25580312	656	1,00705045	1136	–0,00895149	1616	0,00053169
177	0,26112942	657	1,00533999	1137	–0,00892346	1617	0,00051553
178	0,26647748	658	1,00359618	1138	–0,00888519	1618	0,00049897
179	0,27184703	659	1,00181613	1139	–0,00883670	1619	0,00048206
180	0,27723785	660	0,99999673	1140	–0,00877839	1620	0,00046487
181	0,28264967	661	0,99813477	1141	–0,00871058	1621	0,00044748
182	0,28808086	662	0,99622793	1142	–0,00863388	1622	0,00042996
183	0,29352832	663	0,99427571	1143	–0,00854936	1623	0,00041241
184	0,29898979	664	0,99227814	1144	–0,00845826	1624	0,00039492
185	0,30446379	665	0,99023501	1145	–0,00836179	1625	0,00037759
186	0,30994292	666	0,98815128	1146	–0,00826124	1626	0,00036049
187	0,31541664	667	0,98603857	1147	–0,00815807	1627	0,00034371
188	0,32087942	668	0,98390898	1148	–0,00805372	1628	0,00032732
189	0,32632772	669	0,98177413	1149	–0,00794953	1629	0,00031137
190	0,33176291	670	0,97964151	1150	–0,00784572	1630	0,00029587
191	0,33718641	671	0,97751528	1151	–0,00774156	1631	0,00028080
192	0,34259612	672	0,97539999	1152	–0,00763634	1632	0,00026612
193	0,34799346	673	0,97329751	1153	–0,00752929	1633	0,00025183
194	0,35338857	674	0,97119933	1154	–0,00741941	1634	0,00023789
195	0,35878843	675	0,96909179	1155	–0,00730556	1635	0,00022428
196	0,36419504	676	0,96696152	1156	–0,00718664	1636	0,00021097
197	0,36960630	677	0,96479824	1157	–0,00706184	1637	0,00019797
198	0,37501567	678	0,96259840	1158	–0,00693107	1638	0,00018530
199	0,38042067	679	0,96036028	1159	–0,00679443	1639	0,00017297
200	0,38582069	680	0,95808180	1160	–0,00665200	1640	0,00016100
201	0,39121276	681	0,95576295	1161	–0,00650428	1641	0,00014942
202	0,39659312	682	0,95340622	1162	–0,00635230	1642	0,00013827
203	0,40195993	683	0,95101436	1163	–0,00619718	1643	0,00012757
204	0,40731155	684	0,94859030	1164	–0,00603995	1644	0,00011736
205	0,41264382	685	0,94614009	1165	–0,00588133	1645	0,00010764
206	0,41795277	686	0,94367232	1166	–0,00572169	1646	0,00009841
207	0,42323670	687	0,94119555	1167	–0,00556143	1647	0,00008969
208	0,42849480	688	0,93871796	1168	–0,00540085	1648	0,00008145
209	0,43372753	689	0,93624630	1169	–0,00523988	1649	0,00007369
210	0,43893452	690	0,93378636	1170	–0,00507828	1650	0,00006641
211	0,44411398	691	0,93134465	1171	–0,00491582	1651	0,00005958
212	0,44927117	692	0,92892076	1172	–0,00475220	1652	0,00005320
213	0,45441882	693	0,92649974	1173	–0,00458693	1653	0,00004725
214	0,45956191	694	0,92406255	1174	–0,00441953	1654	0,00004171
215	0,46470167	695	0,92159041	1175	–0,00424950	1655	0,00003659
216	0,46983016	696	0,91907411	1176	–0,00407681	1656	0,00003186
217	0,47493636	697	0,91651711	1177	–0,00390204	1657	0,00002752
218	0,48001827	698	0,91392425	1178	–0,00372581	1658	0,00002357
219	0,48507480	699	0,91130056	1179	–0,00354874	1659	0,00002000

Продолжение таблицы А.16

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
220	0,49010240	700	0,90865471	1180	–0,00337115	1660	0,00001679
221	0,49509781	701	0,90599838	1181	–0,00319318	1661	0,00001392
222	0,50005986	702	0,90334350	1182	–0,00301494	1662	0,00001140
223	0,50499037	703	0,90069934	1183	–0,00283652	1663	0,00000918
224	0,50989790	704	0,89806435	1184	–0,00265797	1664	0,00000726
225	0,51478708	705	0,89543132	1185	–0,00247934	1665	0,00000562
226	0,51965805	706	0,89279335	1186	–0,00230066	1666	0,00000424
227	0,52450975	707	0,89014496	1187	–0,00212197	1667	0,00000309
228	0,52933955	708	0,88748403	1188	–0,00194331	1668	0,00000217
229	0,53414668	709	0,88480945	1189	–0,00176471	1669	0,00000143
230	0,53893113	710	0,88211997	1190	–0,00158620	1670	0,00000088
231	0,54369178	711	0,87941558	1191	–0,00140787	1671	0,00000048
232	0,54842731	712	0,87669794	1192	–0,00122989	1672	0,00000020
233	0,55313757	713	0,87396891	1193	–0,00105244	1673	0,00000004
234	0,55782259	714	0,87123030	1194	–0,00087567	1674	–0,00000004
235	0,56248253	715	0,86848394	1195	–0,00069976	1675	–0,00000006
236	0,56711762	716	0,86573164	1196	–0,00052487	1676	–0,00000004
237	0,57172819	717	0,86297523	1197	–0,00035115	1677	0,00000000
238	0,57631468	718	0,86021649	1198	–0,00017875	1678	0,00000002
239	0,58087761	719	0,85745725	1199	–0,00000782	1679	0,00000000
240	0,58719976	720	0,85474342	1200	0,00000779	1680	0,00000000
241	0,59173064	721	0,85193656	1201	0,00017701	1681	0,00000002
242	0,59623644	722	0,84911455	1202	0,00034552	1682	0,00000000
243	0,60071719	723	0,84627969	1203	0,00051313	1683	–0,00000004
244	0,60517294	724	0,84343424	1204	0,00067966	1684	–0,00000005
245	0,60960372	725	0,84058046	1205	0,00084492	1685	–0,00000004
246	0,61400958	726	0,83772057	1206	0,00100873	1686	0,00000004
247	0,61839056	727	0,83485680	1207	0,00117093	1687	0,00000019
248	0,62274670	728	0,83199134	1208	0,00133133	1688	0,00000045
249	0,62707805	729	0,82912621	1209	0,00148978	1689	0,00000083
250	0,63138475	730	0,82626143	1210	0,00164611	1690	0,00000134
251	0,63566700	731	0,82339529	1211	0,00180023	1691	0,00000201
252	0,63992500	732	0,82052619	1212	0,00195211	1692	0,00000285
253	0,64415895	733	0,81765147	1213	0,00210172	1693	0,00000387
254	0,64836893	734	0,81476433	1214	0,00224898	1694	0,00000510
255	0,65255499	735	0,81185593	1215	0,00239383	1695	0,00000654
256	0,65671715	736	0,80891701	1216	0,00253618	1696	0,00000821
257	0,66085548	737	0,80594452	1217	0,00267593	1697	0,00001011
258	0,66497005	738	0,80294885	1218	0,00281306	1698	0,00001227
259	0,66906094	739	0,79994431	1219	0,00294756	1699	0,00001468
260	0,67312824	740	0,79694485	1220	0,00307942	1700	0,00001735
261	0,67717199	741	0,79396166	1221	0,00320864	1701	0,00002030
262	0,68119219	742	0,79100220	1222	0,00333502	1702	0,00002352
263	0,68518882	743	0,78807349	1223	0,00345816	1703	0,00002702

Продолжение таблицы А.16

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
264	0,68916187	744	0,78518123	1224	0,00357762	1704	0,00003080
265	0,69311129	745	0,78231422	1225	0,00369297	1705	0,00003486
266	0,69703698	746	0,77944709	1226	0,00380414	1706	0,00003918
267	0,70093884	747	0,77655407	1227	0,00391140	1707	0,00004379
268	0,70481679	748	0,77361369	1228	0,00401499	1708	0,00004866
269	0,70867071	749	0,77062281	1229	0,00411524	1709	0,00005382
270	0,71250047	750	0,76758806	1230	0,00421242	1710	0,00005924
271	0,71630596	751	0,76451506	1231	0,00430678	1711	0,00006495
272	0,72008705	752	0,76141145	1232	0,00439859	1712	0,00007093
273	0,72384360	753	0,75828860	1233	0,00448799	1713	0,00007719
274	0,72757549	754	0,75515892	1234	0,00457487	1714	0,00008373
275	0,73128256	755	0,75203479	1235	0,00465908	1715	0,00009053
276	0,73496463	756	0,74892561	1236	0,00474045	1716	0,00009758
277	0,73862141	757	0,74583682	1237	0,00481857	1717	0,00010488
278	0,74225263	758	0,74277342	1238	0,00489277	1718	0,00011240
279	0,74585799	759	0,73974008	1239	0,00496235	1719	0,00012010
280	0,74943730	760	0,73673754	1240	0,00502666	1720	0,00012796
281	0,75299039	761	0,73376310	1241	0,00508546	1721	0,00013596
282	0,75651711	762	0,73081444	1242	0,00513877	1722	0,00014406
283	0,76001729	763	0,72788616	1243	0,00518662	1723	0,00015226
284	0,76349062	764	0,72496070	1244	0,00522904	1724	0,00016053
285	0,76693670	765	0,72201426	1245	0,00526648	1725	0,00016886
286	0,77035516	766	0,71902283	1246	0,00529956	1726	0,00017725
287	0,77374564	767	0,71596990	1247	0,00532895	1727	0,00018571
288	0,77710790	768	0,71285541	1248	0,00535532	1728	0,00019424
289	0,78044169	769	0,70968427	1249	0,00537929	1729	0,00020286
290	0,78374678	770	0,70646064	1250	0,00540141	1730	0,00021156
291	0,78702291	771	0,70319589	1251	0,00542228	1731	0,00022037
292	0,79026979	772	0,69991077	1252	0,00544196	1732	0,00022928
293	0,79348715	773	0,69662714	1253	0,00545981	1733	0,00023825
294	0,79667471	774	0,69336592	1254	0,00547515	1734	0,00024724
295	0,79983215	775	0,69013742	1255	0,00548726	1735	0,00025621
296	0,80295914	776	0,68694302	1256	0,00549542	1736	0,00026509
297	0,80605536	777	0,68378420	1257	0,00549899	1737	0,00027385
298	0,80912047	778	0,68066143	1258	0,00549732	1738	0,00028241
299	0,81215417	779	0,67757157	1259	0,00548986	1739	0,00029072
300	0,81515616	780	0,67450951	1260	0,00547633	1740	0,00029874
301	0,81812616	781	0,67147030	1261	0,00545664	1741	0,00030643
302	0,82106389	782	0,66844879	1262	0,00543067	1742	0,00031374
303	0,82396915	783	0,66543949	1263	0,00539849	1743	0,00032065
304	0,82684176	784	0,66243677	1264	0,00536061	1744	0,00032715
305	0,82968154	785	0,65943505	1265	0,00531757	1745	0,00033325
306	0,83248830	786	0,65642755	1266	0,00526993	1746	0,00033895
307	0,83526186	787	0,65340591	1267	0,00521822	1747	0,00034425

Продолжение таблицы А.16

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
308	0,83800204	788	0,65036160	1268	0,00516300	1748	0,00034917
309	0,84070866	789	0,64728630	1269	0,00510485	1749	0,00035374
310	0,84338156	790	0,64417440	1270	0,00504432	1750	0,00035796
311	0,84602058	791	0,64102268	1271	0,00498194	1751	0,00036187
312	0,84862556	792	0,63782771	1272	0,00491822	1752	0,00036549
313	0,85119636	793	0,63458757	1273	0,00485364	1753	0,00036883
314	0,85373292	794	0,63130628	1274	0,00478862	1754	0,00037194
315	0,85623523	795	0,62799109	1275	0,00472309	1755	0,00037479
316	0,85870326	796	0,62464879	1276	0,00465675	1756	0,00037736
317	0,86113701	797	0,62128816	1277	0,00458939	1757	0,00037963
318	0,86353649	798	0,61792203	1278	0,00452067	1758	0,00038154
319	0,86590173	799	0,61456438	1279	0,00445003	1759	0,00038306
320	0,86823275	800	0,61122915	1280	0,00437688	1760	0,00038411
321	0,87052968	801	0,60792802	1281	0,00430063	1761	0,00038462
322	0,87279275	802	0,60466971	1282	0,00422062	1762	0,00038453
323	0,87502220	803	0,60146257	1283	0,00413609	1763	0,00038373
324	0,87721829	804	0,59831460	1284	0,00404632	1764	0,00038213
325	0,87938130	805	0,59522876	1285	0,00395060	1765	0,00037965
326	0,88151157	806	0,59220375	1286	0,00384863	1766	0,00037621
327	0,88360940	807	0,58923859	1287	0,00374044	1767	0,00037179
328	0,88567517	808	0,58632936	1288	0,00362600	1768	0,00036636
329	0,88770954	809	0,58346064	1289	0,00350540	1769	0,00035989
330	0,88971328	810	0,58061078	1290	0,00337934	1770	0,00035244
331	0,89168716	811	0,57775874	1291	0,00324885	1771	0,00034407
332	0,89363199	812	0,57488246	1292	0,00311486	1772	0,00033488
333	0,89554856	813	0,57195790	1293	0,00297849	1773	0,00032497
334	0,89743771	814	0,56896078	1294	0,00284122	1774	0,00031449
335	0,89930025	815	0,56586637	1295	0,00270458	1775	0,00030361
336	0,90113740	816	0,56266594	1296	0,00257013	1776	0,00029252
337	0,90295086	817	0,55937186	1297	0,00243867	1777	0,00028133
338	0,90474240	818	0,55599898	1298	0,00231005	1778	0,00027003
339	0,90651380	819	0,55256299	1299	0,00218399	1779	0,00025862
340	0,90826684	820	0,54909184	1300	0,00206023	1780	0,00024706
341	0,91000335	821	0,54562376	1301	0,00193766	1781	0,00023524
342	0,91172515	822	0,54219742	1302	0,00181460	1782	0,00022297
343	0,91343416	823	0,53884728	1303	0,00168938	1783	0,00021004
344	0,91513276	824	0,53559047	1304	0,00156050	1784	0,00019626
345	0,91682357	825	0,53243453	1305	0,00142701	1785	0,00018150
346	0,91850924	826	0,52938894	1306	0,00128831	1786	0,00016566
347	0,92019170	827	0,52645052	1307	0,00114365	1787	0,00014864
348	0,92187129	828	0,52358958	1308	0,00099297	1788	0,00013041
349	0,92354778	829	0,52076862	1309	0,00083752	1789	0,00011112
350	0,92522116	830	0,51795080	1310	0,00067884	1790	0,00009096
351	0,92688597	831	0,51510761	1311	0,00051845	1791	0,00007014

Продолжение таблицы А.16

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
352	0,92852960	832	0,51222179	1312	0,00035760	1792	0,00004884
353	0,93013861	833	0,50927733	1313	0,00019720	1793	0,00002718
354	0,93169897	834	0,50625944	1314	0,00003813	1794	0,00000530
355	0,93319114	835	0,50317073	1315	–0,00011885	1795	–0,00001667
356	0,93458502	836	0,50002767	1316	–0,00027375	1796	–0,00003871
357	0,93587626	837	0,49685021	1317	–0,00042718	1797	–0,00006090
358	0,93694276	838	0,49364116	1318	–0,00057975	1798	–0,00008331
359	0,93825562	839	0,49048690	1319	–0,00073204	1799	–0,00010600
360	0,93882222	840	0,48726128	1320	–0,00088453	1800	–0,00012902
361	0,93910780	841	0,48404889	1321	–0,00103767	1801	–0,00015244
362	0,93944183	842	0,48090875	1322	–0,00119192	1802	–0,00017631
363	0,93981497	843	0,47783482	1323	–0,00134747	1803	–0,00020065
364	0,94021434	844	0,47481564	1324	–0,00150411	1804	–0,00022541
365	0,94062629	845	0,47184024	1325	–0,00166151	1805	–0,00025052
366	0,94103714	846	0,46889391	1326	–0,00181932	1806	–0,00027594
367	0,94144084	847	0,46595836	1327	–0,00197723	1807	–0,00030159
368	0,94184042	848	0,46301611	1328	–0,00213493	1808	–0,00032740
369	0,94223966	849	0,46005089	1329	–0,00229210	1809	–0,00035332
370	0,94264206	850	0,45705924	1330	–0,00244849	1810	–0,00037928
371	0,94304859	851	0,45404822	1331	–0,00260415	1811	–0,00040527
372	0,94345831	852	0,45102447	1332	–0,00275928	1812	–0,00043131
373	0,94387033	853	0,44799543	1333	–0,00291410	1813	–0,00045741
374	0,94428390	854	0,44497138	1334	–0,00306879	1814	–0,00048357
375	0,94469895	855	0,44196397	1335	–0,00322332	1815	–0,00050978
376	0,94511572	856	0,43898547	1336	–0,00337759	1816	–0,00053599
377	0,94553441	857	0,43604105	1337	–0,00353145	1817	–0,00056217
378	0,94595520	858	0,43312057	1338	–0,00368470	1818	–0,00058827
379	0,94637816	859	0,43020942	1339	–0,00383722	1819	–0,00061423
380	0,94680335	860	0,42729337	1340	–0,00398892	1820	–0,00064002
381	0,94723080	861	0,42436272	1341	–0,00413972	1821	–0,00066562
382	0,94766054	862	0,42141388	1342	–0,00428967	1822	–0,00069100
383	0,94809253	863	0,41844400	1343	–0,00443889	1823	–0,00071616
384	0,94852674	864	0,41545081	1344	–0,00458749	1824	–0,00074110
385	0,94896314	865	0,41244014	1345	–0,00473571	1825	–0,00076584
386	0,94940178	866	0,40942464	1346	–0,00488366	1826	–0,00079036
387	0,94984276	867	0,40641716	1347	–0,00503137	1827	–0,00081465
388	0,95028618	868	0,40342874	1348	–0,00517887	1828	–0,00083869
389	0,95073213	869	0,40046292	1349	–0,00532610	1829	–0,00086245
390	0,95118056	870	0,39751923	1350	–0,00547302	1830	–0,00088590
391	0,95163139	871	0,39459758	1351	–0,00561965	1831	–0,00090901
392	0,95208451	872	0,39169692	1352	–0,00576598	1832	–0,00093176
393	0,95253992	873	0,38881435	1353	–0,00591199	1833	–0,00095413
394	0,95299770	874	0,38594643	1354	–0,00605766	1834	–0,00097608
395	0,95345799	875	0,38308980	1355	–0,00620300	1835	–0,00099758

Продолжение таблицы А.16

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
396	0,95392092	876	0,38024146	1356	–0,00634801	1836	–0,00101862
397	0,95438653	877	0,37739896	1357	–0,00649273	1837	–0,00103918
398	0,95485472	878	0,37455986	1358	–0,00663727	1838	–0,00105924
399	0,95532539	879	0,37172187	1359	–0,00678170	1839	–0,00107879
400	0,95579847	880	0,36888463	1360	–0,00692617	1840	–0,00109783
401	0,95627397	881	0,36604937	1361	–0,00707084	1841	–0,00111635
402	0,95675201	882	0,36321735	1362	–0,00721583	1842	–0,00113434
403	0,95723273	883	0,36038967	1363	–0,00736129	1843	–0,00115181
404	0,95771618	884	0,35756668	1364	–0,00750735	1844	–0,00116873
405	0,95820232	885	0,35474832	1365	–0,00765415	1845	–0,00118510
406	0,95869103	886	0,35193455	1366	–0,00780184	1846	–0,00120091
407	0,95918218	887	0,34912542	1367	–0,00795060	1847	–0,00121615
408	0,95967573	888	0,34632129	1368	–0,00810058	1848	–0,00123082
409	0,96017172	889	0,34352258	1369	–0,00825195	1849	–0,00124490
410	0,96067026	890	0,34072974	1370	–0,00840487	1850	–0,00125838
411	0,96117144	891	0,33794323	1371	–0,00855950	1851	–0,00127125
412	0,96167526	892	0,33516354	1372	–0,00871607	1852	–0,00128350
413	0,96218157	893	0,33239114	1373	–0,00887480	1853	–0,00129511
414	0,96269026	894	0,32962648	1374	–0,00903596	1854	–0,00130610
415	0,96320119	895	0,32686967	1375	–0,00919978	1855	–0,00131643
416	0,96371437	896	0,32412042	1376	–0,00936650	1856	–0,00132610
417	0,96422988	897	0,32137919	1377	–0,00953635	1857	–0,00133509
418	0,96474782	898	0,31864044	1378	–0,00970931	1858	–0,00134334
419	0,96526824	899	0,31588373	1379	–0,00988421	1859	–0,00135069
420	0,96579106	900	0,31309909	1380	–0,01005916	1860	–0,00135711
421	0,96631614	901	0,31028631	1381	–0,01023208	1861	–0,00136272
422	0,96684334	902	0,30745528	1382	–0,01040130	1862	–0,00136768
423	0,96737257	903	0,30462678	1383	–0,01056627	1863	–0,00137225
424	0,96790390	904	0,30180656	1384	–0,01072678	1864	–0,00137649
425	0,96843740	905	0,29899424	1385	–0,01088259	1865	–0,00138042
426	0,96897315	906	0,29619082	1386	–0,01103348	1866	–0,00138404
427	0,96951112	907	0,29339717	1387	–0,01117933	1867	–0,00138737
428	0,97005119	908	0,29061333	1388	–0,01132004	1868	–0,00139041
429	0,97059318	909	0,28783935	1389	–0,01145552	1869	–0,00139317
430	0,97113697	910	0,28507563	1390	–0,01158573	1870	–0,00139565
431	0,97168253	911	0,28232266	1391	–0,01171065	1871	–0,00139785
432	0,97222994	912	0,27958067	1392	–0,01183025	1872	–0,00139976
433	0,97277928	913	0,27684984	1393	–0,01194454	1873	–0,00140137
434	0,97333058	914	0,27413017	1394	–0,01205352	1874	–0,00140267
435	0,97388375	915	0,27142157	1395	–0,01215722	1875	–0,00140366
436	0,97443863	916	0,26872396	1396	–0,01225572	1876	–0,00140432
437	0,97499505	917	0,26603737	1397	–0,01234911	1877	–0,00140464

Окончание таблицы А.16

n	$w_{LD}(n)$	N	$w_{LD}(n)$	n	$w_{LD}(n)$	n	$w_{LD}(n)$
438	0,97555292	918	0,26336211	1398	–0,01243749	1878	–0,00140461
439	0,97611230	919	0,26069855	1399	–0,01252102	1879	–0,00140423
440	0,97667326	920	0,25804700	1400	–0,01259985	1880	–0,00140347
441	0,97723589	921	0,25540830	1401	–0,01267419	1881	–0,00140235
442	0,97780016	922	0,25278329	1402	–0,01274437	1882	–0,00140084
443	0,97836592	923	0,25017211	1403	–0,01281078	1883	–0,00139894
444	0,97893300	924	0,24757451	1404	–0,01287379	1884	–0,00139664
445	0,97950127	925	0,24498713	1405	–0,01293350	1885	–0,00139388
446	0,98007071	926	0,24240740	1406	–0,01298972	1886	–0,00139065
447	0,98064139	927	0,23983550	1407	–0,01304224	1887	–0,00138694
448	0,98121342	928	0,23727200	1408	–0,01309086	1888	–0,00138278
449	0,98178684	929	0,23471866	1409	–0,01313556	1889	–0,00137818
450	0,98236156	930	0,23217624	1410	–0,01317644	1890	–0,00137317
451	0,98293743	931	0,22964458	1411	–0,01321357	1891	–0,00136772
452	0,98351428	932	0,22712346	1412	–0,01324707	1892	–0,00136185
453	0,98409205	933	0,22461258	1413	–0,01327697	1893	–0,00135556
454	0,98467078	934	0,22211202	1414	–0,01330334	1894	–0,00134884
455	0,98525056	935	0,21962197	1415	–0,01332622	1895	–0,00134170
456	0,98583146	936	0,21714290	1416	–0,01334570	1896	–0,00133415
457	0,98641348	937	0,21467522	1417	–0,01336194	1897	–0,00132619
458	0,98699650	938	0,21221877	1418	–0,01337510	1898	–0,00131784
459	0,98758037	939	0,20977323	1419	–0,01338538	1899	–0,00130908
460	0,98816497	940	0,20733693	1420	–0,01339276	1900	–0,00129991
461	0,98875030	941	0,20490860	1421	–0,01339708	1901	–0,00129031
462	0,98933647	942	0,20248823	1422	–0,01339816	1902	–0,00128031
463	0,98992356	943	0,20007615	1423	–0,01339584	1903	–0,00126990
464	0,99051163	944	0,19767358	1424	–0,01339014	1904	–0,00125912
465	0,99110062	945	0,19528091	1425	–0,01338116	1905	–0,00124797
466	0,99169038	946	0,19289781	1426	–0,01336903	1906	–0,00123645
467	0,99228079	947	0,19052347	1427	–0,01335382	1907	–0,00122458
468	0,99287177	948	0,18815661	1428	–0,01333545	1908	–0,00121233
469	0,99346341	949	0,18579693	1429	–0,01331381	1909	–0,00119972
470	0,99405581	950	0,18344441	1430	–0,01328876	1910	–0,00118676
471	0,99464907	951	0,18110010	1431	–0,01326033	1911	–0,00117347
472	0,99524320	952	0,17876595	1432	–0,01322880	1912	–0,00115988
473	0,99583812	953	0,17644344	1433	–0,01319457	1913	–0,00114605
474	0,99643375	954	0,17413400	1434	–0,01315806	1914	–0,00113200
475	0,99702997	955	0,17183905	1435	–0,01311968	1915	–0,00111778
476	0,99762671	956	0,16956003	1436	–0,01307987	1916	–0,00110343
477	0,99822386	957	0,16729836	1437	–0,01303906	1917	–0,00108898
478	0,99882134	958	0,16505547	1438	–0,01299769	1918	–0,00107448
479	0,99941903	959	0,16283278	1439	–0,01295623	1919	–0,00105995

Дифференциальный масштабный коэффициент для таблиц индекса

Т а б л и ц а А.17 — Таблица перехода 0 (дифференциалный масштабный коэффициент для индекса)

Разность	Индекс	Разность	Индекс	Разность	Индекс	Разность	Индекс	Разность	Индекс	Разность	Индекс	Разность	Индекс	Разность	Индекс
0	68	16	87	32	46	48	25	64	9	80	40	96	96	112	112
1	69	17	88	33	47	49	19	65	10	81	43	97	97	113	113
2	70	18	89	34	48	50	20	66	12	82	44	98	98	114	114
3	71	19	72	35	49	51	14	67	13	83	45	99	99	115	115
4	75	20	90	36	50	52	15	68	17	84	52	100	100	116	116
5	76	21	73	37	51	53	16	69	18	85	53	101	101	117	117
6	77	22	65	38	41	54	11	70	21	86	63	102	102	118	118
7	78	23	66	39	42	55	7	71	22	87	56	103	103	119	119
8	79	24	58	40	35	56	8	72	26	88	64	104	104	120	120
9	80	25	67	41	36	57	5	73	27	89	57	105	105	121	121
10	81	26	59	42	37	58	2	74	28	90	74	106	106	122	122
11	82	27	60	43	29	59	1	75	31	91	91	107	107	123	123
12	83	28	61	44	38	60	0	76	32	92	92	108	108	124	124
13	84	29	62	45	30	61	3	77	33	93	93	109	109	125	125
14	85	30	54	46	23	62	4	78	34	94	94	110	110	126	126
15	86	31	55	47	24	63	6	79	39	95	95	111	111	127	127

Таблица А.18 – Таблица перехода 1 (индекс для дифференциального масштабного коэффициента)

Разность	Индекс	Разность	Индекс	Разность	Индекс	Разность	Индекс	Разность	Индекс	Разность	Индекс	Разность	Индекс	Разность	Индекс
0	60	16	53	32	76	48	34	64	88	80	9	96	96	112	112
1	59	17	68	33	77	49	35	65	22	81	10	97	97	113	113
2	58	18	69	34	78	50	36	66	23	82	11	98	98	114	114
3	61	19	49	35	40	51	37	67	25	83	12	99	99	115	115
4	62	20	50	36	41	52	84	68	0	84	13	100	100	116	116
5	57	21	70	37	42	53	85	69	1	85	14	101	101	117	117
6	63	22	71	38	44	54	30	70	2	86	15	102	102	118	118
7	55	23	46	39	79	55	31	71	3	87	16	103	103	119	119
8	56	24	47	40	80	56	87	72	19	88	17	104	104	120	120
9	64	25	48	41	38	57	89	73	21	89	18	105	105	121	121
10	65	26	72	42	39	58	24	74	90	90	20	106	106	122	122
11	54	27	73	43	81	59	26	75	4	91	91	107	107	123	123
12	66	28	74	44	82	60	27	76	5	92	92	108	108	124	124
13	67	29	43	45	83	61	28	77	6	93	93	109	109	125	125
14	51	30	45	46	32	62	29	78	7	94	94	110	110	126	126
15	52	31	75	47	33	63	86	79	8	95	95	111	111	127	127

Таблицы для *TwinVQ*

Т а б л и ц а А.19 — Сборник кодов *LSP* для базового кодера

Имя файла	Содержание	Режим	Число элементов	Число векторов
20*b*19*s*48*bs*	*LSP*	*core*	20	64+16+2

VN	*EN*	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0,1174	0,2629	0,4047	0,5952	0,7249	0,8803	1,0465	1,1687	1,3174	1,4778	1,6125	1,779
	12	1, 8951	2,0787	2,2225	2,3546	2,5002	2,6572	2,8355	2,9783				
1	0	0,1828	0,3386	0,4116	0,5621	0,721	0,8888	1,0088	1,1501	1,3288	1,4473	1,5727	1,7789
	12	1,8824	2,0304	2,1998	2,3585	2,4955	2,6511	2,814	2,9605				
2	0	0,1293	0,3249	0,4623	0,6582	0,7948	0,9477	1,1068	1,2104	1,3098	1,4293	1,569	1,757
	12	1,8739	2,0543	2,2133	2,363	2,5169	2,6593	2,8147	2,9676				
3	0	0,1359	0,3157	0,477	0,634	0,6863	0,797	1,0202	1,2062	1,3237	1,4243	1,571	1,788
	12	1,9209	2,0588	2,1726	2,3476	2,5309	2,6718	2,8079	2,9564				
4	0	0,1311	0,2912	0,4227	0,6131	0,7523	0,9053	1,0771	1,2221	1,3648	1,534	1,6759	1,8169
	12	1,902	2,0612	2,2039	2,3483	2,4994	2,6446	2,803	2,9602				
5	0	0,1264	0,2699	0,3763	0,581	0,7562	0,9114	1,0854	1,2163	1,373	1,5429	1,6488	1,7651
	12	1,8514	2,0338	2,1947	2,3444	2,4975	2,6497	2,8091	2,9622				
6	0	0,1437	0,2588	0,3554	0,5655	0,7454	0,9205	1,0799	1,171	1,3025	1,446	1,5693	1,7406
	12	1,8593	2,0314	2,1916	2,3495	2,5033	2,648	2,8094	2,9634				
7	0	0,1114	0,2943	0,4312	0,5984	0,7299	0,8952	1,0942	1,2548	1,3628	1,4482	1,5589	1,7495
	12	1,8897	2,0738	2,2136	2,3605	2,5199	2,6703	2,8201	2,9603				
8	0	0,0948	0,2009	0,3228	0,52	0,6759	0,8336	1,0034	1,169	1,3346	1,4869	1,6169	1,7833
	12	1,8962	2,0733	2,2408	2,3867	2,5308	2,6702	2,822	2,9715				
9	0	0,1354	0,2516	0,3765	0,5569	0,6437	0,7706	0,9587	1,1167	1,2847	1,448	1,5884	1,7576
	12	1,8759	2,0557	2,2131	2,3606	2,5117	2,6561	2,8135	2,9681				
10	0	0,1457	0,3447	0,4823	0,6299	0,7071	0,8266	1,0111	1,159	1,3168	1,4711	1,6095	1,7785
	12	1,8965	2,0814	2,2277	2,3749	2,5471	2,6981	2,8343	2,9631				
11	0	0,1308	0,2559	0,3849	0,5877	0,6716	0,7929	0,9911	1,1467	1,2875	1,4331	1,5804	1,7507
	12	1,8768	2,0533	2,2073	2,3571	2,5075	2,6526	2,8114	2,9668				
12	0	0,11	0,2342	0,3655	0,5678	0,7011	0,8539	1,0156	1,1344	1,2878	1,4475	1,5857	1,7553
	12	1,8788	2,0568	2,209	2,3575	2,5091	2,6533	2,8127	2,9664				
13	0	0,1276	0,2569	0,3779	0,5755	0,7111	0,8433	1,012	1,1497	1,2934	1,4561	1,5821	1,7393
	12	1,8597	2,0338	2,1981	2,3514	2,5005	2,6466	2,8099	2,9643				
14	0	0,1449	0,2648	0,3846	0,5961	0,7299	0,8612	1,0124	1,1361	1,276	1,4033	1,5325	1,7194
	12	1,8567	2,038	2,1982	2,3487	2,5037	2,6505	2,8093	2,9663				
15	0	0,1191	0,2724	0,3804	0,562	0,7344	0,8313	1,0047	1,1589	1,2462	1,4182	1,5826	1,7253
	12	1,8671	2,0428	2,1839	2,3524	2,4969	2,6468	2,8092	2,9619				
16	0	0,1605	0,3428	0,4476	0,5915	0,7067	0,8606	1,0336	1,1538	1,3018	1,4734	1,6169	1,7796
	12	1,9074	2,1093	2,2537	2,3646	2,5005	2,6536	2,8113	2,9592				
17	0	0,198	0,3043	0,3485	0,5137	0,6684	0,836	1,034	1,18	1,3348	1,4823	1,6125	1,7743
	12	1,8685	2,0274	2,1877	2,3437	2,5005	2,6494	2,8088	2,966				
18	0	0,1522	0,2816	0,3968	0,5617	0,7004	0,8729	1,0133	1,1174	1,2474	1,4405	1,6196	1,7822
	12	1,8452	2,0062	2,1803	2,3318	2,484	2,6414	2,8079	2,9638				
19	0	0,0838	0,2169	0,3715	0,6054	0,7642	0,8653	0,9821	1,1144	1,2896	1,4547	1,6024	1,7732
	12	1,8876	2,0665	2,2032	2,3498	2,507	2,6526	2,8113	2,9673				
20	0	0,175	0,3555	0,4484	0,5662	0,6469	0,7965	1,0043	1,1502	1,302	1,4691	1,6097	1,774
	12	1,8883	2,0712	2,2111	2,3544	2,5066	2,6538	2,813	2,9669				
21	0	0,1553	0,2518	0,3129	0,494	0,6681	0,84	1,0047	1,1351	1,2916	1,4545	1,6012	1,7689
	12	1,8857	2,0635	2,2054	2,3529	2,5061	2,6522	2,8115	2,9686				
22	0	0,151	0,2829	0,373	0,5356	0,672	0,8278	1,0133	1,1413	1,2788	1,4408	1,5875	1,7644
	12	1,8764	2,0482	2,1994	2,3476	2,5006	2,6488	2,8092	2,9658				
23	0	0,1242	0,2415	0,3427	0,5296	0,6996	0,8341	0,9758	1,1223	1,2826	1,4479	1,5923	1,7595
	12	1,8802	2,0585	2,1987	2,3476	2,5022	2,6495	2,8098	2,9672				
24	0	0,137	0,2369	0,3319	0,4979	0,6176	0,7802	0,9803	1,1446	1,3162	1,4684	1,6044	1,7707
	12	1,8874	2,063	2,2188	2,3651	2,5155	2,6591	2,8162	2,9694				

Продолжение таблицы А.19

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
25	0	0,1082	0,2604	0,3933	0,5205	0,6302	0,8223	1,0015	1,1183	1,2792	1,4528	1,5918	1,7571
	12	1,8807	2,0589	2,1995	2,3482	2,5029	2,6504	2,8099	2,967				
26	0	0,1655	0,3502	0,4789	0,6442	0,7489	0,8833	1,0259	1,1458	1,276	1,4114	1,5562	1,7398
	12	1,8638	2,0451	2,2086	2,3589	2,5122	2,6577	2,8144	2,9662				
27	0	0,1398	0,2609	0,3662	0,539	0,6898	0,8598	1,0172	1,1053	1,2365	1,4228	1,5771	1,7457
	12	1,8741	2,0544	2,2075	2,3562	2,5101	2,6536	2,8122	2,9669				
28	0	0,0999	0,2563	0,3139	0,4686	0,6594	0,8453	1,0207	1,155	1,2962	1,4475	1,5626	1,7421
	12	1,8998	2,0971	2,2265	2,3679	2,5087	2,6455	2,7782	2,9457				
29	0	0,1138	0,2129	0,3229	0,5337	0,6762	0,8288	1,0247	1,1512	1,2682	1,4174	1,5696	1,7451
	12	1,8698	2,0503	2,2105	2,3574	2,5101	2,6545	2,8123	2,9678				
30	0	0,1464	0,2629	0,3682	0,5432	0,6492	0,8349	1,066	1,1853	1,2746	1,4003	1,5387	1,7244
	12	1,8658	2,0481	2,2015	2,3521	2,5052	2,6515	2,8106	2,967				
31	0	0,1281	0,2253	0,3181	0,5448	0,738	0,8919	1,0314	1,1421	1,2714	1,4252	1,5776	1,7548
	12	1,8818	2,06	2,2127	2,3616	2,5133	2,6548	2,8141	2,9675				
32	0	0,1671	0,2982	0,3779	0,5418	0,6953	0,8556	1,0118	1,1498	1,3094	1,4649	1,6072	1,7726
	12	1,8847	2,0659	2,2223	2,3653	2,5163	2,6591	2,8135	2,9657				
33	0	0,1584	0,313	0,4545	0,612	0,7032	0,8315	1,0019	1,0994	1,2239	1,4114	1,5721	1,7441
	12	1,8778	2,0618	2,1931	2,3407	2,4986	2,6482	2,8099	2,9659				
34	0	0,1343	0,3271	0,4606	0,6457	0,8085	0,9335	1,0208	1,1017	1,2672	1,4591	1,6018	1,7899
	12	1,9118	2,0595	2,2023	2,3582	2,512	2,6632	2,8272	2,9739				
35	0	0,1772	0,3363	0,4214	0,5756	0,727	0,8934	1,0098	1,0963	1,2554	1,4493	1,5827	1,7606
	12	1,892	2,0618	2,1981	2,3514	2,5063	2,6505	2,8097	2,9658				
36	0	0,1629	0,3196	0,422	0,5608	0,6796	0,8588	1,007	1,1124	1,3047	1,4725	1,5731	1,7454
	12	1,8934	2,0498	2,1861	2,3488	2,5041	2,6429	2,8095	2,9683				
37	0	0,196	0,3317	0,4323	0,6092	0,6915	0,8002	0,9693	1,1223	1,2892	1,4404	1,5735	1,7424
	12	1,8701	2,0495	2,201	2,3498	2,5061	2,6522	2,8101	2,9654				
38	0	0,1468	0,3363	0,4748	0,6645	0,7704	0,8472	0,9688	1,1161	1,2929	1,4648	1,6101	1,7753
	12	1,8918	2,0788	2,2212	2,3697	2,5156	2,6535	2,8097	2,9669				
39	0	0,2093	0,3679	0,4453	0,613	0,7225	0,8482	1,0341	1,1537	1,2889	1,4605	1,5785	1,7303
	12	1,8466	2,0314	2,1931	2,3403	2,4949	2,6475	2,8109	2,9672				
40	0	0,099	0,2279	0,3506	0,5357	0,6608	0,8117	0,993	1,1395	1,3025	1,4607	1,5989	1,7658
	12	1,883	2,062	2,2171	2,3647	2,5148	2,6583	2,8151	2,9689				
41	0	0,1526	0,2818	0,3926	0,5601	0,6776	0,7982	0,9554	1,0972	1,2681	1,4457	1,5934	1,7644
	12	1,8867	2,0678	2,2277	2,3753	2,5217	2,661	2,8151	2,9671				
42	0	0,1664	0,2865	0,3753	0,5542	0,7075	0,8239	0,9482	1,1018	1,2804	1,4409	1,5888	1,7569
	12	1,876	2,0564	2,2125	2,3592	2,5109	2,6547	2,8128	2,9673				
43	0	0,1497	0,2776	0,404	0,6093	0,6903	0,8241	0,9759	1,0756	1,2293	1,4062	1,564	1,7444
	12	1,869	2,0471	2,2058	2,3548	2,5082	2,6541	2,8123	2,9663				
44	0	0,1856	0,3246	0,416	0,5572	0,6681	0,8443	1,0393	1,1434	1,2551	1,4226	1,5953	1,7619
	12	1,8762	2,0626	2,2266	2,3616	2,5056	2,6552	2,8174	2,9657				
45	0	0,1963	0,3169	0,3824	0,5766	0,6988	0,8159	1,013	1,111	1,2497	1,4341	1,5718	1,7518
	12	1,8737	2,051	2,2107	2,3543	2,5119	2,6529	2,812	2,9667				
46	0	0,1856	0,3454	0,4117	0,5668	0,7447	0,8623	0,9852	1,169	1,2915	1,3994	1,5688	1,7447
	12	1,8442	2,0565	2,2031	2,3417	2,5085	2,6514	2,8045	2,9678				
47	0	0,166	0,2923	0,4013	0,5757	0,6739	0,8321	1,0109	1,1455	1,2546	1,3844	1,5564	1,7446
	12	1,8702	2,0453	2,2067	2,3576	2,5086	2,6522	2,8103	2,9659				
48	0	0,1399	0,3315	0,4603	0,6454	0,7743	0,9153	1,0794	1,2339	1,3972	1,5176	1,5984	1,7353
	12	1,8565	2,0569	2,2083	2,3508	2,5053	2,6524	2,8188	2,9722				
49	0	0,1288	0,3059	0,4501	0,6671	0,7666	0,8348	0,9684	1,1432	1,3353	1,5161	1,6186	1,7363
	12	1,8551	2,0594	2,207	2,3614	2,5017	2,6432	2,8081	2,9672				
50	0	0,1283	0,3108	0,4497	0,6447	0,7964	0,8913	0,9786	1,1139	1,3089	1,4954	1,6582	1,7989
	12	1,8601	2,0313	2,2017	2,357	2,5284	2,6688	2,8041	2,953				
51	0	0,1353	0,3106	0,4598	0,6512	0,7234	0,8098	0,9998	1,1857	1,3597	1,486	1,5771	1,7548
	12	1,9111	2,1113	2,2261	2,3281	2,4854	2,6665	2,8452	2,9846				
52	0	0,1604	0,3143	0,4318	0,6003	0,7132	0,8728	1,0448	1,1722	1,3083	1,4632	1,5988	1,7629
	12	1,8779	2,0539	2,2067	2,3572	2,5069	2,6503	2,8111	2,9627				
53	0	0,1028	0,2826	0,45	0,6391	0,7665	0,9555	1,1078	1,1585	1,2412	1,4131	1,5947	1,7693
	12	1,8957	2,0951	2,2557	2,3628	2,503	2,6596	2,8153	2,9674				

Окончание таблицы А.19

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
54	0	0,1349	0,3392	0,4855	0,6061	0,6773	0,8411	1,0665	1,1875	1,2882	1,4453	1,6329	1,8118
	12	1,8896	2,035	2,201	2,3721	2,5096	2,6424	2,8095	2,9693				
55	0	0,2062	0,3369	0,3912	0,5671	0,7037	0,8188	1,0099	1,1641	1,3288	1,4701	1,5703	1,7315
	12	1,8461	2,022	2,1849	2,3378	2,497	2,6473	2,8072	2,9652				
56	0	0,1611	0,2965	0,3994	0,576	0,7066	0,8527	1,0117	1,1601	1,3263	1,4822	1,6103	1,7697
	12	1,885	2,037	2,1847	2,3501	2,5087	2,6489	2,8095	2,963				
57	0	0,174	0,3077	0,3979	0,5763	0,6674	0,7658	0,9572	1,1408	1,3099	1,4713	1,611	1,7712
	12	1,8662	2,0278	2,1799	2,3338	2,4943	2,6458	2,8068	2,9654				
58	0	0,1644	0,2687	0,3549	0,558	0,7462	0,8957	0,9979	1,1157	1,2975	1,4555	1,5783	1,722
	12	1,8323	2,0177	2,1885	2,3429	2,5006	2,6499	2,8095	2,9659				
59	0	0,1396	0,3211	0,4387	0,591	0,7217	0,8955	1,059	1,1835	1,346	1,5024	1,6307	1,8283
	12	1,9458	2,0646	2,1776	2,3532	2,5179	2,6556	2,8191	2,9694				
60	0	0,182	0,288	0,3636	0,5345	0,6327	0,7978	1,0031	1,1387	1,2978	1,4558	1,5979	1,7627
	12	1,8847	2,061	2,1793	2,354	2,5294	2,6627	2,8095	2,9644				
61	0	0,1252	0,2473	0,3853	0,5624	0,7105	0,8498	0,9762	1,1477	1,309	1,4242	1,5993	1,7636
	12	1,8327	2,0352	2,2026	2,332	2,5015	2,6514	2,8022	2,9628				
62	0	0,2073	0,3438	0,3918	0,5472	0,7194	0,8508	1,0187	1,176	1,2857	1,4605	1,6015	1,7364
	12	1,8728	2,0492	2,1888	2,3526	2,5016	2,6466	2,8089	2,9626				
63	0	0,1434	0,2665	0,369	0,5565	0,6735	0,8049	1,0074	1,1739	1,3292	1,4451	1,5527	1,7176
	12	1,8464	2,0366	2,1979	2,3497	2,5038	2,6518	2,8105	2,9653				
64	0	0,0561	0,0312	0,0155	–0,0042	0,0325	0,0171	0,0047	0,0264	0,0483	0,0216	–0,0111	–0,0329
	12	0,0229	0,0045	–0,0032	0,0015	0,0081	0,0198	0,0181	0,0193				
65	0	0,0423	0,0088	–0,0058	–0,0414	–0,0039	0,0202	0,0252	0,0543	0,0252	–0,0252	–0,009	–0,003
	12	0,0404	0,0114	0,068	0,1005	0,0661	0,0072	–0,0267	–0,0069				
66	0	–0,0397	–0,0683	–0,0203	-0,0272	0,0078	0,004	–0,0156	0,0067	0,0163	0,0187	0,0435	0,0378
	12	0,0811	0,0438	–0,0306	–0,0252	–0,0009	0,0161	0,0134	0,0158				
67	0	0,02	–0,0239	–0,0162	–0,01	0,0278	0,0185	0,0115	0,0347	0,0359	0,024	0,0296	0,0116
	12	0,0473	0,0197	0,0079	0,0177	0,0266	0,043	0,0641	0,0436				
68	0	–0,0128	0,0234	0,0498	–0,0345	–0,0268	0,0514	0,0364	0,0499	0,0485	0,0428	0,0564	0,0481
	12	0,0919	0,0574	0,013	0,0282	0,0017	–0,0071	0,0073	0,0156				
69	0	–0,0074	0,0132	0,0242	–0,0196	0,0217	0,0452	0,0326	0,0172	0,0195	0,0434	0,0361	–0,0285
	12	–0,0001	0,0035	0,0074	–0,0221	–0,0314	–0,0046	0,0049	0,0143				
70	0	–0,0005	–0,0231	0	–0,0433	–0,019	–0,0067	–0,0218	0,0094	0,0235	0,0337	0,0322	–0,0196
	12	0,0049	–0,0037	0,0272	–0,0061	–0,0058	0,021	0,017	0,0165				
71	0	0,0121	–0,0081	0,0233	0,0268	0,0422	–0,0014	–0,0065	0,0172	0,013	0,0054	0,0201	0,0062
	12	0,0431	0,016	0,0193	0,0402	0,0449	0,0152	–0,0066	0,0082				
72	0	–0,0388	–0,0291	0,0236	–0,0205	–0,0011	–0,0001	–0,0203	0,0054	0,0064	–0,0207	–0,0137	–0,0211
	12	0,0169	–0,0032	–0,0093	–0,0004	0,0357	0,0354	–0,0014	–0,0013				
73	0	–0,0215	–0,0158	0,0584	0,0477	0,0426	0,0422	0,0241	0,0117	–0,0025	–0,0027	0,0182	0,0029
	12	0,0412	0,0151	0,0549	0,0632	0,0653	0,0757	0,0695	0,0421				
74	0	0,0022	0,0325	0,0862	0,0287	0,0114	0,013	0,0068	–0,0033	–0,0203	–0,0141	0,0073	–0,0058
	12	0,0333	0,0088	0,0676	0,0589	0,0051	–0,0155	–0,0006	0,0127				
75	0	–0,0292	–0,0195	0,0575	0,0156	0,0006	0,0107	–0,0123	0,0147	0,0253	0,0278	0,0644	0,0385
	12	0,0303	–0,0152	–0,0222	0,0136	0,0206	0,0226	0,0215	0,0208				
76	0	–0,0255	–0,0425	0,0049	0,0131	0,0372	0,039	–0,015	–0,0178	0,0075	0,0045	0,0157	0,002
	12	0,0357	0,0057	0,0122	0,0176	0,0198	0,0277	0,0226	0,0208				
77	0	0,0567	0,052	0,0237	–0,0588	–0,0326	–0,0352	–0,0394	0,0085	0,0105	0,0036	0,0152	0,002
	12	0,0395	0,0105	0,028	0,0407	0,0652	0,0661	0,0186	–0,008				
78	0	0,032	0,0601	0,0632	–0,0162	–0,0058	0	–0,0441	–0,0256	–0,0027	–0,0019	0,017	0,0029
	12	0,0321	0,0081	0,0478	0,026	0,0004	0,0197	0,0281	0,0243				
79	0	0,0268	0,0004	0,0397	0,0076	–0,0044	0,0231	0,0238	0,0318	–0,0023	–0,0106	0,0283	0,0012
	12	0,0055	–0,0158	0,0285	0,032	0,0307	0,0348	0,0273	0,0219				
80	0	0,4987	0,4924	0,4713	0,4427	0,4144	0,4168	0,3813	0,3695	0,3672	0,3619	0,3276	0,3173
	12	0,3115	0,278	0,3223	0,2951	0,2826	0,2568	0,2145	0,2527				
81	0	0,3519	0,3181	0,2607	0,206	0,1473	0,1563	0,1073	0,0633	0,0178	0,0178	–0,03	–0,0538
	12	–0,0466	–0,0529	0,0389	–0,024	–0,0405	–0,1279	–0,1221	0,0012				

Таблица А.20 — Сборник кодов *LSP* для масштабируемого кодера

Имя файла	Содержание	Режим	Число элементов	Число векторов
20*b*19*s*48*sc*	*LSP*	*enhance*	20	64+16+2

VN	*EN*	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0,1567	0,293	0,4518	0,6063	0,7601	0,9216	1,067	1,2147	1,3665	1,5191	1,6681	1,8001
	12	1,9483	2,1034	2,2565	2,3958	2,5371	2,6784	2,8201	2,9648				
1	0	0,1363	0,2451	0,3925	0,524	0,6463	0,8015	0,9244	1,0729	1,2474	1,4157	1,5815	1,7372
	12	1,8908	2,0462	2,2112	2,3506	2,4974	2,653	2,7995	2,9567				
2	0	0,1649	0,2806	0,4123	0,5608	0,7206	0,8699	0,9973	1,1528	1,3042	1,45	1,6105	1,7487
	12	1,8897	2,0482	2,2069	2,3509	2,499	2,6508	2,7986	2,9581				
3	0	0,1559	0,2632	0,4053	0,5361	0,7076	0,8445	0,9292	1,0251	1,1718	1,3625	1,5449	1,7128
	12	1,879	2,0404	2,2004	2,3505	2,5068	2,6627	2,82	2,9759				
4	0	0,1529	0,2209	0,3506	0,4867	0,6205	0,7836	0,8989	1,0645	1,2688	1,3758	1,4801	1,6433
	12	1,8303	2	2,1712	2,3409	2,5026	2,6545	2,817	2,9744				
5	0	0,1433	0,2288	0,3578	0,4937	0,6217	0,7847	0,9273	1,0626	1,2336	1,3957	1,5397	1,704
	12	1,8573	2,0034	2,1756	2,3409	2,5	2,6518	2,807	2,9664				
6	0	0,1664	0,2349	0,3465	0,4486	0,5634	0,7389	0,9071	1,0751	1,243	1,3872	1,5295	1,7175
	12	1,8303	1,9659	2,1052	2,2759	2,4874	2,6327	2,7668	2,9633				
7	0	0,1458	0,2419	0,3502	0,4939	0,6698	0,8162	0,9193	1,0387	1,2036	1,3958	1,5755	1,7351
	12	1,8583	1,9894	2,163	2,3414	2,4937	2,6384	2,7951	2,9633				
8	0	0,1491	0,2608	0,4017	0,5588	0,719	0,8763	1,0197	1,1674	1,3192	1,477	1,6349	1,7749
	12	1,9286	2,0907	2,2376	2,3789	2,5253	2,6725	2,8259	2,975				
9	0	0,1812	0,2982	0,4199	0,549	0,6944	0,7986	0,8602	0,9987	1,2019	1,4017	1,5757	1,7307
	12	1,8879	2,05	2,1715	2,319	2,4852	2,6498	2,8074	2,9718				
10	0	0,164	0,2604	0,3885	0,5396	0,6878	0,8403	0,9959	1,1211	1,2562	1,4133	1,5712	1,7299
	12	1,8956	2,0557	2,2103	2,3572	2,5108	2,6634	2,8211	2,9763				
11	0	0,1566	0,297	0,4632	0,5803	0,6433	0,7481	0,9195	1,1048	1,2769	1,4439	1,6033	1,7564
	12	1,9081	2,0646	2,2041	2,349	2,4886	2,6269	2,7689	2,9472				
12	0	0,1467	0,2153	0,361	0,5554	0,7325	0,8634	0,9628	1,1097	1,3069	1,4554	1,5611	1,6825
	12	1,8505	2,0227	2,1789	2,3352	2,4982	2,6573	2,815	2,9725				
13	0	0,1285	0,2004	0,354	0,5039	0,6163	0,7581	0,9083	1,0782	1,2559	1,4238	1,5749	1,724
	12	1,8709	2,0279	2,1951	2,3786	2,541	2,6805	2,8318	2,9801				
14	0	0,1463	0,2438	0,3903	0,5417	0,6666	0,7941	0,9412	1,1156	1,295	1,4665	1,6099	1,722
	12	1,8562	2,0184	2,1976	2,3507	2,5054	2,6602	2,8182	2,9738				
15	0	0,1344	0,1615	0,2467	0,4187	0,6347	0,81	0,9542	1,1096	1,2665	1,4258	1,5829	1,7285
	12	1,8835	2,0436	2,1891	2,3415	2,4915	2,6198	2,7569	2,9449				
16	0	0,187	0,3036	0,4315	0,535	0,6405	0,7804	0,9596	1,1519	1,3195	1,4341	1,5404	1,6821
	12	1,8442	2,009	2,1585	2,3114	2,4752	2,6417	2,8005	2,9638				
17	0	0,1751	0,2719	0,3828	0,5296	0,7036	0,9034	1,0637	1,1863	1,3125	1,4169	1,5111	1,6425
	12	1,8131	1,9929	2,1476	2,3055	2,4739	2,6423	2,8053	2,9703				
18	0	0,1547	0,3035	0,4669	0,5746	0,672	0,8425	1,033	1,1782	1,2909	1,4494	1,6306	1,7793
	12	1,8982	2,038	2,2171	2,378	2,4998	2,6364	2,808	2,9789				
19	0	0,16	0,2923	0,4369	0,5654	0,6968	0,8422	0,9896	1,1608	1,3081	1,4465	1,5856	1,7046
	12	1,8411	1,9932	2,1526	2,3165	2,4797	2,6425	2,8061	2,968				
20	0	0,1562	0,2721	0,4059	0,5534	0,7148	0,8458	0,9977	1,1582	1,3127	1,5004	1,6237	1,7315
	12	1,912	2,0342	2,1649	2,3464	2,4693	2,6224	2,7995	2,9671				
21	0	0,0886	0,1482	0,3145	0,5203	0,7025	0,8746	1,0101	1,1515	1,3107	1,4647	1,6207	1,7669
	12	1,9143	2,0666	2,1859	2,3295	2,4918	2,6541	2,8129	2,9723				
22	0	0,1526	0,2884	0,4275	0,5815	0,7543	0,9188	1,0653	1,2176	1,3491	1,4997	1,69	1,8325
	12	1,9063	1,9897	2,1563	2,3462	2,4988	2,6451	2,7962	2,9644				
23	0	0,1488	0,2703	0,4265	0,5861	0,744	0,8934	1,0286	1,1766	1,3575	1,5352	1,646	1,6996
	12	1,8178	2,0112	2,2095	2,3733	2,4985	2,632	2,7964	2,9644				
24	0	0,1572	0,2429	0,3593	0,5062	0,6562	0,8095	0,961	1,118	1,2825	1,4421	1,6008	1,7497
	12	1,901	2,0565	2,208	2,3977	2,5566	2,6866	2,8352	2,9804				
25	0	0,1483	0,2405	0,3764	0,5123	0,6309	0,7726	0,9575	1,1598	1,2968	1,3939	1,5335	1,6989
	12	1,8671	2,0255	2,1687	2,3193	2,4833	2,6474	2,8075	2,9693				

Продолжение таблицы А.20

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
26	0	0,14	0,2833	0,4514	0,588	0,7017	0,838	1,0121	1,1987	1,3348	1,4524	1,6014	1,769
	12	1,9371	2,0673	2,2007	2,3651	2,5405	2,6912	2,8146	2,9573				
27	0	0,147	0,2935	0,4763	0,6216	0,7255	0,8457	0,9767	1,1172	1,2633	1,4177	1,5811	1,744
	12	1,8958	2,0268	2,1556	2,3108	2,476	2,6424	2,8053	2,9671				
28	0	0,1452	0,2348	0,3705	0,5244	0,6837	0,8371	0,9823	1,1497	1,3086	1,4343	1,5756	1,734
	12	1,8896	2,043	2,1931	2,3458	2,5031	2,6589	2,818	2,9747				
29	0	0,1752	0,2676	0,3921	0,5289	0,6385	0,8015	1,0014	1,1711	1,3076	1,4502	1,5964	1,7216
	12	1,8779	2,0451	2,1952	2,3347	2,4832	2,6366	2,7888	2,9543				
30	0	0,1758	0,2815	0,3937	0,5605	0,6842	0,7915	0,9867	1,1111	1,2982	1,4349	1,5369	1,7311
	12	1,8721	2,0534	2,1813	2,31	2,4778	2,6298	2,7919	2,9531				
31	0	0,1484	0,2614	0,406	0,5621	0,6949	0,838	1,0034	1,1408	1,3059	1,4644	1,5745	1,7082
	12	1,8647	2,0263	2,1836	2,3269	2,4801	2,6403	2,7939	2,9558				
32	0	0,1859	0,2648	0,405	0,5544	0,7034	0,853	0,9523	1,0875	1,2349	1,3468	1,4844	1,6538
	12	1,8288	1,9979	2,1465	2,3027	2,4674	2,6434	2,8068	2,9675				
33	0	0,1523	0,2676	0,4172	0,5676	0,6782	0,7917	0,938	1,0911	1,2527	1,4017	1,5428	1,7005
	12	1,8617	2,0219	2,1783	2,3305	2,4898	2,6492	2,809	2,9696				
34	0	0,1429	0,2452	0,3811	0,5172	0,6436	0,78	0,9157	1,0533	1,2193	1,3916	1,5431	1,6877
	12	1,8483	2,0113	2,1709	2,3314	2,4933	2,6543	2,8159	2,9774				
35	0	0,1429	0,2755	0,4623	0,6348	0,7367	0,8006	0,9024	1,082	1,2703	1,4362	1,597	1,7439
	12	1,8992	2,0596	2,2391	2,3879	2,5199	2,6538	2,8044	2,9714				
36	0	0,1755	0,2674	0,3797	0,5093	0,6128	0,768	0,9463	1,0888	1,2435	1,3842	1,5028	1,653
	12	1,8138	1,9719	2,1481	2,3143	2,4863	2,643	2,8036	2,9727				
37	0	0,1028	0,2019	0,4564	0,5772	0,6168	0,7817	1,0249	1,1519	1,2362	1,3784	1,5467	1,7077
	12	1,8748	2,0347	2,1629	2,3126	2,4791	2,633	2,7958	2,9645				
38	0	0,1552	0,2539	0,3842	0,4973	0,6185	0,7823	0,9783	1,2079	1,3287	1,4312	1,5682	1,6595
	12	1,7912	1,9505	2,1305	2,3021	2,4812	2,6502	2,8077	2,973				
39	0	0,144	0,2506	0,4179	0,5561	0,6573	0,7961	0,9659	1,1313	1,2651	1,3966	1,5404	1,691
	12	1,85	2,0157	2,1777	2,334	2,4927	2,6505	2,8105	2,9723				
40	0	0,1231	0,2362	0,392	0,5375	0,6822	0,839	0,9828	1,1363	1,2963	1,4559	1,6114	1,7592
	12	1,9089	2,0622	2,2153	2,3634	2,5155	2,6669	2,8236	2,9771				
41	0	0,1644	0,3065	0,4254	0,5421	0,7221	0,8851	0,9748	1,1407	1,3215	1,4231	1,5704	1,757
	12	1,8728	2,0114	2,1897	2,3369	2,4832	2,6524	2,7991	2,9625				
42	0	0,1541	0,2444	0,3696	0,4892	0,584	0,7198	0,9029	1,0817	1,2504	1,4125	1,5675	1,7227
	12	1,8809	2,0445	2,2064	2,3566	2,5114	2,6657	2,8226	2,9783				
43	0	0,1685	0,2822	0,4073	0,531	0,678	0,826	0,9442	1,0943	1,2424	1,3927	1,5633	1,721
	12	1,8769	2,0336	2,1866	2,3397	2,4986	2,6565	2,8165	2,9738				
44	0	0,1595	0,2721	0,4006	0,518	0,6605	0,8306	0,9808	1,1277	1,3003	1,4401	1,5503	1,6986
	12	1,8766	2,0486	2,2281	2,3824	2,5148	2,6498	2,7995	2,9672				
45	0	0,1576	0,2128	0,3353	0,4837	0,5821	0,6951	0,8633	1,013	1,2138	1,4012	1,5417	1,7026
	12	1,8413	2,004	2,1891	2,342	2,5054	2,6629	2,8164	2,9719				
46	0	0,1623	0,2444	0,3882	0,541	0,6619	0,8141	0,9687	1,1125	1,2987	1,4352	1,5293	1,6206
	12	1,7733	1,9671	2,1671	2,3417	2,5032	2,6485	2,8044	2,975				
47	0	0,1645	0,2494	0,4065	0,5624	0,7081	0,8553	0,9783	1,117	1,2605	1,3666	1,5037	1,6825
	12	1,8595	2,0285	2,2063	2,3573	2,512	2,6642	2,8209	2,9768				
48	0	0,1412	0,2599	0,4089	0,5763	0,7575	0,9404	1,0963	1,2039	1,2875	1,4104	1,5831	1,7469
	12	1,9057	2,0655	2,2198	2,3592	2,5008	2,6519	2,8075	2,9645				
49	0	0,1506	0,2249	0,3354	0,4702	0,6394	0,8209	0,9874	1,1137	1,2158	1,3713	1,5632	1,7234
	12	1,8511	2,0006	2,1659	2,3298	2,4952	2,6518	2,8204	2,9774				
50	0	0,1427	0,2519	0,4087	0,5698	0,73	0,8871	1,0136	1,1402	1,2784	1,4188	1,5715	1,7279
	12	1,8821	2,0373	2,1915	2,3442	2,5019	2,6578	2,8168	2,9741				
51	0	0,1546	0,2464	0,3871	0,5358	0,6887	0,8628	1,0297	1,1593	1,2707	1,3877	1,5348	1,7029
	12	1,8544	2,0102	2,1646	2,3194	2,485	2,647	2,8099	2,971				
52	0	0,1563	0,2624	0,3977	0,5517	0,7151	0,8666	0,9931	1,1265	1,2623	1,419	1,5703	1,7052
	12	1,8638	2,0244	2,1786	2,333	2,4926	2,6513	2,8121	2,9714				
53	0	0,1756	0,3055	0,4336	0,5642	0,6999	0,8519	1,0053	1,1666	1,3314	1,4941	1,6493	1,7804
	12	1,9026	2,041	2,1964	2,3556	2,5057	2,6603	2,8194	2,972				

Окончание таблицы А.20

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
54	0	0,1584	0,3036	0,4517	0,5956	0,7302	0,8717	1,0275	1,1963	1,3597	1,5018	1,6108	1,7443
	12	1,9113	2,0817	2,223	2,3474	2,489	2,6508	2,8187	2,9778				
55	0	0,1863	0,2831	0,4032	0,5544	0,7006	0,8505	1,0197	1,1847	1,3216	1,427	1,5489	1,695
	12	1,8516	2,0187	2,152	2,3037	2,4735	2,6409	2,804	2,9682				
56	0	0,1365	0,2121	0,3269	0,4585	0,6069	0,7824	0,9466	1,1138	1,282	1,4431	1,6006	1,7496
	12	1,9023	2,0596	2,2231	2,3686	2,5198	2,6706	2,8259	2,9791				
57	0	0,1418	0,2012	0,3235	0,4318	0,5561	0,7305	0,9193	1,0594	1,2154	1,3873	1,5191	1,6691
	12	1,8355	2,0049	2,1635	2,3106	2,4786	2,6498	2,7995	2,9615				
58	0	0,1298	0,2535	0,4368	0,6313	0,8039	0,9111	0,9697	1,0732	1,2481	1,4316	1,5946	1,7482
	12	1,8986	2,0577	2,219	2,3698	2,5216	2,6719	2,829	2,9839				
59	0	0,164	0,2683	0,3859	0,5464	0,721	0,8791	0,9947	1,1248	1,2728	1,42	1,5777	1,6956
	12	1,8111	1,9719	2,1449	2,334	2,5168	2,6671	2,83	2,982				
60	0	0,16	0,2608	0,4067	0,5504	0,6802	0,817	0,9717	1,1255	1,2888	1,4648	1,6376	1,8126
	12	1,9239	2,0146	2,1353	2,3044	2,4523	2,618	2,7863	2,9651				
61	0	0,1642	0,2736	0,3797	0,5014	0,6611	0,8243	0,9655	1,1203	1,2834	1,4431	1,604	1,7534
	12	1,9015	2,0469	2,1684	2,318	2,484	2,6484	2,8086	2,9696				
62	0	0,15	0,273	0,4163	0,5685	0,7419	0,9107	1,0222	1,1298	1,2903	1,4624	1,6191	1,7762
	12	1,9303	2,0511	2,1871	2,3528	2,5078	2,6608	2,8287	2,9768				
63	0	0,1448	0,2603	0,3473	0,4684	0,6635	0,8312	0,9606	1,124	1,2812	1,3949	1,5717	1,7249
	12	1,9085	2,0455	2,1565	2,29	2,4086	2,6323	2,8375	2,9776				
64	0	0,0383	0,0567	0,0096	0,0058	0,0176	0,0025	0,0011	−0,0004	0,0199	0,0127	−0,0218	−0,0168
	12	0,0079	0,0142	−0,0017	0,0067	0,0106	0,0125	0,01	0,0094				
65	0	0,0081	0,0043	−0,0335	−0,0402	−0,012	0,0213	0,042	0,0507	0,0238	−0,0191	−0,0249	−0,0022
	12	0,0109	0,0117	0,0497	0,0557	0,0464	0,035	0,0275	0,019				
66	0	−0,0261	−0,048	−0,046	−0,014	0,0035	0,0071	0,0125	0,0034	−0,0005	0,0004	0,0096	0,034
	12	0,0524	0,0495	−0,0303	−0,0281	−0,008	0,0009	−0,0004	0,0039				
67	0	0,0102	−0,0001	−0,0296	−0,0006	0,0327	0,0346	0,0448	0,0414	0,0344	0,0273	0,0219	0,0264
	12	0,0257	0,0208	−0,012	−0,0134	0,0141	0,0377	0,0392	0,0162				
68	0	0,0104	0,0071	0,011	0,0152	−0,0102	0,0115	0,0289	0,0372	0,0476	0,0562	0,0607	0,0667
	12	0,0618	0,0449	0,0256	−0,0128	−0,0182	0,0032	0,005	0,0107				
69	0	−0,0177	0,0338	0,0173	0,0015	0,0224	0,0144	0,034	0,0447	0,0526	0,0358	−0,0029	−0,0166
	12	−0,0123	−0,0052	0,0066	−0,0246	−0,0336	−0,0143	−0,0047	0,0054				
70	0	−0,0163	0,0226	0,0214	−0,0236	−0,0324	−0,0239	−0,0003	0,0098	−0,0045	−0,0015	0,0045	0,0126
	12	0,0129	0,0104	−0,0208	−0,0085	0,0011	0,0066	0,0228	0,0289				
71	0	0,0096	0,0184	0,0142	0,0422	0,0419	−0,0037	0,015	0,0267	0,0238	0,0191	0,0134	0,0177
	12	0,0218	0,0207	0,041	0,0594	0,0599	0,0113	−0,0264	−0,0092				
72	0	−0,0428	−0,0164	−0,002	−0,0084	-0,0008	0,0143	0,0203	0,0009	−0,0134	−0,0132	−0,0215	−0,0289
	12	−0,0202	−0,0083	0,0009	0,0297	0,0171	−0,015	−0,0062	0,0079				
73	0	−0,0259	0,013	0,0487	0,0691	0,0466	0,035	0,0465	0,0052	−0,0155	0,0023	0,0134	0,0146
	12	0,0142	0,011	0,029	0,0326	0,0437	0,0669	0,0835	0,0498				
74	0	−0,0035	0,0321	0,054	0,0382	0,0055	0,0101	0,0146	−0,0031	−0,0324	−0,0381	−0,0189	0,0044
	12	0,0177	0,0185	0,0471	0,0252	−0,0066	−0,0072	−0,0129	−0,0052				
75	0	−0,0373	−0,0028	0,0375	0,0295	−0,0016	−0,0017	−0,0022	0,0084	0,0219	0,0322	0,0421	0,0369
	12	0,0094	−0,0019	0,0096	0,0242	0,0324	0,031	0,0363	0,0306				
76	0	−0,0433	−0,0333	0,001	0,0338	0,034	0,0181	−0,0051	−0,0337	−0,017	0,0115	0,0188	0,0132
	12	0,011	0,0114	0,0232	0,025	0,0232	0,0203	0,015	0,0128				
77	0	0,0552	0,0672	−0,0093	−0,0535	−0,022	−0,0204	−0,0284	−0,0194	−0,0177	−0,0228	−0,0217	−0,0114
	12	−0,0043	−0,0033	−0,0012	0,0158	0,0533	0,0514	0,0011	−0,0072				
78	0	0,0209	0,0653	0,0635	0,0541	0,0327	−0,0325	−0,0418	−0,0221	−0,002	0,0078	0,0096	0,0202
	12	0,0285	0,0278	0,0574	0,0428	−0,0096	−0,011	0,026	0,0358				
79	0	0,0146	0,062	0,0578	0,0008	−0,0285	0,0054	0,0183	0,0136	0,0081	0,001	−0,0056	0,0145
	12	−0,0033	−0,0204	0,038	0,0141	0,0198	0,0573	0,031	−0,0055				
80	0	0,4989	0,4993	0,499	0,4995	0,4999	0,4985	0,4973	0,499	0,4977	0,4971	0,4977	0,4995
	12	0,4969	0,4986	0,4996	0,4996	0,4999	0,4989	0,4937	0,4906				
81	0	0,2664	0,3126	0,3112	0,3132	0,2987	0,2609	0,2724	0,2535	0,2363	0,2679	0,2746	0,2536
	12	0,2708	0,269	0,2831	0,2939	0,262	0,2428	0,2456	0,2665				

Таблица А.21 — Сборник кодов 0 чередования *VQ* для базового кодера (тип окна: длинное)

Имя файла	Содержание	Режим	Число элементов	Число векторов
cdmdct 0	*MDCT*	Базовый длинный	20	32

VN	*EN*	0	1	2	3	4	5	6	7	8	9	10	11
0	0	–165,2	44,8	83,7	–31,8	–54,7	79,8	104,3	–11322,7	–46,7	–5,7	139,8	–38,4
	12	–59,7	–101,2	–5,6	134,4	34	113	–738,8	80,1				
1	0	17,2	4,2	148,2	54,7	54,2	–11,9	74,7	35,2	73,9	90	9523,9	73,2
	12	1,1	–47,8	–5,7	–966,6	60	120,8	–189,5	–59,4				
2	0	395,5	136	5816,1	–64,1	–89,2	17,3	371,2	–9,6	11,2	–10,6	–31,9	27,1
	12	–41,5	31,5	–496,7	–138,5	–2108,3	–272,2	–99,2	262,8				
3	0	–1136,4	–142,9	73,5	–92	–544,9	–28,6	–79,1	–85,1	136,3	–5,4	–122,4	–81,2
	12	–6,2	5,8	–123,2	11554,5	4,3	–71,2	–22,1	–203,1				
4	0	–56	14,4	30	–2,6	29,7	–150	96,5	71,3	24,7	–130,8	–7,1	5682,9
	12	–7,6	43,1	15,6	35,9	–219,7	–84,9	–1051,1	647,5				
5	0	209,2	–93,8	–105,8	–18,2	51,5	22,9	–38,3	88	4,6	–38,3	–118,2	328,6
	12	12156,2	–111	–47	109,5	–3,2	116,4	–156,3	51				
6	0	69,1	–74,3	15,4	184,4	–165,6	–97,8	23,8	12,2	–12330,6	–72,9	57,2	42
	12	–55,7	61,3	–107,8	98,3	167,9	32,8	146	6,3				
7	0	58,7	728	–3068,5	–263,4	–353,1	–398,7	2696,7	95,2	–202,9	–28,2	–5,6	45,7
	12	22,5	13,1	–66,7	42	–269,3	–467,2	–208,8	2168				
8	0	17,8	–3857,2	–214,1	221,1	–3488,2	83,3	–162,9	–260,7	3,5	–46,1	32,8	–73,8
	12	–1	–53	397,9	227,4	–961,9	286,9	–229	364,3				
9	0	–7050,2	4,7	168,8	–40,4	51,1	–143,8	–141,4	–56,7	–159,6	79,5	–8,1	23,2
	12	2,3	–51,7	54,7	186,3	6,3	44,6	–91,9	54,7				
10	0	–329,7	–636,1	207,9	115	–178,2	–103,3	65	–120,6	–4683,1	–975,1	–77,1	85,8
	12	53,5	42	184,9	–51,3	355,2	–112,8	–321,5	79,8				
11	0	14,4	32,5	1394,3	–151,4	48,1	–35,6	293,8	4278,1	–118	173,6	21	–36,7
	12	–7,3	37,1	975,2	253,8	213,2	476,8	–1456	–23,3				
12	0	255,3	–5356,6	89	351	429,9	175,9	381,5	39,8	–315,5	43,2	57,3	196
	12	9,8	33,2	1035	92,4	1186	31,7	145,9	–27,5				
13	0	–2680,3	260,6	3250,2	–8,1	–326,9	12	–191,2	237,9	–30,4	–41,6	28,7	10,7
	12	–20,2	11,9	–456,6	146,7	50	–546	–1884,2	295,5				
14	0	144,6	–204,4	166,4	6474,3	–373,7	49	117	–49	–2,2	9,8	–7,6	–70,1
	12	88,3	–27,5	201,1	102,2	58,5	34	95,8	36,3				
15	0	953,3	23,7	296,6	91,9	50,3	–338,5	6794,1	7	0,2	77,2	27,9	96,8
	12	99,2	11,7	–48,9	7,5	138,1	39,4	426,9	26				
16	0	–1,2	–408,9	–306,6	–234,8	89	–555,9	22	106,6	98	11,5	–4,6	–15,3
	12	–52,1	58,4	11645,8	–104,3	–52	–66,8	162,1	–38,7				
17	0	–2503	–167,9	–131,7	159,9	75,5	–314,6	3624,7	–25,9	229,1	–144,9	30,6	–191,3
	12	51,7	–35,8	–69,2	–178,6	–196,6	202,2	165,9	–165,6				
18	0	14,1	–8789,4	–46	49,4	–14,4	112,4	–30,6	82,4	–160,4	21,3	74	50,2
	12	82,7	53,1	59,1	–15,9	–285,9	91,9	145,8	240,5				
19	0	–331,2	–1859,9	19,9	343,1	–63,4	3389,2	541,5	485,4	129,3	214,5	29,4	19
	12	33,4	–15,1	–1094,3	–528,6	266,3	–185,5	–478,7	–185,8				
20	0	–123	3966,6	–17,5	4279,8	–15,7	–47,5	239,8	76,4	9,8	–3,7	–2	29,3
	12	49,4	–109,1	182,2	–47,5	158	–80,9	542	–328,9				
21	0	–213,5	492,4	–408,5	2968,7	99,5	–651,8	–379,6	728,1	–230,4	192,2	84,5	–130
	12	–20,3	–78,7	–502,3	–74	–11514,1	–172,9	–882	–26,7				
22	0	–401,2	120,3	23,1	–233,7	3403	–332,7	47,4	23,8	190,7	–735,1	131,1	–83
	12	–30,7	–147,3	–263,5	306,2	–54,8	11629,9	308,2	–528				
23	0	–308,5	1931,8	659,7	126	–4247,4	–208	347,9	207,5	76	–238,6	184	–58
	12	–52,3	–23,7	194,3	165,8	46,6	400,7	–1299,6	–120,9				
24	0	–3467,8	–4196,2	–49	–28,7	–107,2	–172,7	–355,4	210,5	–72	–118,8	41,6	–159,4
	12	77,6	–97,3	–136,3	47,9	–402,7	132,7	–101,5	–288,8				
25	0	–71	517,2	–413,7	–1414	–374,7	–86,5	–285,7	322,6	–473	351,9	436,9	–71,8
	12	62,9	–4541,1	–314,8	271,7	25,2	250,9	–436,7	500,3				
26	0	94,6	–66,5	311,2	3960,3	3911,6	528,5	309,7	–213,5	74,9	52,6	28,3	–52,8
	12	–48,6	–101,2	–18,1	–283	671,4	–38,7	282,7	39,6				
27	0	692,1	–48,2	–87,8	112,4	43,7	7489	–30,8	-63,6	87,2	–13,4	59,8	25,1
	12	–82,8	4,7	–162	26,4	–94,8	–86	190,6	–353,1				
28	0	–4033,6	182,3	–237,9	–49	–181,6	453,2	–508,9	–56,1	–86,6	–51,4	90,5	–4,4
	12	–19,8	53,4	–97,7	–284,1	180,3	263,8	3747	0,8				
29	0	1019,4	–1791,3	15,8	252,2	83	–3434,4	132,8	–87,3	93,9	199,1	–149,5	–30,6
	12	14,1	31,4	–17,3	828,3	76,8	–403	21,1	541,3				
30	0	7,4	–0,4	10,4	165,8	–24,5	49,6	54	–88,5	96,7	–12001,5	–61,2	2,8
	12	43,7	–39,3	292,9	–87,4	–132,9	–36,3	538,9	45,2				
31	0	83,6	–75,6	37,8	–16,5	10805,3	65,4	–23,6	17	17,1	–73,1	21,4	10,9
	12	64	19,7	46	84,1	–112,2	–55,3	244,2	134,2				

Таблица А.22 — Сборник кодов 1 чередования *VQ* для базового кодера (тип окна: длинное)

Имя файла	Содержание	Режим	Число элементов	Число векторов
cdmdct 1	*MDCT*	Базовый длинный	20	32

VN	*EN*	0	1	2	3	4	5	6	7	8	9	10	11
0	0	235,3	67,3	50,9	–70,8	141	–11826,5	–180,2	–63,5	101,1	–67,7	–52,8	–26,3
	12	3,1	9,5	–213,3	91,1	–19,6	–208,5	185,1	155,8				
1	0	–2551,4	–264,8	68,9	–3066,5	–340,7	139,4	130,4	26,9	60,2	35,7	–4,1	74,7
	12	–35,7	10,4	–215,4	–28,2	182,3	258,3	272,9	–157,2				
2	0	625,1	–205,8	260,9	–388,8	–46,3	–37,3	40,9	–162,9	–59,8	–66,2	–57,3	–204,2
	12	–46,2	54,2	121,4	–154,5	–12125	83,4	421,2	168,3				
3	0	–209,8	227	–229,9	–47	6981,7	–61,2	43,9	–60,2	–68,3	50,4	34,3	–0,2
	12	16,8	10,1	–144,3	–9,7	–127,6	–128,7	–320,2	195,7				
4	0	3358,8	295,2	–171,2	–205,5	–2008,6	–196,7	25,4	–112,9	–50,6	–304,3	–65	–49,7
	12	–54	–35,7	–2190	54,1	–279	491,8	37,6	–327,2				
5	0	–11,3	–341,2	158,5	133,3	–158,6	2,8	–11,8	–38,9	6211,1	–519,8	–58,8	–26,8
	12	17,3	–57,9	341,8	–294,8	20,2	38,5	378,2	101,3				
6	0	66	85,5	1894,5	3096,2	–1994,1	40	–103,1	–369,4	–93,3	–65,3	151,1	34
	12	12,1	–27,7	–488,9	–136,9	39,4	1058,2	385,7	370,2				
7	0	–8	–8,6	–44,9	18,7	–73,3	50,5	–4	–61,6	–14,4	1,7	5132,2	–25,8
	12	147,1	42,4	18,2	2672,9	182,8	–251,4	–329,7	378,2				
8	0	–4,6	165,2	–55,1	9383,7	117,1	–95,3	–115,3	18,8	–0,1	59,2	70,4	–45,5
	12	21,6	–8,3	29,7	–51,5	294,9	–117,1	–209,5	–44,2				
9	0	–214,8	–80,8	47,6	–433,9	371,7	5001,4	–330,3	27,5	37,4	–44	–255,9	–153,2
	12	63,1	–12,2	–152,5	552,9	216,9	–145	–903,1	512,9				
10	0	1342,8	–240,9	–920,2	98,8	–132,1	–192,2	–158,8	3771,8	–21,2	–158,4	–48,4	107,4
	12	60,7	–10,7	–13,8	–227,2	240,3	478,1	319,2	496,1				
11	0	4,8	–5600,1	492	159,7	–55,6	–26,9	–318,6	121	149,8	91,1	126,5	–130,8
	12	–63,2	89,2	–228,1	–49	132,7	–101,9	95,8	–197,4				
12	0	356,8	2654,8	–358,9	114	103,3	85,6	22,1	–37,8	106,3	1,2	–15	–49,8
	12	–72,9	170,8	–66	36,2	2617,9	–162,5	503,6	1981,6				
13	0	–9,4	127,7	23,1	–66,9	–122,5	–198,8	101,7	56,8	41,2	89,5	–60,9	30,3
	12	5664,6	159,7	213,8	–162,9	–59,2	84,3	–33,3	–259,1				
14	0	–103,3	42,4	–1	–180,8	44,4	743,2	4688,1	28,2	–81,9	51,3	69,5	61,2
	12	–106,5	–11,6	–37	–126,5	–27,4	164,2	–136,7	–88,4				
15	0	–1042,2	–313,7	–154,2	–414,9	650,3	–2755,9	–161,8	364,9	103,8	27,5	–94,9	–172,6
	12	–75,1	108,1	–7700	–310,2	161,1	–56,1	–341,6	–34,1				
16	0	–54,2	–3008,6	–3909,4	–85,6	–62,9	–56,7	108,9	40	267,7	–151,7	144,4	–63,7
	12	–221,4	–67,8	193,4	55,7	572,8	–265,8	–565,6	–407,7				
17	0	–63,3	231,6	–3415,5	601	575,7	1192,1	–742,8	47	–427,7	29,6	58,8	46,8
	12	–87,6	101,1	–843,8	314,8	–1669,2	–178,3	–315,2	–230,1				
18	0	558	–233,4	130,9	5,3	42,9	–227,1	63,8	–7070,4	–35,4	27,3	–70,9	–14,2
	12	30,6	127,3	499,7	11,2	343,1	789,6	860,6	325,5				
19	0	–3560,6	3198,6	–27,2	19	3,2	–363,7	52,9	257,5	–150,4	128,9	–159,8	–58,2
	12	1,2	17,3	–15,3	28,7	–376,5	–186,5	121,6	–72,7				
20	0	–1971,3	–21,1	–60,6	3407	–171,8	35,7	–15,5	65,3	173,8	–58,9	2,5	70,6
	12	–48,5	–24,8	306,4	156,2	476,7	–327,6	–2828,6	–905,5				
21	0	–580,4	–214,9	–1765,9	76,6	–4029,4	–5,1	–172,3	–113	51,4	–152	15	–37,4
	12	27,1	–56,2	266,6	–408	589,2	298,9	941,4	–136				
22	0	297,5	–16,5	62,7	–49,7	3,1	97,1	–11428,2	64,4	11,5	64	32,8	–114,3
	12	2,9	10	–32,2	117,6	–48,1	68,9	293,1	14,3				
23	0	–49,9	2192,6	578,3	–4411,8	–44,1	134,5	–64,7	55,5	–10,8	–3,7	–90,7	3,1
	12	–46,5	92,6	431	139,3	–407,1	15	–167,1	–108,2				
24	0	28,8	–7,1	121	–63,5	79,6	–35,6	–14	45,6	–201,3	5649,4	–13,3	–12,9
	12	43,6	26,2	155,9	–306,9	–134,7	22,8	556	–21				
25	0	6,7	–62,4	–23,1	790,1	199,2	–93,1	72,3	–203,9	–26,1	–374,6	–437,6	219,8
	12	–139,7	–9142,4	30,3	–176,5	119,9	–146,4	–595	–391,8				
26	0	2816,7	122,3	3753,5	–33,5	242,4	245,3	233,1	–340	17,7	–215,5	40,5	–58,3
	12	6,6	29,7	–254,2	–2,6	675,4	–347,6	506,2	227,3				
27	0	–682,7	–375,4	5715,8	196,7	236,8	214	21,2	229,6	–232,8	–41,2	–54,1	2,8
	12	–74,8	–148,1	–103,5	–55,2	298,6	–263,6	393,1	–462,4				
28	0	–10807,7	112,1	–283,5	173,9	368,5	172,3	46,9	208	51,4	226,4	72,3	108,1
	12	–76,5	37	543,5	–277,6	–291	–178,8	–246	228,4				
29	0	–58,2	–42,1	–36,1	–25,6	–78,7	315,2	193,4	11,3	47,5	46	44,7	12181
	12	–7,5	–10	–31,3	106	255,7	78,9	571,9	–518,5				
30	0	–102,9	20,3	–9260,7	53,2	55,9	133,6	125,5	202,7	–66,6	–62,6	–7,3	2,2
	12	–0,6	56,8	–6,7	–14,7	–340,4	–160,4	–419,5	–241,2				
31	0	5816,2	60,8	–128,6	–76,2	350	99,9	–81,6	133,1	–103,3	–19,4	56	–81
	12	–87,3	–14,7	276,2	194,7	–148,2	129,3	–495,4	–270,1				

Таблица А.23 — Сборник кодов 0 чередования *VQ* для базового кодера (тип окна: короткое)

Имя файла	Содержание	Режим	Число элементов	Число векторов
cdmdct 2	*MDCT*	Базовый короткий	17	32

VN	*EN*	0	1	2	3	4	5	6	7	8	9	10	11
0	0	16384	–534	296,2	242,6	–1011,7	–156,3	–25,8	–720	135,3	–288,9	–197,1	339,2
	12	12,6	484,7	68,3	1615,2	–524,1							
1	0	–7,6	–264,3	–165	453,1–	–8874–	–37,8	918,3	–257,6	183,8	–144,9	115,8	63,6
	12	352,4	137,9	322,3	630,2	5343							
2	0	15,7–	337,7–	–1005,3	4231,3	–2785,4	214	–3,1	311,3	–296,8	99,8	–496,3	186,7
	12	33,9	685,2	642,6	633,4	4888,4							
3	0	149,2	–252,5	–238,4–	–191,4–	45,6	154,2	–498,2	–4264,6	–1516,9	985,7	–939,2	–419,5
	12	1279,1	581,1	1809	951,5	1015,2							
4	0	239–	–72–	–104,5–	882,2	–73,4–	24,2	–187,9	141,5	–9608	337,9	253,2	433,7
	12	533,1	1421,2	2313,7	877,7	1429,2							
5	0	233,8–	–356,3	–704,4–	218,4	–766,8–	–226,7	–7255,4	–139,7	229,3	–439,2	–523	230,3
	12	173,3	1689	806,6	2720	1824,9							
6	0	–651,9	677,9–	–425–	–7670,7	–637,7	168,6	–751,9	–37,3	–471,9	193	137,8	293
	12	367,1	910,5	542	–4284,2	5243,1							
7	0	–45,2–	609–	–135,9	–740,3–	2550,1	3678,4	1197	774,5	–1783,1	355,6	–191,8	–292,4
	12	293,4	196,9	199,3	2391,9	429,2							
8	0	819,6–	–838,5–	–374,1	–704,7	–329,2–	310,2	404	–204,9	94,1	1035,5	–487,3	1144,9
	12	0,1	8832	1367,5	120,2	1549,2							
9	0	–274,8	–263,5	590,7–	723,8	420,4–	–272,4	346,6	–478,9	3294,9	1890,3	–1243,6	–200,1
	12	96,6	232,9	3382,5	953,2	1167,4							
10	0	–113,3–	–594,2	–9768,3	80,5–	–315,6–	12,5	281,2	–97,8	–37,7	427,1	310,7	–30,2
	12	90,6	92,8	909,2	684,6	954,4							
11	0	–92,3–	141–673	576,4	–24,4–	56,3	–1025,6	115,2	–8990,2	176,6	144,3	86,6	–190,4
	12	1808,1		452,2	29	3351,3							
12	0	152,2–	2705,2	2093,3	–1395,5	27,3–	–76,7	835,9	–54,8	–363,6	690,4	–316,8	–151,5
	12	539,3	492,3	2115	777,4	3297,1							
13	0	–266,9	–272,8–	–464,7–	262,5	0,4–	11,9	441,1	–267,7	–33,5	361,3	4513,6	–476,3
	12	12,5	138,8	965,3	2941	760,6							
14	0	–5805,6	289,6–	469,5–	1712,9–	–169,3	150,8	–1835,6	734,9	90	551,8	–236,2	–323,6
	12	–172,4	5589,5	2574,5	1050,4	1301,2							
15	0	–3543,8	329,1	–598,6	–274,4–	1067–	973,2	856,7	549,2	–31,2	–82,9	–907,1	671,3
	12	–115,7	2434,7	2131,6	931,5	1369,2							
16	0	–355,6	242,4	–267,9	182,5	177,8	–134	85,2	806,1	117,7	–121,4	–738,8	–6574
	12	–813,8	–615,8	–632,4	390,3	1416,5							
17	0	–18,8	247,7	2109,5–	–2071,1	–3397	1029,1	474,4	–52,4	620,7	63,1	518,2	–206,5
	12	288,5	1795,3	2903	–2856,7	1327,6							
18	0	260,8	496,6–	695,4–	–1262,8	409	–521,2	–2073	1709	1929,4	69,5	–1172,1	–1298,6
	12	3453,2	77,8	1448,3	1117,8	1309,2							
19	0	3500,7	–3781,1	337,1	350,8–	620,7	–166,5	507,2	–305	–15,5	–51	–343,1	310
	12	21,2	1445,3	1738,2	576	1721,7							
20	0	–32,2–	–54,2	–1286,9	76,5	–1713,8	–455,4	3055,5	1436,6	345,3	–55	94,7	–580,7
	12	39,9	1640,6	2128	625,4	–2599,5							
21	0	–178,5	–715,7–	–315,2–	–671,3	–1472,1	4815,1	–753,8	83,8	317,6	136	559,3	–192,3
	12	690	2273,6	2537,7	1253	–3324,6							
22	0	591	–267,2	–299,1–	–485,2	–423,5	–38,2	674,1	–2357,8	1102,6	–3306,7	–757,9	–302,3
	12	564,8	2143,9	754,7	2681	611,6							
23	0	–525,9–	–124,6	–189,3–	111,3–	427,3–	–505,8	–420,1	127,8	–364,7	–7157,5	–47,4	226,7
	12	380,5	195,8	1269	18,3	204,7							
24	0	–492,2–	–4261,3	1940,7–	1797,6–	341,4–	601,3	611,3	27,5	252,7	151,2	390,8	–54,1
	12	1216,6	2894,2	3176	1731,3	1433,6							
25	0	–12834,3	–93,1	–374,5	287	–1899,4	35,7	–715,3	–307,9	10,6	78,9	1096,8	34,3
	12	103,6	2076,4	615,2	1465,3	758							
26	0	1855,5–	683,3	–272,6–	–279,2	818,5–	–796,5	85,7	57,3	161,8	–7,5	271,7	–197,5
	12	6743	2581,4	1422,1	175	1154,1							
27	0	–129,1–	–379,4–	–4225,6	–2308,9	247,6	246,7	–478,2	234,3	327,2	–193,4	–380,8	165,5
	12	322,4	1003,3	–706,7	135,4	2359,3							
28	0	–8970,6	625,4–	316,8	–108,1	625,2	1014,1	934,6	–815,1	–137,6	–365,9	–392,8	–204,2
	12	–130,7	195,9	1439,7	3056,3	603,4							
29	0	–90–	1303,7	–1332,6	167,8	–600,4	2072	–1718,3	213,6	125	–37,1	21,4	106,7
	12	3498,4	2942,1	–421,4	1483	5828,9							
30	0	478,7	7218,4	129,5	141,4	732,8	36,2	296,8	–233,7	72,8	121,4	–214,2	–431,6
	12	2,4	741,3	1486,1	1177,2	1714,4							
31	0	–4432,8	–852,9	–2584,9	–89,8–	–780,6–	–722	–88,5	–1466,4	32,9	161,5	–65,6	–642,8
	12	207,1	735,6	–5723,4	841,8	361,4							

Таблица А.24 — Сборник кодов 1 чередования *VQ* для базового кодера (тип окна: короткое)

Имя файла	Содержание	Режим	Число элементов	Число векторов
cdmdct 3	*MDCT*	Базовый короткий	17	32

VN	*EN*	0	1	2	3	4	5	6	7	8	9	10	11
0	0	210,8	–13952,3	–184,9	–354	562,2	506,7	–239	–758,3	–199,4	205,1	32,8	–1105,2
	12	–674,6	577,5	–224,6	–325,9	–1149,9							
1	0	–413,1	83,5	–336,7	–1688	–1187,9	–3898,7	–171,5	897,3	–485,8	695,3	643,6	–319,2
	12	221,8	–3611,9	3212	–1658,3	–2700,4							
2	0	–331,7	418,3	–46,9	126	865,1	463,8	–6325,3	271,8	176,6	–314,8	–18,7	–555,5
	12	795,3	–474,9	628,7	–2206,3	55,7							
3	0	–4269	–392,4	3219,7	–539,8	14,4	–774,7	–262,6	–302,9	46,9	70,7	–63,6	120,8
	12	59,4	811,7	4138,9	–9,2	112,3							
4	0	767,6	–710,4	–7343,7	108,3	660,1	–256,2	–137,4	–77,8	–77,4	276,4	–25	325,9
	12	46,3	–578,7	2070,7	1799,3	1214,2							
5	0	–141,8	3251,9	–1072,7	343,9	–593,7	–29	191,1	1221,3	–369,9	–22,8	–530,7	1110,2
	12	–1079,3	–47,4	–3613,6	–3544,2	1237,2							
6	0	–241	877,9	–36,2	125,9	253,3	8498,6	–447,2	–525,9	685,6	502,4	–196,4	1546,3
	12	755,3	382	305,2	–797,4	–1413							
7	0	849,4	–216,3	375,9	–12034,9	–438	–444,1	250,6	80,4	1147,9	193,8	413,3	66,6
	12	–56,1	1401	–1495,1	–445,3	–475,5							
8	0	3884,2	4137,2	298,4	3,6	326	225,8	367,8	–384,3	147	–7,4	358	–281,7
	12	447,3	423,2	1928,4	–3456,7	–2995,4							
9	0	482	292,9	–1031,8	47,7	–3246,4	248,2	666,6	–548,3	–470,8	430,4	–840,3	–332
	12	156,3	–5991,7	–387,9	–641,7	–316,6							
10	0	–231,9	–198,8	–307,8	452,8	12,6	–418,5	88,7	–41	358,8	–388,4	–428,7	4450,3
	12	384,5	737,2	–4,5	4933,9	1082,1							
11	0	5421,3	346,1	127	3133,9	–586,4	76,3	585,8	–319	147	–67	–700,3	206,2
	12	93,4	–4386,2	443,2	682,7	2974							
12	0	–303	–58,5	–133,6	609,8	–339,1	–79,9	112,4	577,4	3997,3	–635,8	275	–193,2
	12	573,9	1187,4	1662,3	1321,8	527,8							
13	0	102,2	–1280,8	–539,2	–4552,8	–744,6	694,4	–49	94,7	–22	–5,5	–577,7	–254,2
	12	114,7	–531,6	–3841,1	900,7	3404,9							
14	0	–573,8	117,5	–2	259,8 –	–324,1	–48,6	–1858,6	–3521,6	94,6	–161,4	908,1	–404
	12	–416,5	361,2	–792	303,6	–1569							
15	0	128,5	–2026,6	78,6	–476,9	–975,5	–358,2	–1236,6	3351,7	–75	44,7	636,2	67,4
	12	6,3	2449,2	–374,2	–126,5	–3959,9							
16	0	–30,5	–640,4	1339,6	1507,5	–493,6	–403,1	22,8	–368,6	–1413,1	–2039,5	–1884,3	–1548,5
	12	–654,6	2228,7	545,6	1654,6	–4512,5							
17	0	–9067,5	–617,7	–460,9	264,7	1549,6	54,8	–82,9	204,5	22	90,4	120,8	421,4
	12	–36,4	–445,4	524,9	811,8	1014,5							
18	0	6359	–1291,6	–175,5	–279,3	1966	183,8	–1376,2	–322,4	91,9	124,1	987,5	–377,8
	12	466	2217,7	–1671,1	–1155,7	–88,9							
19	0	–108,4	–170,5	–2132,2	3481,2	–207,9	1148,2	–640,2	202,6	191	324	103,3	–949,2
	12	3242,5	–123,1	129,5	–2247,7	125,2							
20	0	18,2	–309,1	–693,2	–63,8	5058,6	1085,2	1284,9	–674,8	395,6	253,9	–412,8	–290,1
	12	37,7	–810	1208,4	–624,7	–8533,2							
21	0	66,8	103,7	136	296,5	275,6	86,2	–373,8	–81,1	5	–404,3	6575,4	–38,4
	12	417,6	183,3	1674,7	–342,1	1390,6							
22	0	97,2	–6,6	764,7	417,9	339,4	–160,5	–22	–560,2	16,2	4354,3	–714,5	–186,2
	12	–792,8	5761,3	–2868,7	426	264,2							
23	0	–267,9	–69,1	165,6	–718,6	–826,1	2625,5	3479,2	511,1	528,3	172	259,7	–380,7
	12	507,4	278,7	803,8	373,8	870,7							
24	0	598 –	–1202,7	5005,8	–1061,1	1246,4	27,8	–6,3	358	–147,8	–604,8	–50,5	–175,6
	12	131,6	–2995,7	–1675,1	–1108,7	1871,5							
25	0	–2505,2	–282,2	145,1	399,2	1227,1	–1103,3	1568,9	–203	–65,6	104	427,1	–180,8
	12	102,3	–109,7	–2583,3	1662,1	1983,7							
26	0	60	3970,3	4094,5	844,5	–1026,5	114,6	–830,2	–38,5	568,2	–555,8	–49,7	242,7
	12	853,8	–1240,9	151,6	418,9	485,3							
27	0	58,5	56,9	156	37,8 –	–305,9	–437,6	548,6	651,9	–6621,1	284,5	–295,6	–333,5
	12	1616,7	1988	1497,3	51,3	2570,1							
28	0	–275,8	8445,6	–1183,1	849,5	–337,1	48,2	–161,5	–270,5	431,1	24,4	507,4	17,2
	12	–571,8	580,6	–2151,6	674,3	–1505							
29	0	–43,4	–348,7	2831,3	459,9 –	–784,6	1878,2	–188,4	–91,6	–819,7	1119,6	344,8	709,9
	12	160,9	1134,7	–777,8	431,3	63							
30	0	120,6	–343,5	15,8	–248,1	143,5	528,9	761,2	–6620,7	476,2	–192,8	669,9	546,6
	12	467,6	–65,3	2492,2	916,1	–781,6							
31	0	1232,8	–125,6	–52,2	–361,1	669,9	–600,5	359,3	125	–195,9	–437,1	–282,8	216,6
	12	6506,1	1893,6	–117,7	–55,5	7375,8							

Таблица А.25 — Сборник кодов 0 чередования VQ для масштабируемого кодера (тип окна: длинное)

Имя файла	Содержание	Режим	Число элементов	Число векторов
scmdct 0	*MDCT*	Расширенный длинный	28	32

VN	*EN*	0	1	2	3	4	5	6	7	8	9	10	11
0	0	128,3	–932,2	–269,6	–417,6	–94,2	340,3	–83,8	–8130,1	144,4	249	156,5	–315,9
	12	–95,4	404,7	–662,3	–283,8	94	208,8	–97,2	–56,2	97,9	396,1	182	–135
	24	–38,2	306,2	91,3	106,5								
1	0	201,2	–4276	1196,9	392,1	–5216,8	598,6	376,6	131,8	56,3	–2237	894,1	–832,2
	12	–256,5	–506,3	481,2	321,3	393,9	–91,1	103,8	452	170,5	235,3	269	459
	24	166,1	–664	–17,3	339								
2	0	2617	85,4	–167,7	–1804,1	4414,1	498,8	1432,6	–1359,2	632,5	–4937,2	–415,2	174,3
	12	–1008,8	300	112,1	70	158	–152,1	275,7	–391,9	50,5	136	–80	–413,4
	24	9	468,8	46,9	325,5								
3	0	301,2	137,9	–1784,8	45,9	–823,1	4,9	144,8	154	322,9	156,4	51,7	–162,2
	12	275,2–	319,9	–146,4	–2,5	221,7	35,6	–16,2	688	148,6	348,9	8192	–112,7
	24	629,1	218,2	373,9	72,8								
4	0	–3885,1	–446,4	–519,5	566,8	662,8	–1059,8	5754,8	251,3	–416,1	–798,9	–728,2	697,9
	12	–679,3–	–473,4	–192,3	–159,6	–119,9	–81,8	–66,8	–37,8	–143,7	–33,9	–20	40
	24	93,6	–47,5	–192,2	–117,1								
5	0	754	275,7	–345,7	129	259,6	–803,1	–85,8	374	–198,4	323,5	470	1314,9
	12	956,8	–319,2	–30,2	1585,1	–12,5	–123,9	–8192	–343,6	–109,5	–101,9	130	–686,1
	24	384	–180,1	–413,2	–23,9								
6	0	2396,1	355,6	–207,3	1833,8	–10,6	–1321,6	7373	–158	144,5	–284,1	93,2	–317
	12	–545,6	–237,8	–271,6	–132,6	91,9	–17,1	–94	374	460,8	–193,5	–409,3	92,1
	24	–235,6	88,2	96,8	–121,1								
7	0	–651,8	–2127,6	655,4	7212,4	–323,1	–477,9	1955,8	303,2	–420,9	1250,9	–27,1	–552,6
	12	59,2	–45,7	71,2	–178,6	107,9	–89,2	88,7	–236,6	–200,7	253,5	–280,3	265,6
	24	–188	–223,3	–172,2	338,6								
8	0	–353,3	–490,6	53,3	1929,3	–386	7211,5	669,8	–267	–1117,5	–390,9	–264,1	583,9
	12	1183,9	1311,3	–79,4	–333,2	24,9	159,3	–210,4	116,9	–194,3	52	–82	–105,1
	24	239,1	134,9	–69	–116,3								
9	0	14,3	348,3	868,3	–63,9	321,9	–2221,2	53,3	215,6	–7890,4	–3,5	233,1	163,6
	12	499,4	131	–0,4	–131,6	13,7	–230,4	221,1	–61,7	–189,2	–252	117,3	426,7
	24	89,1	–323,3	209,5	16,4								
10	0	–221	–173,4	59,4	–119,5	116,1	–19,6	45,3	96,6	152	–19	195,5	141,4
	12	2,7	–32,6	125,8	–104,3	–8192	12,6	–132,6	264,3	–136,2	–180,6	76,3	–98,8
	24	141,3	–164,8	–325,5	128,1								
11	0	–3222,5	377,5	4063,5	834,4	1458,5	2100,7	378,5	73,9	–311,6	–200,6	–139,7	–125,7
	12	581,7	259,8	151,5	82	–30,6	–35,4	–247,7	402	6035,4	–166,7	–100,8	–75,2
	24	284,6	–53,3	–318,5	232,8								
12	0	1721,7	–4502,2	–852,4	1847,9	46,2	–228,1	1950,1	–221	–671,4	1833,4	–4362,2	1558,5
	12	–1244,3	20,6	34,9	294,9	144	202	–90,3	–813,7	–142,5	117,9	260,1	–37,7
	24	1205,4	–476	–1403,6	36,6								
13	0	–7301,4	1080,6	–901,3	97,1	–380,6	996,1	–699,1	1045,8	–514,7	920,1	283	–1027,7
	12	–166,3	371,3	190,4	89,6	–808,9	–141,1	118,3	169,7	–481,7	–209,4	92,2	27,4
	24	374,3	–135,2	477	15,8								
14	0	–147,3	4031,9	1435,8	–46,5	–5117,2	81	885,1	–1175,5	–1107,2	–887,5	–763,1	504,5
	12	–304,1	419,4	–131,7	–258,4	44,3	–169,2	–284,9	–656,5	–168,8	249,9	92,2	434,8
	24	399,3	2742,8	–2269,8	–422,2								
15	0	55	2056,7	41,7	644,4	–94,3	–222,1	403,9	173,3	78,1	285,3	213	–356,7
	12	–415,1	–64,4	8192	27,7	129,3	–171,8	124,2	–475,3	–119,8	553,9	168,1	405,4
	24	231,8	–115,4	357,8	–11,8								

Окончание таблицы А.25

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
16	0	–4373,6	–4707,6	–234,2	–154,3	–32,1	544,7	–566,6	–793,3	514,2	542,4	–440,2	–49,7
	12	595,3	–94	102,5	178,9	–85,8	–285,5	–218,5	143,9	31,4	98,4	180,4	–214,9
	24	3,5	262,8	–484,5	–88								
17	0	290	–8192	–278,4	297,9	146,5	–0,5	–588,2	–248,6	–199,9	–520,7	497,6	475,9
	12	118,3	–89	107,1	–91,6	–86,8	91,8	–77,5	233,9	–58,3	394,4	–145,9	–215,6
	24	265,4	–149,4	–72,8	135,5								
18	0	–110,7	–1113	–290,1	4806,1	1204,9	–730,4	–5190	400,1	–593	–1403,1	–340,4	677,9
	12	29,2	–122,1	20,3	320,7	79,6	181,3	373,8	20	–197,5	–354,6	–208,2	129,6
	24	–98,4	103,4	252,8	–91,8								
19	0	378,7	137	–1140	–592,3	–251,4	–532,3	1373,9	238,4	1098,1	–27,9	403,4	32,6
	12	8140,6	–375,4	–137,7	–83,2	–147,7	168,8	57	360,3	179,8	–195,7	–220,1	–226,8
	24	170,5	230,1	–163	–56,9								
20	0	–987,2	30,1	866,7	526,6	8093,1	70,7	–55,7	–699,4	–1164,6	–316,1	1375,7	189
	12	22,1	–498,7	347,6	–16,3	22,9	–76,5	0,5	97,8	–269,5	74	–329,3	416,5
	24	51,1	–208,5	–687,8	–355,3								
21	0	630,7	–1598,8	–450,4	–489,3	168	1335,9	–243,7	841	–433,4	–23,2	561,6	413,4
	12	17,6	–575,4	49,2	–282,7	–210	–28,5	12,3	228,6	70,7	7983,5	43,6	–92,4
	24	–191,5	124,7	–265,7	–638,3								
22	0	–890,3	548,5	–274,7	1330,1	–126	104,6	497,7	85,7	217,6	21,2	8192	370,7
	12	–722,8	0,7	–477,6	20,6	1,1	–385,2	46,2	–386,8	–204,8	–104,7	–154,2	–134,7
	24	131,3	–264,4	–509,6	–312,1								
23	0	–3520,7	1720,9	–920,1	–958,3	633,2	–725,3	–1123,3	–240,1	2405	–1595,2	–2771,2	–464
	12	–995,5	–2664,6	–962	–1557,6	529,4	176,4	–432,5	26,9	–311,8	68,5	–222,6	162,7
	24	4052	–508,5	–260,2	–5318,8								
24	0	–1135,5	–686,1	–738,4	27	225,8	–2268,9	219,7	608,7	276,7	586,2	–316,9	43,1
	12	–28,4	8192	415,8	–84,4	174,8	–61,7	–40,6	–286,4	148,3	101,6	–161	–82,4
	24	110,8	268,8	–79,3	121,3								
25	0	–311,8	–137,4	–670,6	–884,3	113,8	–261,8	–146,9	–241,6	–489,4	155,7	–64	8192
	12	–116,1	–97	–146,1	357,8	7,3	254,2	220,7	–1634,1	353,3	–134,5	–26,2	–222,5
	24	–382,3	4,3	112,5	–157,4								
26	0	1862,8	–495,5	6418,9	–149,9	340,6	–2238,5	196,6	–164,2	–1437,8	435,9	710,5	–514,6
	12	169,9	–348,1	–74	106	138,6	–177,7	21,5	283,2	–545,6	89,3	–63,1	431
	24	764,4	–249,3	–414,9	–587,9								
27	0	–41,7	270,5	–359,4	–10,9	–27,3	–261	142,7	77,9	14,5	19,9	345,9	52
	12	–78,3 –	81	122,3	102,6	3,7	8192	–30,7	–356,9	–97,6	73,5	–216,4	–302,7
	24	153,8	42,4	2,2	98								
28	0	–583,8	365	–4062,4	1,6	1181	833,6	650,5	–1345,9	–5367,4	231,7	93,2	–2805,4
	12	366,5	–317,6	–158,5	116,1	–199,2	–92,6	223,6	817	174,4	–114,9	17,9	–1200,2
	24	148,8	–180,4	–170	–80,1								
29	0	–1043	–79,6	–359,2	–298,3	–661,8	–1090,7	128,3	–837,1	–1261,5	–8192	216,3	141,4
	12	–51,2	582,6	205	182,4	–69,7	335,1	110,7	118,4	–108	–75,7	274,5	–77,5
	24	18,7	–89,6	179,5	–99,4								
30	0	709,8	–1626,7	–165,6	1293,3	19,2	–1666,5	133,5	138,4	–806,8	–440,8	518,5	696,7
	12	75	129,9	–23,3	–8192	38,3	–374,9	–127,1	–946,8	129,2	–201	–42,1	251,4
	24	293	430,6	–511,5	637								
31	0	–710,9	4344,6	–134,9	5527,1	–120,1	713,5	394,8	–680,5	404	66,3	336,9	478
	12	–302,1	–30,3	27,7	318	–498,2	23	85,9	–151,9	–74,7	–77,4	–225,1	242,8
	24	780,9	–393	656,9	715,2								

Таблица А.26 — Сборник кодов 1 чередования *VQ* для масштабируемого кодера (тип окна: длинное)

Имя файла	Содержание	Режим	Число элементов	Число векторов
scmdct 1	*MDCT*	Расширенный длинный	28	32

VN	*EN*	0	1	2	3	4	5	6	7	8	9	10	11
0	0	–561,2	–713,8	–16,7	1735,2	215	–8119,3	–300	–116,6	–321,4	830,9	355,2	–58,3
	12	280,3	464,6	167,6	47,9	–11,4	96,1	137,8	–252,5	137,8	–303,4	–98,5	14,5
	24	–225,3	160,7	–12	–58,6								
1	0	–775	315,3	–620,2	–992,4	–12,8	–1335,7	143,6	–1198,4	–1288,1	8015,5	204,3	158,9
	12	–591,5	731,1 –	144,8	179,3	93,2	145,5	38,9	257,6	101,6	–67,5	50,9	11,4
	24	99,4	210,4	–40	–58,5								
2	0	–3915	–782,6	–1029,4	4961	534,9	–181,7	606,6	369,6	–1139,2	–1963,3	–426,4	–941,3
	12	259 –	276,8	121,6	84,6	202	449,1	144,5	205,4	–366,1	–335,5	20,2	82,8
	24	357,4	119,7	–107,2	–358,5								
3	0	–1610,2	–1283,8	–1449,3	437,8	–267,9	5562,3	–2744,8	–305,7	278	2618,1	326,5	–169,6
	12	–2019,1	–1045,5	321,3	281,7	–564,1	159,5	358,6	–793,5	–231,2	–47,2	172,5	195,9
	24	–473,5	609,3	–268,5	–18,1								
4	0	203,1	1157,9	–332	8192	246,5	–60,8	–900,1	285,3	878,8	–932,6	72,4	223,6
	12	206,3 –	117,7	195,7	–205,6	50,3	–217,2	–262,4	197,6	33,2	–89,9	313,1	270,3
	24	78,6	208,8	135,8	–92								
5	0	–182,8	3165,2	500,3	–3345,9	631,5	–220,1	–219,9	4608,3	–3178,4	–399,9	–312,2	378,8
	12	–559,3 –	196,1	354,2	24,5	397,2	24,5	–122,3	–579,3	–29,6	–351,2	29	306
	24	518,6	329,2	259	–361,8								
6	0	3560,4	–556,8	901,4	–1009,7	–3115,6	–1781,1	20,4	–448,8	–2473,8	–1018,8	–3821,8	–655
	12	–247,1 –	231,3 –	–350,5	–76,9	–2524,5	–11,8	601,6	–42,9	503,4	11,8	–98,8	563,1
	24	855,1	613,3	877,7	–836,3								
7	0	99,3	803,3	–350,6	–103,9	556,6	–2584,9	501,2	24,7	7101,9	42,6	83,8	–947,8
	12	630,4 –	491,2 –	–69	3,7	–182,6	–165,1	372,6	64,4	120,3	–103,6	51,5	3,1
	24	483,5	39,7	254,2	315,2								
8	0	–684,1	–186,1	1408	467,3	516,3	647,9	–1823,9	–209,9	–1533,2	214,2	–735,8	–223,6
	12	7325,9 –	349,4 –	210,1	66,3	160,8	63,6	–153,7	–434,8	–205,1	214	479,9	138,6
	24	90,4	36,2	260,9	–82								
9	0	326	510,6	664,5	933,7	–171,2	614,6	264,1	528,9	97,5	–301,1	157	6638,7
	12	–70,2	209,6 –	74,7	–734,7	91,6	–496,3	–258,8	4242	–162,9	–179,7	101,5	346,6
	24	250,4	181,2	165	–184								
10	0	536,5	1332,9	164	1325	–231,2	–62	–8136,5	13,3	–8,8	–183,4	–21,5	–156,9
	12	–748,1 –	78,5 –	–84	–435,3	12,9	–236	–118,6	182,9	–26,3	52,5	–93,9	402,3
	24	43,3	265,1	249,3	96,7								
11	0	2532,6	–1240,8	4875,4	–395,9	744,2	4754,6	–212,9	538,6	38,7	–791,5	311,9	–400
	12	–141,9	–34,1 –	122,9 –	20,4	310,5	289	–264	34,3	–295,5	13,5	–98,2	85,9
	24	155,5	421,3	420,2	–1113,5								
12	0	–854,1	–4737,3	–4964,8	–42,6	36	271	402,8	216	–361,4	–1031,3	896,6	303,3
	12	–882,9	146,9	407,8	–57,6	140,3	–62,1	–208,2	–299	62,6	–104	–54	–22
	24	125	541	167,3	62,6								
13	0	–37,3	–320,7	–479,8	–1002	6982,4	428,5	429,6	1185,6	270,6	–38,5	–2430	–216,1
	12	–258,1 –	1022,9	7,5	31,9	145,5	–95,5	57,1	–248,5	–362,5	386,9	–647,1	–296,8
	24	215,8	657,7	158	–170,8								
14	0	–936,6	–6447,3	1963,8	–43,3	621,1	–589,6	790,4	34,8	192	–681,5	–75,1	–1008,5
	12	–592,1 –	372,8	–407,7 –	–232,1	84,5	48	735,8	–190,7	–189	–36,1	–20,3	69
	24	379,5	24,9	290,1	–187,7								
15	0	–903	–538,2	–466,7	82,9	323,1	–2047,7	296	732,5	–316,9	625,5	–87,2	19,2
	12	117 –	–8038,9	371,8 –	126,6	–22,1	–100,3	–177,6	–126,4	260,5	0,5	0,6	34,8
	24	285,4	103,3	347,5	389,4								

Окончание таблицы А.26

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
16	0	1747,9	2234,3	–277,5	–109,8	–171,4	–878,5	–34,1	–6668,5	–277,8	–71,8	–929,3	77,1
	12	–55,3	–834,8	1929,4	4,5	18,3	59,5	–95,9	305,9	158,7	59,7	248,2	–149,5
	24	128	–65,1	–1,6	213,8								
17	0	–739,1	–212,2	400,8	70,1	–189,8	683,7	40,3	–532	3	–695	–1249,8	–1958,4
	12	–1494	142,9	–97,3	–2738,4	–112,9	159	–7679,3	260,8	–267,6	–203,5	147,6	1002,2
	24	–304,2	317,3	792,7	427,9								
18	0	3208,6	–221,6	–1632,8	444,5	–382,3	–171,5	130,6	5996,9	1416,9	299,8	–641,1	–1334,1
	12	–135	–295,7	999,2	–588,2	–109,4	0,8	45,1	241,5	716	75,3	181,6	–63,4
	24	608	302,1	–2,8	186,6								
19	0	–4161,7	–205,9	–505	–5407	132,5	913,8	441,8	–43,6	281,5	–884	369	306,5
	12	236,6	16,9	321,3	–136	–285,3	–57	117,7	172,7	–341,1	–286,4	518,2	–36,8
	24	–165,2	38,2	–128	2036,2								
20	0	4768,2	651,6	–1323,6	–21,5	3555,3	2592,6	672,9	–1588,3	776,8	1531,6	170,8	–1284,5
	12	–351,6	29,5	637,3	–571,3	401,2	–84,3	–197,4	–192,6	304,2	–91,6	–144,7	–112,9
	24	135,2	–150,4	49,5	492,3								
21	0	566,2	193,6	–8192	–1008,5	311	–703,7	143,8	–322,4	–345,2	493,5	–122,1	–583,3
	12	–72,1	189,9	–399,4	–105	209	–118,6	97,1	220,5	479,3	–14,2	198,2	–104,4
	24	202,3	83,1	12,5	226,3								
22	0	–529,2	317,2	–3662,3	–2057,5	–4863,5	1738	495,1	783,9	9,1	105,2	–514,3	84,7
	12	258,6	538,8	134	–14,6	359,9	–259,7	–39,4	–164,9	525,2	19,7	–3801,9	78,5
	24	–237,3	–526,4	466,5	20,8								
23	0	1203,4	–1301,8	348,1	–1694,5	–83,5	–667,8	–69,4	–248,8	–806,8	–32,7	7341	–343,4
	12	80	–278,2	174,6	–342,1	–312,2	498,4	–382,4	–16,3	252,4	246,8	1,9	111,2
	24	321,4	77,7	248,6	–645,2								
24	0	–2744	–226,8	5684,9	–3196,3	201	–1143,2	–68,9	48	625,6	–12,2	–1007,2	–911,8
	12	–619,4	413,5	–113,2	45,1	334,9	15,2	–7,5	363,2	–1040	–165,1	230,6	–309,3
	24	510,4	–149,2	–171,9	454,3								
25	0	482,8	2739,8	–178,6	648,8	19,1	–422	129,6	155,7	–32,2	218,2	255,1	–766,4
	12	–511,5	–441,8	–7895,9	–95,4	88,4	–507,1	–117,5	–160,8	110,5	478,6	–43,8	375,2
	24	–422,4	–239,4	273	43,8								
26	0	459,8	2520,3	–3700,9	–219	604,4	–398,3	–3011,9	–165,8	611	–3859,3	549,1	–468,7
	12	–54,6	378,4	259,5	–6,5	291	215,1	225	–25,2	51,5	94,8	–151,8	–382,5
	24	–18,1	–1051,7	–4932,4	174,7								
27	0	–7985,6	–471,4	536,2	–173,5	565,7	67,8	802,8	–1040,1	832,5	–106,7	–361,2	453,6
	12	40,1	–622,4	–140,1	–63,5	383	–80,7	–149,3	–162,5	–72,1	84,6	–199,5	84,3
	24	–575,8	–136	87,8	–713,6								
28	0	350,8	–1145,1	–210,1	1038,6	–18,8	–1098	210,8	210,4	–802,5	–696,9	212,9	381,2
	12	–524,5	73,2	–33,8	8192	130,9	–435,1	–880,7	–549,6	133	–97,3	76,1	177,6
	24	–12	261	–168,5	177,8								
29	0	–3820,4	4929	586,4	357,3	208,7	–1327,9	525,2	–426,1	–290	–84,7	–283,1	–992,5
	12	–777	727,9	522,5	–130,6	–31,9	58,6	655,4	–233,7	–2,1	1969,1	–24,5	122,3
	24	–1434,3	–529,6	–127,7	399,8								
30	0	–253,6	580,2	3009,4	2563,5	–2683,8	34,2	–168,6	349	322,5	–399,6	626,4	–743,8
	12	–938,8	151,5	8,4	–310,6	199,1	116,5	181,6	–192,6	–90	93,5	178,1	–6795,7
	24	77,2	689,1	269,7	216,1								
31	0	–74,9	2728,3	143	1142,2	181,9	3492,9	5593,3	631	–296,2	–136,5	464,6	–234,4
	12	639,1	–401,2	206,2	–0,4	244,2	–132,8	323,2	–611,7	–270,7	–438	756,8	194,9
	24	30,6	–370,5	–55,2	–210,9								

Таблица А.27 — Сборник кодов 0 чередования *VQ* для масштабируемого кодера (тип окна: короткое)

Имя файла	Содержание	Режим	Число элементов	Число векторов
scmdct 2	*MDCT*	расширенный короткий	24	32

VN	*EN*	0	1	2	3	4	5	6	7	8	9	10	11
0	0	613	−1918,4	−968,9	892,6	419	79,3	−77,8	−240,8	162,1	493,8	−1223,7	720
	12	181,3	−1005,3	−10,1	1160,7	464,2	−1110,3	138,9	−7755,7	−603,2	−621	779,9	−144,7
1	0	199,2	449,8	−147,5	−650,1	474,8	−258,1	635,7	−975,8	768,7	−1061,6	606,4	−437,7
	12	−535,1	214,6	−934,9	−8192	−104,9	84	−393,5	−261,4	26,9	1021,8	−110,7	−818
2	0	−2191,5	516,5	−512,3	−7841,8	−1863,1	930,8	296,5	−854,1	321,5	−104,4	745,6	1428,2
	12	−71,9	−406,6	549,5	1065,5	1224,3	−899,9	−418,3	4	−588,7	125,3	−942,3	−1317,2
3	0	−5741,3	1175,1	−3780,8	−1979,1	3997,5	2363,4	1948,9	−1075,8	270,2	1113,4	−1514,8	36,5
	12	−215,7	1244,3	1454,6	−629,9	526,6	−31,9	960,9	−292,7	1256,4	1770,9	−1638,3	−1141,1
4	0	1869,4	1185,6	515,5	−381	−1273	−942,5	−988,9	−595,9	−954	−663,4	−547,1	731,1
	12	−360,1	−93,3	7036,1	−1416	1070,4	−2318,2	314,9	440,7	72,7	−60,8	−1731,8	1086,1
5	0	−535,5	−152,7	−523,3	1189,4	100,9	−560,4	−269,2	−1528,7	−482	−672,3	204,8	−1138,3
	12	−85,9	−914,3	−266,2	1683,5	−411,3	−522,6	427,8	436,3	166,2	8119	−1297,1	173,8
6	0	1996,2	1353,1	2318,5	4260	−3061,5	2192,3	702	−2385,7	1653,8	1494,3	−1128,3	807,4
	12	−1586,9	1139,7	−1618,1	217,8	−808,5	827	−756,3	1528	493,9	−1344,4	−4055,4	−118,7
7	0	164	620,7	−359,6	201,5	−932,1	8192	−101,1	−1723,8	−94,3	−950,4	198	903,7
	12	−602,2	−198,7	−514,4	1396,4	−756,6	−429	370,1	−249,4	−59,5	221,6	−1645,2	1282,2
8	0	−959,2	−2799,8	2031,5	1081,8	3374,3	3582,4	603,2	777,2	−1314	2646,5	340	−1727,3
	12	1645,3	173,9	−558,3	−807,2	−645,4	−2908,5	−525,6	2070	269,2	−86,7	3020,5	−6239,7
9	0	−474	−489,6	511,5	−707,2	247,4	1316,2	−45,1	−823,5	−8016,4	−1546,4	236,2	−143,7
	12	143,7	−385,4	−375,3	−1493,8	716,5	334,3	−156,9	658,5	−48,5	−1005,8	−2546,6	320,7
10	0	896,7	−1018,5	−8192	−1013,5	−380,1	657,3	94,6	−263,9	421,1	−783,9	−149	−427,7
	12	555,6	264,4	393,1	−322,4	255,5	−175	−628,9	673,4	1279,4	−641	−1016,1	−1884,3
11	0	−513,2	−335,3	−37,1	−805,8	1171,8	262,9	−1791,4	−8099,1	815,1	−783,7	296,2	−601,5
	12	−192	86,9	−3,9	974	2015,1	161,7	−1157	−490,5	−1972,9	−985,8	381,7	525,7
12	0	−2340,4	−5661,8	4040	−2018	620,7	−1192,3	82,4	−2415	−531,9	−408,1	1410,8	−124,5
	12	1669,3	1686	351,5	−150,9	−46,5	775,2	−550	−189,9	1333,1	33,9	−1745,5	−417,2
13	0	65	22,3	770,3	−610	1222,8	625,5	923,8	611,1	778,8	−72,4	−793,9	−8138,9
	12	−652,8	−443,5	−326,3	372	−835,5	−932,2	−438	636,3	552,6	−645,1	1783,7	−1578,1
14	0	−1400	−98,5	3057	3613,4	2282,3	2572,4	3484,4	−3571,9	481,7	−3392,1	1297	1167,4
	12	161	−1064,2	−313,3	1599,9	−6,9	470,8	653,7	136,3	2168,9	1038,2	856,7	363
15	0	−785,2	−384,5	−166,7	305,1	−465,8	−565,9	−1368,1	−287,8	52,2	207,1	2756,1	487,7
	12	422,8	−1724,3	−646,7	219,3	52,2	−7309,5	−398,8	871,1	614,9	−965,9	−94,9	−1695,4
16	0	3931	−7212,4	−30,7	−853,7	−141,4	−138,3	−196	295,3	−78,4	364	388,5	−1395,2
	12	−1237	1038,9	703,5	87,8	546,8	−9,3	−696,1	−777,8	−78,2	−475,6	224	−268,8
17	0	2059	2976,1	2990,4	−3869,7	−1417	1535,6	−2343,5	−983,8	−969,7	1746,4	149,8	2876,3
	12	−1391,5	−262,8	−1512,9	980,8	−1948,6	−752,1	−564,2	1323,3	1,1	−518,9	2387,7	617,5
18	0	263,7	78,7	663,8	−310,6	−2441,4	−1218,3	−481,4	1743,2	514,8	−483,3	1094,8	655,3
	12	269,5	146,2	−913	−744,6	7872,2	2233,3	401,5	948,2	−1182,8	1871,3	459,9	−672,7
19	0	23,6	461,8	754,2	−411,9	451,3	245,3	−8192	−294,2	−192,5	−819,3	−791,4	−1601,9
	12	−430,8	−131,7	573,7	1347,4	−345,2	46,1	1877,7	−798,3	−42,5	−84,7	96	1252,6
20	0	381,9	3174,6	−608,4	−2156,8	−3019,3	781	4767,2	−1575,8	920,3	−660,6	−779,4	−2567,7
	12	1139,5	647,3	98,2	2453,2	−1062,3	−608,9	−1438,9	1269,2	−1554,3	−979,2	−244	1879,8
21	0	−502,3	−843,6	631,9	843,7	708,5	−992	−471,3	−1323,4	788,6	−952,4	833,7	1104,1
	12	555,9	27,4	1181,2	−183,5	291	803,2	−106,9	−517	−504,5	108,5	8192	2984,5
22	0	433,6	246	−1589,2	−817,6	−708,4	−1522,4	−1661,6	1227,3	−1094,2	−491,1	237	−887,9
	12	1532,3	1569,8	−299,8	−219,7	−517,8	407,6	414,2	122,3	7957,8	−501,4	−406	202
23	0	−4188,3	247,2	437,9	1234,2	−3749,7	−5089,5	−159,7	−401	1524,7	−1408	−3081,6	−60,8
	12	1318,4	−63	−1692,2	−71	−575,4	−1530,1	−6,8	838,7	−913,4	−431,7	1032,4	−2078,2
24	0	656,5	1578,5	−2239,9	764,9	1307,3	−4063	2133,6	−4375,1	−2720,8	−82,3	−4,4	1379,8
	12	1452,4	−794,8	−1084,7	−720,3	−1685,5	−105,5	1770,7	1205,6	−635,2	799,4	1327,4	−776,7
25	0	406	353,4	1207,9	100,3	−554,4	−363,5	812,3	−219,5	368,2	−818	−1283,4	115,5
	12	949,7	−348,7	864,5	−336,3	326,9	−779,5	7874,7	−142,3	1076,4	−848	−246,6	−3475,1
26	0	−8192	−3,9	−103,4	−300,1	−451,7	−318,6	197,6	139,9	−167,5	−386,7	1758,9	755,7
	12	75,4	−465,2	329,6	557,2	−219	526,8	896,1	435,7	−509,5	65,4	−407,4	−1614,7
27	0	246,7	−2,2	−202	763,1	−8192	985,9	776,4	51,7	74,6	180,7	−360	463,2
	12	226	−83,2	1488,2	−1249,9	−1186,7	167,1	−211,1	−562,4	75,1	1007,6	400,6	−2326,2
28	0	336	441,6	215,6	104,9	−216,9	860,4	−1256,2	505,1	596,4	−7828	542,6	−218,9
	12	−1139,2	175,8	−1623	383,9	66,5	−230,2	−732,1	505,2	417,1	587	210,5	−1437
29	0	572,1	−128	823	−312,2	59,7	100,7	−720,4	695,3	−790,7	−1235,8	891,5	1663,5
	12	347,7	7774,4	−121,7	−516,7	−906,6	929,5	−315,7	−397,6	−1038,2	−1197,8	−92,7	−929
30	0	−1050,2	−1276,6	−428,5	−141,9	88,4	−1400,9	34,3	763,6	−203,7	−854,5	−713,6	13,7
	12	−7944,3	265,6	−520,4	−532,2	1207,4	−1606,8	480,2	−31,7	−2769,1	−386,9	−52,8	−45,7
31	0	−4196,5	−786,1	−474,8	2859,1	−3151,7	3652,5	−2296	1179,6	−1717,4	1482,6	−2578	−401,3
	12	1222,3	522,7	124,4	2285,2	−180	−849,8	−180,3	1514,7	−1489,2	−946	1162,9	1043,9

Таблица А.28 — Сборник кодов 1 чередования *VQ* для масштабируемого кодера (тип окна: короткое)

Имя файла	Содержание	Режим	Число элементов	Число векторов
scmdct 3	*MDCT*	расширенный короткий	28	32

VN	*EN*	0	1	2	3	4	5	6	7	8	9	10	11
0	0	–1587,4	–8151,2	466,2	476,5	1310,7	–36,4	–835,2	–385,1	–329,8	–983,2	–724,4	–861,8
	12	–354,7	–992,7	197,9	554,3	–77,3	207,1	–977,7	–660	–331,6	602,2	706,1	–1584,4
1	0	–495	61,2	–263,6	1248,6	–1573,1	–713,3	2527,9	–6633,3	–696,5	247,3	1391,3	–289,7
	12	–324,9	489	–1072	–459,6	–211,1	–736	385,2	–1296	–37,3	1785,9	–258,4	–459
2	0	433	–524,1	264,9	–41,6	7135,3	1978,2	105,5	–52,2	–396,5	–306,8	–1282,4	412,9
	12	–21,3	1616,7	–46,3	–844,6	–1902,3	1368,5	–133,6	211,3	1114,3	457,7	2710,2	–1426,3
3	0	1621,1	–382,5	941,6	1426,9	–4604,8	2534,3	–1221,5	267,8	–4996,4	–1924,1	763,4	1417,4
	12	254,9	–620,3	325,2	358,6	669,7	–799,6	529,5	–319,1	–364	–423,1	–2416,2	–1041
4	0	–357,9	860,6	339,1	143,7	–188,7	1506,9	–903,6	–898,6	6330,3	–3820,3	829,1	–11,5
	12	220,8	–477,8	1134,2	253	383	–1115,3	–1149,9	848,9	–1113,5	107,8	–868,4	295
5	0	–8192	–591,2	–299,6	524,8	1471,7	73,1	–869,7	–211,1	–162,2	139,5	–1121,5	–1009,6
	12	91,9	160	–270,5	67,1	–275,7	–861,7	–722,4	–488,8	–623,8	698,3	697,8	–264,1
6	0	–818,4	786	–608,1	–61,4	1509,7	7783,9	–37,6	2065,4	–638,7	198,9	35,8	–824,1
	12	791,2	739	605,3	–1152,7	–372,9	424,6	1346,8	–18,8	–625,8	604,6	–384,4	284,6
7	0	3029,7	–1750,6	4506,6	–870	513,2	2301,4	927,1	2530,7	228,1	–2474,1	1344,5	–78,5
	12	153,2	–1658,2	–537,2	–280,5	642,4	2369,1	500,4	–584,8	–170,5	–303,1	1704	136,9
8	0	–1569,8	–1288,6	201,5	–1157,1	–466,5	797,8	371,5	–1819,5	537,5	131,8	–3898,4	5382,6
	12	1775,5	–542,9	–1048,7	–370,1	–635,7	–952,7	–997,7	–41,5	726	11,2	–81,3	13,6
9	0	–2450	1788,6	–1971,3	614,4	–787,8	–345,9	–1830,9	4736,7	154,9	76,6	1477,9	576,4
	12	1549,4	–873,6	–1628,5	–5,8	–1817,4	1087,2	–119,7	–4009,5	–362,6	–2510	–3032	–1122,4
10	0	1489,3	5731,5	4239,4	396,9	1859,3	–1909,5	302	–790,5	1461,5	–1084	–510	–1498,7
	12	788,4	–259	965	–536,7	428,1	622,7	–227,5	–1693,9	–98,6	217,7	–852,7	–1826,4
11	0	727,1	–471,3	–3938	4596,7	155,4	2508,6	–1160,6	–959,7	1337,4	–1124,4	782	747,3
	12	169,4	–577,7	2942,3	–638,7	1307,7	1236,2	–449,5	–649,7	294,4	1321,4	–699,9	364,8
12	0	–3489,7	–557,6	1321	–1750,9	–324,9	–704,9	5375,6	2709,4	343,7	–3100,4	–1881,5	172,9
	12	–1917	847,7	679,8	–268,1	974,4	27,3	–60,4	–1128,4	12,1	–2983,5	–354,1	945
13	0	254,3	1387,7	1209,8	598,9	277,1	–934,4	–7093,7	1923,9	1015,7	–357,3	–390,3	1251,3
	12	–946,7	480	208,4	1134,5	750	–716,6	–726,8	794,7	–336	75,3	807	–2576,3
14	0	1188,4	–248,5	718	–1366,1	–3130,4	–1170,6	–101,4	1117,5	1004,6	7,1	–1020,9	63,7
	12	–483,1	–1065	878,5	–1709,5	–6818,4	1088,2	363,8	–686,1	–1432,5	–405,5	343,5	–391,3
15	0	–945,3	2896,2	362,4	–2761,9	2342,6	805,7	–619,8	490,8	–2910	–798,1	–59,8	409,3
	12	–1168,1	149,5	550,7	4936,1	–563,9	–328,6	–320	616	1443,7	381,4	–1751,2	–738,3
16	0	1408,8	–324,5	–250,1	–42,1	–718,3	654,3	608	324,9	–863,8	596,2	–2048,1	–1469,7
	12	1793,5	–211,2	–15,2	394,4	992,8	–6976,7	1115,8	–1024,9	823,3	216,9	–1298,7	2454
17	0	–2586,3	–2557,9	–1035,1	–2009,8	–1809,3	–5664,7	–661,2	–987,2	–261,8	–731,4	806,3	972,9
	12	–2113,7	1892	1078,4	1751,2	–443,4	1183,3	1097,3	570,2	–57,2	–956	–1259,4	53
18	0	1515,5	941,4	–381,6	1674,6	–81,4	121,6	2131,6	–556,5	1081,9	1872,4	–816,3	–1076,9
	12	496,7	7226,8	–416,5	–382,9	508,1	1137,3	–1334,6	790,6	1024,6	613,2	–441,6	–1065,9
19	0	–406,3	380,5	681	–1482,5	–2079,6	–1901,1	–3583,9	–3315,9	–447,2	–63,2	–691	–1701,2
	12	578,3	–2632,7	663,9	30,6	1611,2	3718,7	867,1	1683,5	50	–1270,9	–1571,8	–1539,7
20	0	321,7	–618,9	1318,2	–2858,6	–95,2	109,2	1145,3	1812	–278,3	2607,3	–359,3	1530,9
	12	–1664,7	678	2133,4	–1815,8	1561,3	56,1	–134,9	–12,2	–342,6	6635,7	2642,9	2408,3
21	0	757,9	–357,8	798,2	492,1	–1311,4	742,9	4154,7	105,9	571,6	2169,7	903,3	334,5
	12	–3191,2	–1928,7	–1051,4	1526,9	245,2	–1750,8	–1104,9	1049,2	5111,9	–1193,6	–521,4	–734
22	0	–1838,4	1235,6	1095,1	2725,7	2497,5	–1924,5	1151,5	603,4	–1416,1	2696,4	–268,5	–1621,3
	12	399	–529,9	2481,5	–123,4	2178,4	857,2	–991,2	3837	–385,4	–212,5	–3028,1	4468,3
23	0	1330,4	2003,6	–5625,7	–1541,8	709,1	–203,4	722,3	468,9	1065,2	1010,1	1321,9	653,3
	12	–1503,7	–890,1	126,5	595,1	1292	–120,9	867,1	309	–1041,5	–1233,9	4458,3	1747,7
24	0	3786,4	166,8	659,1	–7165,8	736,4	760,8	–157,3	–832,6	–726,2	–558,5	–1335,9	–1448,4
	12	1227,1	310,8	–1051,7	–528	–791,4	629,3	33,1	–1012,9	–632,4	–129,1	–499,5	1025,3
25	0	–1451,7	246,9	244,9	149,8	5,9	156,5	–51,5	–109,4	478,2	306,5	7042,5	253,4
	12	122,7	110,6	–2,7	–947,8	–1074,1	318,6	81,2	44,4	–1188,6	980,2	–1001,7	1166,1
26	0	–1751,7	1177,9	–1175,2	–2322,5	–4825,7	1590,5	258	229,3	–92	167,1	71,5	–2142,2
	12	–2665,3	1556,1	–2104,4	374,6	1627,4	–847,9	1737,1	110,3	594,9	1048,1	–177,5	–207,9
27	0	121	–145,1	–561,2	420,7	–152,6	–871,5	–1213,1	–294,2	–868,6	807,8	–220,4	981,4
	12	89,8	–313,7	–7218,6	424,1	–315,4	–975,3	–199,4	–893,3	283,3	–108,3	775,2	785,3
28	0	2464,9	–3039,7	–673,7	188,1	1483,1	1089,5	–446,4	283	648,4	2184,9	–1143,1	576,4
	12	1052,9	281	–2367,4	1505,7	–702,3	628,9	4176,2	1901,7	–1451,3	–460	–4274,9	174,8
29	0	9,6	362,4	362,1	802,5	497,8	–914,2	–1697,2	–45,5	–1122,9	648,9	720,2	916,9
	12	–371,9	588,3	162,3	–6838,7	–849,4	–560,4	316,3	180,4	–229,1	–2966,6	–638,9	437,5
30	0	3819,9	685,6	–662,9	1849,7	1157,1	–2943,1	2673	–153,4	–1106	–1384,2	869,4	1226,8
	12	–513,6	–352	1064,3	4032,2	375,9	–824,9	–1518,7	1262	–2905,1	160	–315,9	–3058
31	0	–125,4	–618,8	379,6	–840,3	–975,8	–1430,1	1370,9	583	–452,9	–73	–374,4	–1129
	12	7405	175,4	123,6	–458,8	1692,4	–318,4	–1535,9	521,6	–277,1	–690,6	899,4	3196,9

Таблица А.29 — Сборник кодов масштаба *Bark*

Имя файла	Содержание	Режим	Число элементов	Число векторов
*Fcd*I	огибающий *bark*	Базовый расширенный	6	64

VN	0	1	2	3	4	5
0	2,59553	1,59708	–0,21402	–0,17514	–0,09628	–0,26982
1	–0,31846	–0,28922	–0,32107	1,15714	1,03967	–0,03709
2	2,80202	–0,48079	2,15678	–0,37126	–0,36071	–0,16559
3	0,19375	–0,32843	–0,30811	0,27793	–0,2106	–0,07445
4	2,2074	–0,43946	–0,3363	1,40869	0,0118	0,04888
5	–0,50163	–0,3072	–0,34648	–0,36553	0,08881	0,66861
6	–0,5478	3,10285	2,923	–0,42525	–0,30431	–0,12124
7	0,36265	–0,59101	–0,40694	–0,37653	–0,02823	8
8	0,6108	–0,2944	0,84234	0,50441	0,15898	–0,19708
9	–0,26828	–0,42775	–0,55847	8	0,26906	0,09807
10	0,58163	0,69724	–0,38205	–0,32305	–0,04204	–0,18015
11	–0,1531	–0,52465	–0,5112	–0,46322	8	0,60858
12	8	–0,37894	–0,41379	–0,57177	–0,46141	–0,54807
13	–0,3409	–0,46601	–0,40139	2,77429	–0,30628	–0,29932
14	–0,00875	1,00366	0,35027	–0,14719	–0,12343	–0,01017
15	–0,42826	–0,25287	0,5889	0,00455	–0,18978	0,72595
16	–0,41718	–0,42836	0,44293	–0,26096	0,67149	–0,15131
17	–0,48094	0,40551	0,63479	0,10576	0,60523	0,00965
18	0,5875	0,17574	0,22913	–0,07386	–0,25141	–0,28609
19	1,54175	–0,38932	–0,26886	–0,37004	1,26474	–0,04734
20	–0,30997	–0,45256	2,71793	–0,29703	–0,34868	–0,27166
21	0,80891	–0,39427	–0,36544	1,05681	–0,26354	–0,07337
22	1,69802	0,38677	–0,19213	0,04237	–0,07601	–0,16281
23	0,82683	0,70298	1,21216	–0,23461	–0,02006	–0,24785
24	–0,4438	–0,46604	–0,44381	–0,40097	1,97582	0,05232
25	0,71883	0,61561	–0,1844	0,68805	0,00106	–0,01607
26	–0,38766	8	–0,06823	–0,22391	–0,52152	–0,42881
27	–0,4812	–0,4593	–0,4612	1,54307	–0,31462	0,33229
28	0,72907	–0,31141	–0,38468	–0,36775	–0,3587	0,7
29	0,88654	2,10076	–0,1964	–0,23175	–0,15172	–0,15909
30	–0,51999	–0,53691	–0,52863	–0,4753	0,54786	–0,32601
31	–0,4771	2,06467	1,01919	–0,28684	–0,14439	–0,05744
32	–0,57224	0,42109	0,5242	–0,2856	–0,1884	–0,15752
33	–0,24585	–0,44169	8	–0,43055	–0,40749	–0,43133
34	0,20221	–0,25994	–0,31787	–0,30064	0,6737	0,05431
35	0,32692	–0,4254	–0,45128	–0,45458	–0,16592	–0,3783
36	1,42145	–0,42477	–0,43237	–0,33452	–0,32365	–0,35176
37	–0,52165	–0,47377	0,32662	–0,34749	–0,24129	–0,18172
38	–0,46545	0,29674	–0,31146	0,41441	–0,12466	–0,07952
39	–0,26585	0,07241	–0,26056	–0,2953	–0,24943	–0,22879
40	0,34838	–0,3777	0,48323	–0,31334	–0,0997	–0,15367
41	–0,36403	–0,40947	1,43126	–0,48284	–0,33652	0,15962
42	–0,50425	–0,42948	–0,31036	0,28637	0,32368	0,02942
43	–0,44723	2,90793	–0,35379	–0,37063	–0,16951	–0,00506
44	–0,41686	–0,31531	0,3233	0,63998	–0,13565	–0,25397
45	–0,57783	–0,55613	–0,52567	–0,35549	–0,35857	0,13331
46	–0,5269	2,92107	–0,39287	2,50676	0,83325	–0,34838
47	–0,35037	1,58353	–0,14216	–0,14513	1,25833	0,02395
48	–0,54618	–0,51853	–0,5225	0,6915	–0,34809	–0,32408
49	0,91424	–0,39755	–0,23708	0,06298	0,32884	–0,27221
50	–0,26031	–0,34193	1,87043	–0,21861	1,45874	–0,0149
51	2,71047	–0,44695	–0,35421	–0,35093	–0,21284	0,18185
52	–0,54924	–0,36134	2,97081	2,32648	0,01315	0,14583

Окончание таблицы А.29

VN	0	1	2	3	4	5
53	–0,51477	–0,40314	1,23469	0,2071	0,00622	–0,15786
54	–0,37564	0,82963	0,77552	0,72145	–0,21951	–0,21676
55	1,59584	–0,35791	0,89497	–0,26614	–0,12674	–0,13373
56	–0,42762	1,49492	–0,31584	0,6556	–0,06384	–0,07354
57	–0,38684	0,71379	–0,43055	–0,24995	–0,29525	0,42418
58	–0,55264	–0,52329	–0,57869	–0,60007	–0,61091	–0,56922
59	–0,46465	0,55867	–0,28994	–0,14188	0,53955	–0,1124
60	–0,40785	–0,46448	1,26074	1,48856	0,10128	–0,10592
61	–0,427	0,70964	–0,30237	1,50654	0,0617	–0,27741
62	–0,43966	1,39518	–0,39417	–0,44686	–0,35716	–0,39151
63	–0,43383	0,9626	1,83882	–0,37505	–0,235	–0,05237

Таблица А.30 — Сборник кодов периодического пикового компонента

Имя	Содержание	Режим	Число элементов	Число векторов
pcdl	*PPC*	базовый длинный	10	64+64

VN	0	1	2	3	4	5	6	7	8	9
0	1,28888	0,26874	3,16758	1,37315	0,49535	0,94071	1,44487	0,37004	–0,70932	–0,05462
1	–0,02234	–0,20001	0,19604	0,26626	0,75466	–0,81913	0,03708	–0,29647	–1,78632	–0,18008
2	–2,19075	0,13333	2,93822	–0,22869	0,54111	0,51598	0,29629	–1,60571	0,48978	–0,40555
3	–0,16435	–1,43562	0,12102	0,1109	0,83013	0,16508	–0,07873	2,78031	0,15479	–0,28546
4	3,06554	–1,41708	–1,88582	0,14353	–1,66046	0,26375	0,08646	–0,41044	–0,23005	0,47026
5	–0,11899	–0,14149	0,03797	0,01927	3,72378	0,16278	0,28207	0,15989	–0,17563	0,48802
6	–0,19166	0,05232	0,68887	0,24894	–0,48014	0,01794	–0,09456	–0,03353	–3,71716	–0,37879
7	–0,08722	0,69099	0,52234	0,74945	–0,62316	0,58827	0,30028	1,0085	0,24395	0,06055
8	0,12958	–0,22199	0,96387	3,13907	–1,96687	0,53834	–0,69258	–0,04179	–0,39869	–0,03142
9	–0,02278	0,45226	–0,46875	0,03839	2,35515	–0,02039	–0,03662	–0,33315	–0,3716	–3,22201
10	1,50588	–0,78015	1,43802	–0,42078	–1,56427	–0,32711	–1,81516	–0,1517	–1,39942	–0,70676
11	0,03549	0,97029	0,17778	2,40233	–0,1207	1,79612	–0,09612	–3,32378	–0,06238	–0,47122
12	0,11414	–0,06499	1,95345	–0,03737	–0,08911	1,86812	0,03298	0,3419	–0,9445	0,26631
13	–1,20479	–0,70412	0,30736	1,66473	0,25032	–1,62769	0,26336	–2,39109	1,66663	–1,01706
14	0,27057	0,28062	4,60481	0,14478	–0,04332	–0,26043	–0,15716	–0,52842	1,15402	0,02068
15	–0,02468	–2,63793	–0,06035	2,17716	0,55198	1,33668	0,18469	0,73687	–1,59535	0,47395
16	0,08763	–1,03259	0,29432	0,23395	–0,21428	0,13532	–4,58495	0,5589	1,19783	–1,8095
17	–1,87625	1,93644	0,2355	0,60058	1,07984	–0,13744	–0,89547	–0,8501	0,80741	–0,42413
18	0,23718	–1,82628	2,62266	0,48302	0,5773	0,00497	–0,88499	0,11386	0,50195	–1,61181
19	2,29405	0,76792	0,77194	0,45336	–0,15374	–1,68579	0,40538	–0,73821	–0,52721	0,27069
20	0,31311	0,67546	0,05131	–0,53788	0,75649	–1,92602	0,44063	–3,20197	0,24121	–1,52231
21	0,38234	–0,34155	–2,34677	0,44454	–0,12667	–0,01386	0,83684	–3,42385	0,27861	–0,62057
22	–2,294	–0,00411	0,10751	–2,0974	0,55445	–1,90704	0,46176	–0,22817	–1,05712	–0,31803
23	0,08838	0,104	0,07782	0,28709	0,77533	0,5377	–3,49315	–0,15122	–0,71822	–0,55433
24	–0,12964	–0,15924	0,08426	4,31671	0,35474	–0,3812	0,08329	–0,5072	0,22773	0,10575
25	0,37323	0,11307	–0,26726	1,19933	–0,2052	3,97109	0,23754	0,42287	–0,17526	1,6363
26	–0,8932	0,98569	–0,17186	–0,66099	0,34743	0,31512	0,56153	0,03641	–0,2279	–0,32669
27	0,49296	1,50705	0,11747	3,64671	–0,08886	0,95113	–0,10827	0,24228	–0,2188	0,09239
28	–0,05516	–0,36735	0,06542	–0,10454	–0,64008	–0,07052	–0,50515	2,55652	–0,94717	–1,30125
29	0,09405	0,74533	–1,13331	0,37229	–0,38024	0,17472	0,61134	0,20712	–0,10988	–0,49257
30	–0,11187	–0,17222	–1,73824	–0,13261	–0,52195	0,62213	–0,08559	0,17988	3,15089	–0,24704

Продолжение таблицы А.30

VN	0	1	2	3	4	5	6	7	8	9
31	0,01548	3,74936	0,08098	–0,20425	–0,05489	–0,31726	0,69418	0,32569	–0,1634	0,50865
32	–0,17257	0,28592	2,00649	0,53473	2,57991	0,33445	–0,33589	–0,71256	0,51466	2,46521
33	–0,09483	1,08439	0,03998	0,11961	1,85047	0,00631	–0,45111	–0,71269	0,15621	1,16161
34	–0,02632	2,00056	–0,32356	–0,56487	0,10923	1,26427	–0,2674	2,41749	–0,16922	–0,8848
35	–0,0544	0,16837	2,56225	0,1894	–0,22447	–0,39572	0,49943	0,11457	0,35493	0,89878
36	–0,26616	–0,43364	0,3558	0,05177	0,50149	–0,24278	0,17047	–0,12116	5,45574	–0,15218
37	0,06203	0,23433	–0,99429	0,12958	–0,628	–3,51385	–0,24958	0,06293	–0,89521	0,2426
38	3,72107	1,23981	–0,88335	0,46989	–0,3001	–0,42813	0,41867	1,0639	0,31391	–0,57627
39	–1,12063	0,44526	1,91996	0,34752	–0,524	–0,6746	–2,59639	0,11581	–0,20243	0,28307
40	–0,03832	–0,01316	–0,03846	–0,007	0,04931	–0,09357	0,06964	–0,00049	0,21041	–0,08892
41	0,10125	–0,03012	–0,03539	–2,37631	0,28491	2,4899	–0,34378	–0,73052	0,52019	–0,10068
42	1,15802	0,7305	0,86786	–0,15069	2,3452	–0,69111	0,57534	1,29397	0,15267	–0,16541
43	–0,12218	–0,03125	–0,01363	–1,5446	–0,27404	–0,62664	0,46486	–0,44052	0,17302	0,77978
44	–0,03625	–0,33463	–1,59515	–0,42669	–0,3637	3,45674	0,35785	0,34057	0,26874	–0,66518
45	0,12284	0,01668	–0,36273	0,00005	0,05147	–0,29728	–0,00085	5,10734	0,41718	–0,05075
46	–0,20474	1,11663	–1,4853	–1,6009	–0,02503	–0,24016	–0,09167	2,05302	0,41858	–0,52989
47	–1,22133	–0,80464	3,2915	0,29576	–1,01545	0,85362	0,25387	1,69078	0,05201	–0,3414
48	0,10775	–0,30053	–0,19873	0,20553	0,38424	0,2795	0,33175	0,10667	–0,43378	5,0211
49	0,06324	–1,1164	–0,34474	–0,03852	2,47664	0,30992	0,34303	–3,0838	–0,26944	–0,15003
50	–3,87998	0,08673	–0,92651	–0,28277	0,7912	0,06152	0,05675	–0,04378	–0,15055	–0,49744
51	0,06965	–0,24747	–0,30479	0,44796	–0,38039	0,30899	–2,49186	0,15352	–0,07609	2,77728
52	0,03771	–0,07037	1,44405	–0,08627	0,01465	0,65797	–0,20831	–0,02516	3,82295	0,58679
53	–0,28911	0,00076	2,25478	–2,67418	1,08246	0,20249	0,62566	–0,04001	–0,20703	–0,15767
54	1,76926	0,21839	0,4214	–0,18353	–0,23904	0,25714	–0,23697	0,07767	0,53228	0,22643
55	0,19622	–0,064	0,10897	0,79608	–1,20257	–1,10434	0,30441	0,49781	3,11769	–0,29082
56	–0,31973	–1,36087	1,1628	0,16043	–1,01046	2,64181	–0,19644	–0,27073	–0,77847	–0,42009
57	0,04065	–2,4468	–0,45823	0,51819	0,08004	–1,72072	–0,52	0,59359	0,37168	0,78352
58	–2,06255	0,4418	–1,32138	2,39634	–0,89775	1,60514	–1,52266	–0,22682	–0,10606	–0,5798
59	–3,31681	0,85384	–0,27833	0,29965	–1,55139	–0,12387	0,08035	–0,41261	0,26349	–0,21486
60	–0,91388	–0,98165	–1,13409	–1,47777	0,23383	1,1217	–0,27391	–2,83883	0,37536	–0,83449
61	–0,30317	0,03241	–2,27905	0,39328	–0,10539	2,62031	–0,21607	–0,11414	–3,2875	0,16852
62	0,05213	2,18937	0,84015	1,33077	–0,05358	1,3409	–0,09204	0,09985	0,93062	0,59825
63	0,12392	1,59708	0,9337	–0,90451	–0,12543	–2,01707	0,37513	1,94555	0,23099	–0,70055
64	2,13905	0,10191	1,24373	0,51358	1,63431	–0,08351	0,30739	–1,16675	–0,35279	–0,37096
65	0,0812	1,07058	0,15907	–0,09428	–0,14151	–0,4618	–0,21325	0,45456	0,66027	3,37816
66	–1,32206	–0,4821	1,43573	1,74563	2,01948	–1,56545	0,17182	0,45271	0,0628	–0,91456
67	–2,3898	–2,35197	–0,22554	0,33998	–0,79031	0,06321	–0,01377	1,07176	1,19673	0,2861
68	1,24864	–0,63868	–1,0763	–0,28812	1,14537	1,80661	–0,15262	2,16631	1,67133	–0,29882
69	–0,05193	–1,29291	–0,85556	–0,03597	4,35258	–0,32884	0,79375	0,41554	0,35156	–0,57719
70	–0,1198	–0,05545	0,43083	–0,10895	–2,37948	–0,49303	2,75568	–0,04768	–2,33369	–0,07613
71	0,21686	1,41341	–0,42804	–2,70242	0,61944	–0,06463	–0,43017	–0,1903	–2,0755	–0,07512
72	–0,02184	0,4818	0,88869	4,23426	0,32367	–0,16813	–0,0971	–0,93254	0,52898	0,05603
73	–0,15532	0,52294	–1,60824	–0,18328	3,23327	0,1135	–1,45172	–0,0493	–0,23632	–0,25873
74	–0,05867	–2,59416	0,8312	0,31796	–0,30826	0,2373	2,77116	0,02446	0,55313	–0,22707
75	–0,51455	1,30626	–0,77779	2,71737	0,5061	–1,37105	0,15878	0,48821	0,25587	1,37404
76	–0,12972	0,16455	2,0636	0,25143	0,28128	2,95291	0,08582	0,09976	–0,25123	–0,30165
77	–0,04632	–1,3975	–0,14369	0,86661	0,47791	–3,43773	–0,08283	0,40818	–0,40507	1,10291
78	–0,0182	–0,15582	1,34311	0,06751	–0,14121	–0,53351	0,03831	0,5043	2,10788	–0,58678
79	0,39477	–3,04864	–0,45931	2,01978	0,04376	–0,52536	–0,31599	–0,12492	–0,10121	0,11121

Окончание таблицы А.30

VN	0	1	2	3	4	5	6	7	8	9
80	–0,03452	–0,03057	–0,09894	0,03226	0,01192	–0,48283	–4,65939	–0,11003	–0,3613	0,079
81	–1,31754	0,0574	–0,2971	0,08453	–0,30721	0,86434	2,45968	2,51238	1,23547	–0,50231
82	1,50294	–0,24216	2,28121	–0,90775	1,04895	0,21859	–2,06629	0,08329	0,34822	0,1581
83	0,36466	0,70677	–0,918	2,47546	3,79165	–0,24031	–0,37853	–0,32197	–0,37636	–0,02901
84	–0,12627	–0,04525	–0,82218	–2,26543	–0,84347	–1,03939	0,26653	–2,47079	1,23837	0,75649
85	0,24064	–0,62543	–2,94637	0,47164	1,58428	0,23203	–0,59003	0,3325	–0,13853	0,14422
86	–0,17463	–0,12583	0,85598	–0,17722	1,54264	0,05216	–0,08885	0,88151	0,46255	–0,48946
87	–0,06676	0,36942	0,58086	0,11589	–1,18536	–0,12857	–2,80603	0,00658	1,06034	–0,58457
88	–0,1561	–0,18171	–0,63499	2,6653	0,46641	0,59787	0,30223	–0,53096	1,29943	–0,98973
89	–0,00976	1,04356	0,16697	1,99034	0,23572	2,65679	–0,00325	1,03831	–0,0701	–0,77931
90	0,13381	0,49307	0,04619	–1,68299	0,19455	0,52731	0,14125	0,99717	0,43132	–0,97925
91	–0,1563	1,8791	0,19313	0,23078	0,30029	1,5847	0,36806	–0,91846	–0,10561	2,76858
92	–0,03141	–0,27603	–1,25022	0,76867	0,37696	–1,86725	0,45545	2,35843	0,57733	–0,51505
93	0,19081	–0,02002	0,89507	–0,17124	0,07392	0,07283	–0,47206	–0,03043	–0,34827	–4,29424
94	–0,29	0,04272	–1,72027	–0,11905	0,27361	–1,60983	0,1195	–0,37867	3,50315	0,36605
95	–0,1328	4,18273	0,04841	0,61437	0,60526	–0,23885	–0,05236	–0,82974	–0,29851	0,51121
96	0,03186	–0,24892	0,25614	–0,17325	3,12521	–0,12148	0,07243	–0,3332	–0,04557	–0,77374
97	–0,02997	2,20189	0,2339	0,05403	3,47503	–0,14415	0,23471	0,75477	0,00188	0,06228
98	–1,11961	0,43678	0,61581	–3,07064	–0,02222	–0,20063	–0,19664	–0,1743	1,29354	0,31722
99	0,80714	–0,31437	3,21672	0,05728	0,42811	–0,01624	0,4669	–0,14758	0,79094	–0,49991
100	–0,04883	0,18854	–0,0422	–0,24325	0,33396	0,58096	0,07173	0,18625	4,80098	–0,04774
101	0,12902	0,04455	–0,58451	0,22529	0,18288	–3,31484	0,20874	0,1439	–2,33226	–0,00004
102	1,4006	2,53879	2,23308	–0,40397	–0,91191	0,30424	–0,59309	–0,28013	–0,53681	–0,72096
103	0,14981	–0,34273	–0,1096	–2,08103	–0,55452	1,22105	–2,32346	0,71767	–0,04973	0,88489
104	–0,06296	0,00779	0,37141	–0,07311	–0,43599	–0,69232	0,37183	0,59265	–0,20391	0,76627
105	–0,04811	0,81816	–0,06811	–3,60488	–0,30926	1,2619	–0,25378	–2,67757	0,01421	0,15661
106	–0,18735	–0,29123	–0,1443	–0,17479	–0,2036	1,97977	–0,22341	–0,17369	–2,54577	–0,41017
107	–0,15575	0,64521	–0,20741	–0,06303	2,55173	0,08849	–1,3168	–0,0852	–3,59554	–0,16002
108	0,2538	0,42032	–2,50621	0,13451	–0,06898	2,41402	0,09969	–0,09772	1,45985	0,45849
109	0,19057	–0,22542	–0,16274	–0,19255	–0,085	0,1524	–0,06344	3,54373	–0,19634	0,42005
110	–0,00046	1,3075	0,44001	0,29739	0,12153	0,45681	–0,24122	3,97823	1,01631	–0,5718
111	–1,0474	0,76204	0,79817	0,02734	–0,85256	0,27437	–0,36512	0,45476	–0,34809	–0,26607
112	0,39011	–0,04813	0,28325	–0,40521	–0,26705	0,11646	–1,00205	–0,09658	0,39184	6,38045
113	0,18221	–3,1436	0,56642	–0,41265	2,10605	0,00959	0,32737	–0,30618	–0,46143	0,62372
114	–3,41695	–0,29748	0,93423	0,2968	–0,22186	–0,04393	0,4199	–0,02121	–0,0349	0,24145
115	0,08292	–0,34821	–0,98374	0,76693	–2,30082	0,11379	–2,00866	0,10299	–1,80156	0,49759
116	0,09801	0,02398	0,87932	–0,29853	–0,23085	0,60271	0,283	0,17743	–0,0655	0,11407
117	0,0495	–2,7049	0,93988	–0,84697	–0,47967	–0,45333	–0,34552	–0,60481	0,77467	1,79495
118	2,31525	–1,35283	1,51663	0,46045	–0,94423	0,66598	0,14087	–1,23472	0,07825	0,23723
119	–0,01891	0,01146	0,05543	–0,00939	–0,25309	–0,05408	–0,08464	–0,08331	0,28796	–0,23134
120	0,14456	–1,82728	0,54162	–1,45394	0,65876	2,88044	0,04801	1,52839	0,4177	0,24366
121	0,00192	–1,51934	–0,28914	0,27953	–0,43474	0,3359	0,34038	0,09476	–0,16812	0,55648
122	1,58963	0,62054	–1,19708	0,95618	–0,19293	0,13055	–0,46467	–1,08598	1,0294	–0,00537
123	–1,63945	1,27057	1,17362	0,77949	–0,25979	1,31919	1,09939	–0,59425	–1,04889	0,33754
124	–0,36973	–1,0016	0,18249	0,95907	0,72926	2,32241	0,60848	–2,12186	–0,24255	–0,65101
125	–0,00988	–0,04139	–0,76083	0,14922	0,57784	3,97886	–0,30668	–0,34279	–0,84522	0,11366
126	–0,12785	1,334	0,11998	0,17243	–1,82888	0,49492	0,09101	–0,99747	0,39561	–1,90151
127	0,1718	1,22115	0,41649	2,03242	–0,20785	–2,41295	–0,75	1,09243	0,65831	–2,08726

Таблицы для *ER BSAC*

Таблица А.31 — Параметры *cband_si_type*

cband_si_type	*max_cband_si_len*	Наибольший *cband_si*		Модель, перечисленная в	
		Нулевой *cband*	Другой *cband*	Нулевой *cband*	Другой *cband*
0	6	6	4	Таблица А.51	Таблица А.44
1	5	6	6	Таблица А.51	Таблица А.45
2	6	8	4	Таблица А.51	Таблица А.44
3	5	8	6	Таблица А.51	Таблица А.45
4	6	8	8	Таблица А.51	Таблица А.46
5	6	10	4	Таблица А.51	Таблица А.44
6	5	10	6	Таблица А.51	Таблица А.45
7	6	10	8	Таблица А.51	Таблица А.46
8	5	10	10	Таблица А.51	Таблица А.47
9	6	12	4	Таблица А.51	Таблица А.44
10	5	12	6	Таблица А.51	Таблица А.45
11	6	12	8	Таблица А.51	Таблица А.46
12	8	12	12	Таблица А.51	Таблица А.48
13	6	14	4	Таблица А.51	Таблица А.44
14	5	14	6	Таблица А.51	Таблица А.45
15	6	14	8	Таблица А.51	Таблица А.46
16	8	14	12	Таблица А.51	Таблица А.48
17	9	14	14	Таблица А.51	Таблица А.49
18	6	15	4	Таблица А.51	Таблица А.44
19	5	15	6	Таблица А.51	Таблица А.45
20	6	15	8	Таблица А.51	Таблица А.46
21	8	15	12	Таблица А.51	Таблица А.48
22	10	15	15	Таблица А.51	Таблица А.50
23	8	16	12	Таблица А.51	Таблица А.48
24	10	16	16	Таблица А.51	Таблица А.50
25	9	17	14	Таблица А.51	Таблица А.49
26	10	17	17	Таблица А.51	Таблица А.50
27	10	18	18	Таблица А.51	Таблица А.50
28	12	19	19	Таблица А.51	Таблица А.50
29	12	20	20	Таблица А.51	Таблица А.50
30	12	21	21	Таблица А.51	Таблица А.50
31	12	22	22	Таблица А.51	Таблица А.50

Таблица А.32 — Параметры модели масштабного коэффициента

scf_model	Наибольший дифференциальный *ArModel*	Модель, перечисленная в
0	0	Не используется
1	3	Таблица А.37
2	7	Таблица А.38
3	15	Таблица А.39
4	15	Таблица А.40
5	31	Таблица А.41
6	31	Таблица А.42
7	63	Таблица А.43

Таблица А.33 — Параметры *cband_si BSAC*

cband_si	Плоскость *MCB*	Таблица, перечисленная	*cband_si*	Плоскость *MCB*	Таблица, перечисленная
0	0	—	12	6	Таблица А.67
1	1	Таблица А.56	13	7	Таблица А.68
2	1	Таблица А.57	14	7	Таблица А.69
3	2	Таблица А.58	15	8	Таблица А.70
4	2	Таблица А.59	16	9	Таблица А.71
5	3	Таблица А.60	17	10	Таблица А.72
6	3	Таблица А.61	18	11	Таблица А.73
7	4	Таблица А.62	19	12	Таблица А.74
8	4	Таблица А.63	20	13	Таблица А.75
9	5	Таблица А.64	21	14	Таблица А.76
10	5	Таблица А.65	22	15	Таблица А.77
11	6	Таблица А.66			

Таблица А.34 — Позиция величины вероятности в таблице вероятностей

			h	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
			g	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
			f	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
			e	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
a	*b*	*c*	*d*																
0	x	x	x		0	15	22	29	32	39	42	45							
1	x	x	0		1	16	23	30					46	53	56	59			
	x	x	1		2	17	24	31					46	53	56	59			
2	x	0	0		3	18			33	40			47	54			60	63	
	x	0	1		4	19			33	40			48	55			60	63	
	x	1	0		5	20			34	41			47	54			60	63	
	x	1	1		6	21			34	41			48	55			60	63	
3	0	0	0		7		25		35		43		49		57		61		64
	0	0	1		8		25		36		43		50		57		62		64
	0	1	0		9		26		35		43		51		58		61		64
	0	1	1		10		26		36		43		52		58		62		64
	1	0	0		11		27		37		44		49		57		61		64
	1	0	1		12		27		38		44		50		57		62		64
	1	1	0		13		28		37		44		51		58		61		64
	1	1	1		14		28		38		44		52		58		62		64

где:

I — спектральный индекс;

$$a = i \% 4;$$

b — вырезанный бит (*i*–3) ых спектральных данных, значение которых то же самое, как значение *i*-ых спектральных данных;

c — вырезанный бит (*i*–2) ых спектральных данных, значение которых то же самое, как значение *i*-ых спектральных данных;

d — вырезанный бит (*i*–1) ых спектральных данных, значение которых то же самое как значение *i*-ых спектральных данных;

e — являются ли старшие биты (*i*–*a*+3) ых спектральных данных, значение которых больше, чем значение из *i*-ых спектральных данных, ненулевыми (1) или нулевыми (0);

f — являются ли старшие биты (*i*–*a*+2) ых спектральных данных, значение которых больше, чем значения из *i*-ых спектральных данных, ненулевыми (1) или нулевыми (0);

g — являются ли старшие биты (*i*–*a*+1)ых спектральных данных, значение которых больше, чем значения из *i*-ых спектральных данных, ненулевыми (1) или нулевыми (0);

h — являются ли сташие биты (*i*–*a*)ых спектральных данных, значение которых больше, чем значения из *i*-ых спектральных данных, ненулевыми (1) или нулевыми (0).

Таблица А.35 — Минимальная вероятность (*min_p*0) в пропорции к доступной длине уровня

Доступная длина	1	2	3	4	5	6	7	8	9	10	11	12	13
(Шестнадцатеричный) *min_p*0	2000	1000	800	400	200	100	80	40	20	10	8	4	2

Таблица А.36 — Максимальная вероятность (*max_p*0) в пропорции к доступной длине уровня

Доступная длина	1	2	3	4	5	6	7	8	9	10	11	12	13
(Шестнадцатеричный) *min_p*0	2000	3000	3800 0	3*C*0 0	3*E*0 0	3*F*0 0	3*F*8 0	3*FC* 0	3*FE* 0	3*FF* 8	3*FF C*	3*FF*	3*FFE*

Таблица А.37 — Арифметическая модель масштабного коэффициента 1

Размер	Кумулятивные частоты (шестнадцатеричные)			
4	752	3*cd*	14*d*	0

Таблица А.38 — Арифметическая модель масштабного коэффициента 2

Размер	Кумулятивные частоты (шестнадцатеричные)							
8	112*f*	*de*7	*a*8*b*	7*c*1	47*a*	23*a*	*d*4	0

Таблица А.39 — Арифметическая модель масштабного коэффициента 3

Размер	Кумулятивные частоты (шестнадцатеричные)							
16	1*f*67	1*c*5*f*	18*d*8	1555	1215	*eb*4	*adc*	742
	408	1*e*6	*df*	52	32	23	c	0

Таблица А.40 — Арифметическая модель масштабного коэффициента 4

Размер	Кумулятивные частоты (шестнадцатеричные)							
16	250*f*	22*b*8	2053	1*deb*	1*b*05	186*d*	15*df*	12*d*9
	*f*77	*c*01	833	50*d*	245	8*c*	33	0

Таблица А.41 — Арифметическая модель масштабного коэффициента 5

Размер	Кумулятивные частоты (шестнадцатеричные)							
32	8*a*8	74*e*	639	588	48*c*	3*cf*	32*e*	272
	1*bc*	13*e*	*e*4	97	69	43	2*f*	29
	20	1*b*	18	15	12	*f*	*d*	*c*
	a	9	7	6	4	3	1	0

Таблица А.42 — Арифметическая модель масштабного коэффициента 6

Размер	Кумулятивные частоты (шестнадцатеричные)							
32	*c*2a	99*f*	809	6*ec*	603	53*d*	491	40*e*
	394	30*a*	2*a*5	259	202	1*bc*	170	133
	102	*c*9	97	73	4*f*	37	22	16
	f	*b*	9	7	5	3	1	0

Таблица А.43 — Арифметическая модель масштабного коэффициента 7

Размер	Кумулятивные частоты (шестнадцатеричные)							
64	3*b*5*e*	3*a*90	39*d*3	387*c*	3702	3566	33*a*7	321*c*
	2*f*90	2*cf*2	29*fe*	26*fa*	23*e*4	20*df*	1*e*0*d*	1*ac*4
	1804	159*a*	131*e*	10*e*7	*e*5*b*	*c*9*c*	*b*78	*a*21
	8*fd*	7*b*7	6*b*5	62*c*	55*d*	4f6	4*d*4	44*b*
	38*e*	2*e*2	29*d*	236	225	1*f*2	1*cf*	1*ad*
	19*c*	179	168	157	146	135	123	112
	101	*f*0	*df*	*ce*	*bc*	*ab*	9*a*	89
	78	67	55	44	33	22	11	0

Таблица А.44 — Арифметическая модель *cband_si* 0

Размер	Совокупные (шестнадцатеричные) частоты				
5	3*ef*6	3*b*59	1*b*12	12*a*3	0

Таблица А.45 — Арифметическая модель *cband_si* 1

Размер	Совокупные (шестнадцатеричные) частоты						
7	3*d*51	33*ae*	1*cff*	*fb*7	7*e*4	22*b*	0

Таблица А.46 — Арифметическая модель *cband_si* 2

Размер	Совокупные (шестнадцатеричные) частоты							
9	3*a*47	2*aec*	1*e*05	1336	*e*7*d*	860	5*e*0	44*a*
	0							

Таблица А.47 — Арифметическая модель *cband_si* 3

Размер	Совокупные (шестнадцатеричные) частоты							
11	36*be*	27*ae*	20*f*4	1749	14*d*5	*d*46	*ad*3	888
	519	20*b*	0					

Таблица А.48 — Арифметическая модель *cband_si* 4

Размер	Совокупные (шестнадцатеричные) частоты							
13	3983	2*e*77	2*b*03	1*ee*8	1*df*9	1307	11*e*4	*b*4*d*
	94c	497	445	40	0			

Таблица А.49 — Арифметическая модель *cband_si* 5

Размер	Совокупные (шестнадцатеричные) частоты							
15	306*f*	249*e*	1*f*56	1843	161*a*	102*d*	*f*6*c*	*c*81
	*af*2	7*a*8	71*a*	454	413	16	0	

Таблица А.50 — Арифметическая модель *cband_si* 6

Размер	Совокупные (шестнадцатеричные) частоты							
23	31*af*	2001	162*d*	127*e*	*f*05	*c*34	*b*8*f*	*a*61
	955	825	7*dd*	6*a*9	688	55*b*	54*b*	2*f*7
	198	77	10	*c*	8	4	0	

Таблица А.51 — Арифметическая модель *cband_si* для нулевой полосы кодирования

Размер	Совокупные (шестнадцатеричные) частоты							
23	3*ff*8	3*ff*0	3*fe*8	3*fe*0	3*fd*7	3*f*31	3*cd*7	3*bc*9
	3074	2*bc*f	231*b*	13*db*	*d*51	603	44*c*	80
	30	28	20	18	10	8	0	

Таблица А.52 — Модель *MC_used*

Размер	Совокупные (шестнадцатеричные) частоты	
2	*2CCD*	0

Таблица А.53 — Модель *stereo_info*

Размер	Совокупные (шестнадцатеричные) частоты		
4	3666	1000	666,0

Таблица А.54 — Арифметическая модель noise_*flag*

Размер	Совокупные (шестнадцатеричные) частоты	
2	2000	0

Таблица А.55 — Арифметическая модель *noise_mode*

Размер	Совокупные (шестнадцатеричные) частоты			
4	3000	2000	1000	0

Таблица А.56 — Таблица вероятности *BSAC* 1 (Плоскость *MCB* = 1)

Значение	Значение вероятности символа, '0' (Шестнадцатеричный)							
1	3900	3*a*00	2*f*00	3*b*00	2*f*00	3700	2*c*00	3*b*00
	3000	3600	2*d*00	3900	2*f*00	3700	2*c*00	

Таблица А.57 — Таблица вероятности *BSAC* 2 (Плоскость *MCB* = 1)

Значение	Значение вероятности символа, '0' (Шестнадцатеричный)							
1	2800	2800	2500	2900	2600	2700	2300	2*a*00
	2700	2800	2400	2800	2500	2600	2200	

Таблица А.58 — Таблица вероятности *BSAC* 3 (Плоскость *MCB* = 2)

Значение	Декодируемые старшие биты	Значение вероятности символа, '0' (Шестнадцатеричный)							
2	Ноль	3*d*00	3*d*00	3300	3*d*00	3300	3*b*00	3300	3*d*00
		3200	3*b*00	3100	3*e*00	3700	3*c*00	3300	
1	Ноль	3700	3*a*00	2800	3*b*00	2600	2*c*00	2400	3*a*00
		2500	2*b*00	2400	3100	2300	2900	2300	3000
		2*c*00	1*d*00	2200	1*a*00	1*c*00	1600	2700	2200
		1*a*00	1*d*00	1900	1*c*00	1*e*00	2*c*00	2400	1900
		1*e*00	1*f*00	1*c*00	2*b*00	2400	2900	2700	2400
		1300	1*a*00	2000	1800	2300	2500	1*f*00	2*c*00
		2300	3600	2800	3100	2500	1400	1200	1800
		1400	2100	2200	1000	1*e*00	3000	2600	1200
		2200							
	Ненулевой	3100							

Таблица А.59 — Таблица вероятности *BSAC* 4 (Плоскость *MCB* = 2)

Значение	Декодируемые старшие биты	Значение вероятности символа, '0' (Шестнадцатеричный)							
2	Ноль	3900	3*a*00	2*e*00	3*a*00	2*f*00	3400	2*a*00	3*a*00
		3000	3500	2*c*00	3600	2*b*00	3100	2500	
1	Ноль	1*e*00	1*d*00	1*c*00	1*d*00	1*c*00	1*d*00	1*b*00	1*d*00
		1*e*00	1*e*00	1*a*00	1*e*00	1*c*00	1*d*00	1*b*00	1*a*00
		1*a*00	1800	1800	1800	1700	1700	1800	1*a*00
		1700	1700	1900	1800	1600	1700	1600	1500
		1700	1800	1600	1*c*00	1700	1900	1700	1500
		1*c*00	1500	1600	0*f*00	1800	1400	1700	1*a*00
		1*a*00	1*e*00	1800	1*c*00	1*b*00	1500	1300	1500
		1400	1600	1500	1700	1600	1*b*00	1800	1400
		1400							
	Ненулевой	3600							

Таблица А.60 — Таблица вероятности *BSAC* 5 (Плоскость *MCB* = 3)

Значение	Декодируемые старшие биты	Значение вероятности символа, '0' (Шестнадцатеричный)							
3	Ноль	3*d*00	3*d*00	3200	3*d*00	3300	3*d*00	3600	3*d*00
		3500	3*c*00	3500	3*f*00	3*b*00	3*f*00	3*d*00	
2	Ноль	3*c*00	3*d*00	2*b*00	3*d*00	2900	3500	2*c*00	3*d*00
		2*b*00	3400	2*b*00	3800	2*b*00	3700	2*a*00	3900
		3400	2400	2*a*00	1*c*00	1*f*00	1600	3500	2500
		1*a*00	2*a*00	2200	2*b*00	2*a*00	3500	2600	1*a*00
		2600	2500	2700	3500	2*d*00	3800	3200	2*e*00
		1800	1600	2900	2500	3100	2*c*00	2300	3600
		3000	3*c*00	3300	3*b*00	3400	1700	1*a*00	1*c*00
		1900	2900	2*a*00	2400	2700	3*c*00	3600	1*d*00
		3100							
	Ненулевой	3100							
1	Ноль	3400	3800	2700	3900	2700	2*f*00	2200	3800
		2500	2*d*00	2000	3300	2000	2900	1*e*00	2*b*00
		2300	1*a*00	1*a*00	1*b*00	1800	1700	1*e*00	1*c*00
		1*b*00	1*c*00	1*b*00	1*a*00	1800	1*d*00	1*b*00	1800
		1900	1*b*00	1*a*00	1*d*00	1*e*00	1*f*00	1*b*00	1*e*00
		1200	1400	1*a*00	1300	1*c*00	1*b*00	1900	2000
		1*e*00	3000	2900	2*d*00	2500	1300	1700	1400
		1300	1*e*00	1*f*00	1100	1900	2100	1*e*00	1500
		1*a*00							
	Ненулевой	2*a*00	2*b*00	2800					

Таблица А.61 — Таблица вероятности *BSAC* 6 (Плоскость *MCB* = 3)

Значение	Декодируемые старшие биты	Значение вероятности символа, '0' (Шестнадцатеричный)							
3	Ноль	3800	3*a*00	2*d*00	3*a*00	2*d*00	3600	2*d*00	3*a*00
		2*d*00	3600	2*b*00	3*a*00	2800	3600	2700	
2	Ноль	2*b*00	3000	2500	2*f*00	2600	2*d*00	2400	3000
		2500	2*b*00	2400	2*d*00	2500	2800	2500	2*a*00
		2900	2300	2200	1*e*00	1*b*00	1900	2600	2300
		1*f*00	1*d*00	2200	1*b*00	1800	2100	2100	1*d*00
		1*d*00	1*f*00	1*f*00	2900	2600	2*a*00	2100	2300
		1800	1*a*00	1*d*00	2000	1*c*00	1*a*00	1*e*00	2900
		2800	2*f*00	2300	2*f*00	2600	1*d*00	1700	1*d*00
		1*c*00	1*e*00	2100	1700	2200	2300	2300	1400
		1*a*00							
	Ненулевой	3000							
1	Ноль	1900	1900	1900	1*b*00	1700	1*b*00	1*a*00	1000
		1900	1600	1800	1*e*00	1900	1*a*00	1700	1*b*00
		1700	1500	1500	1500	1700	1400	1900	1700
		1600	1600	1200	1300	1200	1600	1500	1500
		1300	1600	1600	1*c*00	1400	1700	1600	1400
		1400	1400	1500	1400	1300	1300	1500	1800
		1600	1*f*00	1*a*00	1*e*00	1800	1700	1600	1600
		1300	1400	1300	1100	1500	1600	1500	1200
		1300							
	Ненулевой	2*b*00	2800	2700					

Таблица А.62 — Таблица вероятности *BSAC* 7 (Плоскость *MCB* = 4)

Значение	Декодируемые старшие биты	Значение вероятности символа, '0' (Шестнадцатеричный)							
MCB	Ноль	3*d*00	3*d*00	3500	3*e*00	3500	3*f*00	3*b*00	3*e*00
		3200	3*f*00	3*a*00	3*f*00	3*d*00	3*f*00	3*b*00	
MCB-1	Ноль	3*f*00	3*f*00	3200	3*f*00	3500	3*e*00	3700	3*f*00
		2*d*00	3*c*00	3000	3*f*00	3700	3*e*00	3400	3*f*00
		3900	2600	2*f*00	1*e*00	2400	1500	3700	3100
		1*b*00	2600	2300	3*a*00	3900	3*e*00	2*b*00	2200
		2800	2*f*00	2500	3*e*00	3700	3*e*00	3*d*00	3900
		1*a*00	3300	2500	2800	3*c*00	3800	2*c*00	3*d*00
		3800	3*f*00	3*b*00	3*f*00	3*a*00	1*e*00	1*b*00	1800
		1800	3*b*00	3*a*00	1200	2*f*00	3*f*00	3*b*00	1*b*00
		3500							
	Ненулевой	2*F*00							
MCB-2	Ноль	3*c*00	3*e*00	3000	3*e*00	3100	3*a*00	3100	3*d*00
		2*c*00	3900	2*e*00	3*c*00	2*d*00	3*c*00	3100	3*d*00
		3100	2100	2*c*00	2600	2800	1*d*00	2*b*00	2800
		2800	2400	2200	2100	2300	2*d*00	2500	1*f*00
		2100	2b00	2700	3200	2d00	3400	2a00	3500
		1800	1800	1*f*00	1*e*00	2*e*00	2*a*00	2400	3000
		2*b*00	3*e*00	3*d*00	3*d*00	3*a*00	1*e*00	2*b*00	2600
		1900	3400	3500	1*c*00	2600	3300	2*a*00	1*c*00
		2*b*00							
	Ненулевой	2800	2900	2400					
Другие	Ноль	3500	3*b*00	2900	3*b*00	2*a*00	3100	2700	3*b*00
		2600	2*f*00	2400	3400	2300	2*d*00	2000	3300
		2700	1*c*00	2400	1*c*00	1*c*00	1900	2700	2800
		1*b*00	1*d*00	2000	1*b*00	1*a*00	2300	1*d*00	1700
		1*e*00	2400	2100	2*b*00	2100	2800	2000	2300
		1*b*00	1500	1*b*00	1400	1*a*00	1*a*00	2000	2*a*00
		2200	3700	2*f*00	3200	2*a*00	1700	1700	1600
		1900	2500	2300	1500	1900	2500	2200	1400
		1*b*00							
	Ненулевой	2*d*00	2500	2300	2500	2500	2600	2400	

Таблица А.63 — Таблица вероятности *BSAC* 8 (Плоскость *MCB* = 4)

Значение	Декодируемые старшие биты	Значение вероятности символа, '0' (Шестнадцатеричный)							
MCB	Ноль	3*b*00	3*c*00	3400	3*c*00	3400	3000	3*b*00	3*c*00
		3200	3*a*00	3100	3*c*00	3000	3900	3*f*00	
MCB-1	Ноль	3500	3800	2*c*00	3900	2*c*00	3400	2*b*00	3800
		2*e*00	3400	2*d*00	3600	2*a*00	3300	2800	3100
		3100	2600	2900	2000	2300	2*c*00	2*d*00	2600
		2000	2600	2300	2500	2100	1*f*00	2400	1*d*00
		2500	2400	2400	3000	2800	3000	2900	2200
		1*e*00	1*c*00	2500	1*d*00	2300	2300	2500	3300
		2*c*00	3700	2*b*00	3400	2*c*00	1*e*00	1*c*00	2100
		1*b*00	2900	2*a*00	1*d*00	2600	3200	2*a*00	2000
		2400							
	Ненулевой	3200							
MCB-2	Ноль	2900	2*e*00	2600	2*f*00	2600	2*d*00	2600	2*e*00
		2500	2*b*00	2600	2*f*00	2300	2*a*00	2300	2800
		2800	2100	2400	2000	2000	1*b*00	2400	1*f*00
		1*c*00	2100	2200	1*d*00	1*c*00	1*f*00	1*c*00	1900
		1*e*00	2100	2100	2900	2200	2300	2100	1*c*00
		1*a*00	1*a*00	2100	2100	1*c*00	1*c*00	1*f*00	2700
		2500	2*d*00	2700	2*a*00	2300	1*c*00	1*d*00	1*a*00
		1*a*00	1*b*00	1*d*00	1800	2000	2300	1*f*00	1900
		1*c*00							
	Ненулевой	2*b*00	2900	2800					
Другие	Ноль	1*c*00	1*e*00	1*b*00	1*e*00	1*c*00	1*e*00	1900	1*a*00
		1*f*00	1*f*00	1900	2000	1*a*00	1*f*00	1700	1*b*00
		1*a*00	1900	1800	1900	1800	1600	1900	1*a*00
		1900	1700	1800	1700	1800	1600	1700	1400
		1600	1800	1*a*00	1*c*00	1*c*00	1*c*00	1700	1700
		1500	1500	1600	1600	1500	1400	1700	1*b*00
		1*a*00	2300	1*c*00	1*d*00	1*a*00	1600	1600	1500
		1400	1800	1500	1300	1700	1900	1600	1400
		1400							
	Ненулевой	2800	2500	2500	2700	2500	2600	2500	

Таблица А.64 — Таблица вероятности *BSAC* 9 (Плоскость MCB = 5)

Значение	Декодируемые старшие биты	Значение вероятности символа, '0' (Шестнадцатеричный)
MCB	Ноль	3*d*00 3*e*00 3300 3*e*00 3500 3*e*00 3700 3*e*00 3400 3*e*00 3500 3*f*00 3*d*00 3*f*00 3*c*00
MCB-1	Ноль	Таблица вероятности *BSAC* 7 (Смотри таблицу А.62)
	Ненулевой	2*e*00
MCB-2	Ноль	Таблица вероятности *BSAC* 7 (Смотри таблицу А.62)
	Ненулевой	2900 2*a*00 2700
MCB-3	Ноль	Таблица вероятности *BSAC* 7 (Смотри таблицу А.62)
	Ненулевой	2*d*00 2500 2400 2500 2400 2500 2300
Другие	Ноль	Таблица вероятности *BSAC* 7 (Смотри Таблицу А.62)
	Ненулевой	2800 2500 2300 2300 2200 2200 2200 2200 2200 2200 2200 2100 2000 2200 2100 2000

Таблица А.65 — Таблица вероятности *BSAC* 10 (Плоскость MCB = 5)

Значение	Декодируемые старшие биты	Значение вероятности символа, '0' (Шестнадцатеричный)
MCB	Ноль	3*b*00, 3*c*00, 3400, 3*c*00, 3200, 3900, 2*e*00, 3*d*00, 3400, 3900, 2*f*00, 3*c*00, 2*d*00, 3700, 2*d*00
MCB-1	Ноль	Таблица вероятности *BSAC* 8 (Смотри таблицу А.63)
	Ненулевой	3100
MCB-2	Ноль	Таблица вероятности *BSAC* 8 (Смотри таблицу А.63)
	Ненулевой	2*b*00, 2*a*00, 2900
MCB-3	Ноль	Таблица вероятности *BSAC* 8 (Смотри таблицу А.63)
	Ненулевой	2700, 2600, 2500, 2500, 2500, 2200, 2200
другие	Ноль	Таблица вероятности *BSAC* 8 (Смотри таблицу А.63)
	Ненулевой	2200, 2300, 2300, 2300, 2200, 2300, 2200, 2300, 2200, 2200, 2200, 2200, 2200, 2000, 2100, 2200

Таблица А.66 — Таблица вероятности *BSAC* 11

То же самое как таблица вероятности *BSAC* 9, но плоскость *MCB* = 6

Таблица А.67 — Таблица вероятности *BSAC* 12

То же самое как Таблица вероятности *BSAC* 10, но плоскость *MCB* = 6

Таблица А.68 — Таблица вероятности *BSAC* 13

То же самое как Таблица вероятности *BSAC* 9, но плоскость *MCB* = 7

Таблица А.69 — Таблица вероятности *BSAC* 14

То же самое как таблица вероятности *BSAC* 10, но плоскость *MCB* = 7

Таблица А.70 — Таблица вероятности *BSAC* 15

То же самое как таблица вероятности *BSAC* 9, но плоскость *MCB* = 8

Таблица А.71 — Таблица вероятности *BSAC* 16

То же самое как таблица вероятности *BSAC* 9, но плоскость *MCB* = 9

Таблица А.72 — Таблица вероятности *BSAC* 17

То же самое как таблица вероятности *BSAC* 9, но плоскость *MCB* = 10

Таблица А.73 — Таблица вероятности *BSAC* 18

То же самое как таблица вероятности *BSAC* 9, но плоскость *MCB* = 11

Таблица А.74 — Таблица вероятности *BSAC* 19

То же самое как таблица вероятности *BSAC* 9, но плоскость *MCB* = 12

Таблица А.75 — Таблица вероятности *BSAC* 20

То же самое как таблица вероятности *BSAC* 9, но плоскость *MCB* = 13

Таблица А.76 — Таблица вероятности *BSAC* 21

То же самое как таблица вероятности *BSAC* 9, но плоскость *MCB* = 14

Таблица А.77 — Таблица вероятности *BSAC* 22

То же самое как таблица вероятности *BSAC* 9, но плоскость *MCB* = 15

Таблицы для *SBR*

Таблицы Хаффмана *SBR*
Функция *sbr_huff_dec ()* используется как:
данные = sbr_huff_dec (t_huff, codeword),
где *t_huff* является выбранной таблицей Хаффмана, и *codeword* является словом, считанным из полезной нагрузки потока битов. Возвращенное значение *data* является индекс таблицы Хаффмана, соответствующим определенной кодовой комбинации, с вычтенным наибольшим абсолютным значением (*LAV*) таблицы.

Таблица А.78 — Обзор таблиц Хаффмана

Наименование таблицы	*df_env_flag*	*df_noise_flag*	*amp_res*	*LAV*	Примечания
t_huffman_env_1_5dB	0	*dc*	0	60	Примечание 1
f_huffman_env_1_5dB	1	*dc*	0	60	
t_huffman_env_bal_1_5dB	0	*dc*	0	24	
f_huffman_env_bal_1_5dB	1	*dc*	0	24	
t_huffman_env_3_0dB	0	*dc*	1	31	
f_huffman_env_3_0dB	1	*dc*	1	31	
t_huffman_env_bal_3_0dB	0	*dc*	1	12	
f_huffman_env_bal_3_0dB	1	*dc*	1	12	
t_huffman_noise_3_0dB	*dc*	0	*dc*	31	
f_huffman_noise_3_0dB	*dc*	1	*dc*	31	Примечание 2
t_huffman_noise_bal_3_0dB	*dc*	0	*dc*	12	
f_huffman_noise_bal_3_0dB	*dc*	1	*dc*	12	Примечание 2

Примечание 1 — *dc* (безразличное), указывает, что переменная не релевантна.
Примечание 2 — Таблицы Хаффмана *f_huffman_noise_3_0dB* и *f_huffman_noise_bal_3_0dB* являются такими же как *f_huffman_env_3_0dB* и *f_huffman_env_bal_3_0dB*, соответственно.

Таблица А.79 — *t_huffman_ env*_1_5*dB*

Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)
0	0x00000012	0x0003*FFD*6	61	0x00000003	0x00000004
1	0x00000012	0x0003*FFD*7	62	0x00000004	0x0000000*C*
2	0x00000012	0x0003*FFD*8	63	0x00000005	0x0000001*C*
3	0x00000012	0x0003*FFD*9	64	0x00000006	0x0000003*C*
4	0x00000012	0x0003*FFDA*	65	0x00000007	0x0000007*C*
5	0x00000012	0x0003*FFDB*	66	0x00000008	0x000000*FC*
6	0x00000013	0x0007*FFB*8	67	0x00000009	0x000001*FC*
7	0x00000013	0x0007*FFB*9	68	0x0000000*A*	0x000003*FD*
8	0x00000013	0x0007*FFBA*	69	0x0000000*C*	0x00000*FFA*
9	0x00000013	0x0007*FFBB*	70	0x0000000*D*	0x00001*FF*8
10	0x00000013	0x0007*FFBC*	71	0x0000000*E*	0x00003*FF*6
11	0x00000013	0x0007*FFBD*	72	0x0000000*E*	0x00003*FF*8
12	0x00000013	0x0007*FFBE*	73	0x0000000*F*	0x00007*FF*5
13	0x00000013	0x0007*FFBF*	74	0x00000010	0x0000*FFEF*
14	0x00000013	0x0007*FFC*0	75	0x00000011	0x0001*FFE*8
15	0x00000013	0x0007*FFC*1	76	0x00000010	0x0000*FFF*2
16	0x00000013	0x0007*FFC*2	77	0x00000013	0x0007*FFD*4
17	0x00000013	0x0007*FFC*3	78	0x00000013	0x0007*FFD*5
18	0x00000013	0x0007*FFC*4	79	0x00000013	0x0007*FFD*6
19	0x00000013	0x0007*FFC*5	80	0x00000013	0x0007*FFD*7
20	0x00000013	0x0007*FFC*6	81	0x00000013	0x0007*FFD*8
21	0x00000013	0x0007*FFC*7	82	0x00000013	0x0007*FFD*9
22	0x00000013	0x0007*FFC*8	83	0x00000013	0x0007*FFDA*
23	0x00000013	0x0007*FFC*9	84	0x00000013	0x0007*FFDB*
24	0x00000013	0x0007*FFCA*	85	0x00000013	0x0007*FFDC*
25	0x00000013	0x0007*FFCB*	86	0x00000013	0x0007*FFDD*
26	0x00000013	0x0007*FFCC*	87	0x00000013	0x0007*FFDE*
27	0x00000013	0x0007*FFCD*	88	0x00000013	0x0007*FFDF*
28	0x00000013	0x0007*FFCE*	89	0x00000013	0x0007*FFE*0
29	0x00000013	0x0007*FFCF*	90	0x00000013	0x0007*FFE*1
30	0x00000013	0x0007*FFD*0	91	0x00000013	0x0007*FFE*2
31	0x00000013	0x0007*FFD*1	92	0x00000013	0x0007*FFE*3
32	0x00000013	0x0007*FFD*2	93	0x00000013	0x0007*FFE*4
33	0x00000013	0x0007*FFD*3	94	0x00000013	0x0007*FFE*5
34	0x00000011	0x0001*FFE*6	95	0x00000013	0x0007*FFE*6
35	0x00000012	0x0003*FFD*4	96	0x00000013	0x0007*FFE*7
36	0x00000010	0x0000*FFF*0	97	0x00000013	0x0007*FFE*8
37	0x00000011	0x0001*FFE*9	98	0x00000013	0x0007*FFE*9
38	0x00000012	0x0003*FFD*5	99	0x00000013	0x0007*FFEA*
39	0x00000011	0x0001*FFE*7	100	0x00000013	0x0007*FFEB*
40	0x00000010	0x0000*FFF*1	101	0x00000013	0x0007*FFEC*
41	0x00000010	0x0000*FFEC*	102	0x00000013	0x0007*FFED*
42	0x00000010	0x0000*FFED*	103	0x00000013	0x0007*FFEE*

Окончание таблицы А.79

Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)
43	0x00000010	0x0000*FFEE*	104	0x00000013	0x0007*FFEF*
44	0x0000000*F*	0x00007*FF*4	105	0x00000013	0x0007*FFF*0
45	0x0000000*E*	0x00003*FF*9	106	0x00000013	0x0007*FFF*1
46	0x0000000*E*	0x00003*FF*7	107	0x00000013	0x0007*FFF*2
47	0x0000000*D*	0x00001*FFA*	108	0x00000013	0x0007*FFF*3
48	0x0000000*D*	0x00001*FF*9	109	0x00000013	0x0007*FFF*4
49	0x0000000*C*	0x00000*FFB*	110	0x00000013	0x0007*FFF*5
50	0x0000000*B*	0x000007*FC*	111	0x00000013	0x0007*FFF*6
51	0x0000000*A*	0x000003*FC*	112	0x00000013	0x0007*FFF*7
52	0x00000009	0x000001*FD*	113	0x00000013	0x0007*FFF*8
53	0x00000008	0x000000*FD*	114	0x00000013	0x0007*FFF*9
54	0x00000007	0x0000007*D*	115	0x00000013	0x0007*FFFA*
55	0x00000006	0x0000003*D*	116	0x00000013	0x0007*FFFB*
56	0x00000005	0x0000001*D*	117	0x00000013	0x0007*FFFC*
57	0x00000004	0x0000000*D*	118	0x00000013	0x0007*FFFD*
58	0x00000003	0x00000005	119	0x00000013	0x0007*FFFE*
59	0x00000002	0x00000001	120	0x00000013	0x0007*FFFF*
60	0x00000002	0x00000000			

Таблица А.80 — *f_huffman_ env*_1_5*dB*

Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)
0	0x00000013	0x0007*FFE7*	61	0x00000003	0x00000004
1	0x00000013	0x0007*FFE*8	62	0x00000004	0x0000000*D*
2	0x00000014	0x000*FFFD*2	63	0x00000005	0x0000001*D*
3	0x00000014	0x000*FFFD*3	64	0x00000006	0x0000003*D*
4	0x00000014	0x000*FFFD*4	65	0x00000008	0x000000*FA*
5	0x00000014	0x000*FFFD*5	66	0x00000008	0x000000*FC*
6	0x00000014	0x000*FFFD*6	67	0x00000009	0x000001*FB*
7	0x00000014	0x000*FFFD7*	68	0x0000000*A*	0x000003*FA*
8	0x00000014	0x000*FFFD*8	69	0x0000000*B*	0x000007*F*8
9	0x00000013	0x0007*FFDA*	70	0x0000000*B*	0x000007*FA*
10	0x00000014	0x000*FFFD*9	71	0x0000000*B*	0x000007*FB*
11	0x00000014	0x000*FFFDA*	72	0x0000000*C*	0x00000*FF*9
12	0x00000014	0x000*FFFDB*	73	0x0000000*C*	0x00000*FFB*
13	0x00000014	0x000*FFFDC*	74	0x0000000*D*	0x00001*FF8*
14	0x00000013	0x0007*FFDB*	75	0x0000000*D*	0x00001*FFB*
15	0x00000014	0x000*FFFDD*	76	0x0000000*E*	0x00003*FF*8
16	0x00000013	0x0007*FFDC*	77	0x0000000*E*	0x00003*FF*9
17	0x00000013	0x0007*FFDD*	78	0x00000010	0x0000*FFF*1
18	0x00000014	0x000*FFFDE*	79	0x00000010	0x0000*FFF*2

Окончание таблицы А.80

Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)
19	0x00000012	0x0003*FFE*4	80	0x00000011	0x0001*FFEA*
20	0x00000014	0x000*FFFDF*	81	0x00000011	0x0001*FFEB*
21	0x00000014	0x000*FFFE*0	82	0x00000012	0x0003*FFE*1
22	0x00000014	0x000*FFFE*1	83	0x00000012	0x0003*FFE*2
23	0x00000013	0x0007*FFDE*	84	0x00000012	0x0003*FFEA*
24	0x00000014	0x000*FFFE*2	85	0x00000012	0x0003*FFE*3
25	0x00000014	0x000*FFFE*3	86	0x00000012	0x0003*FFE*6
26	0x00000014	0x000*FFFE*4	87	0x00000012	0x0003*FFE*7
27	0x00000013	0x0007*FFDF*	88	0x00000012	0x0003*FFEB*
28	0x00000014	0x000*FFFE*5	89	0x00000014	0x000*FFFE*6
29	0x00000013	0x0007*FFE*0	90	0x00000013	0x0007*FFE*2
30	0x00000012	0x0003*FFE*8	91	0x00000014	0x000*FFFE*7
31	0x00000013	0x0003*FFE*1	92	0x00000014	0x000*FFFE*8
32	0x00000012	0x0003*FFE*0	93	0x00000014	0x000*FFFE*9
33	0x00000012	0x0003*FFE*9	94	0x00000014	0x000*FFFEA*
34	0x00000011	0x0001*FFEF*	95	0x00000014	0x000*FFFEB*
35	0x00000012	0x0003*FFE*5	96	0x00000014	0x000*FFFEC*
36	0x00000011	0x0001*FFEC*	97	0x00000013	0x0007*FFE*3
37	0x00000011	0x0001*FFED*	98	0x00000014	0x000*FFFED*
38	0x00000011	0x0001*FFEE*	99	0x00000014	0x000*FFFEE*
39	0x00000010	0x0000*FFF*4	100	0x00000014	0x000*FFFEF*
40	0x00000010	0x0000*FFF*3	101	0x00000014	0x000*FFFF*0
41	0x00000010	0x0000*FFF*0	102	0x00000013	0x0007*FFE*4
42	0x0000000*F*	0x00007*FF*7	103	0x00000014	0x000*FFFF*1
43	0x0000000*F*	0x00007*FF*6	104	0x00000012	0x0003*FFEC*
44	0x0000000*E*	0x00003*FFA*	105	0x00000014	0x000*FFFF*2
45	0x0000000*D*	0x00001*FFA*	106	0x00000014	0x000*FFFF*3
46	0x0000000*D*	0x00001*FF*9	107	0x00000013	0x0007*FFE*5
47	0x0000000*C*	0x00000*FFA*	108	0x00000013	0x0007*FFE*6
48	0x0000000*C*	0x00000*FF*8	109	0x00000014	0x000*FFFF*4
49	0x0000000*B*	0x000007*F*9	110	0x00000014	0x000*FFFF*5
50	0x0000000*A*	0x000003*FB*	111	0x00000014	0x000*FFFF*6
51	0x00000009	0x000001*FC*	112	0x00000014	0x000*FFFF*7
52	0x00000009	0x000001*FA*	113	0x00000014	0x000*FFFF*8
53	0x00000008	0x000000*FB*	114	0x00000014	0x000*FFFF*9
54	0x00000007	0x0000007*C*	115	0x00000014	0x000*FFFFA*
55	0x00000006	0x0000003*C*	116	0x00000014	0x000*FFFFB*
56	0x00000005	0x0000001*C*	117	0x00000014	0x000*FFFFC*
57	0x00000004	0x0000000*C*	118	0x00000014	0x000*FFFFD*
58	0x00000003	0x00000005	119	0x00000014	0x000*FFFFE*
59	0x00000002	0x00000001	120	0x00000014	0x000*FFFFF*
60	0x00000002	0x00000000			

Таблица А.81 — *t_huffman_ env_bal_*1_5*Db*

Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)
0	0x00000010	0x0000*FFE*4	25	0x00000002	0x00000002
1	0x00000010	0x0000*FFE*5	26	0x00000004	0x0000000*E*
2	0x00000010	0x0000*FFE*6	27	0x00000006	0x0000003*E*
3	0x00000010	0x0000*FFE*7	28	0x00000008	0x000000*FE*
4	0x00000010	0x0000*FFE*8	29	0x0000000*B*	0x000007*FD*
5	0x00000010	0x0000*FFE*9	30	0x0000000*C*	0x00000*FFD*
6	0x00000010	0x0000*FFEA*	31	0x0000000*F*	0x00007*FF*0
7	0x00000010	0x0000*FFEB*	32	0x00000010	0x0000*FFE*3
8	0x00000010	0x0000*FFEC*	33	0x00000010	0x0000*FFF*5
9	0x00000010	0x0000*FFED*	34	0x00000010	0x0000*FFF*6
10	0x00000010	0x0000*FFEE*	35	0x00000010	0x0000*FFF*7
11	0x00000010	0x0000*FFEF*	36	0x00000010	0x0000*FFF*8
12	0x00000010	0x0000*FFF*0	37	0x00000010	0x0000*FFF*9
13	0x00000010	0x0000*FFF*1	38	0x00000010	0x0000*FFFA*
14	0x00000010	0x0000*FFF*2	39	0x00000011	0x0001*FFF*6
15	0x00000010	0x0000*FFF*3	40	0x00000011	0x0001*FFF*7
16	0x00000010	0x0000*FFF*4	41	0x00000011	0x0001*FFF*8
17	0x00000010	0x0000*FFE*2	42	0x00000011	0x0001*FFF*9
18	0x0000000*C*	0x00000*FFC*	43	0x00000011	0x0001*FFFA*
19	0x0000000*B*	0x000007*FC*	44	0x00000011	0x0001*FFFB*
20	0x00000009	0x000001*FE*	45	0x00000011	0x0001*FFFC*
21	0x00000007	0x0000007*E*	46	0x00000011	0x0001*FFFD*
22	0x00000005	0x0000001*E*	47	0x00000011	0x0001*FFFE*
23	0x00000003	0x00000006	48	0x00000011	0x0001*FFFF*
24	0x00000001	0x00000000			

Таблица А.82 — *f_huffman_ env_bal*_1_5*dB*

Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)
0	0x00000012	0x0003*FFE*2	25	0x00000003	0x00000006
1	0x00000012	0x0003*FFE*3	26	0x00000005	0x0000001*E*
2	0x00000012	0x0003*FFE*4	27	0x00000006	0x0000003*E*
3	0x00000012	0x0003*FFE*5	28	0x00000009	0x000001*FE*
4	0x00000012	0x0003*FFE*6	29	0x0000000B	0x000007*FD*
5	0x00000012	0x0003*FFE*7	30	0x0000000C	0x00000*FFE*
6	0x00000012	0x0003*FFE*8	31	0x0000000F	0x00007*FFA*
7	0x00000012	0x0003*FFE*9	32	0x00000010	0x0000*FFF*6
8	0x00000012	0x0003*FFEA*	33	0x00000012	0x0003*FFF*1
9	0x00000012	0x0003*FFEB*	34	0x00000012	0x0003*FFF*2
10	0x00000012	0x0003*FFEC*	35	0x00000012	0x0003*FFF*3
11	0x00000012	0x0003*FFED*	36	0x00000012	0x0003*FFF*4
12	0x00000012	0x0003*FFEE*	37	0x00000012	0x0003*FFF*5
13	0x00000012	0x0003*FFEF*	38	0x00000012	0x0003*FFF*6
14	0x00000012	0x0003*FFF*0	39	0x00000012	0x0003*FFF*7
15	0x00000010	0x0000*FFF*7	40	0x00000012	0x0003*FFF*8
16	0x00000011	0x0001*FFF*0	41	0x00000012	0x0003*FFF*9
17	0x0000000*E*	0x00003*FFC*	42	0x00000012	0x0003*FFFA*
18	0x0000000*B*	0x000007*FE*	43	0x00000012	0x0003*FFFB*
19	0x0000000*B*	0x000007*FC*	44	0x00000012	0x0003*FFFC*
20	0x00000008	0x000000*FE*	45	0x00000012	0x0003*FFFD*
21	0x00000007	0x0000007*E*	46	0x00000012	0x0003*FFFE*
22	0x00000004	0x0000000*E*	47	0x00000013	0x0007*FFFE*
23	0x00000002	0x00000002	48	0x00000013	0x0007*FFFF*
24	0x00000001	0x00000000			

Таблица А.83 — *t_huffman_ env*_3_0*dB*

Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)
0	0x00000012	0x0003*FFED*	32	0x00000003	0x00000006
1	0x00000012	0x0003*FFEE*	33	0x00000005	0x0000001*E*
2	0x00000013	0x0007*FFDE*	34	0x00000007	0x0000007*E*
3	0x00000013	0x0007*FFDF*	35	0x00000009	0x000001*FE*
4	0x00000013	0x0007*FFE*0	36	0x0000000*B*	0x000007*FD*
5	0x00000013	0x0007*FFE*1	37	0x0000000*D*	0x00001*FFB*
6	0x00000013	0x0007*FFE*2	38	0x0000000*E*	0x00003*FF*9
7	0x00000013	0x0007*FFE*3	39	0x0000000*E*	0x00003*FFC*
8	0x00000013	0x0007*FFE*4	40	0x0000000*F*	0x00007*FFA*
9	0x00000013	0x0007*FFE*5	41	0x00000010	0x0000*FFF*6
10	0x00000013	0x0007*FFE*6	42	0x00000011	0x0001*FFF*5
11	0x00000013	0x0007*FFE*7	43	0x00000012	0x0003*FFEC*
12	0x00000013	0x0007*FFE*8	44	0x00000013	0x0007*FFED*
13	0x00000013	0x0007*FFE*9	45	0x00000013	0x0007*FFEE*
14	0x00000013	0x0007*FFEA*	46	0x00000013	0x0007*FFEF*
15	0x00000013	0x0007*FFEB*	47	0x00000013	0x0007*FFF*0
16	0x00000013	0x0007*FFEC*	48	0x00000013	0x0007*FFF*1
17	0x00000011	0x0001*FFF*4	49	0x00000013	0x0007*FFF*2
18	0x00000010	0x0000*FFF*7	50	0x00000013	0x0007*FFF*3
19	0x00000010	0x0000*FFF*9	51	0x00000013	0x0007*FFF*4
20	0x00000010	0x0000*FFF*8	52	0x00000013	0x0007*FFF*5
21	0x0000000*E*	0x00003*FFB*	53	0x00000013	0x0007*FFF*6
22	0x0000000*E*	0x00003*FFA*	54	0x00000013	0x0007*FFF*7
23	0x0000000*E*	0x00003*FF*8	55	0x00000013	0x0007*FFF*8
24	0x0000000*D*	0x00001*FFA*	56	0x00000013	0x0007*FFF*9
25	0x0000000*C*	0x00000*FFC*	57	0x00000013	0x0007*FFFA*
26	0x0000000*B*	0x000007*FC*	58	0x00000013	0x0007*FFFB*
27	0x00000008	0x000000*FE*	59	0x00000013	0x0007*FFFC*
28	0x00000006	0x0000003*E*	60	0x00000013	0x0007*FFFD*
29	0x00000004	0x0000000*E*	61	0x00000013	0x0007*FFFE*
30	0x00000002	0x00000002	62	0x00000013	0x0007*FFFF*
31	0x00000001	0x00000000		0x00000003	0x00000006

Таблица А.84 — *f_huffman_ env*_3_0*dB*

Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)
0	0x00000014	0x000*FFFF*0	32	0x00000003	0x00000006
1	0x00000014	0x000*FFFF*1	33	0x00000005	0x0000001*E*
2	0x00000014	0x000*FFFF*2	34	0x00000008	0x000000*FC*
3	0x00000014	0x000*FFFF*3	35	0x00000009	0x000001*FC*
4	0x00000014	0x000*FFFF*4	36	0x0000000*A*	0x000003*FC*
5	0x00000014	0x000*FFFF*5	37	0x0000000*B*	0x000007*FC*
6	0x00000014	0x000*FFFF*6	38	0x0000000*C*	0x00000*FFC*
7	0x00000012	0x0003*FFF*3	39	0x0000000*D*	0x00001*FFC*
8	0x00000013	0x0007*FFF*5	40	0x0000000*E*	0x00003*FFA*
9	0x00000013	0x0007*FFEE*	41	0x0000000*F*	0x00007*FF*9
10	0x00000013	0x0007*FFEF*	42	0x0000000*F*	0x00007*FFA*
11	0x00000013	0x0007*FFF*6	43	0x00000010	0x0000*FFF*8
12	0x00000012	0x0003*FFF*4	44	0x00000010	0x0000*FFF*9
13	0x00000012	0x0003*FFF*2	45	0x00000011	0x0001*FFF*6
14	0x00000014	0x000*FFFF*7	46	0x00000011	0x0001*FFF*7
15	0x00000013	0x0007*FFF*0	47	0x00000012	0x0003*FFF*5
16	0x00000011	0x0001*FFF*5	48	0x00000012	0x0003*FFF*6
17	0x00000012	0x0003*FFF*0	49	0x00000012	0x0003*FFF*1
18	0x00000011	0x0001*FFF*4	50	0x00000014	0x000*FFFF*8
19	0x00000010	0x0000*FFF*7	51	0x00000013	0x0007*FFF*1
20	0x00000010	0x0000*FFF*6	52	0x00000013	0x0007*FFF*2
21	0x0000000*F*	0x00007*FF*8	53	0x00000013	0x0007*FFF*3
22	0x0000000*E*	0x00003*FFB*	54	0x00000014	0x000*FFFF*9
23	0x0000000*C*	0x00000*FFD*	55	0x00000013	0x0007*FFF*7
24	0x0000000*B*	0x000007*FD*	56	0x00000013	0x0007*FFF*4
25	0x0000000*A*	0x000003*FD*	57	0x00000014	0x000*FFFFA*
26	0x00000009	0x000001*FD*	58	0x00000014	0x000*FFFFB*
27	0x00000008	0x000000*FD*	59	0x00000014	0x000*FFFFC*
28	0x00000006	0x0000003*E*	60	0x00000014	0x000*FFFFD*
29	0x00000004	0x0000000*E*	61	0x00000014	0x000*FFFFE*
30	0x00000002	0x00000002	62	0x00000014	0x000*FFFFF*
31	0x00000001	0x00000000			

Таблица А.85 — *t_huffman_ env_bal_3_0dB*

Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)
0	0x0000000*D*	0x00001*FF*2	32	0x00000002	0x00000002
1	0x0000000*D*	0x00001*FF*3	33	0x00000005	0x0000001*E*
2	0x0000000*D*	0x00001*FF*4	34	0x00000006	0x0000003*E*
3	0x0000000*D*	0x00001*FF*5	35	0x00000009	0x000001*FE*
4	0x0000000*D*	0x00001*FF*6	36	0x0000000*D*	0x00001*FF*9
5	0x0000000*D*	0x00001*FF*7	37	0x0000000*D*	0x00001*FFA*
6	0x0000000*D*	0x00001*FF*8	38	0x0000000*D*	0x00001*FFB*
7	0x0000000*C*	0x00000*FF*8	39	0x0000000*D*	0x00001*FFC*
8	0x00000008	0x000000*FE*	40	0x0000000*D*	0x00001*FFD*
9	0x00000007	0x0000007*E*	41	0x0000000*D*	0x00001*FFE*
10	0x00000004	0x0000000*E*	42	0x0000000*E*	0x00003*FFE*
11	0x00000003	0x00000006	43	0x0000000*E*	0x00003*FFF*
12	0x00000001	0x00000000	44		

Таблица А.86 — *f_huffman_ env_bal_3_0dB*

Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)
0	0x0000000*D*	0x00001*FF*7	32	0x00000003	0x00000006
1	0x0000000*D*	0x00001*FF*8	33	0x00000005	0x0000001*E*
2	0x0000000*D*	0x00001*FF*9	34	0x00000006	0x0000003*E*
3	0x0000000*D*	0x00001*FFA*	35	0x00000009	0x000001*FE*
4	0x0000000*D*	0x00001*FFB*	36	0x0000000*C*	0x00000*FFA*
5	0x0000000*E*	0x00003*FF*8	37	0x0000000*D*	0x00001*FF*6
6	0x0000000*E*	0x00003*FF*9	38	0x0000000*E*	0x00003*FFA*
7	0x0000000*B*	0x000007*FC*	39	0x0000000*E*	0x00003*FFB*
8	0x00000008	0x000000*FE*	40	0x0000000*E*	0x00003*FFC*
9	0x00000007	0x0000007*E*	41	0x0000000*E*	0x00003*FFD*
10	0x00000004	0x0000000*E*	42	0x0000000*E*	0x00003*FFE*
11	0x00000002	0x00000002	43	0x0000000*E*	0x00003*FFF*
12	0x00000001	0x00000000	44		

Таблица А.87 — *t_huffman_ nouse*_3_0*dB*

Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)
0	0x0000000*D*	0x00001*FCE*	32	0x00000002	0x00000002
1	0x0000000*D*	0x00001*FCF*	33	0x00000005	0x0000001*E*
2	0x0000000*D*	0x00001*FD*0	34	0x00000008	0x000000*FC*
3	0x0000000*D*	0x00001*FD*1	35	0x0000000*A*	0x000003*F*8
4	0x0000000*D*	0x00001*FD*2	36	0x0000000*D*	0x00001*FCC*
5	0x0000000*D*	0x00001*FD*3	37	0x0000000*D*	0x00001*FE*8
6	0x0000000*D*	0x00001*FD*4	38	0x0000000*D*	0x00001*FE*9
7	0x0000000*D*	0x00001*FD*5	39	0x0000000*D*	0x00001*FEA*
8	0x0000000*D*	0x00001*FD*6	40	0x0000000*D*	0x00001*FEB*
9	0x0000000*D*	0x00001*FD*7	41	0x0000000*D*	0x00001*FEC*
10	0x0000000*D*	0x00001*FD*8	42	0x0000000*D*	0x00001*FCD*
11	0x0000000*D*	0x00001*FD*9	43	0x0000000*D*	0x00001*FED*
12	0x0000000*D*	0x00001*FDA*	44	0x0000000*D*	0x00001*FEE*
13	0x0000000*D*	0x00001*FDB*	45	0x0000000*D*	0x00001*FEF*
14	0x0000000*D*	0x00001*FDC*	46	0x0000000*D*	0x00001*FF*0
15	0x0000000*D*	0x00001*FDD*	47	0x0000000*D*	0x00001*FF*1
16	0x0000000*D*	0x00001*FDE*	48	0x0000000*D*	0x00001*FF*2
17	0x0000000*D*	0x00001*FDF*	49	0x0000000*D*	0x00001*FF*3
18	0x0000000*D*	0x00001*FE*0	50	0x0000000*D*	0x00001*FF*4
19	0x0000000*D*	0x00001*FE*1	51	0x0000000*D*	0x00001*FF*5
20	0x0000000*D*	0x00001*FE*2	52	0x0000000*D*	0x00001*FF*6
21	0x0000000*D*	0x00001*FE*3	53	0x0000000*D*	0x00001*FF*7
22	0x0000000*D*	0x00001*FE*4	54	0x0000000*D*	0x00001*FF*8
23	0x0000000*D*	0x00001*FE*5	55	0x0000000*D*	0x00001*FF*9
24	0x0000000*D*	0x00001*FE*6	56	0x0000000*D*	0x00001*FFA*
25	0x0000000*D*	0x00001*FE*7	57	0x0000000*D*	0x00001*FFB*
26	0x0000000*B*	0x000007*F*2	58	0x0000000*D*	0x00001*FFC*
27	0x00000008	0x000000*FD*	59	0x0000000*D*	0x00001*FFD*
28	0x00000006	0x0000003*E*	60	0x0000000*D*	0x00001*FFE*
29	0x00000004	0x0000000*E*	61	0x0000000*E*	0x00003*FFE*
30	0x00000003	0x00000006	62	0x0000000*E*	0x00003*FFF*
31	0x00000001	0x00000000			

Таблица А.88 — *t_huffman_ nouse_bal_3_0dB*

Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)	Индекс	Длина (шестнадцатеричная)	Кодовая комбинация (шестнадцатеричная)
0	0x00000008	0x000000*EC*	32	0x00000003	0x00000006
1	0x00000008	0x000000*ED*	33	0x00000006	0x0000003*A*
2	0x00000008	0x000000*EE*	34	0x00000008	0x000000*F6*
3	0x00000008	0x000000*EF*	35	0x00000008	0x000000*F7*
4	0x00000008	0x000000*F0*	36	0x00000008	0x000000*F8*
5	0x00000008	0x000000*F1*	37	0x00000008	0x000000*F9*
6	0x00000008	0x000000*F2*	38	0x00000008	0x000000*FA*
7	0x00000008	0x000000*F3*	39	0x00000008	0x000000*FB*
8	0x00000008	0x000000*F4*	40	0x00000008	0x000000*FC*
9	0x00000008	0x000000*F5*	41	0x00000008	0x000000*FD*
10	0x00000005	0x0000001*C*	42	0x00000008	0x000000*FE*
11	0x00000002	0x00000002	43	0x00000008	0x000000*FF*
12	0x00000001	0x00000000	44		

Смешанные таблицы *SBR*

Таблица А.89 — Коэффициенты *c[i]* окна банка *QMF*

i	*c[i]*	*i*	*c[i]*	*i*	*c[i]*
0	0,0000000000	214	0,0019765601	428	0,0117623832
1	–0,0005525286	215	–0,0032086896	429	0,0163701258
2	–0,0005617692	216	–0,0085711749	430	0,0207997072
3	–0,0004947518	217	–0,0141288827	431	0,0250307561
4	–0,0004875227	218	–0,0198834129	432	0,0290824006
5	–0,0004893791	219	–0,0258227288	433	0,0329583930
6	–0,0005040714	220	–0,0319531274	434	0,0366418116
7	–0,0005226564	221	–0,0382776572	435	0,0401458278
8	–0,0005466565	222	–0,0447806821	436	0,0434768782
9	–0,0005677802	223	–0,0514804176	437	0,0466303305
10	–0,0005870930	224	–0,0583705326	438	0,0495978676
11	–0,0006132747	225	–0,0654409853	439	0,0524093821
12	–0,0006312493	226	–0,0726943300	440	0,0550460034
13	–0,0006540333	227	–0,0801372934	441	0,0575152691
14	–0,0006777690	228	–0,0877547536	442	0,0598166570
15	–0,0006941614	229	–0,0955533352	443	0,0619602779
16	–0,0007157736	230	–0,1035329531	444	0,0639444805
17	–0,0007255043	231	–0,1116826931	445	0,0657690668
18	–0,0007440941	232	–0,1200077984	446	0,0674525021
19	–0,0007490598	233	–0,1285002850	447	0,0689664013
20	–0,0007681371	234	–0,1371551761	448	0,0703533073
21	–0,0007724848	235	–0,1459766491	449	0,0715826364

Продолжение таблицы А.89

i	*c[i]*	*i*	*c[i]*	*i*	*c[i]*
22	–0,0007834332	236	–0,1549607071	450	0,0726774642
23	–0,0007779869	237	–0,1640958855	451	0,0736406005
24	–0,0007803664	238	–0,1733808172	452	0,0744664394
25	–0,0007801449	239	–0,1828172548	453	0,0751576255
26	–0,0007757977	240	–0,1923966745	454	0,0757305756
27	–0,0007630793	241	–0,2021250176	455	0,0761748321
28	–0,0007530001	242	–0,2119735853	456	0,0765050718
29	–0,0007319357	243	–0,2219652696	457	0,0767204924
30	–0,0007215391	244	–0,2320690870	458	0,0768230011
31	–0,0006917937	245	–0,2423016884	459	0,0768173975
32	–0,0006650415	246	–0,2526480309	460	0,0767093490
33	–0,0006341594	247	–0,2631053299	461	0,0764992170
34	–0,0005946118	248	–0,2736634040	462	0,0761992479
35	–0,0005564576	249	–0,2843214189	463	0,0758008358
36	–0,0005145572	250	–0,2950716717	464	0,0753137336
37	–0,0004606325	251	–0,3059098575	465	0,0747452558
38	–0,0004095121	252	–0,3168278913	466	0,0741003642
39	–0,0003501175	253	–0,3278113727	467	0,0733620255
40	–0,0002896981	254	–0,3388722693	468	0,0725682583
41	–0,0002098337	255	–0,3499914122	469	0,0717002673
42	–0,0001446380	256	0,3611589903	470	0,0707628710
43	–0,0000617334	257	0,3723795546	471	0,0697630244
44	0,0000134949	258	0,3836350013	472	0,0687043828
45	0,0001094383	259	0,3949211761	473	0,0676075985
46	0,0002043017	260	0,4062317676	474	0,0664367512
47	0,0002949531	261	0,4175696896	475	0,0652247106
48	0,0004026540	262	0,4289119920	476	0,0639715898
49	0,0005107388	263	0,4402553754	477	0,0626857808
50	0,0006239376	264	0,4515996535	478	0,0613455171
51	0,0007458025	265	0,4629308085	479	0,0599837480
52	0,0008608443	266	0,4742453214	480	0,0585915683
53	0,0009885988	267	0,4855253091	481	0,0571616450
54	0,0011250155	268	0,4967708254	482	0,0557173648
55	0,0012577884	269	0,5079817500	483	0,0542452768
56	0,0013902494	270	0,5191234970	484	0,0527630746
57	0,0015443219	271	0,5302240895	485	0,0512556155
58	0,0016868083	272	0,5412553448	486	0,0497385755
59	0,0018348265	273	0,5522051258	487	0,0482165720

Продолжение таблицы А.89

i	*c[i]*	*i*	*c[i]*	*i*	*c[i]*
60	0,0019841140	274	0,5630789140	488	0,0466843027
61	0,0021461583	275	0,5738524131	489	0,0451488405
62	0,0023017254	276	0,5845403235	490	0,0436097542
63	0,0024625616	277	0,5951123086	491	0,0420649094
64	0,0026201758	278	0,6055783538	492	0,0405349170
65	0,0027870464	279	0,6159109932	493	0,0390053679
66	0,0029469447	280	0,6261242695	494	0,0374812850
67	0,0031125420	281	0,6361980107	495	0,0359697560
68	0,0032739613	282	0,6461269695	496	0,0344620948
69	0,0034418874	283	0,6559016302	497	0,0329754081
70	0,0036008268	284	0,6655139880	498	0,0315017608
71	0,0037603922	285	0,6749663190	499	0,0300502657
72	0,0039207432	286	0,6842353293	500	0,0286072173
73	0,0040819753	287	0,6933282376	501	0,0271859429
74	0,0042264269	288	0,7022388719	502	0,0257875847
75	0,0043730719	289	0,7109410426	503	0,0244160992
76	0,0045209852	290	0,7194462634	504	0,0230680169
77	0,0046606460	291	0,7277448900	505	0,0217467550
78	0,0047932560	292	0,7358211758	506	0,0204531793
79	0,0049137603	293	0,7436827863	507	0,0191872431
80	0,0050393022	294	0,7513137456	508	0,0179433381
81	0,0051407353	295	0,7587080760	509	0,0167324712
82	0,0052461166	296	0,7658674865	510	0,0155405553
83	0,0053471681	297	0,7727780881	511	0,0143904666
84	0,0054196775	298	0,7794287519	512	–0,0132718220
85	0,0054876040	299	0,7858353120	513	–0,0121849995
86	0,0055475714	300	0,7919735841	514	–0,0111315548
87	0,0055938023	301	0,7978466413	515	–0,0101150215
88	0,0056220643	302	0,8034485751	516	–0,0091325329
89	0,0056455196	303	0,8087695004	517	–0,0081798233
90	0,0056389199	304	0,8138191270	518	–0,0072615816
91	0,0056266114	305	0,8185776004	519	–0,0063792293
92	0,0055917128	306	0,8230419890	520	–0,0055337211
93	0,0055404363	307	0,8272275347	521	–0,0047222596
94	0,0054753783	308	0,8311038457	522	–0,0039401124
95	0,0053838975	309	0,8346937361	523	–0,0031933778
96	0,0052715758	310	0,8379717337	524	–0,0024826723
97	0,0051382275	311	0,8409541392	525	–0,0018039472

Продолжение таблицы А.89

i	*c[i]*	*i*	*c[i]*	*i*	*c[i]*
98	0,0049839687	312	0,8436238281	526	–0,0011568135
99	0,0048109469	313	0,8459818469	527	–0,0005464280
100	0,0046039530	314	0,8480315777	528	0,0000276045
101	0,0043801861	315	0,8497805198	529	0,0005832264
102	0,0041251642	316	0,8511971524	530	0,0010902329
103	0,0038456408	317	0,8523047035	531	0,0015784682
104	0,0035401246	318	0,8531020949	532	0,0020274176
105	0,0032091885	319	0,8535720573	533	0,0024508540
106	0,0028446757	320	0,8537385600	534	0,0028446757
107	0,0024508540	321	0,8535720573	535	0,0032091885
108	0,0020274176	322	0,8531020949	536	0,0035401246
109	0,0015784682	323	0,8523047035	537	0,0038456408
110	0,0010902329	324	0,8511971524	538	0,0041251642
111	0,0005832264	325	0,8497805198	539	0,0043801861
112	0,0000276045	326	0,8480315777	540	0,0046039530
113	–0,0005464280	327	0,8459818469	541	0,0048109469
114	–0,0011568135	328	0,8436238281	542	0,0049839687
115	–0,0018039472	329	0,8409541392	543	0,0051382275
116	–0,0024826723	330	0,8379717337	544	0,0052715758
117	–0,0031933778	331	0,8346937361	545	0,0053838975
118	–0,0039401124	332	0,8311038457	546	0,0054753783
119	–0,0047222596	333	0,8272275347	547	0,0055404363
120	–0,0055337211	334	0,8230419890	548	0,0055917128
121	–0,0063792293	335	0,8185776004	549	0,0056266114
122	–0,0072615816	336	0,8138191270	550	0,0056389199
123	–0,0081798233	337	0,8087695004	551	0,0056455196
124	–0,0091325329	338	0,8034485751	552	0,0056220643
125	–0,0101150215	339	0,7978466413	553	0,0055938023
126	–0,0111315548	340	0,7919735841	554	0,0055475714
127	–0,0121849995	341	0,7858353120	555	0,0054876040
128	0,0132718220	342	0,7794287519	556	0,0054196775
129	0,0143904666	343	0,7727780881	557	0,0053471681
130	0,0155405553	344	0,7658674865	558	0,0052461166
131	0,0167324712	345	0,7587080760	559	0,0051407353
132	0,0179433381	346	0,7513137456	560	0,0050393022
133	0,0191872431	347	0,7436827863	561	0,0049137603
134	0,0204531793	348	0,7358211758	562	0,0047932560
135	0,0217467550	349	0,7277448900	563	0,0046606460

Продолжение таблицы А.89

i	*c[i]*	*i*	*c[i]*	*i*	*c[i]*
136	0,0230680169	350	0,7194462634	564	0,0045209852
137	0,0244160992	351	0,7109410426	565	0,0043730719
138	0,0257875847	352	0,7022388719	566	0,0042264269
139	0,0271859429	353	0,6933282376	567	0,0040819753
140	0,0286072173	354	0,6842353293	568	0,0039207432
141	0,0300502657	355	0,6749663190	569	0,0037603922
142	0,0315017608	356	0,6655139880	570	0,0036008268
143	0,0329754081	357	0,6559016302	571	0,0034418874
144	0,0344620948	358	0,6461269695	572	0,0032739613
145	0,0359697560	359	0,6361980107	573	0,0031125420
146	0,0374812850	360	0,6261242695	574	0,0029469447
147	0,0390053679	361	0,6159109932	575	0,0027870464
148	0,0405349170	362	0,6055783538	576	0,0026201758
149	0,0420649094	363	0,5951123086	577	0,0024625616
150	0,0436097542	364	0,5845403235	578	0,0023017254
151	0,0451488405	365	0,5738524131	579	0,0021461583
152	0,0466843027	366	0,5630789140	580	0,0019841140
153	0,0482165720	367	0,5522051258	581	0,0018348265
154	0,0497385755	368	0,5412553448	582	0,0016868083
155	0,0512556155	369	0,5302240895	583	0,0015443219
156	0,0527630746	370	0,5191234970	584	0,0013902494
157	0,0542452768	371	0,5079817500	585	0,0012577884
158	0,0557173648	372	0,4967708254	586	0,0011250155
159	0,0571616450	373	0,4855253091	587	0,0009885988
160	0,0585915683	374	0,4742453214	588	0,0008608443
161	0,0599837480	375	0,4629308085	589	0,0007458025
162	0,0613455171	376	0,4515996535	590	0,0006239376
163	0,0626857808	377	0,4402553754	591	0,0005107388
164	0,0639715898	378	0,4289119920	592	0,0004026540
165	0,0652247106	379	0,4175696896	593	0,0002949531
166	0,0664367512	380	0,4062317676	594	0,0002043017
167	0,0676075985	381	0,3949211761	595	0,0001094383
168	0,0687043828	382	0,3836350013	596	0,0000134949
169	0,0697630244	383	0,3723795546	597	–0,0000617334
170	0,0707628710	384	–0,3611589903	598	–0,0001446380
171	0,0717002673	385	–0,3499914122	599	–0,0002098337
172	0,0725682583	386	–0,3388722693	600	–0,0002896981
173	0,0733620255	387	–0,3278113727	601	–0,0003501175
174	0,0741003642	388	–0,3168278913	602	–0,0004095121

Окончание таблицы А.89

i	*c[i]*	*i*	*c[i]*	*i*	*c[i]*
175	0,0747452558	389	–0,3059098575	603	–0,0004606325
176	0,0753137336	390	–0,2950716717	604	–0,0005145572
177	0,0758008358	391	–0,2843214189	605	–0,0005564576
178	0,0761992479	392	–0,2736634040	606	–0,0005946118
179	0,0764992170	393	–0,2631053299	607	–0,0006341594
180	0,0767093490	394	–0,2526480309	608	–0,0006650415
181	0,0768173975	395	–0,2423016884	609	–0,0006917937
182	0,0768230011	396	–0,2320690870	610	–0,0007215391
183	0,0767204924	397	–0,2219652696	611	–0,0007319357
184	0,0765050718	398	–0,2119735853	612	–0,0007530001
185	0,0761748321	399	–0,2021250176	613	–0,0007630793
186	0,0757305756	400	–0,1923966745	614	–0,0007757977
187	0,0751576255	401	–0,1828172548	615	–0,0007801449
188	0,0744664394	402	–0,1733808172	616	–0,0007803664
189	0,0736406005	403	–0,1640958855	617	–0,0007779869
190	0,0726774642	404	–0,1549607071	618	–0,0007834332
191	0,0715826364	405	–0,1459766491	619	–0,0007724848
192	0,0703533073	406	–0,1371551761	620	–0,0007681371
193	0,0689664013	407	–0,1285002850	621	–0,0007490598
194	0,0674525021	408	–0,1200077984	622	–0,0007440941
195	0,0657690668	409	–0,1116826931	623	–0,0007255043
196	0,0639444805	410	–0,1035329531	624	–0,0007157736
197	0,0619602779	411	–0,0955533352	625	–0,0006941614
198	0,0598166570	412	–0,0877547536	626	–0,0006777690
199	0,0575152691	413	–0,0801372934	627	–0,0006540333
200	0,0550460034	414	–0,0726943300	628	–0,0006312493
201	0,0524093821	415	–0,0654409853	629	–0,0006132747
202	0,0495978676	416	–0,0583705326	630	–0,0005870930
203	0,0466303305	417	–0,0514804176	631	–0,0005677802
204	0,0434768782	418	–0,0447806821	632	–0,0005466565
205	0,0401458278	419	–0,0382776572	633	–0,0005226564
206	0,0366418116	420	–0,0319531274	634	–0,0005040714
207	0,0329583930	421	–0,0258227288	635	–0,0004893791
208	0,0290824006	422	–0,0198834129	636	–0,0004875227
209	0,0250307561	423	–0,0141288827	637	–0,0004947518
210	0,0207997072	424	–0,0085711749	638	–0,0005617692
211	0,0163701258	425	–0,0032086896	639	–0,000552528
212	0,0117623832	426	0,0019765601		
213	0,0069636862	427	0,0069636862		

Таблица А.90 — Коэффициенты *c[i]* окна *CLDFB*

i	*c[i]*	*i*	*c[i]*	*i*	*c[i]*	*i*	*c[i]*
0	1,129580193872797e–002	160	–2,726484300186575e–002	320	–3,597475533950260e–003	480	2,666444630171025e–004
1	2,353059744904218e–002	161	–1,711323992709580e–002	321	–3,318857985461042e–003	481	7,872592352725688e–005
2	3,450718748721251e–002	162	–7,298197371320593e–003	322	–3,000422543655664e–003	482	7,095886893185526e–005
3	4,634695977000525e–002	163	2,184256929356781e–003	323	–2,658042081080524e–003	483	5,643103068471008e–005
4	5,918677345174197e–002	164	1,132324047372148e–002	324	–2,292813563887493e–003	484	6,904415362098980e–005
5	7,325978412117062e–002	165	2,012236990754980e–002	325	–1,914114740669928e–003	485	4,694251739991356e–005
6	8,829745229234007e–002	166	2,857528272530154e–002	326	–1,525818616748839e–003	486	3,367998338617662e–005
7	1,042033024802571e–001	167	3,666942822678171e–002	327	–1,156680209049319e–003	487	6,481921021601837e–005
8	1,206924277410051e–001	168	4,439683978044157e–002	328	–7,804546272743493e–004	488	6,582328030188790e–005
9	1,376149808913910e–001	169	5,177964768870787e–002	329	–4,268574601396473e–004	489	–4,256442530773449e–005
10	1,547461142258783e–001	170	5,881296711410786e–002	330	–1,324291707264515e–004	490	4,939392400898679e–005
11	1,719726384566089e–001	171	6,550209046893848e–002	331	1,218226450050751e–004	491	5,272982009116034e–005
12	1,891590407342011e–001	172	7,184073822817207e–002	332	3,189336138130849e–004	492	4,005269212731273e–005
13	2,062605107774960e–001	173	7,783299328224960e–002	333	4,749931197951235e–004	493	2,461876679726978e–005
14	2,232276864673650e–001	174	8,347150698567406e–002	334	5,970696819774243e–004	494	4,469729032194765e–006
15	2,400768261284114e–001	175	8,875756217893037e–002	335	6,673250213055329e–004	495	3,798519731621893e–007
16	2,568176309566753e–001	176	9,368651761350569e–002	336	6,887783835812338e–004	496	1,374896222030490e–006
17	2,734977190313227e–001	177	9,826251129465624e–002	337	6,766320515830324e–004	497	3,965363805500215e–006
18	2,901491317310591e–001	178	1,024804711677230e–001	338	6,944123176012471e–004	498	7,300588863934780e–006
19	3,068186515423912e–001	179	1,063454554357498e–001	339	7,139919634325070e–004	499	1,168894474770061e–005
20	3,235298682841570e–001	180	1,098551252869576e–001	340	7,154123487609100e–004	500	8,563819899447630e–006
21	3,403074146062977e–001	181	1,130180022553412e–001	341	7,376101027486600e–004	501	8,975977837330335e–006
22	3,571527896130669e–001	182	1,158358935177899e–001	342	6,976561203768226e–004	502	2,800455533708622e–005
23	3,740643974275026e–001	183	1,183233335449968e–001	343	5,721223454434728e–004	503	2,015445311139832e–005
24	3,910243970160607e–001	184	1,204854506722672e–001	344	2,934875643581191e–004	504	1,125134651175812e–005
25	4,080154903861317e–001	185	1,223371395264402e–001	345	1,092526149391273e–004	505	5,869707265615299e–006
26	4,250144186334534e–001	186	1,238868653862843e–001	346	6,415402443848103e–004	506	1,013259758329981e–005
27	4,420013942269341e–001	187	1,251477258491527e–001	347	1,194730618383423e–003	507	1,088325131492173e–005
28	4,589582896478246e–001	188	1,261262023246478e–001	348	1,557112059887280e–003	508	7,167101260771279e–006
29	4,758753745532750e–001	189	1,268280540744526e–001	349	1,891971801393744e–003	509	4,840577540089826e–006
30	4,927463828072591e–001	190	1,272498700590511e–001	350	2,225524159129023e–003	510	–1,469933448634890e–005
31	5,095720854151864e–001	191	1,273590703506806e–001	351	2,530906981099261e–003	511	–8,010079089953001e–006
32	5,263554446856779e–001	192	1,274567595465545e–001	352	2,719749515067397e–003	512	–3,299004046633323e–005
33	5,430990601899994e–001	193	1,275561350483646e–001	353	2,729136737522100e–003	513	–4,373302115187172e–005
34	5,598052330684253e–001	194	1,273648326872248e–001	354	2,703019498899013e–003	514	–3,177468256997963e–005
35	5,764734796907189e–001	195	1,269415772180714e–001	355	2,630471852319136e–003	515	–2,976824036182567e–005
36	5,930981800982896e–001	196	1,262995646340671e–001	356	2,470456304276468e–003	516	–2,464228015326852e–005
37	6,096690552916387e–001	197	1,254605188749804e–001	357	2,239142906871446e–003	517	–1,606050838620834e–005
38	6,261725236758639e–001	198	1,244269583009826e–001	358	2,033465291493264e–003	518	–6,261944255489322e–006
39	6,425939632009995e–001	199	1,232131583108813e–001	359	1,948069005335563e–003	519	4,591009581217994e–007

Продолжение таблицы А.90

i	*c[i]*	*i*	*c[i]*	*i*	*c[i]*	*i*	*c[i]*
40	6,589148753746076e–001	200	1,218183974842866e–001	360	1,725029670030533e–003	520	1,395220723090848e–005
41	6,751199626157149e–001	201	1,202545652840080e–001	361	1,417366709895927e–003	521	1,622786214398703e–005
42	6,911981575264606e–001	202	1,185243106889108e–001	362	1,127141815310061e–003	522	–2,043464113212971e–006
43	7,071447728928043e–001	203	1,166399102636992e–001	363	8,089811988213151e–004	523	–1,653463907257247e–006
44	7,229599104052475e–001	204	1,146042249339280e–001	364	4,708009521678285e–004	524	–1,551250801467300e–008
45	7,386515025302785e–001	205	1,124296184976912e–001	365	7,882620739833088e–005	525	–1,907927361317977e–006
46	7,542294504292890e–001	206	1,101215600923314e–001	366	–2,998739993995956e–004	526	–9,607068622268791e–007
47	7,697093346240386e–001	207	1,076972053405737e–001	367	–4,733148292475610e–004	527	–4,636105364510011e–007
48	7,851012620144958e–001	208	1,051641975499523e–001	368	–5,791145447913150e–004	528	–2,765649762593200e–007
49	8,004165237845137e–001	209	1,025397604985405e–001	369	–6,754935404082003e–004	529	–1,922074581855119e–006
50	8,156523162880560e–001	210	9,982957934346254e–002	370	–8,029620210721900e–004	530	–9,897194091136331e–007
51	8,308039608112368e–001	211	9,705239536075722e–002	371	–9,726698841994444e–004	531	–7,873304717454037e–008
52	8,458450064727010e–001	212	9,421624116597689e–002	372	–1,196637962311630e–003	532	2,945239208477290e–008
53	8,607492455327098e–001	213	9,133590931873967e–002	373	–1,292865844760059e–003	533	–2,757610624807679e–006
54	8,754640719350776e–001	214	8,841813387276727e–002	374	–1,146268465739874e–003	534	–1,402925247695813e–005
55	8,899474405744183e–001	215	8,547715661443602e–002	375	–1,040598055074471e–003	535	–9,388962780643742e–006
56	9,041286138017367e–001	216	8,251962055343706e–002	376	–9,767709065548874e–004	536	2,068297421740023e–005
57	9,179666107725365e–001	217	7,955570759229536e–002	377	–9,294665200453614e–004	537	1,496435902895210e–007
58	9,313874086278087e–001	218	7,657649751612349e–002	378	–9,862027119530482e–004	538	6,757014945674924e–009
59	9,443802853939540e–001	219	7,360559211914287e–002	379	–1,047654674829846e–003	539	–2,778618354859861e–007
60	9,568885413848645e–001	220	7,064948295960993e–002	380	–1,099000599887377e–003	540	–1,569003268449803e–006
61	9,690016637782843e–001	221	6,771675107480543e–002	381	–1,151795860160292e–003	541	–1,089500601234349e–006
62	9,807691702375303e–001	222	6,480448458935215e–002	382	–1,194743370333155e–003	542	–9,870547653835426e–007
63	9,927543720639498e–001	223	6,192692754258131e–002	383	–1,250742797799558e–003	543	3,867483283567218e–005
64	1,001463112557766e+000	224	5,911363249658311e–002	384	1,287819050086379e–003	544	–1,232693496472088e–005
65	1,006893331637123e+000	225	5,637219228757212e–002	385	1,263569296641556e–003	545	9,464782951082177e–007
66	1,012508393574432e+000	226	5,368313072045600e–002	386	1,226113111394085e–003	546	8,254429452094225e–007
67	1,017729040219375e+000	227	5,105620793438655e–002	387	1,177515087338257e–003	547	4,883304950437536e–007
68	1,022470190536100e+000	228	4,849284995895640e–002	388	1,122503050159859e–003	548	–2,066961713890010e–007
69	1,026615653698808e+000	229	4,599068181839981e–002	389	1,089428846944533e–003	549	5,158212471036245e–009
70	1,030198648769593e+000	230	4,355568588898841e–002	390	1,054963366189962e–003	550	2,267731106642486e–007
71	1,033205850580933e+000	231	4,125570251909672e–002	391	9,019128558297515e–004	551	–4,880844550713951e–008
72	1,035694432087486e+000	232	3,907137550527191e–002	392	7,847839620863715e–004	552	3,361682183852576e–006
73	1,037683165297586e+000	233	3,696342556744636e–002	393	6,205675927856794e–004	553	4,677015459111491e–006
74	1,039227995800217e+000	234	3,493300140502248e–002	394	3,157663628445906e–004	554	2,820292122791583e–008
75	1,040349586463588e+000	235	3,298151059524886e–002	395	2,556449844935384e–004	555	5,143614846654519e–007
76	1,041086497214721e+000	236	3,110861245410919e–002	396	2,520606580606257e–004	556	3,818588614859347e–009
77	1,041443375950143e+000	237	2,931525594774175e–002	397	2,346980949474655e–004	557	1,737276553950212e–007
78	1,041434355650865e+000	238	2,760090729801069e–002	398	2,060394037017961e–004	558	1,876022048145804e–007
79	1,041043184216171e+000	239	2,597956638848436e–002	399	1,635905995590986e–004	559	–2,986488593070417e–009

Продолжение таблицы А.90

i	*c[i]*	*i*	*c[i]*	*i*	*c[i]*	*i*	*c[i]*
80	1,040262316588456e+000	240	2,443433592149451e–002	400	1,176237128375623e–004	560	–1,409927495646886e–008
81	1,039061496136853e+000	241	2,296470793543091e–002	401	6,193369904730005e–005	561	–6,977078748707401e–008
82	1,037422300157921e+000	242	2,156304510969632e–002	402	3,568554800150508e–005	562	–1,280675520205100e–008
83	1,035311720204252e+000	243	2,023524610221679e–002	403	2,443161189273522e–005	563	–2,222072007942510e–009
84	1,032712952177121e+000	244	1,897505817503749e–002	404	1,334090914042349e–005	564	–1,775191290895584e–009
85	1,029600494883906e+000	245	1,778248750467421e–002	405	2,853437194757816e–006	565	–1,686136654621906e–009
86	1,025966756910904e+000	246	1,665187994388476e–002	406	–1,039263591111469e–004	566	5,818594642226675e–006
87	1,021798805583990e+000	247	1,557759513377242e–002	407	5,144969377044875e–005	567	2,150883991167946e–006
88	1,017100128250049e+000	248	1,456208586604537e–002	408	9,711681816385056e–005	568	2,714879009950152e–007
89	1,011867706519706e+000	249	1,361072086117313e–002	409	2,472023910553232e–005	569	–2,567964804401197e–008
90	1,006109248754940e+000	250	1,270747042064656e–002	410	5,397064424090302e–005	570	2,041128570435378e–006
91	9,998285752401580e–001	251	1,186210743261470e–002	411	6,487880719449901e–005	571	3,262753594084781e–006
92	9,930379854679836e–001	252	1,106958962776399e–002	412	–5,192444140699947e–005	572	3,567581483749161e–006
93	9,857387823493258e–001	253	1,033126278863177e–002	413	–9,204876089551197e–005	573	4,083718802566134e–006
94	9,779405164766706e–001	254	9,640298325700842e–003	414	–1,815837353167847e–004	574	5,364807253588177e–006
95	9,696426101291272e–001	255	8,996371481700806e–003	415	–3,595054179561440e–004	575	4,178050149840223e–006
96	9,608519516143015e–001	256	–8,407748878436545e–003	416	–5,901617707607606e–007	576	5,189086332701670e–006
97	9,515674613550604e–001	257	–7,876393114319395e–003	417	1,831121301698088e–004	577	3,357218747491756e–006
98	9,417975696327747e–001	258	–7,380543918629573e–003	418	9,755685190624611e–005	578	6,310207878018869e–006
99	9,315442093447622e–001	259	–6,925141135202262e–003	419	6,606461762989423e–005	579	5,924001540927652e–006
100	9,208194746232827e–001	260	–6,500502521462604e–003	420	3,799971890923797e–005	580	5,161606640348293e–006
101	9,096310803629866e–001	261	–6,109178606718115e–003	421	4,150075391929448e–005	581	3,377814811745950e–006
102	8,979959173503500e–001	262	–5,741103163221257e–003	422	5,021905476506264e–005	582	1,323267689777069e–006
103	8,859232320517536e–001	263	–5,394569608919965e–003	423	5,861800137434713e–005	583	–1,074716688428712e–007
104	8,734366852542127e–001	264	–5,063851046064050e–003	424	2,126364641291926e–005	584	–3,561585382456484e–006
105	8,605542791988831e–001	265	–4,754191853611012e–003	425	1,181077582797280e–004	585	–4,518603099564185e–006
106	8,472987145504696e–001	266	–4,448993249380505e–003	426	9,990757789944374e–005	586	7,301956971603966e–007
107	8,336863467961255e–001	267	–4,133639756278191e–003	427	1,035782617124906e–004	587	5,891904775161025e–007
108	8,197387292306723e–001	268	–3,811612348723333e–003	428	8,870181845310037e–005	588	2,801882088134371e–008
109	8,054701312929008e–001	269	–3,505531318950422e–003	429	5,533953373249822e–005	589	6,322770332405526e–007
110	7,908995350037713e–001	270	–3,209092846617964e–003	430	1,580188994455254e–005	590	2,542598385847351e–007
111	7,760385598209244e–001	271	–2,927159436740159e–003	431	1,277184430250593e–006	591	1,272704908592385e–007
112	7,609051036128973e–001	272	–2,653818578698405e–003	432	5,009913312943629e–006	592	8,226599990523664e–008
113	7,455111681431031e–001	273	–2,396404013961463e–003	433	1,499170392246774e–005	593	5,433718768789140e–007
114	7,298745530879272e–001	274	–2,152379960589273e–003	434	2,241545750231630e–005	594	4,211177232106135e–007
115	7,140087729493950e–001	275	–1,924844672908215e–003	435	3,628511258723260e–005	595	3,552991527555180e–008
116	6,979336851549095e–001	276	–1,699160580023900e–003	436	2,406516798531014e–005	596	–1,398913109540774e–008
117	6,816667882498023e–001	277	–1,480542563288228e–003	437	2,515118233957011e–005	597	1,356727552196146e–006
118	6,652304141388827e–001	278	–1,283280633901446e–003	438	3,759629789955498e–005	598	–1,706941020342299e–005
119	6,486437667370537e–001	279	–1,131859661378862e–003	439	5,408154543124121e–005	599	1,013575160981381e–005

Окончание таблицы А.90

i	*c[i]*	*i*	*c[i]*	*i*	*c[i]*	*i*	*c[i]*
120	6,319284031798550e–001	280	–9,730460256556873e–004	440	4,493916063285122e–005	600	–2,285562946018590e–005
121	6,151031151692835e–001	281	–7,677634115875747e–004	441	2,806963579578946e–005	601	–8,908041185396514e–008
122	5,981877665956570e–001	282	–5,599347984905645e–004	442	2,364518513682831e–005	602	–9,597515277415496e–009
123	5,811992722116214e–001	283	–3,337966579125254e–004	443	1,260639764582286e–005	603	–3,225913527455964e–007
124	5,641522833259215e–001	284	–9,099722643476421e–005	444	–2,599467772603631e–008	604	1,070242712585309e–006
125	5,470652177576862e–001	285	1,498231621816041e–004	445	–1,774108392496017e–005	605	6,293002327021578e–007
126	5,299509559653194e–001	286	4,366447012116811e–004	446	–5,889276659458115e–006	606	3,575650976036433e–007
127	5,128557121424191e–001	287	6,307841647560053e–004	447	–4,663777919108619e–005	607	2,722295965060517e–005
128	–4,956175421414453e–001	288	6,150316826138937e–004	448	–2,078886359425321e–004	608	8,676848186676888e–006
129	–4,782650346610896e–001	289	8,990255827053560e–004	449	–2,131405580107761e–004	609	3,428660858940255e–007
130	–4,609828932783459e–001	290	1,232134364570107e–003	450	–1,784192600231068e–004	610	4,767793949944890e–007
131	–4,437530233023859e–001	291	1,471167206249042e–003	451	–1,744841754193053e–004	611	3,330981930777764e–007
132	–4,265950246465440e–001	292	1,697652664777771e–003	452	–1,728672507238372e–004	612	2,399696144635756e–007
133	–4,095160467543179e–001	293	1,985825255428654e–003	453	–1,885286127508226e–004	613	7,326611439066549e–009
134	–3,925409172155113e–001	294	2,172866052963961e–003	454	–2,078299015661617e–004	614	1,349943693297681e–007
135	–3,756821671788237e–001	295	1,812176023993582e–003	455	–2,123671573189573e–004	615	–5,393555749348494e–008
136	–3,589626517817934e–001	296	1,344657262814793e–003	456	–2,415166002501312e–004	616	3,629067065524143e–006
137	–3,423942311297658e–001	297	9,373975348172919e–004	457	–2,217025456251449e–004	617	–5,690530948134642e–006
138	–3,259993851088293e–001	298	5,621720998949145e–004	458	–9,907630821710970e–005	618	1,387566465624550e–008
139	–3,097861805973821e–001	299	2,048498552413189e–004	459	–8,039231481768845e–005	619	2,443085172403935e–007
140	–2,937724988593393e–001	300	–2,004822830002534e–004	460	–7,934509417722400e–005	620	1,723217058490933e–009
141	–2,779637821990255e–001	301	–6,169854804735951e–004	461	–5,874199358780108e–005	621	7,391973323448250e–008
142	–2,623749159488041e–001	302	–1,061498982103114e–003	462	–5,449816072329412e–005	622	5,303527922331415e–008
143	–2,470098299603623e–001	303	–1,594860949611097e–003	463	–4,489491034408147e–005	623	–8,883499047404846e–010
144	–2,318815478758375e–001	304	–2,124647831574725e–003	464	–3,498285982359981e–005	624	–3,870536804891648e–009
145	–2,169925682529340e–001	305	–2,621537051750861e–003	465	–1,748284921486958e–005	625	–1,846547564287500e–008
146	–2,023548005388463e–001	306	–3,064311083207632e–003	466	–9,075430772832575e–006	626	–4,244090917065736e–009
147	–1,879711746686855e–001	307	–3,460362845825662e–003	467	–1,052707430241351e–005	627	–4,013524925634108e–009
148	–1,738542127021508e–001	308	–3,794425324215804e–003	468	–6,538878366985722e–006	628	–6,325664562585882e–010
149	–1,600061812296078e–001	309	–4,091032597247918e–003	469	2,206341308073472e–005	629	–6,025110605409611e–010
150	–1,464389150679625e–001	310	–4,369553676668050e–003	470	1,769261935287328e–004	630	1,620171502086309e–006
151	–1,331544923127771e–001	311	–4,554811297024067e–003	471	6,418658561385058e–005	631	5,490569954646963e–007
152	–1,201628679722633e–001	312	–4,663276675479689e–003	472	–8,882305312548962e–005	632	6,355303179925355e–008
153	–1,074630704470568e–001	313	–4,722567636185647e–003	473	–1,721347222211949e–005	633	–5,426597100684762e–009
154	–9,506966959632511e–002	314	–4,704321497976561e–003	474	–6,093372716385583e–005	634	4,292861814894369e–007
155	–8,298103104739203e–002	315	–4,636227793039124e–003	475	–7,679955330373515e–005	635	6,834209542421138e–007
156	–7,120356992726613e–002	316	–4,517190210387324e–003	476	7,194151087015007e–005	636	7,099633014995863e–007
157	–5,973741829536090e–002	317	–4,351667566540186e–003	477	7,245095937243279e–005	637	8,109951846981774e–007
158	–4,859005767016811e–002	318	–4,135130493071822e–003	478	7,870354371072524e–005	638	4,118359768898598e–007
159	–3,775928110298274e–002	319	–3,870851645947402e–003	479	5,822201682995846e–004	639	6,571760029213382e–007

Таблица А.91 — Шумовая таблица $V[\varphi_{re,noise}(i), \varphi_{im,noise}(i)]$

i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$	i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$
0	–0,99948153278296	–0,59483417516607	256	0,99570534804836	0,45844586038111
1	0,97113454393991	–0,67528515225647	257	–0,63431466947340	0,21079116459234
2	0,14130051758487	–0,95090983575689	258	–0,07706847005931	–0,89581437101329
3	–0,47005496701697	–0,37340549728647	259	0,98590090577724	0,88241721133981
4	0,80705063769351	0,29653668284408	260	0,80099335254678	–0,36851896710853
5	–0,38981478896926	0,89572605717087	261	0,78368131392666	0,45506999802597
6	–0,01053049862020	–0,66959058036166	262	0,08707806671691	0,80938994918745
7	–0,91266367957293	–0,11522938140034	263	–0,86811883080712	0,39347308654705
8	0,54840422910309	0,75221367176302	264	–0,39466529740375	–0,66809432114456
9	0,40009252867955	–0,98929400334421	265	0,97875325649683	–0,72467840967746
10	–0,99867974711855	–0,88147068645358	266	–0,95038560288864	0,89563219587625
11	–0,95531076805040	0,90908757154593	267	0,17005239424212	0,54683053962658
12	–0,45725933317144	–0,56716323646760	268	–0,76910792026848	–0,96226617549298
13	–0,72929675029275	–0,98008272727324	269	0,99743281016846	0,42697157037567
14	0,75622801399036	0,20950329995549	270	0,95437383549973	0,97002324109952
15	0,07069442601050	–0,78247898470706	271	0,99578905365569	–0,54106826257356
16	0,74496252926055	–0,91169004445807	272	0,28058259829990	–0,85361420634036
17	–0,96440182703856	–0,94739918296622	273	0,85256524470573	–0,64567607735589
18	0,30424629369539	–0,49438267012479	274	–0,50608540105128	–0,65846015480300
19	0,66565033746925	0,64652935542491	275	–0,97210735183243	–0,23095213067791
20	0,91697008020594	0,17514097332009	276	0,95424048234441	–0,99240147091219
21	–0,70774918760427	0,52548653416543	277	–0,96926570524023	0,73775654896574
22	–0,70051415345560	–0,45340028808763	278	0,30872163214726	0,41514960556126
23	–0,99496513054797	–0,90071908066973	279	–0,24523839572639	0,63206633394807
24	0,98164490790123	–0,77463155528697	280	–0,33813265086024	–0,38661779441897
25	–0,54671580548181	–0,02570928536004	281	–0,05826828420146	–0,06940774188029
26	–0,01689629065389	0,00287506445732	282	–0,22898461455054	0,97054853316316
27	–0,86110349531986	0,42548583726477	283	–0,18509915019881	0,47565762892084
28	–0,98892980586032	–0,87881132267556	284	–0,10488238045009	–0,87769947402394
29	0,51756627678691	0,66926784710139	285	–0,71886586182037	0,78030982480538
30	–0,99635026409640	–0,58107730574765	286	0,99793873738654	0,90041310491497
31	–0,99969370862163	0,98369989360250	287	0,57563307626120	–0,91034337352097
32	0,55266258627194	0,59449057465591	288	0,28909646383717	0,96307783970534
33	0,34581177741673	0,94879421061866	289	0,42188998312520	0,48148651230437
34	0,62664209577999	–0,74402970906471	290	0,93335049681047	–0,43537023883588
35	–0,77149701404973	–0,33883658042801	291	–0,97087374418267	0,86636445711364
36	–0,91592244254432	0,03687901376713	292	0,36722871286923	0,65291654172961
37	–0,76285492357887	–0,91371867919124	293	–0,81093025665696	0,08778370229363
38	0,79788337195331	–0,93180971199849	294	–0,26240603062237	–0,92774095379098
39	0,54473080610200	–0,11919206037186	295	0,83996497984604	0,55839849139647
40	–0,85639281671058	0,42429854760451	296	–0,99909615720225	–0,96024605713970
41	–0,92882402971423	0,27871809078609	297	0,74649464155061	0,12144893606462
42	–0,11708371046774	–0,99800843444966	298	–0,74774595569805	–0,26898062008959

Продолжение таблицы А.91

i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$	i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$
43	0,21356749817493	–0,90716295627033	299	0,95781667469567	–0,79047927052628
44	–0,76191692573909	0,99768118356265	300	0,95472308713099	–0,08588776019550
45	0,98111043100884	–0,95854459734407	301	0,48708332746299	0,99999041579432
46	–0,85913269895572	0,95766566168880	302	0,46332038247497	0,10964126185063
47	–0,93307242253692	0,49431757696466	303	–0,76497004940162	0,89210929242238
48	0,30485754879632	–0,70540034357529	304	0,57397389364339	0,35289703373760
49	0,85289650925190	0,46766131791044	305	0,75374316974495	0,96705214651335
50	0,91328082618125	–0,99839597361769	306	–0,59174397685714	–0,89405370422752
51	–0,05890199924154	0,70741827819497	307	0,75087906691890	–0,29612672982396
52	0,28398686150148	0,34633555702188	308	–0,98607857336230	0,25034911730023
53	0,95258164539612	–0,54893416026939	309	–0,40761056640505	–0,90045573444695
54	–0,78566324168507	–0,75568541079691	310	0,66929266740477	0,98629493401748
55	–0,95789495447877	–0,20423194696966	311	–0,97463695257310	–0,00190223301301
56	0,82411158711197	0,96654618432562	312	0,90145509409859	0,99781390365446
57	–0,65185446735885	–0,88734990773289	313	–0,87259289048043	0,99233587353666
58	–0,93643603134666	0,99870790442385	314	–0,91529461447692	–0,15698707534206
59	0,91427159529618	–0,98290505544444	315	–0,03305738840705	–0,37205262859764
60	–0,70395684036886	0,58796798221039	316	0,07223051368337	–0,88805001733626
61	0,00563771969365	0,61768196727244	317	0,99498012188353	0,97094358113387
62	0,89065051931895	0,52783352697585	318	–0,74904939500519	0,99985483641521
63	–0,68683707712762	0,80806944710339	319	0,04585228574211	0,99812337444082
64	0,72165342518718	–0,69259857349564	320	–0,89054954257993	–0,31791913188064
65	–0,62928247730667	0,13627037407335	321	–0,83782144651251	0,97637632547466
66	0,29938434065514	–0,46051329682246	322	0,33454804933804	–0,86231516800408
67	–0,91781958879280	–0,74012716684186	323	–0,99707579362824	0,93237990079441
68	0,99298717043688	0,40816610075661	324	–0,22827527843994	0,18874759397997
69	0,82368298622748	–0,74036047190173	325	0,67248046289143	–0,03646211390569
70	–0,98512833386833	–0,99972330709594	326	–0,05146538187944	–0,92599700120679
71	–0,95915368242257	–0,99237800466040	327	0,99947295749905	0,93625229707912
72	–0,21411126572790	–0,93424819052545	328	0,66951124390363	0,98905825623893
73	–0,68821476106884	–0,26892306315457	329	–0,99602956559179	–0,44654715757688
74	0,91851997982317	0,09358228901785	330	0,82104905483590	0,99540741724928
75	–0,96062769559127	0,36099095133739	331	0,99186510988782	0,72023001312947
76	0,51646184922287	–0,71373332873917	332	–0,65284592392918	0,52186723253637
77	0,61130721139669	0,46950141175917	333	0,93885443798188	–0,74895312615259
78	0,47336129371299	–0,27333178296162	334	0,96735248738388	0,90891816978629
79	0,90998308703519	0,96715662938132	335	–0,22225968841114	0,57124029781228
80	0,44844799194357	0,99211574628306	336	–0,44132783753414	–0,92688840659280
81	0,66614891079092	0,96590176169121	337	–0,85694974219574	0,88844532719844
82	0,74922239129237	–0,89879858826087	338	0,91783042091762	–0,46356892383970
83	–0,99571588506485	0,52785521494349	339	0,72556974415690	–0,99899555770747
84	0,97401082477563	–0,16855870075190	340	–0,99711581834508	0,58211560180426
85	0,72683747733879	–0,48060774432251	341	0,77638976371966	0,94321834873819

Продолжение таблицы А.91

i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$	i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$
86	0,95432193457128	0,68849603408441	342	0,07717324253925	0,58638399856595
87	–0,72962208425191	–0,76608443420917	343	–0,56049829194163	0,82522301569036
88	–0,85359479233537	0,88738125901579	344	0,98398893639988	0,39467440420569
89	–0,81412430338535	–0,97480768049637	345	0,47546946844938	0,68613044836811
90	–0,87930772356786	0,74748307690436	346	0,65675089314631	0,18331637134880
91	–0,71573331064977	–0,98570608178923	347	0,03273375457980	–0,74933109564108
92	0,83524300028228	0,83702537075163	348	–0,38684144784738	0,51337349030406
93	–0,48086065601423	–0,98848504923531	349	–0,97346267944545	–0,96549364384098
94	0,97139128574778	0,80093621198236	350	–0,53282156061942	–0,91423265091354
95	0,51992825347895	0,80247631400510	351	0,99817310731176	0,61133572482148
96	–0,00848591195325	–0,76670128000486	352	–0,50254500772635	–0,88829338134294
97	–0,70294374303036	0,55359910445577	353	0,01995873238855	0,85223515096765
98	–0,95894428168140	–0,43265504344783	354	0,99930381973804	0,94578896296649
99	0,97079252950321	0,09325857238682	355	0,82907767600783	–0,06323442598128
100	–0,92404293670797	0,85507704027855	356	–0,58660709669728	0,96840773806582
101	–0,69506469500450	0,98633412625459	357	–0,17573736667267	–0,48166920859485
102	0,26559203620024	0,73314307966524	358	0,83434292401346	–0,13023450646997
103	0,28038443336943	0,14537913654427	359	0,05946491307025	0,20511047074866
104	–0,74138124825523	0,99310339807762	360	0,81505484574602	–0,94685947861369
105	–0,01752795995444	–0,82616635284178	361	–0,44976380954860	0,40894572671545
106	–0,55126773094930	–0,98898543862153	362	–0,89746474625671	0,99846578838537
107	0,97960898850996	–0,94021446752851	363	0,39677256130792	–0,74854668609359
108	–0,99196309146936	0,67019017358456	364	–0,07588948563079	0,74096214084170
109	–0,67684928085260	0,12631491649378	365	0,76343198951445	0,41746629422634
110	0,09140039465500	–0,20537731453108	366	–0,74490104699626	0,94725911744610
111	–0,71658965751996	–0,97788200391224	367	0,64880119792759	0,41336660830571
112	0,81014640078925	0,53722648362443	368	0,62319537462542	–0,93098313552599
113	0,40616991671205	–0,26469008598449	369	0,42215817594807	–0,07712787385208
114	–0,67680188682972	0,94502052337695	370	0,02704554141885	–0,05417518053666
115	0,86849774348749	–0,18333598647899	371	0,80001773566818	0,91542195141039
116	–0,99500381284851	–0,02634122068550	372	–0,79351832348816	–0,36208897989136
117	0,84329189340667	0,10406957462213	373	0,63872359151636	0,08128252493444
118	–0,09215968531446	0,69540012101253	374	0,52890520960295	0,60048872455592
119	0,99956173327206	–0,12358542001404	375	0,74238552914587	0,04491915291044
120	–0,79732779473535	–0,91582524736159	376	0,99096131449250	–0,19451182854402
121	0,96349973642406	0,96640458041000	377	–0,80412329643109	–0,88513818199457
122	–0,79942778496547	0,64323902822857	378	–0,64612616129736	0,72198674804544
123	–0,11566039853896	0,28587846253726	379	0,11657770663191	–0,83662833815041
124	–0,39922954514662	0,94129601616966	380	–0,95053182488101	–0,96939905138082
125	0,99089197565987	–0,92062625581587	381	–0,62228872928622	0,82767262846661
126	0,28631285179909	–0,91035047143603	382	0,03004475787316	–0,99738896333384
127	–0,83302725605608	–0,67330410892084	383	–0,97987214341034	0,36526129686425
128	0,95404443402072	0,49162765398743	384	–0,99986980746200	–0,36021610299715

Продолжение таблицы А.91

i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$	i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$
129	–0,06449863579434	0,03250560813135	385	0,89110648599879	–0,97894250343044
130	–0,99575054486311	0,42389784469507	386	0,10407960510582	0,77357793811619
131	–0,65501142790847	0,82546114655624	387	0,95964737821728	–0,35435818285502
132	–0,81254441908887	–0,51627234660629	388	0,50843233159162	0,96107691266205
133	–0,99646369485481	0,84490533520752	389	0,17006334670615	–0,76854025314829
134	0,00287840603348	0,64768261158166	390	0,25872675063360	0,99893303933816
135	0,70176989408455	–0,20453028573322	391	–0,01115998681937	0,98496019742444
136	0,96361882270190	0,40706967140989	392	–0,79598702973261	0,97138411318894
137	–0,68883758192426	0,91338958840772	393	–0,99264708948101	–0,99542822402536
138	–0,34875585502238	0,71472290693300	394	–0,99829663752818	0,01877138824311
139	0,91980081243087	0,66507455644919	395	–0,70801016548184	0,33680685948117
140	–0,99009048343881	0,85868021604848	396	–0,70467057786826	0,93272777501857
141	0,68865791458395	0,55660316809678	397	0,99846021905254	–0,98725746254433
142	–0,99484402129368	–0,20052559254934	398	–0,63364968534650	–0,16473594423746
143	0,94214511408023	–0,99696425367461	399	–0,16258217500792	–0,95939125400802
144	–0,67414626793544	0,49548221180078	400	–0,43645594360633	–0,94805030113284
145	–0,47339353684664	–0,85904328834047	401	–0,99848471702976	0,96245166923809
146	0,14323651387360	–0,94145598222488	402	–0,16796458968998	–0,98987511890470
147	–0,29268293575672	0,05759224927952	403	–0,87979225745213	–0,71725725041680
148	0,43793861458754	–0,78904969892724	404	0,44183099021786	–0,93568974498761
149	–0,36345126374441	0,64874435357162	405	0,93310180125532	–0,99913308068246
150	–0,08750604656825	0,97686944362527	406	–0,93941931782002	–0,56409379640356
151	–0,96495267812511	–0,53960305946511	407	–0,88590003188677	0,47624600491382
152	0,55526940659947	0,78891523734774	408	0,99971463703691	–0,83889954253462
153	0,73538215752630	0,96452072373404	409	–0,75376385639978	0,00814643438625
154	–0,30889773919437	–0,80664389776860	410	0,93887685615875	–0,11284528204636
155	0,03574995626194	–0,97325616900959	411	0,85126435782309	0,52349251543547
156	0,98720684660488	0,48409133691962	412	0,39701421446381	0,81779634174316
157	–0,81689296271203	–0,90827703628298	413	–0,37024464187437	–0,87071656222959
158	0,67866860118215	0,81284503870856	414	–0,36024828242896	0,34655735648287
159	–0,15808569732583	0,85279555024382	415	–0,93388812549209	–0,84476541096429
160	0,80723395114371	–0,24717418514605	416	–0,65298804552119	–0,18439575450921
161	0,47788757329038	–0,46333147839295	417	0,11960319006843	0,99899346780168
162	0,96367554763201	0,38486749303242	418	0,94292565553160	0,83163906518293
163	–0,99143875716818	–0,24945277239809	419	0,75081145286948	–0,35533223142265
164	0,83081876925833	–0,94780851414763	420	0,56721979748394	–0,24076836414499
165	–0,58753191905341	0,01290772389163	421	0,46857766746029	–0,30140233457198
166	0,95538108220960	–0,85557052096538	422	0,97312313923635	–0,99548191630031
167	–0,96490920476211	–0,64020970923102	423	–0,38299976567017	0,98516909715427
168	–0,97327101028521	0,12378128133110	424	0,41025800019463	0,02116736935734
169	0,91400366022124	0,57972471346930	425	0,09638062008048	0,04411984381457
170	–0,99925837363824	0,71084847864067	426	–0,85283249275397	0,91475563922421
171	–0,86875903507313	–0,20291699203564	427	0,88866808958124	–0,99735267083226

Продолжение таблицы А.91

i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$	i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$
172	–0,26240034795124	–0,68264554369108	428	–0,48202429536989	–0,96805608884164
173	–0,24664412953388	–0,87642273115183	429	0,27572582416567	0,58634753335832
174	0,02416275806869	0,27192914288905	430	–0,65889129659168	0,58835634138583
175	0,82068619590515	–0,85087787994476	431	0,98838086953732	0,99994349600236
176	0,88547373760759	–0,89636802901469	432	–0,20651349620689	0,54593044066355
177	–0,18173078152226	–0,26152145156800	433	–0,62126416356920	–0,59893681700392
178	0,09355476558534	0,54845123045604	434	0,20320105410437	–0,86879180355289
179	–0,54668414224090	0,95980774020221	435	–0,97790548600584	0,96290806999242
180	0,37050990604091	–0,59910140383171	436	0,11112534735126	0,21484763313301
181	–0,70373594262891	0,91227665827081	437	–0,41368337314182	0,28216837680365
182	–0,34600785879594	–0,99441426144200	438	0,24133038992960	0,51294362630238
183	–0,68774481731008	–0,30238837956299	439	–0,66393410674885	–0,08249679629081
184	–0,26843291251234	0,83115668004362	440	–0,53697829178752	–0,97649903936228
185	0,49072334613242	–0,45359708737775	441	–0,97224737889348	0,22081333579837
186	0,38975993093975	0,95515358099121	442	0,87392477144549	–0,12796173740361
187	–0,97757125224150	0,05305894580606	443	0,19050361015753	0,01602615387195
188	–0,17325552859616	–0,92770672250494	444	–0,46353441212724	–0,95249041539006
189	0,99948035025744	0,58285545563426	445	–0,07064096339021	–0,94479803205886
190	–0,64946246527458	0,68645507104960	446	–0,92444085484466	–0,10457590187436
191	–0,12016920576437	–0,57147322153312	447	–0,83822593578728	–0,01695043208885
192	–0,58947456517751	–0,34847132454388	448	0,75214681811150	–0,99955681042665
193	–0,41815140454465	0,16276422358861	449	–0,42102998829339	0,99720941999394
194	0,99885650204884	0,11136095490444	450	–0,72094786237696	–0,35008961934255
195	–0,56649614128386	–0,90494866361587	451	0,78843311019251	0,52851398958271
196	0,94138021032330	0,35281916733018	452	0,97394027897442	–0,26695944086561
197	–0,75725076534641	0,53650549640587	453	0,99206463477946	–0,57010120849429
198	0,20541973692630	–0,94435144369918	454	0,76789609461795	–0,76519356730966
199	0,99980371023351	0,79835913565599	455	–0,82002421836409	–0,73530179553767
200	0,29078277605775	0,35393777921520	456	0,81924990025724	0,99698425250579
201	–0,62858772103030	0,38765693387102	457	–0,26719850873357	0,68903369776193
202	0,43440904467688	–0,98546330463232	458	–0,43311260380975	0,85321815947490
203	–0,98298583762390	0,21021524625209	459	0,99194979673836	0,91876249766422
204	0,19513029146934	–0,94239832251867	460	–0,80692001248487	–0,32627540663214
205	–0,95476662400101	0,98364554179143	461	0,43080003649976	–0,21919095636638
206	0,93379635304810	–0,70881994583682	462	0,67709491937357	–0,95478075822906
207	–0,85235410573336	–0,08342347966410	463	0,56151770568316	–0,70693811747778
208	–0,86425093011245	–0,45795025029466	464	0,10831862810749	–0,08628837174592
209	0,38879779059045	0,97274429344593	465	0,91229417540436	–0,65987351408410
210	0,92045124735495	–0,62433652524220	466	–0,48972893932274	0,56289246362686
211	0,89162532251878	0,54950955570563	467	–0,89033658689697	–0,71656563987082
212	–0,36834336949252	0,96458298020975	468	0,65269447475094	0,65916004833932
213	0,93891760988045	–0,89968353740388	469	0,67439478141121	–0,81684380846796
214	0,99267657565094	–0,03757034316958	470	–0,47770832416973	–0,16789556203025

Окончание таблицы А.91

i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$	i	$\varphi_{re,noise}(i)$	$\varphi_{im,noise}(i)$
215	–0,94063471614176	0,41332338538963	471	–0,99715979260878	–0,93565784007648
216	0,99740224117019	–0,16830494996370	472	–0,90889593602546	0,62034397054380
217	–0,35899413170555	–0,46633226649613	473	–0,06618622548177	–0,23812217221359
218	0,05237237274947	–0,25640361602661	474	0,99430266919728	0,18812555317553
219	0,36703583957424	–0,38653265641875	475	0,97686402381843	–0,28664534366620
220	0,91653180367913	–0,30587628726597	476	0,94813650221268	–0,97506640027128
221	0,69000803499316	0,90952171386132	477	–0,95434497492853	–0,79607978501983
222	–0,38658751133527	0,99501571208985	478	–0,49104783137150	0,32895214359663
223	–0,29250814029851	0,37444994344615	479	0,99881175120751	0,88993983831354
224	–0,60182204677608	0,86779651036123	480	0,50449166760303	–0,85995072408434
225	–0,97418588163217	0,96468523666475	481	0,47162891065108	–0,18680204049569
226	0,88461574003963	0,57508405276414	482	–0,62081581361840	0,75000676218956
227	0,05198933055162	0,21269661669964	483	–0,43867015250812	0,99998069244322
228	–0,53499621979720	0,97241553731237	484	0,98630563232075	–0,53578899600662
229	–0,49429560226497	0,98183865291903	485	–0,61510362277374	–0,89515019899997
230	–0,98935142339139	–0,40249159006933	486	–0,03841517601843	–0,69888815681179
231	–0,98081380091130	–0,72856895534041	487	–0,30102157304644	–0,07667808922205
232	–0,27338148835532	0,99950922447209	488	0,41881284182683	0,02188098922282
233	0,06310802338302	–0,54539587529618	489	–0,86135454941237	0,98947480909359
234	–0,20461677199539	–0,14209977628489	490	0,67226861393788	–0,13494389011014
235	0,66223843141647	0,72528579940326	491	–0,70737398842068	–0,76547349325992
236	–0,84764345483665	0,02372316801261	492	0,94044946687963	0,09026201157416
237	–0,89039863483811	0,88866581484602	493	–0,82386352534327	0,08924768823676
238	0,95903308477986	0,76744927173873	494	–0,32070666698656	0,50143421908753
239	0,73504123909879	–0,03747203173192	495	0,57593163224487	–0,98966422921509
240	–0,31744434966056	–0,36834111883652	496	–0,36326018419965	0,07440243123228
241	–0,34110827591623	0,40211222807691	497	0,99979044674350	–0,14130287347405
242	0,47803883714199	–0,39423219786288	498	–0,92366023326932	–0,97979298068180
243	0,98299195879514	0,01989791390047	499	–0,44607178518598	–0,54233252016394
244	–0,30963073129751	–0,18076720599336	500	0,44226800932956	0,71326756742752
245	0,99992588229018	–0,26281872094289	501	0,03671907158312	0,63606389366675
246	–0,93149731080767	–0,98313162570490	502	0,52175424682195	–0,85396826735705
247	0,99923472302773	–0,80142993767554	503	–0,94701139690956	–0,01826348194255
248	–0,26024169633417	–0,75999759855752	504	–0,98759606946049	0,82288714303073
249	–0,35712514743563	0,19298963768574	505	0,87434794743625	0,89399495655433
250	–0,99899084509530	0,74645156992493	506	–0,93412041758744	0,41374052024363
251	0,86557171579452	0,55593866696299	507	0,96063943315511	0,93116709541280
252	0,33408042438752	0,86185953874709	508	0,97534253457837	0,86150930812689
253	0,99010736374716	0,04602397576623	509	0,99642466504163	0,70190043427512
254	–0,66694269691195	–0,91643611810148	510	–0,94705089665984	–0,29580042814306
255	0,64016792079480	0,15649530836856	511	0,91599807087376	–0,98147830385781

Библиография

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

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 *П. М. Смирнов*
Технический редактор *Е. В. Беспрозванная*
Корректор *Л. Я. Митрофанова*
Компьютерная верстка *Т. Ф. Кузнецовой*

Сдано в набор 26.08.2014. Подписано в печать 17.12.2014. Формат 60×84 1/8. Бумага офсетная. Гарнитура Ариал. Печать офсетная. Усл. печ. л. 38,13. Уч.-изд. л. 37,20. Тираж 35 экз. Зак. 1430.

ФГУП «СТАНДАРТИНФОРМ», 123995 Москва, Гранатный пер., 4.
gost.e-ecolog.ru gost.e-ecolog.ru
Набрано и отпечатано в Калужской типографии стандартов, 248021 Калуга, ул. Московская, 256.

gost.e-ecolog.ru