ФЕДЕРАЛЬНОЕ АГЕНТСТВО
ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р 53556.12— 2014

Звуковое вещание цифровое

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

Часть III (MPEG-4 AUDIO)

Масштабируемое кодирование без потерь

ISO/IEC 14496-3:2009
(NEQ)

Издание официальное

Москва
Стандартинформ
2014

gost.e-ecolog.ru

Предисловие

1 РАЗРАБОТАН Техническим комитетом по стандартизации ТК 480 «Связь»

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 480 «Связь»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 17 марта 2014 г. № 149-ст

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

5 ВВЕДЕН ВПЕРВЫЕ

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
2 Термины и определения . . . 1
2.1 Термины . . . 1
2.2 Система обозначений . . . 2
2.3 Определения . . . 2
3 Полезные нагрузки для аудиообъекта . . . 2
4 Семантика . . . 4
5 Инструмент декодера *SLS* . . . 6
5.1 Обзор . . . 6
5.2 Метод передискретизации . . . 6
5.3 *SLS* с ядром масштабируемого *AAC* . . . 6
5.4 Декодирование *lle_single_channel_element (LLE_SCE)* и *lle_channel_pair_element (LLE_CPE)* . . . 6
5.5 Декодирование *lle_data* . . . 9
5.6 Компенсация для остатка *IntMDCT* для раннего завершения декодирования *BPGC/CBAC* . . . 19
5.7 Инверсное отображение ошибки . . . 19
5.8 Целочисленный процесс *Mid/Side* . . . 20
5.9 Целочисленное формирование временного шума (*IntTNS*) . . . 20
5.10 *IntMDCT* и инверсное *IntMDCT* . . . 23
5.11 Вычисление табличных значений на основе компактных таблиц . . . 35
Приложение А (справочное) Описание кодера . . . 37
Приложение В (обязательное) Таблицы для предопределенных коэффициентов с фиксированной точкой . . . 42
Библиография . . . 62

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Звуковое вещание цифровое

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

Ч а с т ь III (MPEG-4 AUDIO)

Масштабируемое кодирование без потерь

Digital sound broadcasting. Coding of signals of sound broadcasting with reduction of redundancy for transfer on digital communication channels. Part III (MPEG-4 audio). Scalable lossless coding

Дата введения — 2015—01—01

1 Область применения

Данный стандарт описывает алгоритм масштабируемого кодирования без потерь *MPEG-4* для аудиосигналов.

2 Термины и определения

2.1 Термины

В этом стандарте используются следующие термины.

Core Layer — Кодер *MPEG-4 GA T/F*, используемый в качестве первого уровня в *SLS*. Поддерживаются типы аудиообъектов *LC AAC, AAC Scalable* (без *LTP*), *ER AAC LC, ER AAC Scalable* и *ER BSAC*.

LLE Layer — Уровень улучшения без потерь, используемый в *SLS* для улучшения качества базового уровня применительно к кодированию без потерь.

Bit Plane — Позиция определенного бита в слове двоичных данных, начиная с 0 как позиции младшего значащего бита (*LSB*). Например, двоичные символы разрядной матрицы с позициями 0, 1, 2, и 3 из слова данных 0×0011 1101 (0×3*d*) будут 1, 0, 1, и 1, соответственно.

BPGC — Код *Golomb* разрядной матрицы.

CBAC — Арифметический код на базе контекста.

LEMC — Код низкоэнергетического режима.

Implicit Band — Полоса масштабного коэффициента, для которой представленные в потоке битов базового уровня квантованные спектральные данные будут использоваться в части определения необходимой дополнительной информации для уровня *LLE*.

Explicit Band — Полоса масштабного коэффициента, для которой квантованные спектральные данные, представленные в потоке битов базового уровня, не будут использоваться в определении необходимой дополнительной информации для уровня *LLE*. Вся дополнительная информация будет явно кодирована в полезной нагрузке *LLE*.

Oversampling Factor (osf) — Отношение между частотами дискретизации уровня *LLE* и базового уровня, возможные значения 1, 2 и 4.

Oversampling Range — Диапазон высокой частоты, охваченый только уровнем *LLE*, включает значения частоты (*osf*-1) *1024 *resp.* (*osf*-1) *128 на окно.

Reserved — Все поля маркированные как *Reserved*. Все поля *Reserved* должны быть обнулены.

Издание официальное

2.2 Система обозначений

Чтобы сделать описание строгим, в этом документе используется следующая система обозначений:
матрицы (и векторы векторов) обозначаются прописными однобуквенными именами, например, *M*;
переменные обозначаются курсивом, например, *variable*;
функции обозначаются как *func* (*x*).

2.3 Определения

DIV (m, n) Целочисленное деление с усечением результата *m/n* до целочисленного значения по направлению к $-\infty$.

$\lfloor * \rfloor$ Операция *floor*. Возвращает самое большое целое число, которое меньше или равно аргументу с вещественным значением.

3 Полезные нагрузки для аудиообъекта

Таблица 1 — Синтаксис *SLSSpecificConfig*

Синтаксис	Количество битов	Мнемоника
SLSSpecificConfig(samplingFrequencyIndex, *channelConfiguration,* *audioObjectType)* *{* *pcmWordLength;* *aac_core_present;* *lle main stream;* *reserved_bit;* *frameLength;* *if (!channelConfiguration){* *program_config_element( );* *}* *}*	 3 1 1 1 3	 *uimsbf* *uimsbf* *uimsbf* *uimsbf* *uimsbf*

Таблица 2 — Полезная нагрузка верхнего уровня для потока *lle*

Синтаксис	Количество битов	Мнемоника
lle_element() *{* *for (ch=0;ch<channel_number;) {* *if (is_channel_pair(ch)) {* *lie channel pair elementO;* *ch += 2;* *} else {* *lle_single channel_element();* *ch++;* *}* *}* *}*		

Таблица 3 — Синтаксис *lle_single_channel_element*

Синтаксис	Количество битов	Мнемоника
lie single channel elementO *{* *lle_individual_channel_stream(1);* *}*		

Таблица 4 — Синтаксис *lle_individual_channel_stream*

Синтаксис	Количество битов	Мнемоника
lie individual channel stream(is first channel) *{* *lle_ics_length_lsb;* *lle_ics_length_msb;* *lie ics length = lie ics length lsb || (lie ics length msb « 8);* *if (is_first_channel) {* *element instance tag;* *}*	 8 8 4	 *uimsbf* *uimsbf* *uimsbf*
lle_reserved_bit *if (lle_main_stream) {* *lle_header(is_first_channel)* *lie side info ( )* *}* *lle_data( );* *byte_align( );* *}*	1	*uimsbf*

Таблица 5 — Синтаксис *lle_channel_pair_element*

Синтаксис	Количество битов	Мнемоника
lle_channel_pair_element0 *{* *lle_individual_channel_stream(1);* *lle_individual_channel_stream(0);* *}*		

Таблица 6 — Синтаксис *lle_header* ()

Синтаксис	Количество битов	Мнемоника
lle_header(is_first_channel) *{* *if (lle_channel_pair_element && common_window &&* *is_first_channel) {* *use_stereo_intmdct;* *}* *if (aac_core_present) {* *band_type_signaling;* *if (band_type_signaling==1) {* *for(g=0;g<num_window_groups;g++) {* *for(sfb=0;sfb<max_sfb;sfb++) {* *band_type[g][sfb];* *}* *}* *}* *} else {* *if (is_first channel) {* *windows_sequence;* *}* *}* *}*	 1 2 1 2	 *uimsbf* *uimsbf* *uimsbf* *uimsbf*

Таблица 7 — Синтаксис *lle_data*

Синтаксис	Количество битов	Мнемоника
lle_data() *{* *BPGC/CBAC data* *LEMC data* *}*	 Изменяется Изменяется	 *bslbf* *bslbf*

Таблица 8 — Синтаксис *lle_side_info*

Синтаксис	Количество битов	Мнемоника
lle_side_info0 *For(g=0;g<num_window_groups;g++) {* *for(sfb=0;sfb<num_sfb+num_osf_sfb;sfb++) {* *if (band_type[g][sfb]==Explicit_Band) {* *vcod_dpcm_max_bp[g][sfb]* *}* *if (max_bp[g][sfb] != -1) {* *vcod_lazy_bp[g][sfb]* *}* *}* *}* *cb_cbac* *}*	 1...17 1...2 1	 *bslbf* *bslbf* *uimsbf*

4 Семантика

Элементы данных:

aac_core_present Указывает, действует ли улучшение без потерь поверх ядра *MPEG-4 GA T/F* (*aac_core_present*=1) или в неосновном режиме (*aac_core_present*=0).

lle_main_stream Указывает, представляет ли собой текущий поток основной поток *LLE*, включая всю необходимую дополнительную информацию или поток расширения *LLE*, который расширяет предыдущий поток *LLE*.

pcmWordlength Длина слова квантования исходной формы сигнала *PCM*.

Таблица 9 — Длина слова исходной формы сигнала *PCM*

pcmWordlength	Длина слова исходной формы сигнала PCM
0	8
1	16
2	20
3	24
4—7	Зарезервировано

frameLength Длина фрейма IntMDCT в уровне LLE.

Таблица 10 — Длина фрейма *IntMDCT*

frameLength	Длина фрейма *IntMDCT*	Фактор передискретизация банка фильтров *IntMDCT* (*osf*)
0	1024	1
1	2048	2
2	4096	4
3—7	Зарезервировано	Зарезервировано

element_instance_tag Уникальный тег экземпляра для синтаксических элементов. Все синтаксические элементы, содержащие теги экземпляра, могут появиться не один раз, но должны иметь уникальный *element_instance_tag* в каждом аудиофрейме. Когда присутствует ядро *MPEG–4 GA T/F*, синтаксические элементы *SLS* и *MPEG–4 GA T/F* из того же звукового канала используют тот же самый *element_instance_tag*.

lle_ics_length Длина потока отдельного канала *LLE* (*LLE_ICS*) для текущего фрейма в байтах.

band_type_signaling По умолчанию, тип полосы для полосы масштабного коэффициента определяется следующим образом: полоса масштабного коэффициента, которая находится в разделе, кодированном нулевым сборником кодов (*ZERO_HCB*), кодированном *Intensity Stereo* (*IS*), или кодированном *Perceptual Noise Substitution* (*PNS*), является *Explicit_Band*. Иначе это *Implicit_Band*.

Полосы масштабного коэффициента выше *max_sfb* и в диапазоне передискретизации всегда являются *Explicit_Band*. Этот тип полосы по умолчанию может быть перезаписан *band_type_signaling* следующим образом:

Таблица 11 — Сигнализация о типе полосы

Значение *band_type_signaling*	Тип полосы
00	Используется по умолчанию
01	Сигнализация о типе полосы для каждого *sfb* следует
10	Все *sfb* являются *Explicit_Band*
11	Зарезервировано

band_type [g] [sfb] Сигнализация о типе полосы для каждой полосы масштабного коэффициента, когда *band_type_signaling* = = 01. Полоса масштабного коэффициента устанавливается в *Explicit_Band*, если *band_type* [*g*] [*sfb*] равно 0.

Таблица 12 — Тип полосы

Значение	Тип полосы
0	*Explicit_Band*
1	Значение по умолчанию

vcod_dpcm_max_bp [g] [sfb] Кодированная максимальная разрядная матрица переменной длины для полосы масштабного коэффициента *sfb* и группового *g*.

vcod_lazy_bp [g] [sfb] Кодированная *lazy* разрядная матрица переменной длины для ненулевой полосы масштабного коэффициента *sfb* и группового *g*.

cb_cbac Индикация таблицы частот, которая будет использоваться в процессе декодирования *LLE*.

Таблица 13 — *cb_cbac* таблица

cb_cbac	Таблица частот
0	*BPGC*
1	*CBAC*

bpgc/cbac_data Двоичный поток битов кодированных спектральных данных остатотка *bpgc/cbac*.

low_energy_mode_data Двоичный поток битов кодированных данных спектра остатка режима *LEMC*.

5 Инструмент декодера *SLS*

5.1 Обзор

Поток базового уровня *MPEG-4 GA* декодируется детерминированным декодером базового уровня. Его вывод, который является детерминированным спектром в области *MDCT*, отправляется в инверсный процесс отображения ошибок. Тем временем остаточный спектр *IntMDCT*, который переносится в потоках уровня *LLE*, декодируется и отправляется в инверсный процесс отображения ошибок, чтобы восстановить спектр *IntMDCT*. Затем вызывается и выполняется в случае необходимости на коэффициентах *IntMDCT* процесс инверсного целочисленного *Mid/Side* (*M/S*) и инверснного целочисленного *TNS*. Наконец, его вывод инверсно преобразовывается при использовании инверсного процесса *IntMDCT*, чтобы получить аудиовыборки *PCM*. Подробное описание каждого процесса дается в последующих разделах.

5.2 Метод передискретизации

Базовому уровню разрешено работать на более низкой частоте дискретизации, чем уровням *LLE*. Следующая таблица показывает некоторые возможные комбинации частоты дискретизации.

Таблица 14 — Пример комбинаций частот дискретизации для уровней *Core* и *LLE*

	Core@ 48 кГц	*Core*@ 96 кГц	*Core*@ 192 кГц
LLE@ 48 кГц	X (*osf* = 1)		
LLE@ 96 кГц	X (*osf* = 2)	X (*osf* = 1)	
LLE@ 192 кГц	X (*osf* = 4)	X (*osf* = 2)	X (*osf* = 1)

Масштабируемость кодека, использующего различные частоты дискретизации, достигается, изменяя длину инверсного *IntMDCT* в декодере, соответственно. В то время, как ядро *AAC* обрабатывает 1024 значения в каждом фрейме, кодек *SLS* должен обработать *osf**1024 значений на фрейм. Это достигается увеличением длины инверсного *IntMDCT* в декодере до *osf**1024 линий спектра. 1024 значения инверсного квантованного спектра из ядра *AAC* добавляются к 1024 низкочастотным значениям спектра остатка *SLS*.

5.3 *SLS* с ядром масштабируемого *AAC*

Если базовым уровнем является *AAC Scalable*, спектральные данные, декодируемые из уровней *SLS*, добавляются к спектральным данным, декодируемым из потоков масштабируемого *AAC* детерминированным инверсным квантизатователем *AAC*. Получающиеся спектральные данные затем обрабатываются инверсным целочисленным *M/S* и инверсным целочисленным процессом *TNS* в случае необходимости. Наконец вывод преобразовывается инверсным *IntMDCT*, чтобы выработать аудиовыборки *PCM*.

5.3.1 Неосновной режим

В неосновном режиме *SLS* работает как автономный кодек без ядра *AAC*. В случае аудиообъектного типа *SLS* это сообщается с помощью *aac_core_present* = 0 для неосновного режима и *aac_core_present* = 1 для режима на базе ядра. В случае неосновного аудиообъектного типа *SLS* это всегда *aac_core_present* = 0.

В неосновном режиме используются следующие значения по умолчанию:

- *window_shape* = 0 (синусное окно);
- если (*lle_channel_pair_element*) *common_window* = 1 (включено);
- если (*use_stereo_intmdct*), все флаги *M/S* включены, иначе все флаги *M/S* выключены;
- если (*window_sequence* = = *EIGHT_SHORT_SEQUENCE*) группировка = {2,2,2,2}.

5.4 Декодирование *lle_single_channel_element* (*LLE_SCE*) и *lle_channel_pair_element* (*LLE_CPE*)

5.4.1Определения

lle_ics_length — Длина потока отдельного канала *LLE* (*LLE_ICS*) в байтах.

vcod_dpcm_max_bp [g] [sfb] — Кодированная максимальная разрядная матрица переменной длины для полосы масштабного коэффициента *sfb* и группы *g*. Этот элемент присутствует только для незначительных полос масштабного коэффициента.

vcod_lazy_bp [g] [sfb] — Кодированная *lazy* разрядная матрица переменной длины для ненулевой полосы масштабного коэффициента *sfb* и группы *g*.

g — Индекс группы.

sfb	Полоса масштабного коэффициента в пределах группы
win	Индекс окна.
bin	Индекс бункера частот
num_window_groups	Число групп окон, которые совместно используют один набор масштабных коэффициентов.
num_sfb	Число полос масштабного коэффициента на короткое окно в случае *EIGHT_SHORT_SQEUENCE*, иначе число полос масштабного коэффициента для длинных окон.
num_osf_sfb	Число полос масштабного коэффициента на окно в диапазоне передискретизации. Диапазон передискретизации покрывается (*osf*-1) *16 полосами с шириной 64 в случае длинных окон, соответственно (*osf*-1) *4 полосы с шириной 32 в случае коротких окон.
max_bp [g] [sfb]	Максимальная разрядная матрица для группового *g* и полосы масштабного коэффициента *sfb*.
lazy_bp [g] [sfb]	Lazy разрядная матрица для группы *g* и полосы масштабного коэффициента *sfb*.
read_bits (n)	Считать n последовательных битов из входного потока битов в порядке *bslbf*.
quant[g] [win] [sfb] [bin]	Квантованные спектральные данные *AAC*.
Interval [g] [win] [sfb] [k]	Интервалы квантования в базовом кодере *AAC*.

5.4.2 Процесс декодирования

5.4.2.1 *LLE_SCE* и *LLE_CPE*

LLE_SCE составляется из *lle_individual_channel_stream* (*LLE_ICS*), в то время как у *LLE_CPE* имеется два *lle_individual_channel_streams* (*LLE_ICS*).

5.4.2.2 Декодирование *LLE_ICS*

В *LLE_ICS* порядок процесса декодирования следующий.

Взять *lle_ ics_len*, информацию со стороны декодируемой *LLE, BPGC/CBAC* и *LEMC*.

Для потока битов *SLS*, составленного из потока *lle_main* (*lle_main_stream* = 1) и многократного (> =1) потока *lle_extension* (*lle_main_stream* = 0) для каждого *LLE_ICS*, создается *lle_data* (), связывая элементы *lle_data* () из потока *lle_main*, и всех доступных потоков *lle_extension*.

Если промежуточный поток *LLE_extension* отсутствует, данные в *lle_data* () последующих потоков не могут использоваться.

5.4.2.3 Восстановление дополнительной информации *BPGC/CBAC*

Поскольку каждая полоса масштабного коэффициента полосы типа *Explicit_Band* передается максимальная разрядная матрица (*max_bp*). Кроме того для каждой полосы масштабного коэффициента разрядная матрица *lazy* (*lazy_bp*) передается, если все остаточные спектральные данные не являются нулевыми для этой полосы масштабного коэффициента (о чем сообщается максимальной разрядной матрицей =-1). *max_bp* кодируется, используя кодированное *DPCM* переменной длины относительно ранее переданной максимальной разрядной матрицы. Первое значение в каждой группе окон кодируется, используя *PCM* на 5 битов. Значение *max_bp* кодируется в унарном представлении. Следующая таблица дает некоторые примеры того, как кодируется значение *DPCM max_bp*.

Таблица 15 — Кодовая комбинация для декодирования значения *DPCM max_bp*

DPCM max_bp	Кодовая комбинация	Длина кодовой комбинации
0	1	1
(*s*)1	01(*s*)	3
(*s*)2	001(*s*)	4
…	…	...
(*s*)10	00000000001(*s*)	12
…	…	...

Различие между *max_bp* и *lazy_bp*, значение которого лежит в пределах диапазона {1, 2, 3}, декодируется следующим образом:

Таблица 16 — Кодовая комбинация для декодирования различия между *max_bp* и *lazy_bp*

max_bp — lazy_bp	Кодовая комбинация	Длина кодовой комбинации
1	10	2
2	0	1
3	11	2

Следующий псевдокод иллюстрирует процесс декодирования для *max_bp* и *lazy_bp*.

```
for (g = 0;g < num_window_groups; g++)
    init = 0;
    for (sfb = 0; sfb <num sfb+num osf sfb;sfb++){
        if (band type[g][sFb]== Explicit Band) {
            if (!init){
                max bp[g][sfb] = read bits(5) - 1;
                        init ++;
                }
            else {
                m = 0;
                while (read_bits(1) == 0) m++;
                if (m) {
                        if (read_bits(1)) m = -m;
                }
                max_bp[g][sfb] = m0 - m;
            }
            m0 = max_bp[g][sfb];
}
if (max_bp[g][sfb]>=0) {
      if (read_bits(1)==0)
            lazy_bp[g][sfb] = max_bp[g][sfb] - 2;
          else {
            if (read_bits(1)= =0)      lazy_bp[g][sfb]  =  max_bp[g][sfb] - 1;
            else lazy_bp[g][sfb] = max_bp[g][sfb] - 3;
            }
        }
    }
```

Для *Implicit_Bands*, *max_bp* [*g*] [*sfb*] вычисляется из порогов квантования квантователя базового уровня следующим образом:

Как первый шаг может быть вычислена максимальная разрядная матрица *M* для каждого спектрального бункера остатка для существенных полос масштабного коэффициента с помощью

$$M\,[g]\,[win]\,[sfb]\,[bin] = INT\,\{log_2[interval\,[g]\,[win]\,[sfb]\,[bin]]\}.$$

где *interval* [*g*] [*win*] [*sfb*] [*bin*] является интервалом квантования, который дается как:

$$interval\,[g]\,[win]\,[sfb]\,[bin] = thr\,(guant\,[g]\,[win]\,[sfb]\,[bin] + 1) - thr\,(guant\,[g]\,[win]\,[sfb]\,[bin] + 1).$$

Здесь *thr* (*x*) и *inv_quant* (*x*) являются соответственно детерминированным порогом квантования и соответствующим детерминированным инверсным квантованием для квантователя *AAC*. Они вычисляются в следующем псевдокоде:

```
If (x==0)
    thr(x)=0;
else
    thr(x) = (thrMantissa(| x |-1, scale res)) « (12+ scale int);
inv_quant(x) = (invQuantMantissa(|x , scale_res)) « (12+ scale_int);
```

где

scale_int = *DIV*(*scale*,4)

scale_res = *scale - scale_int**4, *u*
scale=scale_factor(sfb)+core_scaling_factor+scale_osf-118.
Величина *core_scaling_factor* дается в таблице 17.

Таблица 17 — Таблица для *core_scaling_factor*

Длина слова / Тип *sfb*	16	20	24
Длинное окно (2048), *M/S*	0	16	32
Длинное окно (2048), не *M/S*	2	18	34
Короткое окно (256), *M/S*	6	22	38
Короткое окно (256), не *M/S*	8	24	40

Таблица 18 — Таблица для *scale_osf*

osf	1	2	4
scale_osf	0	2	4

Для полос масштабного коэффициента, кодированных с помощью *IS* или *PNS*, значение *inv_quant* (x) устанавливается в 0.

Максимальная разрядная матрица *max_bp* для каждого *sfb* является максимальным значением *M* для спектральных данных, которые принадлежат этому *sfb*:

max_bp [g] [sfb] = max (M[g][win][sfb][bin])

5.5 Декодирование *lle_data*

5.5.1 Определения

lle_data () — Часть потока битов, которая содержит кодированные данные спектра остатка.

window_group_len [g] — Число окон в каждой группе.

is_lle_ics_eof () — Вспомогательная функция для обнаружения конца *LLE_ICS*.

read_bits (n) — Считать *n* последовательных битов из входного потока битов в порядке *bslbf*. Если достигается конец *LLE_ICS*, по умолчанию возвращается '0'.

cur_bp [g] [sfb] — Текущая декодированная разрядная матрица.

res [g] [win] [sfb] [k] — Восстановленный целочисленный вектор данных спектра остатка.

amp [g] [win] [sfb] [k] — Амплитуда восстановленного целочисленного вектора данных спектра остатка.

sign [g] [win] [sfb] [k] — Знак восстановленного целочисленного вектора данных спектра остатка.

determine_frequency () — Функция для определения вероятности символа '1' согласно таблице частот *CBAC* или *BPGC*.

ambiguity_check (f) — Функция для обнаружения неоднозначности для арифметического декодирования. Аргумент *f* указывает вероятность символа '1'.

terminate_decoding () — Функция для завершения декодирования данных LLE, когда случается неоднозначность.

smart_decoding_cbac_bpgc () — Функция для декодирования дополнительных символов в отсутствии входящих битов в режиме декодирования *cbac/bpgc*. Это декодирование продолжается до точки, где не существует никакой неоднозначности. Оно включает *ambiguity_check (f)* и *terminate_decoding ()*.

smart_decoding_low_energy () — Функция для декодирования дополнительных символов в отсутствии входящих битов в режиме декодирования с низкой энергией. Оно также включает *ambiguity_check (f)* и *terminate_decoding ()*.

5.5.2 Процесс декодирования

5.5.2.1 Краткий обзор

Целочисленный вектор данных спектра остатка декодируется из потока данных *LLE lle_data* (). Во-первых все полосы масштабного коэффициента с *lazy_bp*> 0 являются декодируемыми *BPGC/CBAC*, где

амплитуда и знак данных спектра остатка *res* являются декодируемой разрядной матрицей, начиная с максимальной разрядной матрицы *max_bp* и продолжая с нижними разрядными матрицами до разрядной матрицы 0 для каждой полосы масштабного коэффициента. Потом вызывается декодирование режима низкой энергии, чтобы декодировать остающиеся полосы масштабного коэффициента с *lazy_bp* <= 0.

Декодер *SLS* может обеспечить функциональность мелкоструктурированной масштабируемости (*FGS*), усекая поток битов *LLE*. Кроме того это позволяет декодировать дополнительные символы вне точки усечения, используя свойства арифметического кодирования.

5.5.2.2 Процесс декодирования *BPGC/CBAC*

Процесс декодировани *BPGC* или декодирования *CBAC* выполняется на полосах масштабного коэффициента, для которых *lazy_bp*> 0. Процесс декодирования разрядной матрицы *BPGC/CBAC* используется, чтобы декодировать символы разрядной матрицы для восстановления целочисленных спектральных данных остатка *res*. Процесс декодирования разрядной матрицы стартует с *max_bp* для каждого *sfb* и постепенно продолжается с нижними разрядными матрицами. Для начала разрядная матрица *NUM_BP* сканирует символы разрядной матрицы, которые являются арифметически декодированными, как иллюстрировано в следующем псевдокоде:

```
/* preparing the help element */
for (g=0;g<num_window_groups;g++){
   for (sfb = 0;sfb<num_sfb;sfb++){
        width = swb_offset[g][sfb+1] – swb_offset[g][sfb];
        for (win = 0;win <window_group_len[g];win++) {
           for (bin=0;bin<width;bin++){
               is_sig[g][win][sfb][bin] =
               (quant[g][sfb][win][bin])&&(band_type[g][sfb]==ImplicitBand)?1:0;
               /* sign will be determined implicitly if is_sig == 1 */
               amp[g][win][sfb][bin] = 0;
               sign[g][win][sfb][bin] = 1;
            }
        }
        cur_bp[g][sfb] = max_bp[g][sfb];
           }
      }

   /* BPGC/CBAC decoding */
   while ((max_bp[g][sfb] – cur_bp[g][sfb]<NUM_BP) && (cur_bp[g][sfb] >= 0)){
      for (g=0;g<num_window_groups;g++){
         for (sfb = 0;sfb<num_sfb;sfb++){
            if ((cur_bp[g][sfb]>=0) && (lazy_bp[g][sfb] > 0)){
               width = swb_offset[g][sfb+1] – swb_offset[g][sfb];
               for (win=0;win<window_group_len[g];win++){
                  for (bin=0;bin<width;bin++){
                     if (!is_lle_ics_eof ()){
                        if (interval[g][win][sfb][bin] >
                           amp[g][win][sfb][bin] + (1<<cur_bp[g][sfb])
                     {
                        freq = determine_frequency();
                        sym = decode(freq);
                        amp[g][win][sfb][bin] += sym << cur_bp[g][sfb];
                           /* decode bit-plane cur_bp*/
                        if ((!is_sig[g][win][sfb][bin]) && (sym )) {
                           /* decode sign bit of res if necessary */
                           sign[g][win][sfb][bin] = (decode(freq_sign))? -1:1;
                           is_sig[g][win][sfb][bin] = 1;
                           }
                        }
                     }
```

```
                    else {
                        smart_decoding_cbac_bpgc();
                    }
                }
            }
            cur_bp[g][sfb]- -; /* progress to next bit-plane */
        }
    }
  }
}
```

После этого *BPGC/CBAC* вводит режим декодирования *lazy* после пропуска 2 битов, завершающих строку, где символы разрядной матрицы непосредственно читаются из входного потока битов:

```
/* BPGC/CBAC lazy decoding */
read_bits(1);read_bits(1); /* skip the 2 AC termination string before lazy coding
while (cur_bp[g][sfb] >= 0){
  for (g=0;g<num_window_groups;g++){
    for (sfb = 0;sfb<num_sfb+num_osf_sfb;sfb++){
      if ((cur_bp[g][sfb]>=0) && (lazy_bp[g][sfb] > 0)){
        width = swb_offset[g][sfb+1] – swb_offset[g][sfb];
        for (win=0;win<window_group_len[g];win++){
          for (bin=0;bin<width;bin++)
            { if (!is_lle_ics_eof ()){
              if (interval[g][win][sfb][bin] >
                amp[g][win][sfb][bin] + (1<<cur_bp[g][sfb])
              {
                sym = read_bit();
                amp[g][win][sfb][bin] += sym << cur_bp[g][sfb];
                 /* decode bit-plane cur_bp */
                if ((!is_sig[g][win][sfb][bin]) && (sym )) {
                    /* decode sign bit of res if necessary */
                    sign[g][win][sfb][bin] = (decode(freq_sign))? -1:1;
                    is_sig[g][win][sfb][bin] = 1;
                 }
              }
            }
          }
        }
        cur_bp[g][sfb]- -;
        }
    }
  }
}
```

Значение *NUM_BP* определяется в таблице 19.

Таблица 19 — Значение *NUM_BP*

cb cbac	*NUM BP*
0 *(BPGC)*	4
1 *(CBAC)*	6

Частота присвоения вероятности *freq* в вышеупомянутом процессе декодирования *BPGC/CBAC* является частотой *BPGC freq_bpgc* или частотой *CBAC freq_cbac* в зависимости от того, декодируется ли текущее *LLE_ICS* с таблицей частот *BPGC* или *CBAC*. *freq_bpgc* определяется отношением *cur_bp* к параметру *lazy_bp*, как дано в таблице 20. Биты знака в вышеупомянутом процессе декодирования декодируются с частотой 8192, то есть *freq_sign* = 8192.

Таблица 20 — *freq_bpgc*

cur_bp	Частота *BPGC*
lazy_bp+3	64
lazy_bp+2	964
lazy_bp+1	3277
lazy_bp	5461
<*lazy_bp*	8192

Значение *freq_cbac* определяется контекстом символа разрядной матрицы, который в настоящий момент декодируется. Есть три типа контекста, используемого в *CBAC*.

Контекст 1: полоса частот (*fb*)

Контекст *fb* определяется индексом чередующихся спектральных данных остаточного *IntMDCT* с [*i*], *i*=0,... 1024**osf* - 1 и частотой дискретизации текущего уровня *LLE*, как показано в следующей таблице 21.

Таблица 21 — Контекст полосы частот (*fb*) [элемент разрешения по частоте]

Частота дискретизации / Номер контекста	44100	48000	96000	192000	Прочее
0 (нижняя полоса)	0—185	0—169	0—84	0—42	0—338
1 (средняя полоса)	186—511	170—469	85—234	43—117	339—938
2 (высокая полоса)	> 511	> 469	>234	>117	> 938

Контекст 2: существенное состояние (*ss*)

Для чередующихся спектральных данных остаточного *IntMDCT* с [*i*], *i*=0, ..., 1024* *osf*-1, которые являются несущественными (символы разрядной матрицы *c*[*i*], декодированные до тех пор, пока не являются все нулями), контекст *ss* определяется значением его смежных спектральных данных:

$$sig_cx(i, bp) = \{sig_state(i-2, bp), sig_state(i-1, bp), sig_state(i+1, bp), sig_state(i+1, bp)\}$$

где $sig_state(i, bp)$ определяется как

$$sig_state(i, bp) = \begin{cases} 0\ c[i] \text{ несущественен для } bitplane\ br \\ 1\ c[i] \text{ существенен для } bitplane\ br \end{cases}$$

и $sig_state(i, bp)$ определяется как 0, если *i* меньше 0, или больше длины *IntMDCT*.

Для *c*[*i*], который является существенным, контекст *ss* определяется типом полосы масштабного коэффициента, которую он формирует:

$$sig_core(i) = \begin{cases} 0\ c[i] \text{ из } Explicit_Band \\ 1\ c[i] \text{ из } Implicit_Band \end{cases}$$

Кроме того, для последнего случая контекст *ss* далее определяется согласно значению $quant_interval(i, bp)$, установленному как:

$$guant_interval(i, bp) = \begin{cases} 0 \text{ если } rec_spectrum[i] + 2^{bp+1} \le \text{int}erval[i] \\ 1 \text{ если } rec_spectrum[i] + 2^{bp} \le \text{int}erval[i] < rec_spectrum[i] + 2^{bp+1} \end{cases}$$

Присвоение контексту *ss* обобщается в следующей таблице 22.

Таблица 22 — Контекст состояния значения (*ss*)

Номер контекста	*sig_state(i, cur_bp)*	*sig_cx(i, cur_bp)*	*sig_core (i)*	*quant_jnterval (i)*
0	0	{0,0,0,0}	x	x
1	0	{0,0,0,1} {1,0,0,0}	x	x
2	0	{0,0,1,0} {0,1,0,0}	x	x
3	0	{0,0,1,1} {1,1,0,0}	x	x
4	0	{0,1,0,1} {1,0,1,0}	x	x
5	0	{0,1,1,0}	x	x
6	0	{0,1,1,1} {1,1,1,0}	x	x
7	0	{1,0,0,1}	x	x
8	0	{1,0,1,1} {1,1,0,1}	x	x
9	0	{1,1,1,1}	x	x
10	1	x	0	x
11	1	x	1	0
12	1	x	1	1

Контекст 3: расстояние до *lazy* (*d2l*)

Контекст *d2l* определяется расстоянием *cur_bp* до параметра *lazy_bp* декодируемого в момент символа разрядной матрицы. Подробно назначение перечислено в таблице 23.

Таблица 23 — Расстояние до контекста *lazy* (*D2L*)

Номер контекста	*cur_bp - lazy_bp*
0	<–2
1	–2
2	–1
3	0
4	1
5	2
6	3

Частоты *freq_cbac* для каждого контекста даются в таблице 24.

Таблица 24 — Частоты *freq_cbac*

d2l / *fb**13+*ss*	0	1	2	3	4	5	6
0	8192	7823	7826	6506	4817	2186	1053
1	8192	8344	7983	6440	4202	1362	64
2	8192	8399	8382	7016	4202	1234	64
3	8192	8305	7960	6365	3963	1285	64
4	8192	8335	8146	6655	3746	825	64
5	8192	8473	8244	6726	3929	927	64
6	8192	8398	7919	6098	3581	875	64
7	8192	8359	8028	6382	3459	631	64
8	8192	8192	8192	5461	3277	964	64
9	8192	8333	7481	5288	3076	732	64
10	8192	7658	6898	5145	1424	1636	64
11	8192	5471	5732	6264	4890	1279	93
12	8192	8180	8136	7897	5715	1553	64
13	8192	7242	6876	6083	3604	1214	950
14	8192	7897	7570	6583	3733	1067	900
15	8192	8071	7928	7069	4294	1406	1200
16	8192	8197	7952	6906	4050	1457	1101
17	8192	8278	8039	7094	4160	1381	64
18	8192	8307	8139	7263	4407	1555	64
19	8192	8339	8124	7065	4074	1636	64
20	8192	8213	7918	6827	3787	1161	64
21	8192	8286	8067	6902	3855	1387	64
22	8192	8336	8072	6705	3731	1558	64
23	8192	7636	6962	5036	1985	1037	64
24	8192	5519	5270	5238	4778	1588	219
25	8192	7884	7528	6743	4848	1970	64
26	8192	6084	6323	5929	3321	900	385
27	8192	7862	7618	6728	4409	1431	1302
28	8192	8078	7871	7081	5119	2371	1670
29	8192	8294	8046	7239	5218	2032	967
30	8192	8378	8119	7351	5413	1947	64
31	8192	8378	8207	7491	5624	2444	64
32	8192	8484	8302	7626	5514	2021	64
33	8192	8302	8006	7192	4941	1561	64
34	8192	8464	8246	7510	5217	1780	64
35	8192	8544	8442	7742	4944	2010	64
36	8192	7556	6771	4859	2638	2155	64
37	8192	5916	4780	4713	4239	1240	182
38	8192	7658	7095	5986	3886	1394	64

5.5.2.3 Декодирование кода низкоэнергетического режима (*LEMC*)

Следующий псевдокод иллюстрирует процесс декодирования *LEMC*, который выполняется на полосах масштабного коэффициента, для которых *lazy_bp* <=0.

```
/* low energy mode decoding */
for (g = 0;g < num_window_groups; g++){
  for (sfb = 0; sfb <num_sfb+num_osf_sfb;sfb++){
  if ((cur_bp[g][sfb] >= 0) && (lazy_bp[g][sfb] <= 0))
  {
        width = swb_offset[g][sfb+1] – swb_offset[g][sfb];
for (win=0;win<window_group_len[g];win++){
  for (bin=0;bin<width;bin++){
     res[g][sfb][win][bin] = 0;
      pos = 0;
        /* decoding of binary string and reconstructing res */
        while (1) {
        if (is_lle_ics_eof())smart_decoding_low_energy(pos, res[g][sfb][win][bin]);
        if (decode(freq_silence[pos])==0) break ;
        res[g][sfb][win][bin] ++;
        pos++;
        if (pos>2) pos = 2;
        if (res[g][sfb][win][bin]==(1<<(max_bp[g][sfb]+1))-1) break;
        }
     /* decoding of sign of res */
     if (!is_sig[g][win][sfb][bin]) && res[g][sfb][win][bin]){
        res[g][sfb][win][bin] *= (decode(freq_sign))? -1:1;
        is_sig[g][win][sfb][bin] = 1;
      }
     }
    }
   }
  }
 }
```

Присвоения вероятности для декодирования низкоэнергетического режима, *freq_bpgc* и *freq_silence* даются в таблице 25. Биты знака в вышеупомянутом процессе декодирования декодируются с частотой 8192, то есть *freq_sign* = 8192.

Таблица 25 — *freq_silence*

lazy_bp / Позиция	0	–1	–2	–3
0	12603	9638	6554	3810
1	7447	3344	1820	X
>1	6302	745	552	X

Таблица 26 определяет отображение между двоичной строкой, декодируемой в случае низкоэнергетического режима и остаточными спектральными данными *res*. Бит знака *res* декодируется после того, как декодировался первый ненулевой символ разрядной матрицы.

Таблица 26 — Преобразование *res* в двоичную форму в режиме низкоэнергетического кодирования

Амплитуда *res[g][win][sfb][bin]*	Двоичная строка
0	0
1	1 0
2	1 1 0
3	1 1 1 0
4	1 1 1 1 0
...	...
2^(*max_bp*[*g*][*sfb*]+1)-2	1 1 ... ... 1 0
2^ (*max_bp*[*g*][*sfb*]+1)-1	1 1 ... ... 1 1
pos	0 1 2 3 ...

5.5.2.4 Арифметическое декодирование

Следующий псевдокод иллюстрирует процесс целочисленного арифметического декодирования, используемый в *BPGC/CBAC* и процессе декодирования низкоэнергетического режима.

Определения:

```
#define CODE_WL     16
#define PRE_SHT     14
#define TOP_VALUE (((long)1<<CODE_WL)-1)
#define QTR_VALUE (TOP_VALUE/4+1)
#define HALF_VALUE (2*QTR_VALUE)
#define TRDQTR_VALUE    (3*QTR_VALUE)
```

Инициализация:

```
low = 0;
high = TOP_VALUE;
value = 0;
```

Подпрограмма декодирования

```
int decode(int freq)
{
    range = (long)((high-low)+1);
    cumu = ((long)((value-low)+1)<<PRE_SHT);
    if (cumu<range*freq) {
       sym = 1;
       high = low + (range*freq>>PRE_SHT)-1;
     }
    else   {
       sym = 0;
          low = low + (range*freq>>PRE_SHT);
    }
       for (;;)    {
           if (high<HALF_VALUE) {
       } else if (low>=HALF_VALUE) {
             value -= HALF_VALUE;
             low -= HALF_VALUE;
             high -= HALF_VALUE;
           } else if (low>=QTR_VALUE && high<TRDQTR_VALUE) {
             value -= QTR_VALUE;
               low -= QTR_VALUE;
               high -= QTR_VALUE;
           } else
```

```
            break;
        low = 2*low;
        high = 2*high+1;
        value = 2*value + read_bits(1); /* input next bit from bit-stream */
    }
    return sym;
}
```

5.5.2.5 Интеллектуальное арифметическое декодирование усеченных потоков битов *SLS*

Инеллектуальное арифметическое декодирование обеспечивает эффективный способ декодирования промежуточного уровня, соответствующий данной целевой скорости передачи. Этот алгоритм использует тот факт, что буфер декодирования все еще содержит значимую информацию для арифметического декодирования, даже если не остается битов для подачи в буфер декодирования. Процесс декодирования продолжается, пока существует некоторая неоднозначность в определении символа.

Следующий псевдокод иллюстрирует алгоритм обнаружения неоднозначности в модуле арифметического декодирования. Переменная *num_dummy_bits* представляет число вызовов функции *read_bits (1)* в процессе арифметического декодирования сразу после точки усечения.

```
int ambiguity_check(int freq)
{
    /* if there is no ambiguity, returns 1      */
    /* otherwise, returns 0                     */
    upper = 1<<num_dummy_bits;
    decisionVal = ((high-low)*freq>>PRE_SHT)-value+low-1;
    if(decisionVal>upper || decisionVal<0) return 0;
    else return 1;
}
```

Когда *num_dummy_bits* больше 0, выполняется *smart_decoding_cbac_bpgc ()* или *smart_decoding_low_energy ()*. Чтобы предотвратить ошибки знакового бита, спектральное значение текущей линии спектра должно быть обнулено, когда может произойти неоднозначность при декодировании знакового бита. Все индексные переменные в интеллектуальном процессе декодирования должны быть перенесены из предыдущего процесса арифметического декодирования.

```
smart_decoding_cbac_bpgc()
{
    /* BPGC/CBAC normal decoding with ambiguity detection */
    while ((max_bp[g][sfb] - cur_bp[g][sfb]<LAZY_BP) && (cur_bp[g][sfb] >= 0)){
        for (;g<num_window_groups;g++){
            for (;sfb<num_sfb;sfb++){
                if ((cur_bp[g][sfb]>=0) && (lazy_bp[g][sfb] > 0)){
                    width = swb_offset[g][sfb+1] - swb_offset[g][sfb];
                    for (;win<window_group_len[g];win++){
                        for (;bin<width;bin++){
                            if (interval[g][win][sfb][bin] >
                              amp[g][win][sfb][bin] + (1<<cur_bp[g][sfb])
                            {
                            freq = determine_frequency();
                            if (ambiguity_check(freq)) {
                             /* no ambiguity for arithmetic decoding */
                              sym = decode(freq);
                             amp[g][win][sfb][bin] += sym << cur_bp[g][sfb];
                             /* decode bit-plane cur_bp*/
                             if ((!is_sig[g][win][sfb][bin]) && (sym)) {
                                /* decode sign bit of res if necessary */
                                if (ambiguity_check(freq)) {
                                    sym = decode(freq);
                                    sign[g][win][sfb][bin] = (sym)? -1:1;
                                    is_sig[g][win][sfb][bin] = 1;
```

```
                }
                else {
                    /* discard the decoded symbol prior to sign symbol */
                    amp[g][win][sfb][bin] = 0;
                     terminate_decoding();
                    }
                  }
                 }
                  else terminate_decoding();
                }
               }
              }
              cur_bp[g][sfb] - - ; /* progress to next bit-plane */
             }
            }
           }
          }
         }
         smart_decoding_low_energy(int posPrev,int resPrev)
         {
              /* low energy mode decoding */
              for (;g < num_window_groups; g++){
                 for (; sfb <num_sfb+num_osf_sfb;sfb++){
                    if ((cur_bp[g][sfb] >= 0) && (lazy_bp[g][sfb] <= 0))
                    {
              width = swb_offset[g][sfb+1] – swb_offset[g][sfb];
              for (;win<window_group_len[g];win++){
                 for (;bin<width;bin++){
                    /* Continue to decode after the end of LLE_ICS is reached */
                    res[g][sfb][win][bin] = resPrev; resPrev = 0;
                    pos = posPrev; posPrev = 0;
                    while (1) {
                       /* if ambiguity check is false, discard the spectrum is set to be 0 */
                       if(!ambiguity_check(freq)) res[g][sfb][win][bin] = 0,
                       terminate_decoding();
                       tmp = decode(freq_silence[pos]);
                       if(tmp==0) break;
                        res[g][sfb][win][bin] ++;
                       pos++;
                       if (pos>2) pos = 2;
                       if (res[g][sfb][win][bin]==(1<<(max_bp[g][sfb]+1))-1) break;
                 }
                 /* decoding of sign of res */
                 if (!is_sig[g][win][sfb][bin]) && res[g][sfb][win][bin]){
                 /* if ambiguity check is false,the current spectrum value is set to be 0 */
                 if(!ambiguity_check(freq)) res[g][sfb][win][bin] = 0,
                 terminate_decoding();
                 res[g][sfb][win][bin] *= (decode(freq_sign)) ? -1:1;
                 is_sig[g][win][sfb][bin] = 1;
                }
               }
              }
             }
            }
           }
          }
         }
```

5.6 Компенсация для остатка *IntMDCT* для раннего завершения декодирования *BPGC/CBAC*

Если процесс декодирования *BPGC/CBAC* завершается рано из-за усечения *LLE_ICS, res* компенсируется элементом *res_fill* следующим образом:

```
for (g=0;g<num_window_groups;g++){
    for (sfb = 0;sfb<num_sfb+num_osf_sfb;sfb++){
        width = swb_offset[g][sfb+1] – swb_offset[g][sfb];
        for (win=0;win<window_group_len[g];win++){
            for (bin=0;bin<width;bin++){
                if (is_sig[g][win][sfb][bin]){
                    if (res[g][win][sfb][bin] >= 0) res[g][win][sfb][bin] += (res_fill >>
                    (23-stop_bp[g][win][sfb][bin]));
                    else if (res[g][win][sfb][bin] < 0) res[g][win][sfb][bin] -=
                    (res_fill >> (23-stop_bp[g][win][sfb][bin]));
                    }
                }
            }
        }
    }
```

Здесь *stop_bp[g][win][sfb][bin]* является наивысшей разрядной матрицей, для которой символ разрядной матрицы не декодируется из-за раннего завершения *LLE_ICS*. Значение *res_fill* дается в таблице 27.

Таблица 27 — Значение *res_fill*

stop_bp	*res_fill*
lazy_bp+3	1572608
lazy_bp+2	3079935
lazy_bp+1	5172975
lazy_bp	6990507
<lazy_bp	8388607

5.7 Инверсное отображение ошибки

5.7.1 Принцип

Процесс инверсного отображения ошибки используется, чтобы восстановить спектральные данные *IntMDCT* из остаточных данных *IntMDCT* из уровня *LLE* и квантованных спектральных данных *MDCT* из базового уровня. Этот процесс применяется только в диапазоне непередискретизации. Ввод в инструмент инверсного отображения ошибки является амплитудой *amp* и знаком *sign* спектральных данных остатка *res* и квантованных спектральных данных *quant*. Его выводом являются восстановленные спектральные данные *IntMDCT c*. Процедура инверсного отображения ошибки описывается следующим образом:

```
res[g][win][sfb][bin]= sign[g][win][sfb][bin]* amp[g][win][sfb][bin];
if (quant[g][win][sfb][bin]==0)
    c[g][win][sfb][bin]=res[g][win][sfb][bin];
else
    c[g][win][sfb][bin] = sign(quant[g][win][sfb][bin]) *
                    (res[g][win][sfb][bin]+ref(quant[g][win][sfb][bin]);
```

Чтобы гарантировать кодирование без потерь, в кодере *SLS* должна использоваться следующая процедура отображения ошибки для того же самого спектра частот:

```
if (quant[g][win][sfb][bin]==0)
    res[g][win][sfb][bin]=c[g][win][sfb][bin];
else
    res[g][win][sfb][bin]=sign(quant[g][win][sfb][bin])* c[g][win][sfb][bin]-
ref(quant[g][win][sfb][bin]);
```

Функция *ref(x)* в вышеприведенном процессе детерминированно вычисляется следующим образом

```
if ((sfb is Implicit_Band) then
    ref(x) = thr(abs(x))
else if (sfb is Explicit_Band)
    ref(x) = inv_quant(abs(x))
```

Здесь вычисление *thr*() и *inv_quan* () следует 5.4.2.3.

5.8 Целочисленный процесс *Mid/Side*

Если для левого и правого каналов используется *Mono IntMDCT* (*common_window* = = 0 или *use_stereo_intmdct* = = 0), к полосам масштабного коэффициента должна быть применена инверсная целочисленная обработка *M/S*, где флаг *M/S* включен.

Декодирование *Mid/Side* (*M/S*) выполняется на целочисленных спектральных значениях, используя схему подъема следующим образом:

Шаг 1:

$S = S - NINT\ (c_1 \cdot M);$

Шаг 2:

$M = M - NINT\ (c_2 \cdot S);$

Шаг 3:

$R = M;$

$L = S - NINT\ (c_1 \cdot R),$

где *M, S, R, L* обозначает спектральные данные каналов *Mid, Side, Left* и *Right*, а также

$$c_1 = \left(\cos\frac{\pi}{2} - 1\right) / \sin\frac{\pi}{2} \text{ и } c_2 = \sin\frac{\pi}{4}.$$

Инверсный Stereo *IntMDCT* ожидает спектр *M/S* по умолчанию. Следовательно *M/S* должно быть применено к полосам масштабного коэффициента там, где флаг *M/S* выключается.

Кодирование *Mid/Side* (*M/S*) выполняется на целочисленных спектральных значениях, используя схему подъема следующим образом:

Шаг 1:

$S = R;$

$M = L + NINT\ (c_1 \cdot R);$

Шаг 2:

$S = S + NINT\ (c_1 \cdot M);$

Шаг 3:

$M = M + NINT\ (c_1 \cdot S),$

где *M, S, L, R* обозначают спектральные данные каналов *Mid, Side, Left*, и *Right* и

$$c_1 = \left(\cos\frac{\pi}{2} - 1\right) / \sin\frac{\pi}{2} \text{ и } c_2 = \sin\frac{\pi}{4}.$$

5.9 Целочисленное формирование временного шума (*IntTNS*)

Когда в ядре *AAC* используется формирование временного шума (*TNS*), тот же самый фильтр *TNS* применяется к целочисленным спектральным значениям в *SLS*. Чтобы преобразовать этот фильтр в детерминированный обратимый целочисленный фильтр, требуются следующее :

Для того, чтобы определить коэффициенты *LPC*, используется функция *int_tns_decode_coef* (), как описано в следующем псевдокоде:

```
define SHIFT_INTTNS 21
INT32 tnsInvQuantCoefFixedPoint(coef_res, coef)
{
INT32 intTnsCoef_res3[8] = {-2065292, -1816187, -1348023, -717268,
                            0, 909920, 1639620, 2044572};
INT32 intTnsCoef_res4[16] = {-2088206, -2017095, -1877294, -1673563, -1412842, -1104008,
                             -757579, -385351, 0, 436022, 852989, 1232675, 1558488, 1816187, 1994510,
                             2085664};
    if (coef_res == 3) {
        return intTnsCoef_res3[4+coef];
```

```
        }
        if (coef_res == 4) {
            return intTnsCoef_res4[8+coef];
        }
}
/* Decoder transmitted coefficients for one TNS filter */
int_tns_decode_coef( order, coef_res, *coef, INT32 *a )
{
        INT32 tmp[TNS_MAX_ORDER+1], b[TNS_MAX_ORDER+1];
        /* Inverse quantization */
        for (i=0; i<order; i++) {
          tmp[i+1] = tnsInvQuantCoefFixedPoint(coef_res, coef[i]);
        }
          /* Conversion to LPC coefficients */
          /* worst case for order == 12 and all coefficients == 1:
             6th coefficient raised by 12!/(6!*6!) = 924
             -> 10 bits headroom required -> SHIFT_INTTNS == 21 */
a[0] = 1<<SHIFT_INTTNS;
for (m=1; m<=order; m++)  {
        b[0] = a[0];
        for (i=1; i<m; i++)  {
              b[i] = a[i] + ((((((INT64)tmp[m])*a[m-i])>>(SHIFT_INTTNS-1))+1)>>1);
        }
        b[m] = tmp[m];
        for (i=0; i<=m; i++)  {
            a[i] = b[i];
        }
    }
}
```

Основанная на получающихся коэффициентах *LPC* с фиксированной точкой детерминированная целочисленная версия фильтра *TNS* применяется к целочисленному спектру в декодере. Это делается путем замены функции *tns_ar_filter* () функцией *int_tns_ar_filter* (), описываемой следующим псевдокодом:

```
int_tns_ar_filter(INT32 *spec, size, inc, INT32 *lpc, order )
{
        INT32 y, state[TNS_MAX_ORDER];
        INT64 temp_accu;
        for (i=0; i<order; i++)
            state[i] = 0;
        if (inc == -1)
             spec += size-1;
        for (i=0; i<size; i++) {
            y = *spec;
            temp_accu = 0;
            for (j=0; j<order; j++) {
                temp_accu += ((INT64)lpc[j+1]) * state[j];
            }
            y -= (INT32)( ( ( temp_accu >> (SHIFT_INTTNS-1) ) + 1) >> 1 );
            for (j=order-1; j>0; j—)
                state[j] = state[j-1];
            state[0] = y;
            *spec = y;
            spec += inc;
        }
    }
```

Если используется *StereoIntMDCT*, вместо спектра *L/R* целочисленные спектральные значения представляют спектр *M/S*. В этом случае перед *IntTNS* должно быть применено инверсное целочисленное *M/S*, а после должно быть применено прямое целочисленное *M/S*.

Чтобы гарантировать реконструкцию без потерь, в кодере должен быть применен к целочисленному спектру соответствующий прямой прогноз *LPC*. Это достигается применением к целочисленному спектру функции *int_tns_decode_coef*() и соответствующего прямого фильтра, как описано в следующем псевдокоде:

```
int_tns_filter_encode(length, order, direction, INT32* spec, INT32 *lpc)
{
    INT64 temp_accu;
    if (direction) {
      /* Startup, initial state is zero */
      temp[length-1]=spec[length-1];
      for (i=length-2;i>(length-1-order);i - -) {
            temp[i]=spec[i];
            temp_accu = 0;
            k++;
            for (j=1;j<=k;j++) {
                temp_accu += ((INT64)temp[i+j]) * a[j];
            }
            spec[i] += (INT32)( ( ( temp_accu >> (SHIFT_INTTNS-1) ) + 1) >> 1);
      }
      /* Now filter the rest */
      for  (i=length-1-order;i>=0;i - -) {
            temp[i]=spec[i];
            temp_accu = 0;
            for (j=1;j<=order;j++) {
                temp_accu += ((INT64)temp[i+j]) * a[j];
            }
            spec[i] += (INT32)( ( ( temp_accu >> (SHIFT_INTTNS-1) ) + 1) >> 1 );
      }
    } else {
      /* Startup, initial state is zero */
      temp[0]=spec[0];
      for (i=1;i<order;i++) {
                temp[i]=spec[i];
                temp_accu = 0;
                for (j=1;j<=i;j++) {
                   temp_accu += ((INT64)temp[i-j]) * a[j];
      }
      spec[i] += (INT32)( ( ( temp_accu >> (SHIFT_INTTNS-1) ) + 1) >> 1 );
     }
     /* Now filter the rest */
     for  (i=order;i<length;i++) {
        temp[i]=spec[i];
        temp_accu = 0;
        for (j=1;j<=order;j++) {
            temp_accu += ((INT64)temp[i-j])*a[j];
        }
        spec[i] += (INT32)( ( ( temp_accu >> (SHIFT_INTTNS-1) ) + 1) >> 1 );
      }
    }
  }
```

В случае, если используется *StereoIntMDCT*, целочисленные спектральные значения представляют спектр *M/S* вместо спектра *L/R*. В этом случае инверсное целочисленное *M/S* должно быть применено перед *IntTNS* и прямое целочисленное *M/S* должно быть применено после.

5.10 *IntMDCT* и инверсное *IntMDCT*

5.10.1 Описание

IntMDCT является обратимым целочисленным приближением *MDCT*. Следующий раздел описывает структурную реализацию *MDCT* и *IMDCT*, используемую для прямого и инверсного *IntMDCT*.

В дальнейшем длина фрейма *N* всегда обозначает число новых входных выборок в каждом блоке, которое равно числу значений частоты, таким образом *N* является *osf**1024 или *osf**128.

5.10.2 *MDCT* и инверсный *MDCT* (*IMDCT*)

MDCT определяется выражением

$$X(m) = \sqrt{\frac{2}{N}} \sum_{k=0}^{2N-1} \omega(k) x(k) \cos \frac{(2k+1+N)(2m+1)\pi}{4N}, \text{ где } m = 0,\ldots..,\ N-1$$

N: Длина фрейма (*osf**1024 или *osf**128)

X (m): Значения спектра MDCT

x (k): Входные выборки

ω *(k)*: Функция окна (*Sine* или *KBD*)

IMDCT определяется выражением

$$x(k) = \omega(k) \sqrt{\frac{2}{N}} \sum_{m=0}^{N-1} x(m) \cos \frac{(2k+1+N)(2m+1)\pi}{4N}, \text{ где } k = 0,\ldots..,\ 2N-1$$

Вход в *MDCT* и выход из *IMDCT* имеют 50% перекрытие, то есть *N* выборок. В *IMDCT* вывод двух последующих блоков добавляется в области перекрытия.

5.10.2.1 *MDCT* и *IMDCT* посредством *DCT*-IV

MDCT и *IMDCT* для *IntMDCT* делятся на два блока:

работа с окнами (*Windowing*) и искажение временного домена (*TDA*);

дискретное косинусное преобразование типа IV (*DCT*-IV).

5.10.2.2 Вычисление и работа с окнами блока *TDA*

5.10.2.3 Структура *MDCT* и *IMDCT* для различных последовательностей окон

В *MDCT* блок *Windowing* (работа с окнами) блока *TDA* вычисляется согласно

$$\begin{pmatrix} x(k) \\ x(N-1-k) \end{pmatrix} \mapsto \begin{pmatrix} w(N-1-k & w(k) \\ -w(k) & w(N-1-k \end{pmatrix} \begin{pmatrix} x(k) \\ x(N-1-k \end{pmatrix}, \text{ где } k = 0,\ldots..,\ N/2-1.$$

В *IMDCT* этот блок инвертируется согласно

$$\begin{pmatrix} x(k) \\ x(N-1-k \end{pmatrix} \mapsto \begin{pmatrix} w(N-1-k) & -w(k) \\ w(k) & w(N-1-k \end{pmatrix} \begin{pmatrix} x(k) \\ x(N-1-k \end{pmatrix}, \text{ где } k = 0,\ldots..,\ N/2-1.$$

Операция наложения/добавления уже содержится в этом вычислении.

DCT-IV длины *N* определяется так:

$$x(m) = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} x(k) \cos \frac{(2k+1)(2m+1)\pi}{4N}, \text{ где } m = 0,\ldots..,\ N-1$$

Инверсный *DCT*-IV длины *N* имеет те же самые коэффициенты, это определяется как:

$$x(k) = \sqrt{\frac{2}{N}} \sum_{m=0}^{N-1} x(m) \cos \frac{(2k+1)(2m+1)\pi}{4N}, \text{ где } k = 0,\ldots..,\ N-1.$$

Для вычисления *MDCT* рассматривают вывод двух последующих этапов *Windowing/TDA*. Пусть *x'*(0), ...*x'* (*N*-1) будет выводом этапа *Windowing/TDA* предыдущего блока и *x'*(*N*), ..., *x'* (2*N*-1) будет выводом этапа *Windowing/TDA* текущего блока. Затем *DCT*-IV применяется к значениям *N*

$-x'(N + N/2 - 1),\ -x'(N + N/2 - 2),\ldots..,\ -x'(N),\ -x'(N-1),\ -x'(N-2),\ldots..,\ -x'(N/2).$

То есть используются вторая половина предыдущего блока и первая половина текущего блока. Порядок значений возвращается, и значения умножаются на -1 прежде, чем применить *DCT-IV* длины *N*.

Вторая половина текущего блока выходных значений *Windowing/TDA* должна быть сохранена для следующего блока.

Для вычисления *IMDCT* спектральные значения *MDCT* преобразовываются инверсным *DCT*-IV, вывод умножается на -1, и порядок инвертируется. Затем вторая половина сохраняется для следующего блока, первая половина обрабатывается инверсным блоком *Windowing/TDA* вместе со значениями, сохраненными из предыдущего блока.

5.10.2.4 Структура *MDCT* и *IMDCT* для различных последовательностей окон

На рисунке 1 и 2 структура иллюстрируется для типичных последовательностей окон.

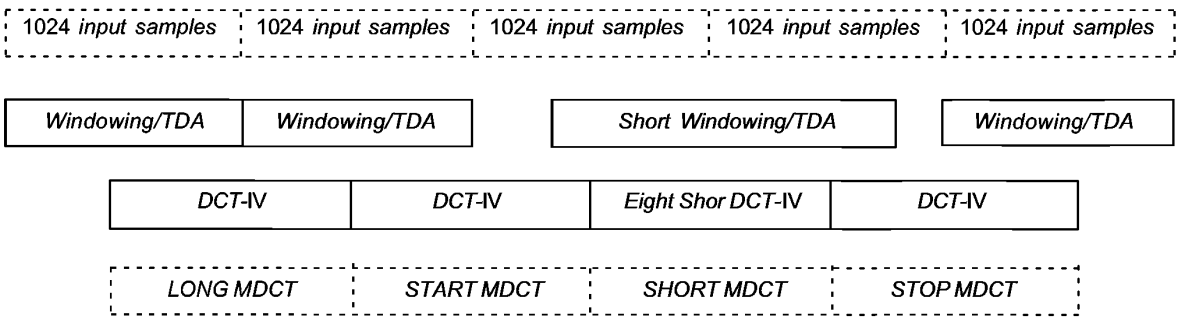


Рис. 1 — Структура *MDCT* для последовательности *LONG, START, SHORT, STOP*

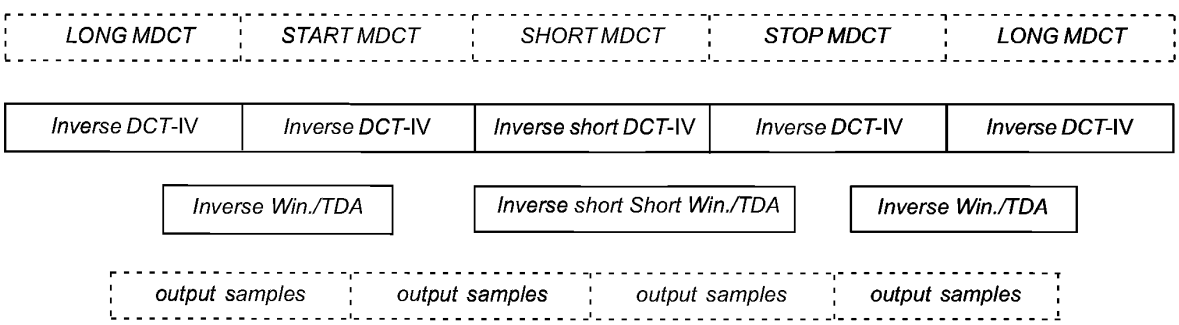


Рис. 2 — Структура *IMDCT* для последовательности *LONG, START, SHORT, STOP, LONG*

5.10.2.5 *IntMDCT*

IntMDCT является инверсным целочисленным приближением *MDCT*. Здесь используются две версии этого преобразования, полагаясь на тот же самый алгоритм:

Mono-IntMDCT: Эта версия обеспечивает спектр *IntMDCT* одного канала.

Stereo-IntMDCT: В случае элемента пары каналов с включенными *common_window* и *use_stereo_intmdct*, используется эта версия. Она обеспечивает спектр *Mid/Side IntMDCT* левого и правого каналов одновременно.

Разложение *MDCT* на шаги подъема

Для *IntMDCT* все вычисления разлагаются на так называемые шаги подъема, позволяя вводить операцию округления, не теряя свойство совершенной реконструкции.

В прямом *IntMDCT* блок *Windowing/TDA* вычисляется по 3*N*/2 шагам подъема:

$$\begin{pmatrix} x(k) \\ x(N-1-k) \end{pmatrix} \mapsto \begin{pmatrix} 1 & -\frac{w(N-1-k)-1}{w(k)} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -w(k) & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{w(N-1-k)-1}{w(k)} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x(k) \\ x(N-1-k \end{pmatrix},$$

где $k = 0,.....,N/2-1$

В инверсном *IntMDCT* блок *Windowing/TDA* вычисляется согласно:

$$\begin{pmatrix} x(k) \\ x(N-1-k) \end{pmatrix} \mapsto \begin{pmatrix} 1 & -\frac{w(N-1-k)-1}{w(k)} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ w(k) & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{w(N-1-k)-1}{w(k)} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x(k) \\ x(N-1-k) \end{pmatrix},$$

где $k = 0,.....,N/2-1$

Эти вычисления математически эквивалентны вычислению, описанному выше, потому что функция окна $w(k)$ выполняет условие *TDAC* $w(k)^2 + w(N-1-k)^2 = 1$, $k=0, N/2-1$.

После каждого шага подъема применяется операция округления, чтобы остаться в целочисленном домене.

Вычисление *Int-DCT*-IV

Для *IntMDCT DCT*- IV вычисляется обратимым целочисленным способом, названным *Int-DCT*-IV. *Int-DCT*-IV длины *N* реализуется так называемыми многомерными шагами подъема. У них имеется следующая общая структура:

$$\begin{pmatrix} I_{N/2} & 0 \\ A & I_{N/2} \end{pmatrix}$$

с матрицей идентичности $I_{N/2}$ размера $N/2$ и произвольной A матрицей $(N/2)\times(N/2)$.

Применение этой блочной матрицы означает, что первая половина входных значений обрабатывается матрицей A и затем добавляется ко второй половине входных значений.

Для целочисленного приближения выходные значения матрицы A, прежде чем их добавить, округляются до целого числа.

Этот процесс может быть инвертирован с помощью

$$\begin{pmatrix} I_{N/2} & 0 \\ -A & I_{N/2} \end{pmatrix}.$$

То есть та же самая матрица A применяется к первой половине значений, и получающиеся значения вычитаются из второй половины входных значений.

Для обратимого целочисленного приближения выходные значения A округляются до целого числа прежде, чем вычесть их.

Чтобы применить эту структуру к *IntMDCT*, *DCT*-IV длины *N* разлагается следующим образом:

$$DCT\ \mathrm{IV}_N = \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix} \begin{pmatrix} 1 & & & & & & cs(0) \\ & \ddots & & & & \iddots & \\ & & \ddots & & \iddots & & \\ & & & 1 & & & \\ & & \ddots & cs(N/2-1) & & \iddots & \\ & \iddots & & & \ddots & & \\ cs(0) & & & & & & 1 \end{pmatrix} \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ & & & s(N/2-1) & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix} \begin{pmatrix} 1 & & & & & & cs(0) \\ & \ddots & & & & \iddots & \\ & & \ddots & & \iddots & & \\ & & & 1 & & & \\ & & \ddots & cs(N/2-1) & & \iddots & \\ & \iddots & & & \ddots & & \\ & & & & & & 1 \end{pmatrix}$$

$$\begin{pmatrix} I & \frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_{N/2} \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ -\frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_{N/2} & I \end{pmatrix} \begin{pmatrix} I & \frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_{N/2} \\ 0 & I \end{pmatrix} \begin{pmatrix} I & -\frac{1}{2}I \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ I & I \end{pmatrix} QP$$

со значениями

$$S(k) = \sin\frac{(2k+1)\pi}{4N},\quad cs(k) = \frac{\cos\frac{(2k+1)\pi}{4N} - 1}{\sin\frac{(2k+1)\pi}{4N}},\quad k = 0,.....,N/2-1$$

и матрицами перестановки P и Q с

$P_{4k,\,4k} = P_{4k+1,\,4k+1} = P_{4k+2,\,4k+3} = P_{4k+3,\,4k+2} = 1,\ k = 0,....,N/4-1$

$P_{k,l} = 0$ в других случаях,

то есть каждая вторая пара значений переставляется, и

$Q_{k,2k} = Q_{N/2+k,2k+1} = 1,\ k = 0,...,N/2-1$

$Q_{k,l} = 0$ в других случаях,

то есть четные индексы располагаются в первой половине, нечетные индексы располагаются во второй половине.

Таким образом *DCT*-IV в основном разлагается на 8 многомерных шагов подъема.

Соответствующее инверсное разложением подъема для инверсного *DCT*-IV дает выражение:

$$DCT\ \mathrm{IV}_N^{-1} = P^{-1}\,Q^{-1}\begin{pmatrix} I & 0 \\ -I & I \end{pmatrix}\begin{pmatrix} I & \frac{1}{2}I \\ 0 & I \end{pmatrix}\begin{pmatrix} I & -\frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_{N/2} \\ 0 & I \end{pmatrix}\begin{pmatrix} I & 0 \\ \frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_{N/2} & I \end{pmatrix}\begin{pmatrix} I & -\frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_{N/2} \\ 0 & I \end{pmatrix}$$

$$\begin{pmatrix} 1 & & & & -cs(0) \\ & \ddots & & \iddots & \\ & & \ddots & & \\ & 1 & & -cs(N/2-1) & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}\begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & 1 & \\ & & s(N/2-1) & & 1 \\ & & & \iddots & \\ -s(0) & & & & 1 \end{pmatrix}\begin{pmatrix} 1 & & & & -cs(0) \\ & \ddots & & \iddots & \\ & & \ddots & & \\ & 1 & & -cs(N/2-1) & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}\begin{pmatrix} I & 0 \\ I & -I \end{pmatrix}$$

Вычисление *Stereo-Int-DCT*-IV, используемого для *Stereo-IntMDCT*.

В случае сигнала стерео, кодированного как *channel_pair_element* с включенными *common_window* и *use_stereo_intmdct, DCT*-IV вычисляется для обоих каналов за один шаг, включая вычисление *M/S*. Это достигается при использовании разложения *Int-DCT*-IV, описанного выше, и исключении трех этапов шагов одномерного подъема и двух перестановок, приводя к:

$$\begin{pmatrix} I & 0 \\ -I & I \end{pmatrix}\begin{pmatrix} I & \frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_N \\ 0 & I \end{pmatrix}\begin{pmatrix} I & 0 \\ -\frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_N & I \end{pmatrix}\begin{pmatrix} I & \frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_N \\ 0 & I \end{pmatrix}\begin{pmatrix} I & -\frac{1}{2}I \\ -I & I \end{pmatrix}\begin{pmatrix} I & 0 \\ I & I \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_N & \frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_N \\ \frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_N & -\frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_N \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\sqrt{2} * I & \frac{1}{2}\sqrt{2} * I \\ \frac{1}{2}\sqrt{2} * I & -\frac{1}{2}\sqrt{2} * I \end{pmatrix}\begin{pmatrix} DCT\ \mathrm{IV}_N & 0 \\ 0 & DCT\ \mathrm{IV}_N \end{pmatrix}$$

Следовательно это упрощение алгоритма *DCT*-IV приводит к интегрированному вычислению матрицы *M/S* и *DCT*-IV для левого и правого каналов. В этом режиме *IntMDCT DCT*-IV работает при длине *N* вместо *N*/2.

Соответствующее инверсное разложение с подъемом для инверсного *Stereo-Int-DCT*-IV дается выражением:

$$\begin{pmatrix} I & 0 \\ -I & I \end{pmatrix}\begin{pmatrix} I & \frac{1}{2}I \\ 0 & I \end{pmatrix}\begin{pmatrix} I & -\frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_N \\ 0 & I \end{pmatrix}\begin{pmatrix} I & 0 \\ \sqrt{2}\,DCT\ \mathrm{IV}_N & I \end{pmatrix}\begin{pmatrix} I & -\frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_N \\ 0 & I \end{pmatrix}\begin{pmatrix} I & 0 \\ I & -I \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_N & \frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_N \\ \frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_N & -\frac{1}{2}\sqrt{2}\,DCT\ \mathrm{IV}_N \end{pmatrix} = \begin{pmatrix} DCT\ \mathrm{IV}_N^{-1} & 0 \\ 0 & DCT\ \mathrm{IV}_N^{-1} \end{pmatrix}\begin{pmatrix} \frac{1}{2}\sqrt{2} * I & \frac{1}{2}\sqrt{2} * I \\ \frac{1}{2}\sqrt{2} * I & -\frac{1}{2}\sqrt{2} * I \end{pmatrix}$$

Формирование шума

В шагах подъема, где обрабатываются сигналы временного интервала, операции округления соединяются с обратной связью по ошибкам, чтобы обеспечить спектральное формирование шума приближения.

Этот шум приближения влияет на эффективность кодирования без потерь в высокочастотной области, где аудиосигналы обычно содержат очень небольшое количество энергии, особенно на частотах дискретизации 96 кГц и выше. Следовательно низкочастотная характеристика шума приближения улучшает эффективность кодирования без потерь.

Используется фильтр формирования шума первого порядка.

Для *IntMDCT* этот фильтр применяется к трем этапам шагов подъема в обработке *Windowing/TDA* и к первому этапу округления обработки *Int-DCT*-IV.

Для инверсного *IntMDCT* тот же самый фильтр применяется к трем этапам шагов подъема в обработке *Windowing/TDA* и к последнему этапу округления обработки инверсного *Int-DCT*- IV.

5.10.3 Алгоритм для *IntMDCT* и инверсного *IntMDCT*

Вычисления

Все операции основаны на вычислениях. Используются следующие форматы:

*INT*32 для ввода, вывода и промежуточных значений, предопределенных коэффициентов с фиксированной точкой;

*INT*64 для умножения с коэффициентами с фиксированной точкой, результаты смещаются и сохраняются в *INT*32 сразу после каждого умножения.

Основные определения для *IntMDCT*:

SINE_DATA_SIZE = 8192

SHIFT = 30

SHIFT_FOR_ERROR_FEEDBACK = 6

Все операции с плавающей запятой в алгоритме выполняются способом с фиксированной точкой. Число дробных битов дается с помощью *SHIFT*.

Необходимые коэффициенты с плавающей точкой для умножения в шагах подъема и для промежуточных вычислений с фиксированной точкой сохраняются как значения с фиксированной точкой в *INT*32:

INT 32*_coeff* = *nearestint* ((1 <<*SHIFT*) **FLOAT_coeff*).

Следующие плавающие коэффициенты сохраняются таким образом:

*sineData [k] = sin (k*pi / (2*SINE_DATA_SIZE)), k=0..., SINE_DATA_SIZE*/2,

определенными в *sineData [SINE_DATA_SIZE*/2+1*]*.

*sineData_cs [k] = (1 - cos (k*pi / (2*SINE_DATA_SIZE))) / sin (k*pi / (2*SINE_DATA_SIZE)),*

k=0,..., *SINE_DATA_SIZE*/2,

определенными в *sineData_cs [SINE_DATA_SIZE*/2+1*]*.

Соответствующие значения для окна *KBD* предопределяются в *KBDWindow [SINE_DATA_SIZE/2]* соответственно *KBDWindow_cs [SINE_DATA_SIZE/2]*.

Основные функции для *IntMDCT*:

```
INT32 multShiftINT32(INT32 x, INT32 y, int shift) {
    return ( (INT32)(((INT64)x*y)>>shift) );
}
INT32 multShiftRoundINT32(INT32 x, INT32 y, int shift) {
    return ( ( multShiftINT32(x,y,shift-1) + 1 ) >>1 );
}
INT32 shiftRoundINT32(INT32 y, INT32 shift) {
    return (((y>>(shift-1))+1)>>1);
}
```

```
INT32 shiftRoundINT32withErrorFeedback(INT32 y, INT32* errorFeedback, int shift) {
    y += *errorFeedback;
    result = shiftRoundINT32(y,shift);
    *errorFeedback = (result << shift) - y;
    return (result);
}
void rotateINT32(int index, INT32 xin, INT32 yin, INT32* xout, INT32* yout) {
    xin += multShiftINT32(-sineData_cs[index], yin, SHIFT);
    yin += multShiftINT32(sineData[index], xin, SHIFT);
    xin += multShiftINT32(-sineData_cs[index], yin, SHIFT);
    *xout = xin;
    *yout = yin;
}

void multHalfSqrt2(INT32* x) {
    *x = multShiftINT32(sineData[SINE_DATA_SIZE/2], *x, SHIFT);
}
void rotatePlusMinusINT32(INT32 xin, INT32 yin, INT32* xout, INT32* yout) {
    xtmp = xin;
    ytmp = yin;
    *xout = xtmp + ytmp;
    *yout = xtmp - ytmp;
}
void rotatePlusMinusNormINT32(INT32 xin, INT32 yin, INT32* xout, INT32* yout) {
    rotatePlusMinusINT32(xin, yin, xout, yout);
    multHalfSqrt2(xout);
    multHalfSqrt2(yout);
    }
void addINT32(INT32* xin, INT32* xout, int N) {
    for (i=0; i<N; i++)
        xout[i] += xin[i];
}
void diffINT32(INT32* xin, INT32* xout, int N) {
    for (i=0; i<N; i++)
        xout[i] -= xin[i];
}
void copyINT32(INT32* xin, INT32* xout, int N) {
    for (i=0; i<N; i++)
        xout[i] = xin[i];
}
void shiftLeftINT32(INT32* x, int N, int shift) {
    for (i=0; i<N; i++)
        x[i] <<= shift;
}
void shiftRightINT32(INT32* x, int N, int shift) {
    for (i=0; i<N; i++)
        x[i] >>= shift;
 }
```

Определения для алгоритма *Windowing/TDA*:

Для прямого *Windowing/TDA*:

direction = 1;

Для инверсного *Windowing/TDA*:

direction = -1;

Алгоритм для прямого соответственно инверсного *Windowing/TDA*:

```
if (windowShape == 0) {
   window = sineData;
   window_cs = sineData_cs;
} else {
   window = KBDWindow;
   window_cs = KBDWindow_cs;
 }
errorFeedback = 0;
for (i=0; i<N/2; i++) {
   tmp = multShiftINT32(-window_cs[(2*i+1)*SINE_DATA_SIZE/(2*N)],
            signal[N-1-i],
            SHIFT - SHIFT_FOR_ERROR_FEEDBACK);
   signal[i] -= direction *
      shiftRoundINT32withErrorFeedback(tmp,
                        &errorFeedback,
                        SHIFT_FOR_ERROR_FEEDBACK);
}
errorFeedback = 0;
 for (i=0; i<N/2; i++) {
   tmp = multShiftINT32(window[(2*i+1)*SINE_DATA_SIZE/(2*N)],
            signal[i],
            SHIFT - SHIFT_FOR_ERROR_FEEDBACK);
signal[N-1-i] -= direction *
   shiftRoundINT32withErrorFeedback(tmp,
                     &errorFeedback,
                     SHIFT_FOR_ERROR_FEEDBACK);
}
errorFeedback = 0;
for (i=0; i<N/2; i++) {
   tmp = multShiftINT32(-window_cs[(2*i+1)*SINE_DATA_SIZE/(2*N)],
                  signal[N-1-i],
                  SHIFT - SHIFT_FOR_ERROR_FEEDBACK);
signal[i] -= direction *
   shiftRoundINT32withErrorFeedback(tmp,
                     &errorFeedback,
                     SHIFT_FOR_ERROR_FEEDBACK);
    }
```

Алгоритм для прямого соответственно инверсного *Int-DCT*-IV:

Входные значения преобразовываются в прямые соответственно инверсные значения *Int-DCT*-IV на месте.

В случае *Mono IntMDCT signal*0[0,..., *N*-1] представляет входные значения одного канала, *signal*1[0,..., *N*/2-1] соответствует верхней половине значений *signal*0 [*N*/2,..., *N*-1].

Если используется *Stereo IntMDCT*, длина *N* является двойной длиной фрейма в алгоритме *Int-DCT*-IV; *signal*0 [0,..., *N*/2-1] представляет входные значения левого канала, а *signal*1 [0,..., *N*/2-1] представляет входные значения правого канала.

Используется временный буфер *liftBuffer* [k], k=0,..., *N*/2-1 значений *INT*32.

Алгоритм для прямого *Int-DCT*-IV:

Перестановка *P*:

```
if (Mono_IntMDCT) {
   for (i=0; i<N; i+=4) {
      (signal0[i+2], signal0[i+3]) = (signal0[i+3], signal0[i+2]);
   }
}
```

```
Перестановка Q:
if (Mono_IntMDCT) {
        for (i=0; i<N; i++) {
            temp[i] = signal[i];
        }
        for (i=0; i<N/2; i++) {
            signal[i] = temp[2*i];
            signal[N/2+i] = temp[2*i+l];
        }
}

Применение шагов подъема:
addINT32(signal0, signal1, N/2);
liftingStep2and3(signal1, liftBuffer, N);
addINT32(liftBuffer, signal0, N/2);
liftingStep4(signal0, liftBuffer, N);
addINT32(liftBuffer, signal1, N/2);
liftingStep5and6(signal1, liftBuffer, N, Mono_IntMDCT);
addINT32(liftBuffer, signal0, N/2);
if (Mono_IntMDCT) {
liftingStep7(signal0, liftBuffer, N);
addINT32(liftBuffer, signal1, N/2);
liftingStep8(signal1, liftBuffer, N);
addINT32(liftBuffer, signal0, N/2);
}
Умножить на -1:
for (k=0; k<N/2; k++) {
signal1[k] *= -1;
}
Алгоритм для обратного "Int DCT- IV":
Умножить на -1:
for (k=0; k<N/2; k++) {
    signal1[k] *= -1;
}
Применить инверсные шаги подъема:
if (Mono_IntMDCT) {
    liftingStep8(signall, liftBuffer, N);
    diffINT32(liftBuffer, signal0, N/2);
    liftingStep7(signal0, liftBuffer, N);
    diffINT32(liftBuffer, signal1, N/2);
}
liftingStep5and6(signall, liftBuffer, N, Mono_IntMDCT);
diffINT32(liftBuffer, signal0, N/2);
liftingStep4(signal0, liftBuffer, N);
diffINT32(liftBuffer, signal1, N/2);
liftingStep2and3(signal1, liftBuffer, N);
diffINT32(liftBuffer, signal0, N/2);
diffINT32(signal0, signal1, N/2);
Инверсная перестановка Q:
if (Mono_IntMDCT) {
    for (i=0; i<N; i++) {
        temp[i] = signal[i];
    }
    for (i=0; i<N/2; i++) {
        signal[2*i] = temp[i];
```

```
        signal[2*i+l] = temp[n/2+i];
    }
}
```

Инверсная перестановка *P*:

```
if (Mono_IntMDCT) {
    for (i=0; i<N; i+=4) {
      (signal0[i+2], signal0[i+3]) = (signal0[i+3], signal0[i+2]);
      }
}
```

Шаги подъема для прямого и инверсного *Int-DCT*-IV:

```
void liftingStep2and3(INT32* signal1, INT32* liftBuffer, int N) {
    copyINT32(signal1, liftBuffer, N/2);
    shiftNormalize = DCTIVsqrt2_fixpt(liftBuffer, N/2) + 1;
    if (shiftNormalize > SHIFT_FOR_ERROR_FEEDBACK) {
        shiftRightINT32(liftBuffer, N/2,
shiftNormalize - SHIFT_FOR_ERROR_FEEDBACK);
        shiftNormalize = SHIFT_FOR_ERROR_FEEDBACK;
    }
    for (k=0; k<N/2; k++) {
        liftBuffer[k] -= signal1[k] << (shiftNormalize - 1);
    }
    errorFeedback = 0;
    for (k=0; k<N/2; k++) {
        liftBuffer[k] = shiftRoundINT32withErrorFeedback(liftBuffer[k],
&errorFeedback, shiftNormalize);
    }
}
void liftingStep4(INT32* signal0, INT32* liftBuffer, int N) {
    copyINT32(signal0, liftBuffer, N/2);
    shiftNormalize = DCTIVsqrt2_fixpt(liftBuffer, N/2);
    for (k=0; k<N/2; k++) {
        liftBuffer[k] = -shiftRoundINT32(liftBuffer[k],shiftNormalize);
    }
}
void liftingStep5and6(INT32* signal1, INT32* liftBuffer, int N, int mono) {
    copyINT32(signal1, liftBuffer, N/2);
    shiftNormalize = DCTIVsqrt2_fixpt(liftBuffer, N/2);
    shiftRightINT32(liftBuffer, N/2, shiftNormalize);
    if (mono) {
        for (k=0; k<N/2; k++)
            liftBuffer[k] += multShiftINT32(-sineData_cs[step*(2*k+1)],
signal1[N/2-1-k],(SHIFT-1));  }
        for (k=0; k<N/2; k++)
            liftBuffer[k] = ((liftBuffer[k]+1)>>1);
        }
void liftingStep7(INT32* signal0, INT32* liftBuffer, int N) {
        for (k=0; k<N/2; k++)
            liftBuffer[N/2-1-k] =
multShiftRoundINT32(sineData[(2*k+1)*SINE_DATA_SIZE/(2*N)],
signal0[k],SHIFT);
}
void liftingStep8(INT32* signal1, INT32* liftBuffer, int N) {
        for (k=0; k<N/2; k++)
            liftBuffer[k] =
multShiftRoundINT32(-sineData_cs[(2*k+1)*SINE_DATA_SIZE/(2*N)],
```

```
    signal1[N/2-1-k],SHIFT);
    }
    Алгоритм для SQRT(2)*DCT-IV:
    Как в прямом, так и в инверсном Int-DCT-IV требуется вычисление SQRT(2)*DCT-IV.
    Это вычисление выполняется детерминированным способом с фиксированной точкой:
    int DCTIVsqrt2_fixpt(INT32 *data, int N) {
        preShift = msbHeadroomINT32(data, N) - 1;
        if (preShift > 15) preShift = 15;
        if (preShift < 0) preShift = 0;
        shiftLeftINT32(data, N, preShift);
        preModulationDCT_fixpt(data, xr, xi, N);
        fftShift = srfft_fixpt(xr, xi, N/2);
        postModulationDCT_fixpt(xr, xi, data, N);
        shiftNormalize = (log2int(N) - 2) / 2 + preShift - fftShift;
        sqrt2Normalize = (log2int(N) - 2) % 2;
        if (sqrt2Normalize) {
            for (i=0; i<N; i++)
                multHalfSqrt2(&data[i]);
    }
    return shiftNormalize;
}
    Предварительная модуляция для DCT-IV:
    void preModulationDCT_fixpt(INT32 *x, INT32 *xr, INT32 *xi, int N) {
        for(i=0;i<N/4;i++) {
            rotateINT32((4*i+1)*SINE_DATA_SIZE/(2*N),
                x[N-1-2*i],x[2*i],
                &xi[i],&xr[i]);
            rotateINT32((4*i+3)*SINE_DATA_SIZE/(2*N),
                x[2*i+1],-x[N-2-2*i],
                &xr[N/2-1-i],&xi[N/2-1-i]);
        }
    }
    Постмодуляция для DCT-IV:
void  postModulationDCT_fixpt (INT32 *xr, INT32 *xi, INT32 *x, int N) {
    x [0] = xr [0];
    x [N-1] =-xi [0];
    for (i=1; i <N/4; i ++) {
            rotateINT32 (2*i*SINE_DATA_SIZE/N,
                xr [i],-xi [i],
                &x [2*i] ,&x [N-2*i-1]);
            rotateINT32 (2*i*SINE_DATA_SIZE/N,
                xr [N/2-i], xi [N/2-i],
                &x [2*i-1] ,&x [N-2*i]);
            }
            rotatePlusMinusNormINT32 (xr [N/4], xi [N/4],
                &x [N/2] ,&x [N/2-1]);
    }
            Split-Radix FFT:
            int srfft_fixpt(INT32 *xr, INT32 *xi, int N) {
              numShifts = 0;
              /* L = 1,2,4,…,N/2 */
              for (L=l; L<N; L*=2) {
                  M = N/L; /* M = N, N/2,...,2 */
                  M2 = M/2;
                  M4 = M2/2;
```

```
    /* L: number of sub-blocks
    M: length of sub-block */
    numShifts += shiftIfRequired(xr, xi, N);
    for (l=0; l<L; l++) {
        /* butterfly on (x[k],x[M2+k]), k = 0,...,N2-1 on each sub-block */
        for (k=0; k<M2; k++) {
    rotatePlusMinusINT32(xr[l*M+k],xr[l*M+M2+k],
                         &xr[l*M+k],&xr[l*M+M2+k]);
    rotatePlusMinusINT32(xi[l*M+k],xi[l*M+M2+k],
                         &xi[l*M+k],&xi[l*M+M2+k]);
          }
    }
    numShifts += shiftIfRequired(xr, xi, N);
    if (M > 2) {
        for (l=0; l<L; l++) {
    if (srfftIndex(l) == 0) {
        /* x[N2+N4+k] -> -j*x[N2+N4+k] , k = 0,...,N4-1 on each sub-block
*/
        for (k=0; k<M4; k++) {
            swap(&xr[l*M+M2+M4+k],&xi[l*M+M2+M4+k]);
            xi[l*M+M2+M4+k] *= -1;
        }
        } else {
        /* complex multiplications */
        for (k = 1; k < M4; k++) {
            rotateINT32(4*k*SINE_DATA_SIZE/(2*M),
                xi[l*M+k],xr[l*M+k],
                &xi[l*M+k],&xr[l*M+k]);
            rotateINT32(4*k*SINE_DATA_SIZE/(2*M),
                -xi[l*M+M2-k],-xr[l*M+M2-k],
                &xr[l*M+M2-k],&xi[l*M+M2-k]);
            }
            for (k = 1; 3*k < M4; k++) {
              rotateINT32(4*3*k*SINE_DATA_SIZE/(2*M),
                xi[l*M+M2+k],xr[l*M+M2+k],
                &xi[l*M+M2+k],&xr[l*M+M2+k]);
              rotateINT32(4*3*k*SINE_DATA_SIZE/(2*M),
                 -xi[l*M+M-k],xr[l*M+M-k],
                &xr[l*M+M-k],&xi[l*M+M-k]);
            }
            for (; 3*k < 2*M4; k++) {
              rotateINT32(4*(M2-3*k)*SINE_DATA_SIZE/(2*M),
                xi[l*M+M2+k],-xr[l*M+M2+k],
                &xr[l*M+M2+k],&xi[l*M+M2+k]);
              rotateINT32(4*(M2-3*k)*SINE_DATA_SIZE/(2*M),
                -xi[l*M+M-k],-xr[l*M+M-k],
                &xi[l*M+M-k],&xr[l*M+M-k]);
             }
            for (; 3*k < 3*M4; k++) {
              rotateINT32(4*(3*k-M2)*SINE_DATA_SIZE/(2*M),
                 -xr[l*M+M2+k],xi[l*M+M2+k],
                 &xi[l*M+M2+k],&xr[l*M+M2+k]);
              rotateINT32(4*(3*k-M2)*SINE_DATA_SIZE/(2*M),
                -xr[l*M+M-k],-xi[l*M+M-k],
                &xr[l*M+M-k],&xi[l*M+M-k]);
            }
```

```
            rotatePlusMinusNormINT32(xi[l*M+M4],xr[l*M+M4],
                        &xr[l*M+M4],&xi[l*M+M4]);
            rotatePlusMinusNormINT32(-xr[l*M+M-M4],xi[l*M+M-M4],
                        &xr[l*M+M-M4],&xi[l*M+M-M4]);
          }
            }
           }
          }
          bit_reverse_fixpt(xr,N);
          bit_reverse_fixpt(xi,N);
          return numShifts;
        }
```

Базовые функции для *FFT*:

```
          int msbHeadroomINT32(INT32 *x, int N) {
            max = 0;
            for (i=0; i<N; i++) {
                  max |= ABS(x[i]);
            }
            return (30-log2int(max));
            }
          int shiftIfRequired(INT32 *xr, INT32 *xi, int N) {
             shiftRequired = 0;
             if ((!msbHeadroomINT32(xr,N))||(!msbHeadroomINT32(xi,N))) {
                   shiftRequired = 1;
                   shiftRightINT32(xr, N, 1);
                   shiftRightINT32(xi, N, 1);
             }
             return shiftRequired;
             }
          void bit_reverse_fixpt(INT32 *x, int N) {
          for (m=1,j=0; m<N-1; m++) {
                  for(k=N>>1; (!((j^=k)&k)); k>>=1);
                  if (j>m) swap(&x[m],&x[j]);
                   }
          }
          srfftIndexTable[32] = {0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
                                 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1};
    int srfftIndex(int l) {
      return srfftIndexTable[(srfftIndexTable[l>>4]<<4)|(l&0xF)];
    }
```

Обработка прямого и инверсного целочисленного *Mid/Side*:

```
  void IntMidSideINT32(INT32* 1, INT32* r) /* L/R -> M/S */
  {
    m = *1;
    s = *r;
    m += multShiftRoundINT32(-sineData cs[SINE DATA SIZE/2], s, SHIFT);
    s += multShiftRoundINT32( sineDataTsiNE_DATA_SIZE/2],   m, SHIFT);
    m += multShiftRoundINT32(-sineData_cs[SINE_DATA_SIZE/2], s, SHIFT);
    *1 = s;
    *r = m;
  }
  void InverseIntMidSideINT32(INT32* 1, INT32* r) /* M/S -> L/R */
  {
    m = *1;
    s = *r;
```

```
    s -= multShiftRoundINT32(-sineData cs[SINE DATA SIZE/2], m, SHIFT);
    m -= multShiftRoundINT32( sineDataTsiNE_DATA_SIZE/2],   s, SHIFT);
    s -= multShiftRoundINT32(-sineData_cs[SINE_DATA_SIZE/2], m, SHIFT);
    *1 = s;
    *r = m;
}
```

5.11 Вычисление табличных значений на основе компактных таблиц

Значения таблиц *sineData, sineData_cs, thrMantissa ()* и *invQuantMantissa ()* вычисляются из компактных таблиц в Приложении Б. Это описывается в следующем псевдокоде:

```
/* interpolate value between v0 = data[0] and v8 = data[8],
   using additionally vm8 = data[-8] and v16 = data[16] */
INT32 interpolateValue1to7(INT32 vm8,
                           INT32 v0,
                           INT32 v8,
                           INT32 v16,
                           INT32 l)
{
   INT32 value;
   INT32 d1, d2, d3;
   d1 = 2*(v8-v0); /* 1 add, 1 shift */
   d2 = v8-vm8; /* 1 add */
   d3 = v16-v0; /* 1 add */
   if (l==1) {
      value = v0 + ( ( 8*d2 - d2 + d1 + 64 ) >> 7 ); /* 4 adds, 2 shifts */
   } else if (l==2) {
      value = v0 + ( ( 2*d2 + d2 + d1 + 16 ) >> 5 ); /* 4 adds, 2 shifts */
   } else if (l==3) {
      value = v0 + ( ( 16*d2 - d2 + 8*d1 + d1 + 64 ) >> 7 ); /* 5 adds, 3 shifts */
   } else if (l==4) {
      value = v0 + ( ( d2 + d1 + 4 ) >> 3 ); /* 3 adds, 1 shifts */
   } else if (l==5) {
      value = v8 - ( ( 16*d3 - d3 + 8*d1 + d1 + 64 ) >> 7 ); /* 5 adds, 3 shifts */
   } else if (l==6) {
      value = v8 - ( ( 2*d3 + d3 + d1 + 16 ) >> 5 ); /* 4 adds, 2 shifts */
   } else if (l==7) {
      value = v8 - ( ( 8*d3 - d3 + d1 + 64 ) >> 7 ); /* 4 adds, 2 shifts */
   }
   return value;
}

INT32 interpolateFromCompactTable(int index, INT32* compactTable)
{
      INT32 value;
      j = index%8;
      k = index/8;
      if (j == 0) {
         value = compactTable[k+1];
         return value;
      }
      value = interpolateValue1to7(compactTable[k],
                                   compactTable[k+1],
                                   compactTable[k+2],
                                   compactTable[k+3],
                                   j);
```

```
    return value;
}
```

Значения для таблиц *sineData* и *sineData_cs* вычисляются, применяя *sineData[index] = interpolateFrom CompactTable(index, sineDataCompact)* и *sineData_cs[index] = interpolateFromCompactTable(index, sineDataCompact_cs).*

Значения для функции *invQuantMantissa* () вычисляются согласно

```
INT32 invQuantMantissa(int quant, int res)
{
    INT32 value;
    INT32 pow_2_quat[4] = {0, 1276901417, 1518500250, 1805811301};
      /* (int)(pow(2.0,res/4.0)*(1<<SHIFT)+0.5) */
    if (quant < MAX_INV_QUANT_TABLE) {
      value = invQuantCompact[quant];
      if (res > 0) {value = multShiftRoundINT32(value, pow_2_quat[res], SHIFT); }
    } else {
      l = quant%8;
      k = quant/8;
    if (l == 0) {value = invQuantMantissa(k, res)<<4; /* 8^(4/3) = 16 = 2^4 */
    } else {
      value = interpolateValue1to7(invQuantMantissa(k-1, res)<<4,
                                   invQuantMantissa(k, res)<<4,
                                   invQuantMantissa(k+1, res)<<4,
                                   invQuantMantissa(k+2, res)<<4,
                                   l);
    }
   }
   return value;
}
```

Значение для функции *thrMantissa* () вычисляется согласно

```
INT32 thrMantissa(quant, res)
{
    INT32 value;
    INT32 invQuant0;
    INT32 invQuant1;
    if (quant < MAX_THR_TABLE) {
      value = thrCompact[res][quant];
    } else {
      invQuant0 = invQuantMantissa(quant,res);
      invQuant1 = invQuantMantissa(quant+1,res);
      value = invQuant1+(((invQuant0-invQuant1)*13)>>5);
    }
    return value;
}
```

Приложение А
(справочное)

Описание кодера

А.1 Обзор

Кодер *SLS* генерирует для данного аудиовхода *PCM* поток битов без потерь, который может декодироваться в разрядно-точную репродукцию *PCM* данного аудио при использовании декодера *SLS*. Кроме того поток без потерь, сгенерированный кодером *SLS*, может быть усеченным, чтобы понизить скорости передачи до скорости передачи базового кодера *MPEG*-4 *GA*. Таким образом получающийся поток битов может декодироваться декодером *SLS*, чтобы образовать репродукцию исходного аудио с потерями таким способом, что лучшая точность сигнала всегда достигается с более высокими скоростями в уровне *LLE*.

А.1.1 Кодирование с передискретизацией

Для процесса кодирования, включающего метод передискретизации, возможны два подхода: субдискретизация в домене *MDCT* и субдискретизация во временном интервале.

Входной сигнал при субдискретизации в домене *MDCT* обрабатывается *IntMDCT* длины *osf**1024, и первые 1024 спектральных значения подаются в кодер *MPEG*-4 *GA*.

Для второго возможного подхода кодирования входной сигнал субдискретизируется во временном интервале и параллельно выполняется полная часть кодирования *AAC*.

А.2 Целочисленный *MDCT*

IntMDCT описывался в нормативной части.

А.3 Группировка и чередование

Существуют два типа окна, которые используются в реализации *SLS*. Они такие же как при работы с окнами *AAC*. Одно является длинным окном с коэффициентами *IntMDCT osf**1024, другое — короткое окно с коэффициентами *IntMDCT osf**128. К оконной аудиопоследовательности применяется целочисленное преобразование. Когда используется короткое окно, набор коэффициентов *osf**1024 *IntMDCT* обрабатывается как матрица 8 частотными коэффициентами *osf**128, представляющими время-частотную эволюцию сигнала на протяжении восьми коротких окон. Тот же самый процесс группировки и чередования, принятый в *AAC*, применяется здесь. Чтобы быть определенными, предположим, что перед чередованием набор с коэффициентов *osf**1024 *IntMDCT* индексируются как *c* [*g*] [*w*] [*b*] [*k*], где

g — индекс групп;

w — индекс окон в пределах группы;

b — индекс полос масштабного коэффициента в пределах окна;

k — индекс коэффициентов в пределах полосы масштабного коэффициента и самый правый индекс изменяется наиболее быстро.

После чередования коэффициенты индексируются как *c* [*g*] [*b*] [*w*] [*k*].

В последующих разделах, когда используется короткое окно, предполагают, что сигнальный процесс выполняется на чередующемся спектре для восьми коротких рамок окна, если не определено иное.

А.4 Целочисленный *Mid/Side*

Если для левого и правого каналов используется *Mono IntMDCT*, там, где флаг *M/S* устанавливается в '1', к полосам масштабного коэффициента должна быть применена целочисленная обработка *M/S*.

Stereo IntMDCT поставляет по умолчанию спектр *M/S*. Следовательно *M/S* должно быть выключено для полос масштабного коэффициента там, где оно не требуется.

А.5 Нормализация перед кодированием *AAC*

После *IntMDCT* масштабный коэффициент *core_scaling* используется, чтобы нормализовать коэффициенты *c*(*k*) *IntMDCT* для каждой полосы масштабного коэффициента *sfb*, чтобы обеспечить базовый входной спектр уровня *AAC c' (k)*.

После *IntMDCT* при нормализации коэффициентов *c(k) IntMDCT* для каждой полосы масштабного коэффициента *sfb* используется масштабный коэффициент *core_scaling*, чтобы обеспечить входной спектр базового уровня *AAC c' (k)*.

$$c' = core_scaling \cdot c,$$

где значение масштабного коэффициента *core_scaling* совместно определяется типом соответствующей полосы масштабного коэффициента и длиной слова входного аудио, данными в следующей таблице:

Таблица А.1 — Значение *core_scaling*

Длина слова / Тип *sfb*	16	20	24
Длинное окно (2048), *M/S*	32	2	1/8
Длинное окно (2048), не *M/S*	$32\sqrt{2}$	$2\sqrt{2}$	$\sqrt{2}/8$
Короткое окно (256), *M/S*	$8\sqrt{2}$	$\sqrt{2}/2$	$\sqrt{2}/32$
Короткое окно (256), не *M/S*	16	1	1/16

После нормализации нормализованный *c'* квантуется базовым квантователем *AAC*, чей индекс квантования вывода *i* затем кодируется по Хаффману. Потом он мультиплексируется с необходимой дополнительной информацией, например масштабным коэффициентом *scale_factor(sfb)*, используемым в квантователе для каждой полосы масштабного коэффициента sfb согласно синтаксису потока битов *AAC*, чтобы генерировать базовый поток битов AAC.

А.6 Отображение ошибок

В уровне *LLE* процедура отображения ошибок используется, чтобы удалить информацию, которая уже была кодирована в базовом уровне. Ввод в модуль отображения ошибок является 1024 коэффициентами *c[sfb][k] IntMDCT* в диапазоне непередискретизации и его соответствующим квантованным значением в базовом кодере *quant[sfb][k].* Его вывод является остаточным спектром *IntMDCT res*.

А.7 Кодер *BPGC/CBAC*

В *SLS* остаточный спектр *IntMDCT res* кодируется кодирующим процессом *BPGC/CBAC*, который состоит из следующих шагов:

определение параметра *BPGC/CBAC*;

кодирование разрядной матрицы остаточных целочисленных спектральных данных;

кодирование низкоэнергетического режима остаточных целочисленных спектральных данных.

А.7.1 Определение параметра *BPGC/CBAC*

Как первый шаг идентифицируются максимальные разрядные матрицы *max_bp* для каждой полосы масштабного коэффициента. Для *Implicit_Band* максимальная разрядная матрица *M* для каждых остаточных спектральных данных может быть вычислена из

$$M[g][win][sfb][bin] = INT\{log_2[interval[g][win][sfb][bin]]\},$$

где *interval*[*g*][*win*][*sfb*][*bin*] является интервалом квантования *AAC*.

Для *Explicit_Band, M* дается как:

$$M[g][win][sfb][bin] = INT\{log_2|res[g][win][sfb][bin]|\},$$

и далее определяем $\log_2 0=-1$ для вышеупомянутого вычисления log. Максимальная разрядная матрица *max_bp* для каждой полосы масштабного коэффициента является максимальным значением *M* для спектральных данных, которые принадлежат *sfb*:

$$max_bp[g][sfb] = max(M[g][win][sfb][bin])$$

После обнаружения *max_bp* для каждой полосы масштабного коэффициента *lazy* выбирается из трех возможных значений *max_bp*-1, *max_bp*-2 и *max_bp*-3.

А.7.2 Кодирование разрядной матрицы остаточных целочисленных спектральных данных

Следующий псевдокод иллюстрирует, как кодируются знак и амплитуда остаточных спектральных данных *IntMDCT res* в потоке данных *BPGC/CBAC*. Элемент справки *M* для незначащей полосы масштабного коэффициента устанавливается в значение *max_bp*, чтобы быть совместимым с процессом декодирования. На полосах масштабного коэффициента, для которых *lazy_bp*> 0, выполняется процесс кодирования *BPGC/CBAC*.

```
/* preparing of help elements */
for (g=0;g<num_window_groups;g++){
    for (sfb = 0;sfb<num_sfb;sfb++){
        width = swb_offset[g][sfb+1] – swb_offset[g][sfb];
        for (win = 0;win <window_group_len[g];win++) {
            for (bin=0;bin<width;bin++)
                is_sig[g][win][sfb][bin]=
((quant[g][sfb][win][bin])&&(band_type[g][sfb]==Implicit_band))?1:0;
            }
        cur_bp[g][sfb] = max_bp[g][sfb];
        }
```

```
}
/* BPGC/CBAC normal coding process */
while ((there exists max_bp[g][sfb]-i >= 0) && (i<LAZY_BP)){
   for (g=0;g<num_window_groups;g++){
      for (sfb = 0;sfb<num_sfb;sfb++){
         if ((cur_bp[g][sfb]>=0) && (lazy_bp[g][sfb] > 0)){
            width = swb_offset[g][sfb+1] – swb_offset[g][sfb];
            for (win=0;win<window_group_len[g];win++){
               for (bin=0;bin<width;bin++){
                  sym = (abs(res[g][win][sfb][bin])&(1<<cur_bp[g][sfb]))?1:0;
                  sgn = (sign(res[g][win][sfb][bin])+1)/2;
                  if (interval[g][win][sfb][bin]>res[g][win][sfb][bin]+(1<<cur_bp[g][sfb]) {
                     encode(sym,freq); /* encode bit-plane cur_bp*/
                     if ((!is_sig[g][win][sfb][bin])&&(sym)){
                        encode(sgn,freq_sign);  /* encode sign bit if necessary * /
                        is_sig[g][win][sfb][bin] = 1;
                  }
               }
            }
         }
      }
      cur_bp[g][sfb]- -;
   }
}
The BPGC/CBAC lazy coding mode is started after the first NUM_BP bit-planes have been coded.
/* BPGC/CBAC lazy coding process */
flush_encode(); /* flush the AC encoder before lazy coding */
while (there exists max_bp[g][sfb]-i >= 0){
   for (g=0;g<num_window_groups;g++){
      for (sfb = 0;sfb<num_sfb;sfb++){
         if ((cur_bp[g][sfb]>=0) && (lazy_bp[g][sfb] > 0)){
            width = swb_offset[g][sfb+1] – swb_offset[g][sfb];
            for (win=0;win<window_group_len[g];win++){
               for (bin=0;bin<width;bin++){
                  sym = (abs(res[g][win][sfb][bin])&(1<<cur_bp[g][sfb]))?1:0;
                  sgn = (sign(res[g][win][sfb][bin])+1)/2;
                  if (interval[g][win][sfb][bin]>res[g][win][sfb][bin]+(1<<cur_bp[g][sfb]) {
                     write_bit(sym); /* encode bit-plane cur_bp*/
                     if ((!is_sig[g][win][sfb][bin])&&(sym)){
                     write_bit(sgn);  /* encode sign bit if necessary */
                     is_sig[g][win][sfb][bin] = 1;
                  }
               }
            }
         }
      }
      cur_bp[g][sfb]- -;
   }
}
```

А.7.3 Кодирование кодом низкоэнергетического режима *(LEMC)*

Для полос масштабного коэффициента с *lazy_bp*=0, -1, -2, и -3 остаточные спектральные данные *res* не кодируются *BPGC/CBAC*. Вместо этого они кодируются процессом кодирования *LEMC*. В низкоэнергетическом режиме амплитуда остаточных спектральных данных *res* сначала конвертируется в двоичный формат, как перечислено в таблице 26. Получающаяся двоичная строка затем кодируется арифметически. Процесс низкоэнергетического режима кодирования выполняется непосредственно после того, как завершается процесс кодирования *BPGC*.

Процесс низкоэнергетического режима кодирования иллюстрируется следующим псевдокодом:

```
for (g = 0;g < num_window_groups; g++){
    for (sfb = 0; sfb <num_sfb+num_osf_sfb;sfb++){
        if ((cur_bp[g][sfb] >= 0) && (lazy_bp[g][sfb] <= 0))
        {
            width = swb_offset[g][sfb+1] – swb_offset[g][sfb];
            for (win=0;win<window_group_len[g];win++){
                pos = 0;
                for (bin=0;bin<width;bin++){
                    if (!is_lle_ics_eof ()){
                        amp = abs(res[g][sfb][win][bin]);
                        sgn = (sign(res[g][sfb][win][bin]) + 1)/2;
                        dumb = amp;
                        while (dumb > 0) { /* binarize and encoding for non-zero res*/
                                encode(1,freq_silence[position]);
                                position++;
                                if (position>2) position = 2;
                                dumb - - ;
                        }
                        if (amp != (1<<(max_bp[g][win][sfb] + 1)) - 1)
                            encode(0,freq_silence[position]); /* encode of terminating 0 */
                        if (amp)
                            encode(sgn,freq_sgn);  /* encode of sign symbol*/
                        }
                    }
                }
            }
        }
    }
}
```

Следующий псевдокод объясняет, как в *BPGC/CBAC* арифметически кодируется двоичный символ и процессы низкоэнергетического режима кодирования.

Определения:

```
#Idefine CODE_WL    16
#Idefine PRE_SHT                14
#Idefine TOP_ VALUE  (((long) I < < CODE_WL)-1)
#Idefine QTR_VALUE              (TOP_VALUE/4+1)
#Idefine HALF_VALUE (2*QTR_VALUE)
#Idefine TRDQTR_VALUE   (3* QTR_VALUE)
```

Инициализация:

```
low = 0;
high = TOP_VALUE;
fbits = 0;
```

Подпрограмма кодирования:

```
void encode(int sym, int freq)
{
    range = (long)(high-low)+1;
    if (sym)
        high = low + (range*freq>>PRE_SHT)-1;
    else
    low = low + (range*freq>>PRE_SHT);
for (;;) {
    if (high<HALF_VALUE) {
    output_bit(0);
while (fbits > 0) {
        output_bit (1);
        fbits - - ;
}
} else if (low>=HALF_VALUE)  {
    output_bit(1);
while (fbits > 0) {
        output_bit (0);
```

```
            fbits - -;
        }
            low -= HALF_VALUE;
            high -= HALF_VALUE;
        } else if (ow>=QTR_VALUE && high<TRDQTR_VALUE)  {
            fbits += 1;
            low -= QTR_VALUE;
            high -= QTR_VALUE;
        } else
        break;
        low = 2*low;
        high = 2*high+1;
    return;
}
    сброс состояния кодирования:
/* flush the state register of AC encoder*/
flush_encode()
{
    fbits += 1;
    if (low < QTR_VALUE)
        output_bit(0);
        while (fbits > 0) {
            output_bit (1);
    fbits - -;
}
else
    output_bit(1);
    while (fbits > 0) {
        output_bit (0);
        fbits - -;
}
```

А.8 Метод усечения потока битов путем его повторного анализа

Полный поток битов *SLS* может быть усечен при любой данной целевой скорости передачи простым способом. Модификация значений *lle_ics_length* не влияет на результаты декодирования *LLE* перед точкой усечения, так как *lle_ics_length* независимо от процедуры декодирования *LLE*. Усечение потока битов может быть выполнено следующим образом:

1. Считать *lle_ics_length* из потока битов;

2. Считать поток битов *LLE*;

3. Вычислить доступную длину фрейма в данной целевой скорости передачи. Самый простой способ вычислить доступную длину фрейма следующий:

target_bits = (int) (target_bitrate/2.*1024.**osf/sampling_rate*+0.5)-16;

target_bytes = (*target_bits*+7)/8.

Переменная *target_bitrate* представляет целевую скорость передачи, бит/с. Переменная *osf* представляет фактор передискретизации. Переменная *sampling_rate* представляет частоту дискретизации входного аудиосигнала, Гц;

4. Обновить *lle_ics_length*, взяв минимум доступной длины фрейма и текущей длины фрейма.

lle_ics_length = min (lle_ics_length, target_bytes);

5. Генерировать усеченный поток битов с обновленным *lle_ics_length.*

Получающийся усеченный поток битов декодируется методом интеллектуального арифметического декодирования, как описано в 5.5.2.5.

Приложение Б
(обязательное)

Таблицы для предопределенных коэффициентов с фиксированной точкой

Таблица Б.1 — *define SINE_DATA_SIZE* 8192

* sin(0,...,pi/4) *and* +-1 */ *INT*32 *sineDataCompact*[515] = {
-1647099, 0, 1647099, 3294193, 4941281, 6588356, 8235416, 9882456, 11529474, 13176464, 14823423, 16470347, 18117233, 19764076, 21410872, 23057618, 24704310, 26350943, 27997515, 29644021, 31290457, 32936819, 34583104, 36229307, 37875426, 39521455, 41167391, 42813230, 44458968, 46104602, 47750128, 49395541, 51040837, 52686014, 54331067, 55975992, 57620785, 59265442, 60909960, 62554335, 64198563, 65842639, 67486561, 69130324, 70773924, 72417357, 74060620, 75703709, 77346620, 78989349, 80631892, 82274245, 83916404, 85558366, 87200127, 88841683, 90483029, 92124163, 93765079, 95405776, 97046247, 98686491, 100326502, 101966277, 103605812, 105245103, 106884147, 108522939, 110161476, 111799753, 113437768, 115075515, 116712992, 118350194, 119987118, 121623759, 123260114, 124896179, 126531950, 128167423, 129802595, 131437462, 133072019, 134706263, 136340190, 137973796, 139607077, 141240030, 142872651, 144504935, 146136880, 147768480, 149399733, 151030634, 152661180, 154291367, 155921191, 157550647, 159179733, 160808445, 162436778, 164064728, 165692293, 167319468, 168946249, 170572633, 172198615, 173824192, 175449360, 177074115, 178698453, 180322371, 181945865, 183568930, 185191564, 186813762, 188435520, 190056834, 191677702, 193298119, 194918080, 196537583, 198156624, 199775198, 201393302, 203010932, 204628085, 206244756, 207860942, 209476638, 211091842, 212706549, 214320755, 215934457, 217547651, 219160334, 20772500, 222384147, 223995270, 225605867, 227215933, 228825464, 230434456, 232042906, 233650811, 235258165, 236864966, 238471210, 240076892, 241682010, 243286558, 244890535, 246493935, 248096755, 249698991, 251300640, 252901697, 254502159, 256102022, 257701283, 259299937, 260897982, 262495412, 264092224, 265688415, 267283981, 268878918, 270473223, 272066891, 273659918, 275252302, 276844038, 278435122, 280025552, 281615322, 283204430, 284792871, 286380643, 287967740, 289554160, 291139898, 292724951, 294309316, 295892988, 297475964, 299058239, 300639811, 302220676, 303800829, 305380268, 306958988, 308536985, 310114257, 311690799, 313266607, 314841679, 316416009, 317989595, 319562433, 321134518, 322705848, 324276419, 325846226, 327415267, 328983538, 330551034, 332117752, 333683689, 335248841, 336813204, 338376774, 339939549, 341501523, 343062693, 344623057, 346182609, 347741347, 349299266, 350856364, 352412636, 353968079, 355522689, 357076462, 358629395, 360181484, 361732726, 363283116, 364832652, 366381329, 367929144, 369476093, 371022173, 372567379, 374111709, 375655159, 377197725, 378739403, 380280190, 381820082, 383359076, 384897167, 386434353, 387970630, 389505993, 391040440, 392573967, 394106570, 395638246, 397168991, 398698801, 400227673, 401755603, 403282588, 404808624, 406333708, 407857835, 409381002, 410903207, 412424444, 413944711, 415464004, 416982319, 418499653, 420016002, 421531363, 423045732, 424559105, 426071480, 427582852, 429093217, 430602573, 432110916, 433618242, 435124548, 436629829, 438134084, 439637307, 441139496, 442640647, 444140756, 445639820, 447137835, 448634799, 450130706, 451625555, 453119340, 454612060, 456103710, 457594286, 459083786, 460572205, 462059541, 463545789, 465030947, 466515010, 467997976, 469479840, 470960600, 472440251, 473918791, 475396216, 476872522, 478347705, 479821764, 481294693, 482766489, 484237150, 485706671, 487175049, 488642281, 490108363, 491573292, 493037064, 494499676, 495961124, 497421405, 498880516, 500338453, 501795212, 503250791, 504705185, 506158392, 507610408, 509061229, 510510853, 511959275, 513406493, 514852502, 516297300, 517740883, 519183248, 520624391, 522064309, 523502998, 524940456, 526376678, 527811662, 529245404, 530677900, 532109148, 533539144, 534967884, 536395365, 537821584, 539246538, 540670223, 542092635, 543513772, 544933630, 546352205, 547769495, 549185496, 550600205, 552013618, 553425732, 554836544, 556246051, 557654248, 559061133, 560466703, 561870954, 563273883, 564675486, 566075761, 567474703, 568872310, 570268579, 571663506, 573057087, 574449320, 575840202, 577229728, 578617896, 580004702, 581390144, 582774218, 584156920, 585538248, 586918198, 588296766, 589673951, 591049748, 592424154, 593797166, 595168781, 596538995, 597907806, 599275210, 600641203, 602005783, 603368947, 604730691, 606091012, 607449906, 608807372, 610163404, 611518001, 612871159, 614222875, 615573145, 616921967, 618269338, 619615253, 620959711, 622302707, 623644239, 624984303, 626322897, 627660017, 628995660, 630329823, 631662503, 632993696, 634323400, 635651611, 636978327, 638303543, 639627258, 640949467, 642270169, 643589359, 644907034, 646223192, 647537830, 648850943, 650162530, 651472587, 652781111, 654088099, 655393548, 656697454, 657999816, 659300629, 660599890, 661897597, 663193747, 664488336, 665781362, 667072820, 668362709, 669651026, 670937767, 672222928, 673506508, 674788504, 676068911, 677347728, 678624950, 679900576, 681174602, 682447025, 683717842, 684987051, 686254647, 687520629, 688784993, 690047736, 691308855,

Продолжение таблицы Б.1

692568348, 693826211, 695082441, 696337036, 697589992, 698841307, 700090977, 701339000, 702585372, 703830092, 705073155, 706314559, 707554301, 708792378, 719859669, 721080937, 722300508, 723518380, 724734549, 725949013, 727161768, 728372813, 729582143, 730789757, 731995651, 733199822, 734402269, 735602987, 736801974, 737999228, 739194745, 740388522, 741580558, 742770848, 743959390, 745146182, 746331221, 747514503, 748696026, 749875788, 751053785, 752230015, 753404474, 754577161, 755748072, 756917205, 758084557, 759250125, 760413906 }
/* (l-cos)/sin (0,...,pi/4) == tan(0,...,pi/8) */ *INT32 sineDataCompact cs*[515] = {
-823550, 0, 823550, 1647101, 2470653, 3294209, 4117769, 4941333, 5764903, 6588480, 7412065, 8235658, 9059261, 9882875, 10706500, 11530138, 12353790, 13177456, 14001138, 14824836, 15648551, 16472285, 17296039, 18119812, 18943607, 19767425, 20591265, 21415130, 22239020, 23062936, 23886880, 24710851, 25534852, 26358882, 27182944, 28007038, 28831164, 29655325, 30479520, 31303752, 32128020, 32952326, 33776671, 34601055, 35425481, 36249948, 37074458, 37899011, 38723609, 39548253, 40372944, 41197681, 42022468, 42847304, 43672190, 44497128, 45322119, 46147163, 46972261, 47797414, 48622624, 49447892, 50273217, 51098602, 51924047, 52749553, 53575122, 54400754, 55226449, 56052210, 56878038, 57703932, 58529894, 59355926, 60182027, 61008200, 61834444, 62660762, 63487153, 64313620, 65140162, 65966781, 66793478, 67620255, 68447111, 69274047, 70101066, 70928168, 71755353, 72582623, 73409979, 74237422, 75064952, 75892571, 76720280, 77548080, 78375971, 79203955, 80032033, 80860205, 81688474, 82516838, 83345301, 84173862, 35002523, 85831284, 86660147, 87489113, 88318182, 89147356, 89976635, 90806021, 91635515, 92465117, 93294829, 94124652, 94954586, 95784632, 96614793, 97445068, 98275458, 99105965, 99936590, 100767333, 101598196, 102429179, 103260284, 104091512, 104922863, 105754339, 106585941, 107417669, 108249525, 109081509, 109913623, 110745868, 111578245, 112410754, 113243397, 114076174, 114909088, 115742138, 116575326, 117408652, 118242119, 119075726, 119909475, 120743367, 121577403, 122411583, 123245910, 124080383, 124915004, 125749775, 126584695, 127419766, 128254990, 129090366, 129925897, 130761582, 131597424, 132433423, 133269580, 134105896, 134942372, 135779010, 136615810, 137452774, 138289901, 139127195, 139964654, 140802281, 141640077, 142478042, 143316178, 144154485, 144992965, 145831619, 146670447, 147509452, 148348633, 149187992, 150027529, 150867247, 151707146, 152547227, 153387491, 154227939, 155068572, 155909392, 156750399, 157591594, 158432979, 159274554, 160116320, 160958280, 161800432, 162642780, 163485323, 164328063, 165171001, 166014138, 166857475, 167701013, 168544753, 169388696, 170232844, 171077197, 171921756, 172766522, 173611498, 174456682, 175302078, 176147685, 176993505, 177839539, 178685788, 179532253, 180378935, 181225835, 182072955, 182920295, 183767856, 184615640, 185463648, 186311880, 187160339, 188009024, 188857937, 189707079, 190556451, 191406055, 192255890, 193105960, 193956264, 194806803, 195657579, 196508594, 197359847, 198211340, 199063074, 199915050, 200767270, 201619735, 202472445, 203325401, 204178606, 205032059, 205885762, 206739717, 207593924, 208448384, 209303098, 210158069, 211013296, 211868780, 212724524, 213580528, 214436793, 215293321, 216150112, 217007167, 217864489, 218722077, 219579933, 220438059, 221296454, 222155121, 223014061, 223873274, 224732763, 225592527, 226452568, 227312888, 228173487, 229034367, 229895528, 230756972, 231618701, 232480714, 233343014, 234205601, 235068477, 235931643, 236795100, 237658849, 238522891, 239387228, 240251860, 241116790, 241982017, 242847543, 243713370, 244579498, 245445929, 246312664, 247179704, 248047050, 248914703, 249782666, 250650937, 251519520, 252388415, 253257624, 254127147, 254996985, 255867141, 256737615, 257608408, 258479521, 259350957, 260222715, 261094797, 261967205, 262839939, 263713002, 264586393, 265460114, 266334167, 267208552, 268083271, 268958326, 269833716, 270709444, 271585511, 272461918, 273338666, 274215756, 275093191, 275970970, 276849095, 277727567, 278606389, 279485560, 280365082, 281244956, 282125184, 283005767, 283886706, 284768003, 285649658, 286531673, 287414049, 288296787, 289179889, 290063356, 290947189, 291831390, 292715959, 293600899, 294486209, 295371892, 296257949, 297144381, 298031190, 298918376, 299805941, 300693886, 301582213, 302470922, 303360016, 304249495, 305139361, 306029615, 306920258, 307811292, 308702717, 309594536, 310486750, 311379359, 312272365, 313165770, 314059575, 314953781, 315848389, 316743401, 317638818, 318534642, 319430873, 320327513, 321224564, 322122027, 323019902, 323918192, 324816898, 325716021, 326615563, 327515524, 328415907, 329316712, 330217941, 331119595, 332021676, 332924185, 333827123, 334730492, 335634293, 336538528, 337443197, 338348303, 339253846, 340159828, 341066251, 341973115, 342880423, 343788175, 344696373, 345605018, 346514112, 347423657, 348333652, 349244101, 350155004, 351066363, 351978180, 352890454, 353803189, 354716385, 355630045, 356544168, 357458757, 358373814, 359289339, 360205334, 361121800, 362038740, 362956154, 363874044, 364792411, 365711256, 366630582, 367550390, 368470680, 369391456, 370312717, 371234466, 372156704, 373079432, 374002652, 374926366, 375850574, 376775279, 377700482, 378626184, 379552387, 380479093, 381406302, 382334016, 383262238, 384190968, 385120208, 386049959, 386980223, 387911001, 388842296, 389774108,

Окончание таблицы Б.1

390706439, 391639290, 392572664, 393506561, 394440984, 395375933, 396311410, 397247417, 398183956, 399121027, 400058633, 400996775, 401935455, 402874673, 403814433, 404754734, 405695580, 406636971, 407578909, 408521396, 409464433, 410408022, 411352164, 412296861, 413242115, 414187927, 415134299, 416081232, 417028728, 417976789, 418925416, 419874612, 420824376, 421774712, 422725621, 423677104, 424629163, 425581800, 426535017, 427488814, 428443194, 429398159, 430353709, 431309847, 432266574, 433223893, 434181804, 435140309, 436099411, 437059110, 438019409, 438980309, 439941811, 440903918, 441866632, 442829953, 443793884, 444758426, 445723581

Таблица Б.2 — *define MAX_INV_QUANT_TABLE* 1025

/* (int)(0.5 + (1«12) * *pow((double)quant,(double)4/(double)*3)) */ *INT32 invQuantCompact[MAX INV QUANT TABLE] = {*
0, 4096, 10321, 17722, 26008, 35020, 44658, 54848, 65536, 76680, 88246, 100204, 112530, 125204, 138207, 151524, 165140, 179043, 193222, 207666, 222365, 237312, 252497, 267915, 283558, 299419, 315494, 331776, 348260, 364942, 381817, 398880, 416128, 433556, 451161, 468940, 486889, 505005, 523285, 541726, 560325, 579080, 597988, 617046, 636253, 655607, 675104, 694742, 714521, 734437, 754490, 774676, 794995, 815445, 836023, 856729, 877561, 898517, 919596, 940797, 962118, 983557, 1005114, 1026788, 1048576, 1070478, 1092493, 1114619, 1136855, 1159201, 1181655, 1204216, 1226883, 1249656, 1272533, 1295513, 1318595, 1341779, 1365063, 1388447, 1411930, 1435511, 1459190, 1482964, 1506835, 1530800, 1554860, 1579013, 1603258, 1627596, 1652025, 1676545, 1701154, 1725853, 1750641, 1775517, 1800480, 1825530, 1850666, 1875888, 1901195, 1926586, 1952062, 1977620, 2003262, 2028986, 2054792, 2080679, 2106646, 2132694, 2158822, 2185029, 2211315, 2237679, 2264122, 2290641, 2317238, 2343911, 2370660, 2397485, 2424385, 2451360, 2478409, 2505533, 2532730, 2560000, 2587343, 2614758, 2642246, 2669805, 2697436, 2725137, 2752909, 2780751, 2808663, 2836645, 2864696, 2892815, 2921003, 2949260, 2977584, 3005975, 3034434, 3062960, 3091552, 3120211, 3148935, 3177726, 3206581, 3235502, 3264487, 3293537, 3322651, 3351829, 3381071, 3410376, 3439744, 3469175, 3498668, 3528224, 3557841, 3587521, 3617262, 3647064, 3676928, 3706852, 3736836, 3766881, 3796986, 3827151, 3857375, 3887658, 3918001, 3948403, 3978863, 4009381, 4039958, 4070593, 4101285, 4132035, 4162842, 4193707, 4224628, 4255606, 4286640, 4317731, 4348878, 4380080, 4411339, 4442653, 4474022, 4505446, 4536925, 4568459, 4600047, 4631689, 4663386, 4695137, 4726941, 4758799, 4790711, 4822675, 4854693, 4886764, 4918887, 4951063, 4983291, 5015571, 5047904, 5080288, 5112724, 5145211, 5177750, 5210340, 5242981, 5275673, 5308416, 5341209, 5374053, 5406947, 5439891, 5472885, 5505929, 5539022, 5572165, 5605357, 5638599, 5671889, 5705229, 5738617, 5772054, 5805540, 5839073, 5872655, 5906285, 5939963, 5973689, 6007463, 6041284, 6075152, 6109068, 6143030, 6177040, 6211097, 6245200, 6279351, 6313547, 6347790, 6382079, 6416414, 6450796, 6485223, 6519696, 6554214, 6588778, 6623388, 6658043, 6692742, 6727487, 6762277, 6797112, 6831991, 6866915, 6901883, 6936896, 6971953, 7007054, 7042199, 7077388, 7112621, 7147897, 7183217, 7218581, 7253988, 7289438, 7324931, 7360467, 7396047, 7431669, 7467334, 7503041, 7538791, 7574584, 7610418, 7646295, 7682214, 7718176, 7754179, 7790224, 7826310, 7862439, 7898609, 7934820, 7971073, 8007367, 8043702, 8080078, 8116495, 8152954, 8189452, 8225992, 8262572, 8299193, 8335854, 8372556, 8409298, 8446080, 8482902, 8519764, 8556666, 8593608, 8630590, 8667611, 8704672, 8741772, 8778912, 8816091, 8853309, 8890567, 8927863, 8965199, 9002573, 9039986, 9077438, 9114929, 9152458, 9190026, 9227632, 9265277, 9302960, 9340681, 9378440, 9416237, 9454072, 9491946, 9529856, 9567805, 9605791, 9643815, 9681877, 9719976, 9758112, 9796285, 9834496, 9872744, 9911029, 9949351, 9987710, 10026106, 10064538, 10103007, 10141513, 10180056, 10218635, 10257251, 10295902, 10334591, 10373315, 10412076, 10450872, 10489705, 10528574, 10567479, 10606419, 10645395, 10684407, 10723455, 10762538, 10801657, 10840811, 10880000, 10919225, 10958485, 10997781, 11037111, 11076477, 11115877, 11155313, 11194783, 11234288, 11273828, 11313403, 11353012, 11392656, 11432334, 11472047, 11511794, 11551576, 11591392, 11631242, 11671126, 11711044, 11750997, 11790983, 11831003, 11871058, 11911145, 11951267, 11991423, 12031612, 12071834, 12112091, 12152380, 12192703, 12233060, 12273450, 12313873, 12354329, 12394818, 12435341, 12475896, 12516485, 12557106, 12597761, 12638448, 12679168, 12719920, 12760706, 12801523, 12842374, 12883257, 12924172, 12965120, 13006100, 13047113, 13088158, 13129235, 13170344, 13211485, 13252658, 13293864, 13335101, 13376370, 13417671, 13459004, 13500369, 13541765, 13583193, 13624652, 13666143, 13707666, 13749220, 13790806, 13832423, 13874071, 13915750, 13957461, 13999203, 14040976, 14082780, 14124615, 14166482, 14208379, 14250307, 14292266, 14334256, 14376276, 14418327, 14460409, 14502522, 14544665, 14586839, 14629043, 14671278, 14713543, 14755838, 14798164, 14840520, 14882906, 14925323, 14967770, 15010246, 15052753, 15095290, 15137857, 15180454, 15223081, 15265737, 15308424, 15351140, 15393886, 15436662, 15479467, 15522302, 15565166, 15608060, 15650984, 15693937, 15736919, 15779931, 15822972, 15866042, 15909142, 15952271, 15995429, 16038616, 16081832, 16125077, 16168351, 16211655, 16254987, 16298348, 16341738, 16385157, 16428604, 16472080, 16515585, 16559119, 16602681, 16646272, 16689892, 16733540, 16777216, 16820921, 16864654, 16908416, 16952206, 16996024, 17039871, 17083745, 17127648, 17171580, 17215539, 17259526, 17303541, 17347585, 17391656, 17435755, 17479883, 17524038, 17568221, 17612431, 17656670, 17700936, 17745230, 17789551, 17833900, 17878277, 17922681, 17967113, 18011572, 18056059, 18100573, 18145115, 18189684, 18234280, 18278903, 18323554, 18368232, 18412937, 18457670, 18502429, 18547216, 18592029, 18636870, 18681737, 18726632, 18771553, 18816502, 18861477, 18906479, 18951508, 18996563, 19041645, 19086754, 19131890, 19177052, 19222241, 19267457, 19312699, 19357967, 19403262, 19448584, 19493932, 19539306, 19584707, 19630134, 19675587, 19721067, 19766572, 19812104, 19857662, 19903247, 19948857, 19994494, 20040156, 20085845, 20131559, 20177300, 20223066, 20268859, 20314677, 20360521, 20406391, 20452287, 20498208, 20544156, 20590128, 20636127, 20682151, 20728201, 20774277, 20820378, 20866504, 20912656, 20958834, 21005037, 21051265, 21097519, 21143798, 21190103, 21236433, 21282788, 21329168, 21375574, 21422004, 21468460, 21514941, 21561448, 21607979, 21654535, 21701117, 21747723, 21794354, 21841011, 21887692, 21934398, 21981129, 22027885, 22074666, 22121472, 22168302,

Окончание таблицы Б.2

22215157, 22262037, 22308941, 22355870, 22402824, 22449802, 22496805, 22543833, 22590885, 22637962, 22685063, 22732188, 22779338, 22826513, 22873711, 22920935, 22968182, 23015454, 23062750, 23110070, 23157415, 23204784, 23252177, 23299594, 23347035, 23394500, 23441990, 23489503, 23537041, 23584602, 23632188, 23679798, 23727431, 23775088, 23822770, 23870475, 23918204, 23965957, 24013733, 24061534, 24109358, 24157206, 24205077, 24252973, 24300892, 24348834, 24396800, 24444790, 24492803, 24540840, 24588901, 24636985, 24685092, 24733223, 24781377, 24829555, 24877756, 24925980, 24974228, 25022499, 25070793, 25119111, 25167452, 25215816, 25264203, 25312613, 25361047, 25409504, 25457984, 25506486, 25555012, 25603562, 25652134, 25700729, 25749347, 25797988, 25846652, 25895339, 25944048, 25992781, 26041537, 26090315, 26139116, 26187940, 26236787, 26285656, 26334548, 26383463, 26432401, 26481361, 26530344, 26579349, 26628377, 26677428, 26726501, 26775597, 26824715, 26873856, 26923019, 26972205, 27021413, 27070644, 27119897, 27169172, 27218469, 27267789, 27317132, 27366496, 27415883, 27465292, 27514724, 27564177, 27613653, 27663151, 27712671, 27762213, 27811777, 27861364, 27910972, 27960603, 28010255, 28059930, 28109627, 28159345, 28209086, 28258848, 28308632, 28358439, 28408267, 28458117, 28507989, 28557882, 28607798, 28657735, 28707694, 28757675, 28807677, 28857701, 28907747, 28957815, 29007904, 29058015, 29108147, 29158301, 29208477, 29258674, 29308893, 29359133, 29409395, 29459678, 29509983, 29560309, 29610656, 29661025, 29711415, 29761827, 29812260, 29862715, 29913190, 29963688, 30014206, 30064745, 30115306, 30165888, 30216492, 30267116, 30317762, 30368429, 30419117, 30469826, 30520556, 30571307, 30622079, 30672873, 30723687, 30774523, 30825379, 30876257, 30927155, 30978075, 31029015, 31079976, 31130958, 31181961, 31232985, 31284030, 31335095, 31386182, 31437289, 31488417, 31539565, 31590735, 31641925, 31693136, 31744367, 31795620, 31846893, 31898186, 31949500, 32000835, 32052191, 32103567, 32154963, 32206380, 32257818, 32309276, 32360755, 32412254, 32463773, 32515313, 32566874, 32618455, 32670056, 32721677, 32773320, 32824982, 32876665, 32928368, 32980091, 33031835, 33083598, 33135383, 33187187, 33239012, 33290856, 33342721, 33394607, 33446512, 33498437, 33550383, 33602349, 33654335, 33706341, 33758367, 33810413, 33862479, 33914565, 33966671, 34018797, 34070943, 34123109, 34175295, 34227501, 34279727, 34331973, 34384238, 34436524, 34488829, 34541154, 34593499, 34645864, 34698248, 34750653, 34803077, 34855521, 34907984, 34960468, 35012971, 35065494, 35118036, 35170598, 35223180, 35275781, 35328402, 35381043, 35433703, 35486383, 35539082, 35591801, 35644539, 35697297, 35750074, 35802871, 35855687, 35908523, 35961378, 36014253, 36067147, 36120061, 36172994, 36225946, 36278917, 36331908, 36384919, 36437948, 36490997, 36544065, 36597153, 36650259, 36703385, 36756530, 36809695, 36862878, 36916081, 36969303, 37022544, 37075804, 37129084, 37182382, 37235700, 37289037, 37342393, 37395767, 37449161, 37502574, 37556006, 37609457, 37662927, 37716416, 37769924, 37823451, 37876997, 37930562, 37984145, 38037748, 38091369, 38145010, 38198669, 38252347, 38306044, 38359760, 38413494, 38467248, 38521020, 38574811, 38628620, 38682449, 38736296, 38790162, 38844046, 38897950, 38951872, 39005812, 39059772, 39113749, 39167746, 39221761, 39275795, 39329847, 39383918, 39438008, 39492116, 39546242, 39600387, 39654551, 39708733, 39762934, 39817153, 39871391, 39925647, 39979921, 40034214, 40088525, 40142855, 40197203, 40251569, 40305954, 40360357, 40414779, 40469219, 40523677, 40578153, 40632648, 40687161, 40741692, 40796242, 40850810, 40905396, 40960000, 41014622, 41069263, 41123922, 41178599, 41233294, 41288007, 41342739, 41397488, 41452256, 41507042, 41561845, 41616667, 41671507, 41726365, 41781241, 41836135, 41891047, 41945977, 42000925, 42055892, 42110876, 42165877, 42220897, 42275935
};

Таблица Б.3 — *define MAX_THR_TABLE* 48

/* *(int) (pow(((double)quant*+1-0.4054),4.0/3.0) * pow(2.0,res/4.0) * (1«12)) */ *INT*32 *thrCompact*[4][*MAX_THR_TABLE*] = {
{2047, 7630, 14603, 22554, 31286, 40680, 50653, 61145, 72109, 83508, 95310, 107489, 120025, 132897, 146088, 159585, 173373, 187441, 201778, 216375, 231223, 246312, 261637, 277189, 292963, 308952, 325150, 341553, 358155, 374952, 391940, 409113, 426469, 444003, 461711, 479592, 497640, 515854, 534230, 552765, 571457, 590304, 609302, 628449, 647743, 667182, 686763, 706485}, {2435, 9074, 17366, 26821, 37206, 48377, 60237, 72714, 85753, 99308, 113343, 127827, 142734, 158042, 173729, 189779, 206176, 222906, 239956, 257315, 274972, 292916, 311140, 329635, 348393, 367407, 386671, 406177, 425921, 445896, 466097, 486520, 507159, 528011, 549071, 570334, 591797, 613457, 635310, 657353, 679581, 701993, 724586, 747356, 770301, 793417, 816704, 840158}, {2896, 10791, 20652, 31896, 44245, 57530, 71635, 86473, 101978, 118098, 134788, 152013, 169741, 187944, 206600, 225687, 245186, 265082, 285358, 306001, 326998, 348338, 370010, 392005, 414312, 436924, 459832, 483029, 506508, 530263, 554286, 578573, 603118, 627915, 652959, 678245, 703770, 729528, 755516, 781729, 808163, 834816, 861683, 888761, 916047, 943538, 971230, 999121}, {3444, 12833, 24559, 37931, 52617, 68416, 85189, 102834, 121273, 140443, 160292, 180775, 201857, 223505, 245690, 268389, 291578, 315237, 339350, 363899, 388869, 414246, 440019, 466175, 492703, 519593, 546835, 574422, 602343, 630592, 659161, 688044, 717232, 746721, 776503, 806574, 836928, 867560, 898465, 929637, 961073, 992769, 1024719, 1056921, 1089370, 1122062, 1154994, 1188162}; }
*INT*32 *KBDWindow[SINE DATA SIZE*/2] = {
0, 157068, 222127, 224278, 314136, 277325, 320246, 323287, 23513, 364876, 399747, 403470, 461694, 439880, 470330, 474628, 33252, 508076, 535676, 540482, 587058, 572042, 597636, 602903, 39500, 633186, 657292, 662984, 703111, 692375, 715334, 721423, 47025, 750182, 772236, 778699, 814650, 807011, 828335, 835153, 56443, 863155, 883884, 891042, 924055, 918837, 939075, 946561, 66007, 974230, 994060, 1001862, 1032698, 1029470, 1048961, 1057069, 75035, 1084669, 1103877, 1112281, 1141453, 1139917, 1158889, 1167583, 0, 1195290, 1214067, 1223044, 1250913, 1250853, 1269469, 1278723, 95586, 1306660, 1325145, 1334670, 1361506, 1362759, 1381138, 1390931, 106629, 1419191, 1437487, 1447543, 1473551, 1475992, 1494226, 1504541, 118311, 1533194, 1551384, 1561954, 1587298, 1590826, 1608988, 1619811, 127425, 1648913, 1667061, 1678134, 1702944, 1707478, 1725626, 1736947, 143394, 1766543, 1784703, 1796268, 1820653, 1826126, 1844309, 1856118, 156519, 1886246, 1904462, 1916512, 1940560, 1946918, 1965177, 1977466, 171004, 2008158, 2026469, 2038996, 2062785, 2069980, 2088350, 2101113, 0, 2132397, 2150833, 2163833, 2187428, 2195422, 2213931, 2227165, 201310, 2259065, 2277655, 2291123, 2314580, 2323339, 2342015, 2355715, 216944, 2388253, 2407021, 2420954, 2444323, 2453818, 2472684, 2486848, 234479, 2520044, 2539012, 2553407, 2576732, 2586938, 2606014, 2620639, 247730, 2654511, 2673699, 2688554, 2711874, 2722770, 2742074, 2757160, 270684, 2791724, 2811149, 2826464, 2849813, 2861381, 2880930, 2896475, 289211, 2931748, 2951426, 2967200, 2990609, 3002832, 3022643, 3038646, 310098, 3074642, 3094588, 3110821, 3134319, 3147184, 3167268, 3183731, 0, 3220464, 3240691, 3257383, 3280996, 3294490, 3314863, 3331785, 352902, 3369268, 3389790, 3406941, 3430692, 3444805, 3465478, 3482860, 374701, 3521107, 3541934, 3559547, 3583458, 3598180, 3619165, 3637007, 399279, 3676030, 3697175, 3715249, 3739341, 3754664, 3775971, 3794276, 418193, 3834087, 3855559, 3874096, 3898388, 3914305, 3935945, 3954715, 449420, 3995325, 4017134, 4036136, 4060645, 4077151, 4099132, 4118368, 474878, 4159789, 4181946, 4201415, 4226156, 4243246, 4265579, 4285282, 503519, 4327526, 4350038, 4369976, 4394967, 4412635, 4435328, 4455502, 0,4498578, 4521455, 4541865, 4567119, 4585362, 4608424, 4629071, 561777, 4672991, 4696241, 4717124, 4742655, 4761471, 4784910, 4806031, 591298, 4850806, 4874437, 4895797, 4921618, 4941003, 4964827, 4986426, 624400, 5032067, 5056086, 5077925, 5104049, 5124002, 5148218, 5170298, 650486, 5216814, 5241229, 5263550, 5289989, 5310508, 5335124, 5357687, 691600, 5405089, 5429908, 5452714, 5479478, 5500563, 5525585, 5548636, 725607, 5596934, 5622162, 5645457, 5672558, 5694207, 5719644, 5743184, 763597, 5792388, 5818034, 5841820, 5869268, 5891481, 5917339, 5941372, 0,5991492, 6017563, 6041844, 6069649, 6092426, 6118711, 6143241, 840613, 6194287, 6220789, 6245567, 6273739, 6297081, 6323800, 6348829, 879548, 6400813, 6427751, 6453031, 6481579, 6505486, 6532646, 6558178, 922879, 6611108, 6638490, 6664275, 6693208, 6717681, 6745288, 6771327, 957774, 6825212, 6853044, 6879338, 6908666, 6933705, 6961765, 6988314, 1010631, 7043166, 7071454, 7098260, 7127990, 7153598, 7182116, 7209180, 1054961, 7265007, 7293757, 7321080, 7351221, 7377398, 7406382, 7433963, 1104111, 7490776, 7519994, 7547836, 7578396, 7605145, 7634599, 7662703, 0, 7720510, 7750203, 7778568, 7809555, 7836877, 7866809, 7895437, 1203568, 7954250, 7984422, 8013315, 8044737, 8072634, 8103048, 8132206, 1253784, 8192033, 8222691, 8252115, 8283979, 8312453, 8343356, 8373048, 1309257, 8433899, 8465048, 8495008, 8527321, 8556375, 8587772, 8618001, 1354767, 8679886, 8711532, 8742031, 8774801, 8804437, 8836334, 8867103, 1421439, 8930032, 8962181, 8993223, 9026458, 9056677, 9089080, 9120395, 1478054, 9184376, 9217034, 9248623, 9282329, 9313135, 9346048, 9377913, 1540381, 9442957, 9476128, 9508269, 9542453, 9573848, 9607278, 9639696, 0, 9705813, 9739503, 9772199, 9806868, 9838856, 9872808, 9905783, 1666355, 9972982,

Продолжение таблицы Б.3

10007197, 10040452, 10075612, 10108196, 10142675, 10176212, 1729907, 10244502, 10279247, 10313066, 10348724, 10381906, 10416917, 10451021, 1799643, 10520412, 10555692, 10590079, 10626241, 10660024, 10695574, 10730247, 1857759, 10800749, 10836570, 10871530, 10908202, 10942590, 10978683, 11013931, 1940530, 11085552, 11121919, 11157455, 11194644, 11229640, 11266282, 11302108, 2011582, 11374859, 11411777, 11447894, 11485607, 11521213, 11558409, 11594818, 2089309, 11668708, 11706182, 11742884, 11781126, 11817347, 11855102, 11892098, 0, 11967137, 12005173, 12042464, 12081241, 12118080, 12156399, 12193986, 2246277, 12270183, 12308786, 12346671, 12385989, 12423450, 12462338, 12500521, 2325417, 12577885, 12617060, 12655542, 12695408, 12733494, 12772956, 12811739, 2411740, 12890280, 12930032, 12969117, 13009536, 13048250, 13088292, 13127680, 2484646, 13207407, 13247741, 13287432, 13328411, 13367756, 13408383, 13448379, 2586010, 13529302, 13570224, 13610526, 13652069, 13692050, 13733267, 13773876, 2673850, 13856004, 13897518, 13938435, 13980549, 14021169, 14062981, 14104208, 2769403, 14187550, 14229661, 14271198, 14313888, 14355152, 14397563, 14439411, 0, 14523978, 14566692, 14608852, 14652124, 14694034, 14737051, 14779525, 2962243, 14865325, 14908646, 14951435, 14995295, 15037855, 15081482, 15124586, 3059421, 15211629, 15255562, 15298984, 15343437, 15386651, 15430893, 15474632, 3164860, 15562927, 15607477, 15651536, 15696587, 15740460, 15785321, 15829700, 3254940, 15919256, 15964429, 16009129, 16054784, 16099319, 16144805, 16189828, 3377591, 16280654, 16326454, 16371800, 16418065, 16463266, 16509381, 16555052, 3484768, 16647158, 16693590, 16739587, 16786466, 16832337, 16879086, 16925409, 3600776, 17018805, 17065874, 17112525, 17160025, 17206569, 17253958, 17300938, 0, 17395633, 17443343, 17490653, 17538778, 17586000, 17634033, 17681675, 3834765, 17777677, 17826034, 17874007, 17922763, 17970667, 18019349, 18067656, 3952629, 18164975, 18213984, 18262625, 18312017, 18360606, 18409942, 18458919, 4079912, 18557564, 18607229, 18656543, 18706575, 18755854, 18805849, 18855500, 4189746, 18955481, 19005807, 19055797, 19106476, 19156449, 19207107, 19257437, 4336577, 19358762, 19409754, 19460425, 19511756, 19562426, 19613753, 19664766, 4465839, 19767445, 19819107, 19870463, 19922450, 19973822, 20025822, 20077523, 4605126, 20181564, 20233902, 20285948, 20338597, 20390674, 20443352, 20495744, 0, 20601158, 20654175, 20706916, 20760231, 20813019, 20866378, 20919467, 4885932, 21026262, 21079964, 21133403, 21187390, 21240891, 21294938, 21348727, 5027325, 21456912, 21511304, 21565445, 21620109, 21674328, 21729067, 21783561, 5179372, 21893145, 21948231, 22003079, 22058425, 22113366, 22168801, 22224005, 5311737, 22334996, 22390781, 22446341, 22502374, 22558040, 22614176, 22670094, 5485830, 22782502, 22838990, 22895266, 22951990, 23008387, 23065229, 23121864, 5640114, 23235699, 23292895, 23349891, 23407312, 23464442, 23521994, 23579352, 5805696, 23694621, 23752530, 23810251, 23868373, 23926241, 23984508, 24042593, 0, 24159306, 24217932, 24276381, 24335209, 24393820, 24452806, 24511622, 6139365, 24629787, 24689135, 24748318, 24807856, 24867213, 24926923, 24986474, 6307318, 25106102, 25166175, 25226096, 25286350, 25346457, 25406896, 25467186, 6487236, 25588284, 25649088, 25709750, 25770725, 25831586, 25892758, 25953792, 6645096, 26076369, 26137908, 26199316, 26261016, 26322636, 26384545, 26446327, 6849716, 26570391, 26632671, 26694829, 26757259, 26819641, 26882292, 26944826, 7032142, 27070387, 27133411, 27196323, 27259488, 27322636, 27386033, 27449325, 7227216, 27576390, 27640163, 27703834, 27767737, 27831656, 27895804, 27959856, 0, 28088436, 28152961, 28217395, 28282042, 28346735, 28411639, 28476456, 7620152, 28606558, 28671841, 28737042, 28802437, 28867908, 28933571, 28999158, 7817874, 29130791, 29196835, 29262808, 29328956, 29395209, 29461636, 29527996, 8028945, 29661168, 29727979, 29794727, 29861633, 29928672, 29995868, 30063005, 8215447, 30197725, 30265307, 30332834, 30400502, 30468331, 30536300, 30604218, 8454023, 30740495, 30808851, 30877162, 30945596, 31014220, 31082965, 31151670, 8667885, 31289511, 31358647, 31427745, 31496950, 31566373, 31635899, 31705393, 8895815, 31844808, 31914726, 31984617, 32054597, 32124822, 32195133, 32265422, 0, 32406418, 32477124, 32547811, 32618570, 32689601, 32760702, 32831788, 9354759, 32974374, 33045872, 33117359, 33188903, 33260743, 33332638, 33404527, 9585624, 33548711, 33621005, 33693295, 33765628, 33838282, 33910975, 33983669, 9831294, 34129460, 34202554, 34275653, 34348777, 34422249, 34495745, 34569249, 10049751, 34716654, 34790552, 34864463, 34938385, 35012678, 35086980, 35161299, 10325866, 35310326, 35385032, 35459760, 35534482, 35609601, 35684713, 35759850, 10574617, 35910508, 35986027, 36061574, 36137102, 36213050, 36288977, 36364936, 10838923, 36517232, 36593568, 36669939, 36746277, 36823057, 36899803, 36976587, 0, 37130530, 37207687, 37284886, 37362037, 37439655, 37517223, 37594837, 11370920, 37750434, 37828416, 37906446, 37984416, 38062874, 38141269, 38219717, 11638455, 38376976, 38455786, 38534652, 38613444, 38692746, 38771972, 38851257, 11922313, 39010187, 39089829, 39169535, 39249153, 39329302, 39409363, 39489490, 12176196, 39650097, 39730576, 39811125, 39891573, 39972575, 40053473, 40134446, 12493565, 40296739, 40378058, 40459454, 40540736, 40622593, 40704334, 40786155, 12780800, 40950142, 41032306, 41114553, 41196673, 41279388, 41361976, 41444650, 13085139, 41610337, 41693349, 41776451, 41859413, 41942991, 42026429, 42109959, 0, 42277355, 42361219, 42445179, 42528987, 42613432, 42697723, 42782114, 13697504, 42951225, 43035945, 43120767, 43205425, 43290739, 43375889, 43461143, 14005370, 43631978, 43717558, 43803245, 43888757, 43974945, 44060955, 44147077, 14331133, 44319643, 44406086, 44492643, 44579013, 44666077, 44752953, 44839946, 14624049,

Продолжение таблицы Б.3

45014250, 45101559, 45188989, 45276220, 45364165, 45451910, 45539777, 14986500, 45715827, 45804007, 45892314, 45980410, 46069238, 46157856, 46246602, 15315939, 46424404, 46513459, 46602645, 46691610, 46781326, 46870820, 46960447, 15664082, 47140009, 47229942, 47320012, 47409850, 47500456, 47590829, 47681342, 0, 47862671, 47953486, 48044443, 48135157, 48226657, 48317914, 48409316, 16364361, 48592418, 48684118, 48775966, 48867560, 48959958, 49052101, 49144395, 16716329, 49329279, 49421867, 49514608, 49607086, 49700385, 49793419, 49886609, 17087820, 50073280, 50166760, 50260399, 50353764, 50447967, 50541894, 50635983, 17423496, 50824449, 50918825, 51013365, 51107620, 51202730, 51297555, 51392547, 17834944, 51582814, 51678089, 51773533, 51868682, 51964703, 52060429, 52156326, 18210408, 52348401, 52444579, 52540930, 52636977, 52733912, 52830542, 52927348, 18606222, 53121238, 53218321, 53315583, 53412531, 53510383, 53607920, 53705638, 0, 53901350, 53999342, 54097518, 54195370, 54294142, 54392590, 54491224, 19402143, 54688764, 54787668, 54886761, 54985521, 55085217, 55184579, 55284131, 19802077, 55483505, 55583325, 55683338, 55783010, 55883631, 55983911, 56084385, 20223198, 56285600, 56386338, 56487275, 56587861, 56689412, 56790612, 56892011, 20605455, 57095073, 57196734, 57298597, 57400100, 57502584, 57604707, 57707035, 21069880, 57911949, 58014536, 58117328, 58219752, 58323172, 58426222, 58529481, 21495267, 58736255, 58839769, 58943494, 59046843, 59151201, 59255180, 59359373, 21942685, 59568013, 59672458, 59777120, 59881395, 59986694, 60091607, 60196737, 0, 60407248, 60512628, 60618228, 60723433, 60829677, 60935525, 61041595, 22842107, 61253984, 61360301, 61466843, 61572981, 61680172, 61786959, 61893972, 23293934, 62108244, 62215502, 62322988, 62430063, 62538204, 62645933, 62753892, 23768642, 62970053, 63078254, 63186687, 63294701, 63403795, 63512468, 63621376, 24201373, 63839432, 63948579, 64057962, 64166919, 64276968, 64386589, 64496448, 24722788, 64716405, 64826501, 64936837, 65046738, 65157746, 65268317, 65379131, 25202046, 65600993, 65712041, 65823333, 65934182, 66046150, 66157675, 66269446, 25705041, 66493220, 66605222, 66717472, 66829272, 66942204, 67054685, 67167415, 0, 67393106, 67506066, 67619277, 67732030, 67845928, 67959367, 68073060, 26715899, 68300673, 68414593, 68528768, 68642478, 68757344, 68871745, 68986402, 27223581, 69215943, 69330825, 69445966, 69560635, 69676473, 69791837, 69907462, 27755858, 70138936, 70254783, 70370893, 70486523, 70603335, 70719665, 70836261, 28242998, 71069672, 71186487, 71303568, 71420163, 71537950, 71655250, 71772818, 28825417, 72008172, 72125957, 72244012, 72361573, 72480339, 72598611, 72717154, 29362516, 72954455, 73073213, 73192244, 73310775, 73430522, 73549767, 73669287, 29925070, 73908541, 74028275, 74148284, 74267787, 74388517, 74508738, 74629238, 0, 74870450, 74991160, 75112151, 75232628, 75354343, 75475543, 75597025, 31055314, 75840199, 75961889, 76083863, 76205316, 76328019, 76450201, 76572667, 31622819, 76817807, 76940480, 77063439, 77185871, 77309564, 77432729, 77556182, 32216641, 77803293, 77926950, 78050897, 78174311, 78298996, 78423146, 78547588, 32762138, 78796674, 78921318, 79046255, 79170652, 79296331, 79421469, 79546901, 33409546, 79797968, 79923600, 80049529, 80174912, 80301587, 80427715, 80554141, 34008443, 80807191, 80933815, 81060737, 81187108, 81314781, 81441901, 81569322, 34634517, 81824360, 81951977, 82079896, 82207256, 82335930, 82464044, 82592462, 0, 82849492, 82978104, 83107021, 83235374, 83365049, 83494160, 83623577, 35892045, 83882603, 84012212, 84142129, 84271476, 84402155, 84532264, 84662681, 36523314, 84923708, 85054316, 85185235, 85315578, 85447263, 85578371, 85709792, 37182616, 85972822, 86104431, 86236354, 86367695, 86500388, 86632498, 86764923, 37790398, 87029961, 87162573, 87295501, 87427842, 87561544, 87694658, 87828090, 38506714, 88095139, 88228755, 88362691, 88496033, 88630747, 88764866, 88899306, 39171324, 89168370, 89302992, 89437937, 89572283, 89708009, 89843136, 89978586, 39864821, 90249667, 90385298, 90521253, 90656605, 90793345, 90929481, 91065942, 0, 91339045, 91475685, 91612653, 91749011, 91886768, 92023914, 92161388, 41257413, 92436515, 92574167, 92712149, 92849516, 92988290, 93126448, 93264937, 41956324, 93542092, 93680757, 93819754, 93958131, 94097924, 94237095, 94376601, 42684967, 94655786, 94795466, 94935480, 95074869, 95215681, 95355868, 95496392, 43358910, 95777610, 95918306, 96059338, 96199740, 96341575, 96482779, 96624321, 44147953, 96907576, 97049289, 97191341, 97332758, 97475615, 97617837, 97760399, 44882109, 98045694, 98188425, 98331498, 98473931, 98617813, 98761054, 98904639, 45646838, 99191975, 99335726, 99479822, 99623272, 99768179, 99912441, 100057048, 0, 100346429, 100491201, 100636320, 100780790, 100926724, 101072008, 101217639, 47182084, 101509066, 101654861, 101801005, 101946495, 102093456, 102239763, 102386420, 47952418, 102679896, 102826715, 102973884, 103120395, 103268386, 103415717, 103563400, 48754151, 103858927, 104006771, 104154968, 104302501, 104451521, 104599878, 104748588, 49498043, 105046169, 105195038, 105344263, 105492820, 105642871, 105792254, 105941993, 50363492, 106241629, 106391526, 106541779, 106691361, 106842444, 106992854, 107143623, 51170914, 107445316, 107596240, 107747524, 107898131, 108050247, 108201685, 108353484, 52010554, 108657236, 108809190, 108961504, 109113138, 109266287, 109418754, 109571585, 0, 109877398, 110030381, 110183727, 110336388, 110490571, 110644069, 110797931, 53695780, 111105807, 111259820, 111414198, 111567888, 111723106, 111877635, 112032529, 54541183, 112342469, 112497514, 112652925, 112807643, 112963897, 113119458, 113275386, 55419608, 113587391, 113743467, 113899913, 114055661, 114212951, 114369543, 114526506, 56237112, 114840578, 114997687, 115155166, 115311945, 115470272, 115627897, 115785894, 57182468, 116102034, 116260176, 116418691,

Продолжение таблицы Б.3

116576500, 116735865, 116894523, 117053556, 58066721, 117371764, 117530940, 117690490, 117849332, 118009734, 118169427, 118329495, 58984785, 118649773, 118809983, 118970569, 119130443, 119291883, 119452610, 119613714, 0, 119936064, 120097308, 120258931, 120419837, 120582316, 120744078, 120906218, 60826977, 121230639, 121392919, 121555578, 121717517, 121881035, 122043832, 122207010, 61750923, 122533503, 122696818, 122860514, 123023486, 123188043, 123351876, 123516090, 62709454, 123844656, 124009007, 124173740, 124337747, 124503342, 124668211, 124833463, 63604069, 125164102, 125329489, 125495259, 125660300, 125826935, 125992839, 126159129, 64632621, 126491841, 126658264, 126825072, 126991147, 127158821, 127325762, 127493089, 65597078, 127827875, 127995334, 128163180, 128330289, 128499003, 128666980, 128835344, 66596874, 129172204, 129340699, 129509583, 129677726, 129847480, 130016493, 130185895, 0, 130524829, 130694360, 130864281, 131033459, 131204252, 131374301, 131544741, 68602600, 131885749, 132056316, 132227275, 132397487, 132569319, 132740404, 132911882, 69608354, 133254963, 133426566, 133598562, 133769809, 133942680, 134114801, 134287316, 70650181, 134632470, 134805109, 134978142, 135150423, 135324333, 135497490, 135671042, 71625208, 136018269, 136191944, 136366014, 136539328, 136714277, 136888469, 137063058, 72739988, 137412357, 137587067, 137762174, 137936522, 138112509, 138287737, 138463362, 73787794, 138814732, 138990478, 139166620, 139342002, 139519026, 139695289, 139871950, 74872384, 140225391, 140402171, 140579350, 140755765, 140933826, 141111124, 141288820, 0, 141644331, 141822146, 142000360, 142177807, 142356905, 142535236, 142713968, 77047722, 143071548, 143250396, 143429645, 143608124, 143788259, 143967623, 144147389, 78138300, 144507037, 144686918, 144867201, 145046712, 145227882, 145408279, 145589079, 79266349, 145950793, 146131707, 146313024, 146493566, 146675770, 146857200, 147039033, 80324848, 147402812, 147584758, 147767107, 147948680, 148131918, 148314380, 148497245, 81528598, 148863088, 149046064, 149229446, 149412049, 149596320, 149779812, 149963709, 82662630, 150331614, 150515621, 150700034, 150883665, 151068969, 151253491, 151438419, 83834796, 151808384, 151993420, 152178864, 152363523, 152549858, 152735409, 152921367, 0, 153293391, 153479456, 153665929, 153851615, 154038981, 154225560, 154412547, 86185256, 154786627, 154973720, 155161221, 155347934, 155536329, 155723935, 155911950, 87363386, 156288084, 156476204, 156664732, 156852471, 157041894, 157230526, 157419568, 88580284, 157797755, 157986900, 158176455, 158365217, 158555667, 158745324, 158935392, 89725040, 159315629, 159505798, 159696378, 159886164, 160077640, 160268321, 160459413, 91020173, 160841698, 161032890, 161224494, 161415302, 161607802, 161799506, 161991621, 92243004, 162375951, 162568166, 162760792, 162952620, 163146144, 163338869, 163532006, 93505207, 163918379, 164111614, 164305262, 164498110, 164692655, 164886400, 165080558, 0, 165468971, 165663225, 165857893, 166051758, 166247324, 166442088, 166637265, 96035657, 167027714, 167222987, 167418673, 167613555, 167810140, 168005920, 168202115, 97303745, 168594599, 168790887, 168987591, 169183488, 169381090, 169577886, 169775097, 98611782, 170169612, 170366915, 170564634, 170761545, 170960163, 171157973, 171356198, 99845266, 171752741, 171951057, 172149789, 172347713, 172547344, 172746167, 172945405, 101233831, 173343972, 173543300, 173743044, 173941978, 174142622, 174342455, 174542705, 102547691, 174943293, 175143630, 175344385, 175544327, 175745982, 175946824, 176148083, 103902040, 176550688, 176752034, 176953797, 177154747, 177357410, 177559259, 177761525, 0, 178166144, 178368496, 178571266, 178773221, 178976890, 179179745, 179383017, 106616633, 179789646, 179993002, 180196776, 180399735, 180604409, 180808267, 181012543, 107976728, 181421177, 181625536, 181830313, 182034273, 182239950, 182444809, 182650086, 109377820, 183060723, 183266082, 183471860, 183676819, 183883496, 184089355, 184295632, 110702154, 184708267, 184914624, 185121400, 185327356, 185535032, 185741888, 185949163, 112185800, 186363791, 186571145, 186778917, 186985868, 187194540, 187402391, 187610661, 113592546, 188027279, 188235626, 188444393, 188652337, 188862003, 189070847, 189280110, 115040759, 189698712, 189908051, 190117810, 190326745, 190537403, 190747237, 190957490, 0, 191378073, 191588401, 191799149, 192009073, 192220720, 192431543, 192642784, 117942868, 193065342, 193276658, 193488393, 193699303, 193911937, 194123745, 194335973, 119396625, 194760501, 194972801, 195185521, 195397415, 195611033, 195823824, 196037036, 120892285, 196463529, 196676812, 196890514, 197103389, 197317988, 197531761, 197745953, 122309207, 198174407, 198388669, 198603351, 198817205, 199032783, 199247534, 199462704, 123889155, 199893114, 200108353, 200324011, 200538841, 200755396, 200971123, 201187269, 125390232, 201619628, 201835841, 202052474, 202268278, 202485807, 202702506, 202919625, 126933608, 203353929, 203571114, 203788718, 204005492, 204223992, 204441662, 204659751, 0, 205095993, 205314147, 205532720, 205750463, 205969930, 206188567, 206407624, 130025760, 206845800, 207064919, 207284458, 207503166, 207723599, 207943200, 208163221, 131574410, 208603324, 208823407, 209043909, 209263579, 209484974, 209705537, 209926519, 133165714, 210368544, 210589587, 210811049, 211031678, 211254032, 211475554, 211697495, 134676547, 212141435, 212363435, 212585854, 212807440, 213030750, 213253227, 213476123, 136353556, 213921973, 214144927, 214368299, 214590839, 214815102, 215038531, 215262380, 137949973, 215710133, 215934037, 216158361, 216381851, 216607063, 216831442, 217056239, 139589358, 217505889, 217730741, 217956012, 218180449, 218406608, 218631933, 218857676, 0, 219309216, 219535013, 219761228, 219986609, 220213710, 220439978, 220666664,

Продолжение таблицы Б.3

142873178, 221120087, 221346826, 221573981, 221800303, 222028344, 222255551, 222483176, 144517498, 222938477, 223166153, 223394246, 223621505, 223850482, 224078626, 224307186, 146205058, 224764357, 224992967, 225221994, 225450187, 225680097, 225909173, 226138666, 147810679, 226597700, 226827241, 227057198, 227286321, 227517161, 227747166, 227977587, 149585024, 228438477, 228668946, 228899830, 229129880, 229361645, 229592576, 229823922, 151277322, 230286661, 230518053, 230749861, 230980835, 231213521, 231445374, 231677642, 153013087, 232142222, 232374535, 232607262, 232839155, 233072760, 233305531, 233538717, 0, 234005131, 234238360, 234472003, 234704812, 234939332, 235173017, 235407117, 156489243, 235875358, 236109499, 236344054, 236577776, 236813206, 237047802, 237282812, 158229531, 237752872, 237987922, 238223385, 238458015, 238694352, 238929855, 239165771, 160013469, 239637642, 239873597, 240109965, 240345500, 240582739, 240819145, 241055963, 161714284, 241529638, 241766494, 242003762, 242240197, 242478335, 242715640, 242953357, 163585731, 243428827, 243666580, 243904744, 244142076, 244381109, 244619309, 244857920, 165373960, 245335177, 245573823, 245812880, 246051104, 246291027, 246530118, 246769619, 167205975, 247248655, 247488190, 247728135, 247967248, 248208058, 248448035, 248688423, 0, 249169229, 249409649, 249650478, 249890475, 250132167, 250373027, 250614296, 170874130, 251096865, 251338165, 251579873, 251820751, 252063320, 252305059, 252547206, 172710184, 253031528, 253273704, 253516288, 253758042, 254001485, 254244097, 254487117, 174590113, 254973185, 255216233, 255459688, 255702313, 255946625, 256190107, 256433996, 176386030, 256921800, 257165715, 257410037, 257653530, 257898706, 258143053, 258387806, 178353819, 258877338, 259122116, 259367300, 259611656, 259857692, 260102900, 260348513, 180237516, 260839763, 261085400, 261331441, 261576655, 261823547, 262069611, 262316080, 182165130, 262809039, 263055530, 263302424, 263548492, 263796234, 264043150, 264290470, 0, 264785129, 265032469, 265280212, 265527129, 265775718, 266023481, 266271646, 186023911, 266767996, 267016181, 267264768, 267512529, 267761960, 268010565, 268259572, 187955004, 268757603, 269006628, 269256053, 269504655, 269754922, 270004365, 270254208, 189930007, 270753911, 271003771, 271254031, 271503468, 271754567, 272004842, 272255517, 191820421, 272756883, 273007573, 273258662, 273508929, 273760855, 274011959, 274263461, 193883250, 274766478, 275017994, 275269908, 275521001, 275773748, 276025675, 276277999, 195861424, 276782659, 277034995, 277287728, 277539642, 277793206, 278045951, 278299092, 197883452, 278805385, 279058537, 279312084, 279564813, 279819190, 280072748, 280326701, 0, 280834616, 281088578, 281342935, 281596475, 281851658, 282106024, 282360784, 201930410, 282870312, 283125080, 283380241, 283634586, 283890571, 284145740, 284401301, 203955281, 284912431, 285167999, 285423959, 285679105, 285935886, 286191853, 286448211, 206023901, 286960932, 287217295, 287474050, 287729991, 287987563, 288244322, 288501471, 208007675, 289015774, 289272927, 289530470, 289787201, 290045559, 290303105, 290561041, 210163689, 291076914, 291334852, 291593178, 291850693, 292109832, 292368160, 292626876, 212234807, 293144309, 293403026, 293662130, 293920425, 294180339, 294439444, 294698935, 214349516, 295217917, 295477408, 295737285, 295996353, 296257037, 296516913, 296777174, 0, 297297694, 297557953, 297818597, 298078434, 298339882, 298600523, 298861548, 218581111, 299383596, 299644618, 299906023, 300166623, 300428830, 300690231, 300952015, 220697954, 301475579, 301737358, 301999519, 302260877, 302523836, 302785992, 303048529, 222858182, 303573597, 303836128, 304099040, 304361150, 304624856, 304887761, 305151046, 224933636, 305677607, 305940884, 306204541, 306467397, 306731845, 306995493, 307259519, 227180422, 307787563, 308051580, 308315976, 308579572, 308844756, 309109142, 309373904, 229342401, 309903419, 310168170, 310433299, 310697630, 310963544, 311228661, 311494154, 231547506, 312025127, 312290608, 312556464, 312821524, 313088162, 313354005, 313620222, 0, 314152643, 314418847, 314685424, 314951208, 315218564, 315485127, 315752063, 235959091, 316285919, 316552839, 316820132, 317086633, 317354702, 317621979, 317889627, 238165546, 318424907, 318692539, 318960541, 319227754, 319496528, 319764513, 320032868, 240414819, 320569560, 320837896, 321106602, 321374520, 321643995, 321912682, 322181737, 242579722, 322719829, 322988864, 323258268, 323526885, 323797054, 324066437, 324336187, 244914311, 324875666, 325145394, 325415489, 325684800, 325955657, 326225730, 326496168, 247164513, 327037022, 327307437, 327578217, 327848215, 328119754, 328390510, 328661631, 249457173, 329203847, 329474943, 329746402, 330017082, 330289296, 330560730, 330832526, 0, 331376093, 331647863, 331919995, 332191349, 332464232, 332736338, 333008804, 254042995, 333553708, 333826147, 334098945, 334370968, 334644514, 334917285, 335190414, 256336148, 335736643, 336009744, 336283202, 336555887, 336830090, 337103519, 337377305, 258671348, 337924847, 338198603, 338472716, 338746057, 339020909, 339294991, 339569427, 260922919, 340118269, 340392674, 340667434, 340941424, 341216920, 341491648, 341766728, 263341785, 342316857, 342591905, 342867305, 343141939, 343418072, 343693439, 343969157, 265677021, 344520559, 344796243, 345072278, 345347548, 345624312, 345900311, 346176661, 268053844, 346729324, 347005637, 347282300, 347558200, 347835588, 348112214, 348389188, 0, 348943098, 349220034, 349497318, 349773842, 350051847, 350329093, 350606685, 272807046, 351161829, 351439382, 351717280, 351994421, 352273037, 352550896, 352829099, 275183433, 353385464, 353663627, 353942133, 354219884, 354499104, 354777569, 355056376, 277600894, 355613949, 355892715, 356171822, 356450177, 356729994, 357009058, 357288464, 279935803, 357847231, 358126593,

Продолжение таблицы Б.3

358406294, 358685246, 358965653, 359245311, 359525307, 282434875, 360085254, 360365206, 360645495, 360925037, 361206028, 361486271, 361766852, 284851411, 362327966, 362608500, 362889370, 363169495, 363451062, 363731885, 364013043, 287308460, 364575310, 364856420, 365137863, 365418565, 365700702, 365982097, 366263825, 0, 366827232, 367108910, 367390921, 367672192, 367954892, 368236852, 368519144, 292221101, 369083676, 369365916, 369648487, 369930321, 370213576, 370496095, 370778943, 294676718, 371344586, 371627382, 371910505, 372192894, 372476699, 372759769, 373043166, 297172237, 373609907, 373893250, 374176920, 374459858, 374744204, 375027818, 375311758, 299586611, 375879582, 376163466, 376447674, 376731154, 377016034, 377300186, 377584661, 302161289, 378153554, 378437971, 378722711, 379006725, 379292133, 379576815, 379861818, 304654854, 380431766, 380716710, 381001975, 381286516, 381572444, 381857649, 382143173, 307187661, 382714161, 382999625, 383285407, 383570468, 383856909, 384142630, 384428668, 0, 385000682, 385286658, 385572950, 385858524, 386145471, 386431700, 386718245, 312250744, 387291270, 387577751, 387864546, 388150626, 388438071, 388724802, 389011845, 314781061, 389585867, 389872846, 390160136, 390446715, 390734651, 391021876, 391309412, 317349911, 391884416, 392171885, 392459663, 392746733, 393035153, 393322865, 393610885, 319839350, 394186857, 394474808, 394763068, 395050622, 395339518, 395627709, 395916206, 322484520, 396493130, 396781558, 397070291, 397358321, 397647686, 397936349, 398225316, 325050328, 398803178, 399092074, 399381273, 399669772, 399959599, 400248726, 400538155, 327653914, 401116941, 401406297, 401695955, 401984916, 402275196, 402564780, 402854664, 0, 403434358, 403724168, 404014276, 404303691, 404594418, 404884452, 405174783, 332857421, 405755370, 406045625, 406336177, 406626039, 406917205, 407207680, 407498451, 335457399, 408079916, 408370610, 408661598, 408951898, 409243495, 409534405, 409825608, 338094354, 410407936, 410699060, 410990476, 411281209, 411573230, 411864566, 412156194, 340653946, 412739369, 413030916, 413322753, 413613909, 413906346, 414198102, 414490147, 343364011, 415074154, 415366116, 415658367, 415949939, 416242784, 416534952, 416827406, 345996776, 417412230, 417704600, 417997256, 418289237, 418582482, 418875054, 419167909, 348665675, 419753535, 420046305, 420339358, 420631740, 420925379, 421218346, 421511595, 0, 422098007, 422391170, 422684613, 422977388, 423271412, 423564768, 423858403, 353998616, 424445585, 424739132, 425032958, 425326118, 425620519, 425914256, 426208269, 356662731, 426796206, 427090130, 427384330, 427677869, 427972639, 428266748, 428561132, 359362090, 429149809, 429444101, 429738668, 430032576, 430327708, 430622182, 430916929, 361986437, 431506329, 431800983, 432095908, 432390179, 432685664, 432980496, 433275597, 364755339, 433865705, 434160712, 434455988, 434750613, 435046445, 435341625, 435637074, 367449310, 436227874, 436523226, 436818845, 437113816, 437409986, 437705508, 438001295, 370177591, 438592772, 438888461, 439184415, 439479725, 439776224, 440072080, 440368198, 0, 440960335, 441256354, 441552635, 441848276, 442145097, 442441278, 442737719, 375628063, 443330501, 443626842, 443923442, 444219404, 444516540, 444813038, 445109795, 378350340, 445703205, 445999859, 446296770, 446593048, 446890489, 447187297, 447484360, 381105955, 448078383, 448375343, 448672557, 448969141, 449266880, 449563990, 449861352, 383789200, 450455972, 450753229, 451050738, 451347621, 451645650, 451943053, 452240706, 386610459, 452835906, 453133453, 453431249, 453728422, 454026733, 454324421, 454622357, 389359443, 455218122, 455515950, 455814024, 456111480, 456410065, 456708030, 457006241, 392140753, 457602554, 457900655, 458199000, 458496730, 458795581, 459093815, 459392293, 0, 459989137, 460287504, 460586112, 460884108, 461183216, 461481711, 461780448, 397696002, 462377808, 462676431, 462975294, 463273548, 463572905, 463871654, 464170640, 400470043, 464768499, 465067371, 465366480, 465664985, 465964583, 466263576, 466562805, 403275360, 467161147, 467460260, 467759607, 468058353, 468358184, 468657414, 468956877, 406011217, 469555686, 469855031, 470154608, 470453588, 470753643, 471053102, 471352791, 408877971, 471952049, 472251619, 472551417, 472850623, 473150895, 473450573, 473750480, 411675374, 474350172, 474649957, 474949970, 475249392, 475549872, 475849763, 476149879, 414502966, 476749988, 477049981, 477350199, 477649830, 477950510, 478250604, 478550921, 0, 479151431, 479451624, 479752038, 480051871, 480352742, 480653032, 480953542, 420149468, 481554436, 481854820, 482155423, 482455448, 482756503, 483056979, 483357674, 422968493, 483958935, 484259502, 484560286, 484860495, 485161724, 485462380, 485763251, 425816587, 486364863, 486665605, 486966560, 487266945, 487568342, 487869168, 488170206, 428598383, 488772153, 489073061, 489374180, 489674733, 489976288, 490277276, 490578474, 431503425, 491180738, 491481804, 491783079, 492083791, 492385496, 492686638, 492987987, 434342291, 493590552, 493891768, 494193190, 494494053, 494795899, 495097187, 495398679, 437209073, 496001528, 496302885, 496604446, 496905451, 497207431, 497508856, 497810482, 0, 498413599, 498715089, 499016780, 499317920, 499620024, 499921578, 500223330, 442932615, 500826698, 501128313, 501430125, 501731391, 502033612, 502335286, 502637156, 445789503, 503240758, 503542489, 503844415, 504145798, 504448127, 504749913, 505051893, 448673123, 505655712, 505957550, 506259582, 506561074, 506863503, 507165392, 507467473, 451493836, 508071492, 508373430, 508675558, 508977151, 509279671, 509581656, 509883829, 454429665, 510488031, 510790061, 511092277, 511393963, 511696565, 511998637, 512300894, 457302721, 512905263, 513207375, 513509671, 513811441, 514114117, 514416267, 514718600, 460201297, 515323119, 515625305, 515927673, 516229518, 516532260,

Продолжение таблицы Б.3

516834480, 517136881, 0, 517741532, 518043784, 518346215, 518648127, 518950927, 519253208, 519555667, 465987069, 520160435, 520462744, 520765229, 521067200, 521370049, 521672383, 521974893, 468874406, 522579760, 522882118, 523184649, 523486670, 523789559, 524091938, 524394490, 471786025, 524999439, 525301837, 525604406, 525906469, 526209390, 526511805, 526814390, 474638329, 527419405, 527721835, 528024433, 528326529, 528629474, 528931917, 529234526, 477597199, 529839589, 530142043, 530444662, 530746782, 531049743, 531352205, 531654831, 480496900, 532259925, 532562394, 532865025, 533167162, 533470129, 533772602, 534075236, 483419622, 534680345, 534982820, 535285455, 535587600, 535890565, 536193040, 536495674, 0, 537100781, 537403254, 537705884, 538008028, 538310983, 538613452, 538916077, 489252317, 539521164, 539823627, 540126244, 540428379, 540731315, 541033770, 541336377, 492162442, 541941428, 542243872, 542546467, 542848585, 543151494, 543453925, 543756507, 495094303, 544361505, 544663921, 544966486, 545268578, 545571451, 545873851, 546176398, 497970621, 546781326, 547083706, 547386233, 547688291, 547991119, 548293479, 548595983, 500944592, 549200824, 549503160, 549805640, 550107655, 550410431, 550712742, 551015195, 503863176, 551619931, 551922215, 552224639, 552526603, 552829317, 553131571, 553433965, 506802193, 554038580, 554340802, 554643163, 554945066, 555247711, 555549900, 555852225, 0, 556456703, 556758855, 557061143, 557362979, 557665545, 557967660, 558269908, 512666111, 558874231, 559176306, 559478513, 559780271, 560082752, 560384784, 560686947, 515591171, 561291098, 561593086, 561895204, 562196877, 562499262, 562801204, 563103274, 518535344, 563707236, 564009129, 564311148, 564612728, 564915010, 565216852, 565518820, 521427896, 566122577, 566424366, 566726279, 567027757, 567329927, 567631662, 567933519, 524408891, 568537053, 568838730, 569140529, 569441897, 569743946, 570045564, 570347302, 527338430, 570950597, 571252154, 571553830, 571855078, 572156999, 572458493, 572760103, 530285745, 573363142, 573664571, 573966115, 574267236, 574569019, 574870380, 575171855, 0, 575774620, 576075911, 576377315, 576678301, 576979938, 577281158, 577582488, 536164898, 578184964, 578486110, 578787365, 579088207, 579389690, 579690759, 579991938, 539096905, 580594107, 580895098, 581196197, 581496886, 581798207, 582099118, 582400135, 542045340, 583001981, 583302810, 583603743, 583904271, 584205421, 584506166, 584807013, 544946197, 585408519, 585709177, 586009937, 586310295, 586611266, 586911836, 587212506, 547926059, 587813654, 588114133, 588414711, 588714892, 589015675, 589316061, 589616545, 550858515, 590217320, 590517611, 590817999, 591117993, 591418580, 591718775, 592019064, 553806042, 592619449, 592919544, 593219733, 593519533, 593819916, 594119910, 594419997, 0, 595019974, 595319865, 595619847, 595919444, 596219615, 596519400, 596819276, 559684274, 597418829, 597718508, 598018274, 598317661, 598617610, 598917179, 599216835, 562615156, 599815948, 600115405, 600414948, 600714115, 601013835, 601313179, 601612607, 565559743, 602211263, 602510492, 602809803, 603108742, 603408224, 603707335, 604006527, 568460885, 604604709, 604903700, 605202771, 605501474, 605800711, 606099580, 606398527, 571431434, 606996220, 607294965, 607593787, 607892246, 608191228, 608489847, 608788542, 574358717, 609385728, 609684219, 609982785, 610280991, 610579711, 610878072, 611176506, 577298335, 611773169, 612071398, 612369698, 612667644, 612966093, 613264188, 613562352, 0, 614158476, 614456434, 614754461, 615052138, 615350308, 615648129, 615946016, 583159435, 616541583, 616839263, 617137009, 617434409, 617732292, 618029829, 618327431, 586081098, 618922426, 619219819, 619517275, 619814389, 620111977, 620409224, 620706532, 589013723, 621300938, 621598036, 621895194, 622192015, 622489300, 622786247, 623083254, 591907097, 623677055, 623973849, 624270702, 624567221, 624864194, 625160834, 625457531, 594860191, 626050710, 626347194, 626643732, 626939942, 627236595, 627532920, 627829298, 597774217, 628421841, 628718005, 629014221, 629310113, 629606438, 629902439, 630198492, 600697834, 630790380, 631086217, 631382103, 631677669, 631973658, 632269328, 632565046, 0, 633156265, 633451765, 633747313, 634042545, 634338191, 634633521, 634928897, 606525650, 635519429, 635814587, 636109788, 636404678, 636699972, 636994954, 637289980, 609430035, 637879810, 638174616, 638469463, 638764003, 639058937, 639353564, 639648231, 612342640, 640237343, 640531789, 640826274, 641120456, 641415022, 641709285, 642003586, 615220218, 642591965, 642886043, 643180157, 643473973, 643768163, 644062055, 644355981, 618147810, 644943610, 645237313, 645531049, 645824491, 646118297, 646411809, 646705353, 621040562, 647292217, 647585536, 647878886, 648171945, 648465360, 648758485, 649051639, 623940169, 649637721, 649930649, 650223604, 650516274, 650809289, 651102019, 651394775, 0, 651980059, 652272588, 652565142, 652857414, 653150022, 653442348, 653734698, 629718732, 654319169, 654611291, 654903435, 655195301, 655487494, 655779409, 656071345, 632597876, 656654988, 656946695, 657238421, 657529874, 657821644, 658113140, 658404655, 635482518, 658987454, 659278737, 659570039, 659861071, 660152409, 660443479, 660734564, 638336356, 661316503, 661607357, 661898224, 662188827, 662479728, 662770363, 663061011, 641230553, 663642075, 663932490, 664222917, 664513083, 664803537, 665093730, 665383934, 644094136, 665964107, 666254076, 666544055, 666833776, 667123776, 667413520, 667703271, 646961870, 668282538, 668572053, 668861576, 669150845, 669440383, 669729669, 670018961, 0, 670597306, 670886360, 671175419, 671464229, 671753298, 672042118, 672330942, 652675506, 672908351, 673196936, 673485523, 673773866, 674062458, 674350806, 674639154, 655521599, 675215611, 675503720, 675791828, 676079696, 676367803, 676655671, 676943536, 658370507, 677519026,

Продолжение таблицы Б.3

677806651, 678094272, 678381658, 678669273, 678956653, 679244028, 661192803, 679818536, 680105670, 680392797, 680679692, 680966808, 681253693, 681540570, 664045926, 682114081, 682400715, 682687341, 682973739, 683260348, 683546730, 683833101, 666872628, 684405601, 684691729, 684977845, 685263738, 685549833, 685835704, 686121563, 669700825, 686693036, 686978650, 687264249, 687549630, 687835203, 688120557, 688405896, 0, 688976327, 689261419, 689546495, 689831356, 690116400, 690401229, 690686040, 675334273, 691255415, 691539979, 691824523, 692108857, 692393365, 692677661, 692961938, 678139715, 693530242, 693814269, 694098275, 694382075, 694666039, 694949796, 695233530, 680945348, 695800748, 696084233, 696367693, 696650950, 696934363, 697217574, 697500758, 683728501, 698066877, 698349811, 698632718, 698915427, 699198281, 699480937, 699763565, 686533139, 700328570, 700610946, 700893293, 701175445, 701457735, 701739829, 702021894, 689315486, 702585769, 702867580, 703149360, 703430949, 703712666, 703994192, 704275685, 692096741, 704838418, 705119657, 705400862, 705681881, 705963018, 706243969, 706524884, 0, 707086459, 707367119, 707647743, 707928183, 708208734, 708489102, 708769432, 697635263, 709329836, 709609910, 709889945, 710169801, 710449758, 710729536, 711009274, 700392720, 711568493, 711847974, 712127413, 712406677, 712686033, 712965215, 713244354, 703147823, 713802373, 714081254, 714360090, 714638756, 714917505, 715196083, 715474615, 705884487, 716031421, 716309695, 716587922, 716865982, 717144116, 717422084, 717700003, 708633549, 718255582, 718533242, 718810852, 719088300, 719365813, 719643163, 719920463, 711364360, 720474801, 720751840, 721028827, 721305655, 721582540, 721859266, 722135939, 714091580, 722689022, 722965434, 723241791, 723517993, 723794242, 724070337, 724346377, 0, 724898193, 725173969, 725449690, 725725259, 726000867, 726276324, 726551723, 719521070, 727102258, 727377393, 727652470, 727927399, 728202359, 728477171, 728751924, 722223526, 729301163, 729575651, 729850078, 730124361, 730398665, 730672826, 730946925, 724921180, 731494857, 731768690, 732042460, 732316090, 732589733, 732863236, 733136675, 727604321, 733683285, 733956457, 734229564, 734502534, 734775509, 735048348, 735321120, 730291088, 735866396, 736138900, 736411337, 736683641, 736955941, 737228109, 737500208, 732963528, 738044136, 738315967, 738587726, 738859358, 739130977, 739402467, 739673887, 735629982, 740216455, 740487605, 740758681, 741029634, 741300565, 741571372, 741842105, 0, 742383300, 742653763, 742924150, 743194417, 743464654, 743734771, 744004811, 740937064, 744544621, 744814390, 745084082, 745353657, 745623193, 745892613, 746161955, 743577876, 746700366, 746969435, 747238425, 747507302, 747776132, 748044849, 748313485, 746211548, 748850485, 749118849, 749387130, 749655303, 749923419, 750191428, 750459353, 748834491, 750994929, 751262580, 751530147, 751797609, 752065007, 752332299, 752599507, 751452665, 753133647, 753400580, 753667426, 753934171, 754200843, 754467415, 754733899, 754060292, 755266590, 755532799, 755798918, 756064940, 756330881, 756596725, 756862479, 756659658, 757393710, 757659188, 757924575, 758189867, 758455071, 758720181, 758985199 };
INT32 KBDWindow cs[SINE DATA SIZEI2] = {
0, 78534, 111064, 112139, 157068, 138663, 160123, 161643, 11756, 182438, 199873, 201735, 230847, 219940, 235165, 237314, 16626, 254038, 267838, 270241, 293529, 286021, 298818, 301452, 19750, 316593, 328646, 331492, 351555, 346187, 357667, 360711, 23513, 375091, 386118, 389349, 407325, 403506, 414168, 417577, 28221, 431578, 441942, 445521, 462028, 459419, 469538, 473280, 33003, 487115, 497030, 500931, 516349, 514735, 524481, 528534, 37517, 542334, 551939, 556141, 570727, 569958, 579445, 583792, 0, 597645, 607034, 611522, 625457, 625427, 634735, 639362, 47793, 653330, 662573, 667335, 680753, 681380, 690569, 695466, 53315, 709596, 718744, 723772, 736776, 737996, 747113, 752271, 59156, 766597, 775692, 780978, 793649, 795413, 804494, 809906, 63713, 824457, 833531, 839068, 851473, 853739, 862814, 868474, 71697, 883272, 892352, 898135, 910327, 913064, 922155, 928060, 78259, 943124, 952232, 958257, 970281, 973460, 982589, 988734, 85502, 1004080, 1013235, 1019499, 1031393, 1034991, 1044176, 1050558, 0, 1066200, 1075418, 1081917, 1093715, 1097712, 1106967, 1113584, 100655, 1129534, 1138829, 1145563, 1157291, 1161671, 1171009, 1177859, 108472, 1194128, 1203512, 1210478, 1222163, 1226911, 1236344, 1243426, 117240, 1260024, 1269508, 1276705, 1288368, 1293471, 1303009, 1310322, 123865, 1327257, 1336851, 1344279, 1355939, 1361387, 1371039, 1378582, 135342, 1395864, 1405577, 1413235, 1424909, 1430693, 1440468, 1448240, 144606, 1465877, 1475716, 1483603, 1495307, 1501419, 1511324, 1519326, 155049, 1537324, 1547297, 1555414, 1567163, 1573595, 1583638, 1591869, 0, 1610236, 1620349, 1628695, 1640502, 1647249, 1657435, 1665896, 176451, 1684638, 1694899, 1703475, 1715351, 1722407, 1732744, 1741435, 187351, 1760558, 1770972, 1779778, 1791734, 1799095, 1809587, 1818509, 199640, 1838020, 1848593, 1857630, 1869676, 1877338, 1887991, 1897144, 209097, 1917050, 1927786, 1937055, 1949200, 1957159, 1967979, 1977364, 224710, 1997669, 2008574, 2018075, 2030330, 2038583, 2049574, 2059192, 237439, 2079902, 2090981, 2100715, 2113086, 2121631, 2132798, 2142650, 251759, 2163772, 2175028, 2184997, 2197493, 2206327, 2217674, 2227761, 0, 2249299, 2260738, 2270943, 2283570, 2292692, 2304223, 2314546, 280888, 2336507, 2348132, 2358574, 2371339, 2380747, 2392467, 2403028, 295649, 2425416, 2437231, 2447911, 2460822, 2470515, 2482427, 2493227, 312200, 2516047, 2528057, 2538977, 2552039, 2562016, 2574124, 2585164,

Продолжение таблицы Б.3

325243, 2608422, 2620630, 2631791, 2645010, 2655270, 2667578, 2678860, 345800, 2702562, 2714971, 2726375, 2739757, 2750299, 2762811, 2774336, 362803, 2798486, 2811100, 2822748, 2836299, 2847124, 2859842, 2871612, 381798, 2896215, 2909038, 2920932, 2934656, 2945763, 2958692, 2970709, 0, 2995770, 3008805, 3020946, 3034849, 3046238, 3059380, 3071645, 420306, 3097169, 3110420, 3122810, 3136896, 3148568, 3161928, 3174442, 439774, 3200435, 3213904, 3226545, 3240819, 3252773, 3266353, 3279120, 461439, 3305585, 3319277, 3332170, 3346637, 3358874, 3372677, 3385697, 478887, 3412641, 3426557, 3439704, 3454369, 3466889, 3480919, 3494194, 505315, 3521621, 3535765, 3549169, 3564034, 3576839, 3591098, 3604631, 527481, 3632545, 3646921, 3660582, 3675653, 3688743, 3703235, 3717026, 552056, 3745433, 3760043, 3773965, 3789245, 3802620, 3817348, 3831400, 0, 3860305, 3875152, 3889335, 3904829, 3918491, 3933457, 3947772, 601784, 3977179, 3992266, 4006713, 4022425, 4036374, 4051582, 4066161, 626892, 4096076, 4111406, 4126119, 4142051, 4156289, 4171741, 4186587, 654629, 4217015, 4232590, 4247570, 4263728, 4278255, 4293955, 4309070, 677384, 4340014, 4355838, 4371088, 4387474, 4402292, 4418242, 4433627, 710720, 4465093, 4481169, 4496690, 4513309, 4528419, 4544621, 4560280, 739027, 4592272, 4608602, 4624397, 4641251, 4656655, 4673113, 4689046, 770191, 4721570, 4738156, 4754228, 4771320, 4787019, 4803735, 4819945, 0, 4853006, 4869852, 4886201, 4903536, 4919531, 4936508, 4952997, 833178, 4986598, 5003707, 5020336, 5037917, 5054210, 5071450, 5088220, 864954, 5122368, 5139741, 5156652, 5174482, 5191074, 5208581, 5225634, 899822, 5260332, 5277973, 5295168, 5313251, 5330144, 5347920, 5365258, 928880, 5400511, 5418423, 5435904, 5454242, 5471437, 5489485, 5507110, 970266, 5542924, 5561109, 5578878, 5597474, 5614974, 5633296, 5651211, 1005792, 5687589, 5706050, 5724110, 5742968, 5760773, 5779372, 5797578, 1044655, 5834526, 5853265, 5871618, 5890740, 5908853, 5927732, 5946231, 0, 5983754, 6002774, 6021421, 6040812, 6059233, 6078394, 6097190, 1123140, 6135292, 6154595, 6173539, 6193201, 6211933, 6231379, 6250472, 1162710, 6289158, 6308748, 6327991, 6347926, 6366971, 6386704, 6406098, 1205872, 6445372, 6465250, 6484795, 6505007, 6524366, 6544389, 6564085, 1242325, 6603953, 6624123, 6643970, 6664462, 6684137, 6704453, 6724453, 1293007, 6764920, 6785383, 6805536, 6826310, 6846303, 6866914, 6887221, 1336927, 6928291, 6949050, 6969511, 6990571, 7010884, 7031792, 7052408, 1384704, 7094085, 7115143, 7135914, 7157262, 7177897, 7199105, 7220032, 0, 7262321, 7283681, 7304764, 7326403, 7347361, 7368872, 7390113, 1481124, 7433019, 7454682, 7476080, 7498013, 7519296, 7541113, 7562668, 1529714, 7606196, 7628166, 7649880, 7672110, 7693721, 7715845, 7737718, 1582434, 7781872, 7804151, 7826184, 7848713, 7870653, 7893087, 7915280, 1627474, 7960066, 7982656, 8005010, 8027841, 8050112, 8072859, 8095374, 1688799, 8140795, 8163699, 8186376, 8209512, 8232117, 8255178, 8278018, 1742389, 8324079, 8347300, 8370302, 8393746, 8416685, 8440065, 8463230, 1800393, 8509937, 8533476, 8556806, 8580560, 8603837, 8627536, 8651031, 0, 8698387, 8722247, 8745907, 8769974, 8793590, 8817611, 8841437, 1917389, 8889448, 8913631, 8937623, 8962006, 8985963, 9010309, 9034468, 1976321, 9083137, 9107647, 9131973, 9156674, 9180974, 9205648, 9230141, 2039963, 9279475, 9304313, 9328975, 9353998, 9378643, 9403646, 9428477, 2094881, 9478479, 9503648, 9528649, 9553994, 9578987, 9604322, 9629493, 2168297, 9680168, 9705670, 9731012, 9756683, 9782025, 9807694, 9833207, 2232929, 9884560, 9910398, 9936082, 9962083, 9987775, 10013782, 10039639, 2302574, 10091673, 10117849, 10143879, 10170211, 10196257, 10222602, 10248806, 0, 10301527, 10328043, 10354421, 10381086, 10407487, 10434174, 10460726, 2442978, 10514139, 10540998, 10567725, 10594726, 10621485, 10648516, 10675419, 2513677, 10729527, 10756731, 10783810, 10811151, 10838268, 10865646, 10892901, 2589701, 10947710, 10975262, 11002695, 11030376, 11057856, 11085582, 11113193, 2655885, 11168706, 11196608, 11224397, 11252422, 11280265, 11308342, 11336310, 2742933, 11392533, 11420787, 11448935, 11477306, 11505514, 11533945, 11562273, 2820076, 11619210, 11647818, 11676326, 11705047, 11733622, 11762408, 11791098, 2902869, 11848753, 11877718, 11906589, 11935661, 11964606, 11993750, 12022803, 0, 12081182, 12110506, 12139742, 12169167, 12198484, 12227989, 12257408, 3069707, 12316514, 12346200, 12375803, 12405584, 12435274, 12465141, 12494929, 3153686, 12554767, 12584816, 12614789, 12644928, 12674994, 12705226, 12735384, 3243648, 12795959, 12826374, 12856718, 12887219, 12917662, 12948261, 12978792, 3322580, 13040107, 13070891, 13101608, 13132472, 13163296, 13194264, 13225170, 3424893, 13287230, 13318384, 13349478, 13380707, 13411912, 13443253, 13474535, 3516109, 13537345, 13568872, 13600343, 13631941, 13663530, 13695244, 13726905, 3613649, 13790470, 13822372, 13854223, 13886191, 13918166, 13950256, 13982299, 0, 14046621, 14078901, 14111134, 14143475, 14175838, 14208307, 14240732, 3810124, 14305818, 14338477, 14371095, 14403810, 14436563, 14469413, 14502224, 3908989, 14568076, 14601117, 14634122, 14667214, 14700360, 14733592, 14766790, 4014529, 14833415, 14866839, 14900232, 14933705, 14967244, 15000861, 15034449, 4107783, 15101849, 15135660, 15169444, 15203298, 15237234, 15271238, 15305218, 4227077, 15373398, 15407597, 15441774, 15476012, 15510346, 15544740, 15579114, 4334013, 15648078, 15682668, 15717240, 15751864, 15786598, 15821384, 15856154, 4447984, 15925907, 15960889, 15995858, 16030871, 16066007, 16101186, 16136354, 0, 16206900, 16242278, 16277645, 16313050, 16348589, 16384165, 16419733, 4677468, 16491076, 16526851, 16562619, 16598417, 16634363, 16670336, 16706307, 4792908, 16778451, 16814625, 16850797, 16886990, 16923344, 16959718, 16996092, 4915750, 17069042, 17105617, 17142194, 17178785, 17215549, 17252325, 17289106, 5024986, 17362866, 17399844, 17436829, 17473819, 17510995, 17548176, 17585365, 5163052, 17659939, 17697322, 17734717, 17772109, 17809699, 17847286, 17884886, 5287436, 17960277, 17998069, 18035874, 18073671,

Продолжение таблицы Б.3

18111677, 18149673, 18187685, 5419600, 18263899, 18302100, 18340319, 18378521, 18416945, 18455352, 18493778, 0, 18570819, 18609432, 18648066, 18686677, 18725521, 18764340, 18803183, 5685619, 18881054, 18920080, 18959132, 18998154, 19037419, 19076654, 19115915, 5819399, 19194620, 19234063, 19273534, 19312968, 19352658, 19392309, 19431991, 5961340, 19511534, 19551395, 19591287, 19631136, 19671251, 19711322, 19751426, 6088294, 19831812, 19872092, 19912408, 19952674, 19993217, 20033708, 20074237, 6246994, 20155469, 20196172, 20236913, 20277597, 20318570, 20359484, 20400439, 6390626, 20482521, 20523648, 20564817, 20605922, 20647326, 20688666, 20730049, 6542812, 20812986, 20854538, 20896136, 20937665, 20979502, 21021268, 21063082, 0, 21146877, 21188858, 21230887, 21272840, 21315112, 21357308, 21399553, 6849031, 21484210, 21526621, 21569084, 21611464, 21654173, 21696800, 21739479, 7002983, 21825002, 21867845, 21910742, 21953552, 21996700, 22039760, 22082875, 7165886, 22169268, 22212545, 22255879, 22299120, 22342708, 22386203, 22429756, 7312364, 22517023, 22560735, 22604508, 22648182, 22692213, 22736144, 22780138, 7493615, 22868282, 22912432, 22956646, 23000754, 23045230, 23089600, 23134035, 7658359, 23223060, 23267650, 23312306, 23356852, 23401774, 23446584, 23491462, 7832458, 23581373, 23626405, 23671505, 23716490, 23761859, 23807113, 23852436, 0, 23943235, 23988710, 24034257, 24079683, 24125502, 24171200, 24216970, 8182656, 24308662, 24354582, 24400577, 24446445, 24492716, 24538860, 24585080, 8358671, 24677667, 24724035, 24770480, 24816793, 24863517, 24910109, 24956779, 8544451, 25050267, 25097084, 25143980, 25190739, 25237919, 25284961, 25332084, 8712322, 25426475, 25473742, 25521092, 25568300, 25615936, 25663430, 25711007, 8918087, 25806305, 25854025, 25901830, 25949488, 25997583, 26045530, 26093564, 9105859, 26189772, 26237947, 26286209, 26334319, 26382874, 26431277, 26479768, 9303809, 26576891, 26625522, 26674243, 26722807, 26771824, 26820684, 26869635, 0, 26967675, 27016764, 27065946, 27114965, 27164446, 27213765, 27263177, 9701863, 27362139, 27411688, 27461331, 27510808, 27560755, 27610534, 27660409, 9901880, 27760296, 27810306, 27860413, 27910350, 27960763, 28011005, 28061345, 10112496, 28162159, 28212633, 28263206, 28313604, 28364486, 28415192, 28465998, 10303676, 28567744, 28618683, 28669723, 28720584, 28771936, 28823108, 28874383, 10535955, 28977063, 29028469, 29079978, 29131303, 29183127, 29234767, 29286511, 10748711, 29390130, 29442004, 29493984, 29545775, 29598073, 29650182, 29702398, 10972488, 29806958, 29859302, 29911754, 29964014, 30016787, 30069366, 30122055, 0, 30227561, 30280376, 30333303, 30386031, 30439281, 30492333, 30545497, 11422346, 30651951, 30705240, 30758641, 30811842, 30865570, 30919096, 30972736, 11648338, 31080141, 31133905, 31187784, 31241457, 31295665, 31349667, 31403785, 11885777, 31512145, 31566385, 31620743, 31674891, 31729580, 31784060, 31838657, 12102224, 31947974, 32002693, 32057531, 32112155, 32167328, 32222286, 32277365, 12363033, 32387643, 32442841, 32498161, 32553263, 32608920, 32664360, 32719921, 12602759, 32831162, 32886842, 32942645, 32998227, 33054370, 33110292, 33166337, 12854362, 33278546, 33334709, 33390997, 33447059, 33503690, 33560095, 33616626, 0, 33729805, 33786452, 33843226, 33899772, 33956891, 34013782, 34070800, 13360017, 34184952, 34242085, 34299347, 34356377, 34413987, 34471364, 34528871, 13613979, 34643999, 34701620, 34759371, 34816887, 34874989, 34932854, 34990851, 13880248, 35106958, 35165068, 35223310, 35281313, 35339908, 35398263, 35456751, 14123942, 35573841, 35632441, 35691176, 35749667, 35808757, 35867603, 35926585, 14415306, 36044659, 36103751, 36162979, 36221961, 36281547, 36340886, 36400362, 14684004, 36519423, 36579009, 36638733, 36698206, 36758289, 36818122, 36878094, 14965442, 36998146, 37058227, 37118447, 37178413, 37238996, 37299324, 37359793, 0, 37480839, 37541416, 37602134, 37662594, 37723677, 37784503, 37845470, 15530906, 37967512, 38028586, 38089804, 38150760, 38212345, 38273668, 38335136, 15814840, 38458177, 38519750, 38581468, 38642922, 38705009, 38766832, 38828801, 16111948, 38952844, 39014917, 39077137, 39139090, 39201682, 39264006, 39326477, 16384883, 39451524, 39514099, 39576822, 39639275, 39702372, 39765199, 39828174, 16708818, 39954228, 40017306, 40080534, 40143488, 40207092, 40270423, 40333903, 17008488, 40460966, 40524549, 40588283, 40651739, 40715852, 40779687, 40843674, 17321765, 40971749, 41035837, 41100078, 41164039, 41228661, 41293002, 41357497, 0, 41486587, 41551182, 41615931, 41680398, 41745530, 41810378, 41875382, 17951039, 42005490, 42070593, 42135852, 42200825, 42266468, 42331826, 42397340, 18266942, 42528467, 42594079, 42659849, 42725330, 42791487, 42857354, 42923380, 18596885, 43055529, 43121652, 43187934, 43253924, 43320595, 43386974, 43453512, 18901054, 43586685, 43653320, 43720115, 43786616, 43853802, 43920693, 43987745, 19259552, 44121945, 44189093, 44256403, 44323415, 44391117, 44458522, 44526089, 19592183, 44661319, 44728981, 44796806, 44864331, 44932550, 45000470, 45068553, 19939284, 45204814, 45272992, 45341333, 45409372, 45478111, 45546546, 45615147, 0, 45752442, 45821135, 45889995, 45958549, 46027807, 46096760, 46165879, 20636326, 46304209, 46373421, 46442799, 46511870, 46581649, 46651120, 46720758, 20986176, 46860127, 46929857, 46999755, 47069343, 47139644, 47209634, 47279793, 21350922, 47420202, 47490452, 47560872, 47630978, 47701802, 47772312, 47842993, 21688300, 47984444, 48055215, 48126157, 48196783, 48268130, 48339162, 48410365, 22083313, 48552862, 48624154, 48695619, 48766766, 48838638, 48910193, 48981919, 22450866, 49125462, 49197278, 49269267, 49340935, 49413334, 49485411, 49557663, 22833740, 49702254, 49774594, 49847108, 49919299, 49992225, 50064827, 50137604, 0, 50283246, 50356111, 50429151, 50501866, 50575320, 50648447, 50721751, 23602441, 50868446, 50941836, 51015404, 51088643, 51162626, 51236279, 51310111, 23988176, 51457860, 51531778, 51605873, 51679639, 51754150, 51828332,

Продолжение таблицы Б.3

51902692, 24389653, 52051497, 52125943, 52200568, 52274860, 52349902, 52424612, 52499501, 24762184, 52649365, 52724339, 52799494, 52874314, 52949888, 53025126, 53100546, 25195611, 53251470, 53326974, 53402660, 53478008, 53554114, 53629883, 53705834, 25600001, 53857820, 53933855, 54010072, 54085950, 54162589, 54238889, 54315372, 26020549, 54468422, 54544988, 54621737, 54698146, 54775319, 54852151, 54929168, 0, 55083282, 55160381, 55237664, 55314603, 55392312, 55469676, 55547226, 26864697, 55702408, 55780040, 55857857, 55935328, 56013573, 56091471, 56169556, 27288205, 56325806, 56403971, 56482324, 56560328, 56639109, 56717542, 56796162, 27728283, 56953482, 57032183, 57111071, 57189609, 57268927, 57347895, 57427051, 28137866, 57585444, 57664679, 57744104, 57823177, 57903033, 57982537, 58062230, 28611535, 58221696, 58301468, 58381430, 58461038, 58541433, 58621474, 58701705, 29054619, 58862245, 58942555, 59023055, 59103199, 59184133, 59264711, 59345481, 29514676, 59507098, 59587945, 59668984, 59749665, 59831139, 59912256, 59993564, 0, 60156259, 60237644, 60319223, 60400442, 60482457, 60564112, 60645961, 30437928, 60809734, 60891659, 60973778, 61055535, 61138092, 61220286, 61302675, 30901033, 61467530, 61549994, 61632654, 61714950, 61798049, 61880784, 61963714, 31381510, 62129650, 62212656, 62295857, 62378693, 62462334, 62545610, 62629082, 31829981, 62796101, 62879648, 62963391, 63046768, 63130952, 63214770, 63298784, 32345637, 63466888, 63550976, 63635262, 63719180, 63803908, 63888268, 63972825, 32829199, 64142014, 64226646, 64311475, 64395934, 64481207, 64566110, 64651211, 33330522, 64821486, 64906661, 64992034, 65077035, 65162853, 65248299, 65333945, 0, 65505308, 65591026, 65676944, 65762488, 65848852, 65934842, 66021032, 34336377, 66193484, 66279747, 66366209, 66452297, 66539207, 66625741, 66712477, 34840822, 66886019, 66972826, 67059835, 67146466, 67233923, 67321002, 67408283, 35363408, 67582917, 67670269, 67757824, 67845000, 67933003, 68020629, 68108456, 35852532, 68284182, 68372080, 68460181, 68547902, 68636453, 68724624, 68812999, 36411818, 68989818, 69078262, 69166910, 69255177, 69344276, 69432993, 69521915, 36937558, 69699828, 69788819, 69878015, 69966827, 70056475, 70145740, 70235209, 37481809, 70414217, 70503755, 70593498, 70682858, 70773054, 70862866, 70952884, 0, 71132987, 71223074, 71313365, 71403272, 71494017, 71584377, 71674943, 38573579, 71856144, 71946778, 72037618, 72128072, 72219367, 72310275, 72401390, 39121012, 72583688, 72674871, 72766260, 72857262, 72949107, 73040564, 73132228, 39687320, 73315624, 73407356, 73499295, 73590845, 73683240, 73775247, 73867461, 40218772, 74051955, 74144236, 74236725, 74328823, 74421770, 74514325, 74607089, 40823223, 74792684, 74885514, 74978553, 75071201, 75164698, 75257803, 75351118, 41392738, 75537812, 75631193, 75724782, 75817979, 75912028, 76005683, 76099548, 41981476, 76287344, 76381274, 76475415, 76569161, 76663761, 76757967, 76852383, 0, 77041280, 77135762, 77230453, 77324749, 77419901, 77514658, 77609625, 43162260, 77799625, 77894657, 77989900, 78084746, 78180450, 78275758, 78371276, 43754227, 78562378, 78657961, 78753756, 78849153, 78945409, 79041268, 79137339, 44365756, 79329544, 79425678, 79522025, 79617972, 79714781, 79811192, 79907814, 44941111, 80101123, 80197809, 80294707, 80391206, 80488568, 80585530, 80682705, 45592138, 80877117, 80974355, 81071806, 81168856, 81266771, 81364284, 81462011, 46206914, 81657528, 81755318, 81853322, 81950923, 82049391, 82147457, 82245736, 46841579, 82442358, 82540700, 82639256, 82737409, 82836430, 82935048, 83033881, 0, 83231608, 83330502, 83429611, 83528316, 83627890, 83727061, 83826446, 48114245, 84025278, 84124725, 84224387, 84323644, 84423772, 84523495, 84623433, 48752170, 84823371, 84923371, 85023585, 85123394, 85224076, 85324352, 85424843, 49410299, 85625887, 85726439, 85827207, 85927569, 86028804, 86129632, 86230677, 50031020, 86432827, 86533932, 86635253, 86736167, 86837956, 86939338, 87040936, 50729901, 87244192, 87345849, 87447724, 87549190, 87651533, 87753468, 87855619, 51391301, 88059981, 88162192, 88264620, 88366639, 88469536, 88572023, 88674728, 52073206, 88880197, 88982961, 89085942, 89188514, 89291964, 89395005, 89498263, 0, 89704838, 89808155, 89911690, 90014814, 90118819, 90222413, 90326224, 53440365, 90533906, 90637776, 90741865, 90845541, 90950100, 91054247, 91158612, 54125548, 91367400, 91471823, 91576465, 91680695, 91785807, 91890507, 91995426, 54831523, 92205321, 92310297, 92415492, 92520274, 92625941, 92731193, 92836665, 55498952, 93047667, 93153196, 93258945, 93364280, 93470500, 93576306, 93682331, 56246823, 93894439, 94000522, 94106824, 94212711, 94319485, 94425844, 94532423, 56956081, 94745637, 94852272, 94959128, 95065568, 95172895, 95279807, 95386939, 57686403, 95601259, 95708447, 95815856, 95922849, 96030729, 96138194, 96245879, 0, 96461306, 96569047, 96677008, 96784554, 96892988, 97001005, 97109243, 59150394, 97325775, 97434069, 97542584, 97650682, 97759669, 97868239, 97977030, 59884000, 98194668, 98303514, 98412582, 98521232, 98630772, 98739895, 98849238, 60638927, 99067981, 99177380, 99287000, 99396203, 99506296, 99615971, 99725867, 61354276, 99945715, 100055666, 100165839, 100275593, 100386239, 100496467, 100606916, 62152124, 100827868, 100938371, 101049097, 101159403, 101270601, 101381381, 101492382, 62910334, 101714439, 101825494, 101936771, 102047629, 102159380, 102270711, 102382264, 63690105, 102605425, 102717032, 102828861, 102940271, 103052574, 103164457, 103276562, 0, 103500826, 103612985, 103725366, 103837327, 103950181, 104062616, 104175273, 65252985, 104400640, 104513350, 104626282, 104738794, 104852201, 104965186, 105078395, 66036035, 105304864, 105418125, 105531609, 105644672, 105758630, 105872166, 105985926, 66840873, 106213497, 106327310, 106441344, 106554958, 106669467, 106783554, 106897865, 67605219, 107126538, 107240900, 107355486, 107469650, 107584710, 107699347, 107814208, 68453875, 108043982, 108158895, 108274032, 108388746, 108504356, 108619544,

Продолжение таблицы Б.3

108734955, 69261986, 108965829, 109081292, 109196979, 109312243, 109428403, 109544141, 109660102, 70092092, 109892076, 110008089, 110124325, 110240139, 110356849, 110473137, 110589648, 0, 110822721, 110939283, 111056069, 111172432, 111289691, 111406528, 111523588, 71755620, 111757760, 111874871, 111992206, 112109118, 112226926, 112344312, 112461921, 72588991, 112697191, 112814851, 112932734, 113050195, 113168552, 113286486, 113404644, 73444549, 113641011, 113759219, 113877651, 113995660, 114114566, 114233048, 114351754, 74258826, 114589217, 114707973, 114826953, 114945510, 115064964, 115183994, 115303248, 75158966, 115541806, 115661110, 115780638, 115899742, 116019743, 116139321, 116259122, 76017779, 116498775, 116618626, 116738701, 116858353, 116978901, 117099026, 117219374, 76898953, 117460121, 117580519, 117701141, 117821339, 117942434, 118063105, 118184000, 0, 118425840, 118546784, 118667952, 118788697, 118910338, 119031556, 119152997, 78664586, 119395929, 119517419, 119639133, 119760424, 119882610, 120004373, 120126360, 79549003, 120370383, 120492419, 120614679, 120736515, 120859247, 120981555, 121104088, 80455938, 121349201, 121471781, 121594586, 121716967, 121840244, 121963097, 122086174, 81320934, 122332377, 122455502, 122578851, 122701777, 122825598, 122948995, 123072617, 82273076, 123319907, 123443577, 123567470, 123690940, 123815304, 123939246, 124063411, 83183240, 124311789, 124436002, 124560438, 124684452, 124809360, 124933844, 125058553, 84116061, 125308017, 125432773, 125557753, 125682309, 125807760, 125932787, 126058038, 0, 126308588, 126433886, 126559408, 126684508, 126810500, 126936070, 127061863, 85984953, 127313496, 127439337, 127565401, 127691042, 127817576, 127943688, 128070022, 86920989, 128322739, 128449121, 128575727, 128701909, 128828984, 128955637, 129082513, 87879803, 129336311, 129463234, 129590380, 129717104, 129844719, 129971912, 130099329, 88796162, 130354208, 130481671, 130609358, 130736622, 130864777, 130992510, 131120466, 89800661, 131376425, 131504428, 131632654, 131760458, 131889152, 132017425, 132145920, 90762675, 132402958, 132531500, 132660265, 132788608, 132917841, 133046652, 133175686, 91747570, 133433801, 133562881, 133692185, 133821067, 133950837, 134080187, 134209759, 0, 134468949, 134598568, 134728409, 134857829, 134988137, 135118024, 135248134, 93720568, 135508399, 135638554, 135768932, 135898890, 136029734, 136160158, 136290805, 94708647, 136552143, 136682836, 136813750, 136944245, 137075625, 137206585, 137337768, 95719690, 137600179, 137731407, 137862857, 137993887, 138125803, 138257299, 138389017, 96687906, 138652499, 138784262, 138916247, 139047813, 139180264, 139312295, 139444547, 97744965, 139709098, 139841396, 139973916, 140106017, 140239001, 140371566, 140504353, 98759177, 140769972, 140902804, 141035858, 141168493, 141302010, 141435109, 141568429, 99796418, 141835115, 141968480, 142102067, 142235235, 142369285, 142502916, 142636769, 0, 142904520, 143038418, 143172537, 143306238, 143440820, 143574984, 143709369, 101874070, 143978183, 144112613, 144247263, 144381497, 144516609, 144651304, 144786221, 102914465, 145056097, 145191058, 145326240, 145461004, 145596647, 145731873, 145867320, 103977935, 146138257, 146273749, 146409460, 146544756, 146680928, 146816684, 146952660, 104998360, 147224657, 147360678, 147496919, 147632744, 147769445, 147905730, 148042236, 106108026, 148315291, 148451840, 148588610, 148724964, 148862193, 148999007, 149136040, 107174637, 149410152, 149547229, 149684527, 149821410, 149959166, 150096507, 150234068, 108264350, 150509234, 150646839, 150784664, 150922074, 151060356, 151198225, 151336312, 0, 151612532, 151750663, 151889014, 152026951, 152165759, 152304154, 152442767, 110446907, 152720038, 152858695, 152997571, 153136035, 153275367, 153414287, 153553426, 111539747, 153831747, 153970929, 154110330, 154249319, 154389175, 154528619, 154668282, 112655697, 154947651, 155087358, 155227282, 155366796, 155507175, 155647143, 155787330, 113728540, 156067745, 156207975, 156348423, 156488460, 156629361, 156769852, 156910561, 114890713, 157192022, 157332774, 157473744, 157614304, 157755727, 157896740, 158037971, 116009779, 158320475, 158461749, 158603240, 158744322, 158886265, 159027799, 159169551, 117151944, 159453097, 159594892, 159736903, 159878506, 160020969, 160163024, 160305295, 0, 160589881, 160732196, 160874727, 161016851, 161159832, 161302406, 161445197, 119439375, 161730821, 161873655, 162016705, 162159348, 162302847, 162445940, 162589249, 120584646, 162875910, 163019261, 163162829, 163305991, 163450007, 163593618, 163737444, 121752987, 164025140, 164169009, 164313093, 164456773, 164601305, 164745433, 164889776, 122878317, 165178504, 165322889, 165467490, 165611687, 165756734, 165901378, 166046236, 124092756, 166335995, 166480896, 166626012, 166770725, 166916286, 167061445, 167206818, 125264196, 167497607, 167643022, 167788652, 167933880, 168079955, 168225628, 168371514, 126458656, 168663331, 168809260, 168955403, 169101145, 169247732, 169393918, 169540318, 0, 169833160, 169979602, 170126258, 170272513, 170419612, 170566310, 170713221, 128850654, 171007087, 171154041, 171301208, 171447976, 171595585, 171742794, 171890217, 130048208, 172185104, 172332569, 172480247, 172627526, 172775644, 172923364, 173071297, 131268715, 173367204, 173515179, 173663366, 173811156, 173959783, 174108013, 174256454, 132446472, 174553380, 174701863, 174850559, 174998859, 175147993, 175296731, 175445681, 133712800, 175743622, 175892614, 176041818, 176190626, 176340266, 176489512, 176638969, 134936400, 176937925, 177087424, 177237134, 177386449, 177536596, 177686348, 177836311, 136182865, 178136279, 178286284, 178436500, 178586322, 178736973, 178887231, 179037699, 0, 179338677, 179489188, 179639908, 179790236, 179941391, 180092153, 180243125, 138678864, 180545112, 180696126,

Продолжение таблицы Б.3

180847350, 180998183, 181149840, 181301107, 181452582, 139928424, 181755575, 181907092, 182058819, 182210156, 182362314, 182514083, 182666060, 141200745, 182970058, 183122077, 183274306, 183426145, 183578804, 183731075, 183883553, 142430742, 184188553, 184341074, 184493802, 184646144, 184799302, 184952073, 185105052, 143748454, 185411052, 185564073, 185717301, 185870143, 186023800, 186177071, 186330548, 145023876, 186637546, 186791066, 186944793, 187098135, 187252290, 187406059, 187560035, 146321933, 187868029, 188022047, 188176271, 188330112, 188484762, 188639029, 188793502, 0, 189102490, 189257005, 189411726, 189566065, 189721210, 189875974, 190030943, 148921121, 190340922, 190495934, 190651150, 190805985, 190961625, 191116884, 191272348, 150222290, 191583317, 191738824, 191894534, 192049865, 192205998, 192361751, 192517709, 151545951, 192829666, 192985666, 193141870, 193297696, 193454321, 193610567, 193767018, 152827880, 194079961, 194236454, 194393150, 194549469, 194706585, 194863324, 195020266, 154196356, 195334192, 195491177, 195648364, 195805176, 195962782, 196120012, 196277445, 155523145, 196592352, 196749828, 196907505, 197064808, 197222903, 197380623, 197538545, 156872264, 197854432, 198012397, 198170563, 198328357, 198486939, 198645149, 198803559, 0, 199120423, 199278876, 199437531, 199595813, 199754882, 199913580, 200072478, 159573599, 200390316, 200549257, 200708398, 200867169, 201026724, 201185908, 201345292, 160925865, 201664103, 201823530, 201983157, 202142415, 202302454, 202462124, 202621993, 162300283, 202941775, 203101687, 203261799, 203421543, 203582065, 203742219, 203902573, 163633725, 204223322, 204383719, 204544314, 204704543, 204865547, 205026185, 205187021, 165052234, 205508737, 205669616, 205830694, 205991406, 206152892, 206314012, 206475330, 166429823, 206798010, 206959371, 207120929, 207282125, 207444090, 207605692, 207767490, 167829367, 208091131, 208252973, 208415012, 208576689, 208739133, 208901215, 209063493, 0, 209388093, 209550415, 209712932, 209875090, 210038011, 210200572, 210363329, 170631595, 210688885, 210851686, 211014681, 211177318, 211340716, 211503755, 211666988, 172034341, 211993500, 212156777, 212320250, 212483365, 212647237, 212810753, 212974463, 173458826, 213301927, 213465681, 213629628, 213793220, 213957567, 214121559, 214285744, 174843260, 214614157, 214778386, 214942808, 215106876, 215271696, 215436162, 215600821, 176310968, 215930182, 216094885, 216259780, 216424323, 216589614, 216754553, 216919685, 177738693, 217249991, 217415167, 217580534, 217745551, 217911313, 218076724, 218242327, 179187923, 218573577, 218739224, 218905062, 219070551, 219236782, 219402664, 219568737, 0,219900928, 220067045, 220233353, 220399313, 220566013, 220732365, 220898907, 182089593, 221232036, 221398623, 221565399, 221731829, 221898996, 222065817, 222232827, 183542106, 222566891, 222733946, 222901190, 223068089, 223235721, 223403010, 223570486, 185015874, 223905485, 224073006, 224240716, 224408083, 224576180, 224743935, 224911877, 186450678, 225247806, 225415793, 225583968, 225751802, 225920362, 226088582, 226256988, 187966663, 226593847, 226762298, 226930937, 227099236, 227268258, 227436942, 227605812, 189443763, 227943596, 228112511, 228281612, 228450376, 228619859, 228789005, 228958337, 190941851, 229297046, 229466423, 229635985, 229805212, 229975155, 230144762, 230314555, 0, 230654185, 230824023, 230994045, 231163734, 231334135, 231504203, 231674455, 193941330, 232015004, 232185302, 232355783, 232525933, 232696791, 232867318, 233038028, 195442806, 233379494, 233550250, 233721189, 233891798, 234063113, 234234097, 234405265, 196964986, 234747645, 234918858, 235090253, 235261321, 235433090, 235604531, 235776154, 198449451, 236119447, 236291116, 236462966, 236634491, 236806713, 236978610, 237150688, 200012704, 237494889, 237667013, 237839318, 238011298, 238183973, 238356324, 238528855, 201538335, 238873963, 239046541, 239219298, 239391733, 239564859, 239737663, 239910646, 203084368, 240256659, 240429688, 240602897, 240775785, 240949361, 241122617, 241296051, 0, 241642965, 241816446, 241990105, 242163446, 242337470, 242511176, 242685059, 206179857, 243032874, 243206804, 243380912, 243554703, 243729175, 243903330, 244077662, 207729411, 244426373, 244600752, 244775308, 244949549, 245124467, 245299070, 245473849, 209299051, 245823454, 245998281, 246173283, 246347972, 246523335, 246698384, 246873609, 210832385, 247224107, 247399379, 247574826, 247749963, 247925769, 248101264, 248276933, 212441823, 248628320, 248804038, 248979929, 249155511, 249331759, 249507699, 249683811, 214015060, 250036085, 250212246, 250388580, 250564607, 250741296, 250917678, 251094233, 215608049, 251447391, 251623994, 251800769, 251977240, 252154368, 252331192, 252508188, 0, 252862227, 253039271, 253216487, 253393400, 253570966, 253748230, 253925665, 218797600, 254280584, 254458068, 254635722, 254813076, 254991080, 255168783, 255346656, 220394271, 255702452, 255880374, 256058466, 256236259, 256414698, 256592839, 256771150, 222010344, 257127819, 257306178, 257484707, 257662938, 257841812, 258020389, 258199135, 223591681, 258556676, 258735471, 258914434, 259093103, 259272410, 259451423, 259630603, 225246150, 259989013, 260168242, 260347639, 260526744, 260706482, 260885929, 261065542, 226865998, 261424818, 261604481, 261784310, 261963849, 262144018, 262323898, 262503942, 228504883, 262864082, 263044177, 263224437, 263404410, 263585008, 263765318, 263945794, 0, 264306794, 264487320, 264668010, 264848414, 265029440, 265210181, 265391085, 231786406, 265752944, 265933899, 266115018, 266295853, 266477305, 266658474, 266839807, 233429164, 267202522, 267383905, 267565450, 267746714, 267928592, 268110189, 268291947, 235090580, 268655515, 268837325, 269019297, 269200989, 269383290, 269565313, 269747497, 236718980, 270111915, 270294151, 270476546, 270658665, 270841389,

Продолжение таблицы Б.3

271023837, 271206444, 238417265, 271571711, 271754370, 271937189, 272119733, 272302879, 272485750, 272668780, 240082661, 273034891, 273217974, 273401214, 273584182, 273767748, 273951041, 274134492, 241766319, 274501446, 274684950, 274868611, 275052002, 275235986, 275419699, 275603570, 0, 275971365, 276155288, 276339369, 276523181, 276707582, 276891715, 277076005, 245137597, 277444636, 277628978, 277813477, 277997709, 278182526, 278367077, 278551784, 246825349, 278921250, 279106009, 279290924, 279475575, 279660806, 279845774, 280030897, 248530953, 280401195, 280586371, 280771700, 280956769, 281142413, 281327796, 281513333, 250205412, 281884460, 282070051, 282255794, 282441279, 282627335, 282813132, 282999082, 251946239, 283371036, 283557040, 283743196, 283929095, 284115561, 284301771, 284488133, 253656059, 284860910, 285047326, 285233893, 285420206, 285607081, 285793703, 285980475, 255383306, 286354073, 286540900, 286727876, 286914601, 287101884, 287288915, 287476096, 0, 287850513, 288037749, 288225133, 288412269, 288599958, 288787398, 288974987, 258842003, 289350219, 289537863, 289725654, 289913199, 290101293, 290289140, 290477135, 260573595, 290853180, 291041230, 291229428, 291417381, 291605878, 291794131, 291982531, 262322176, 292359386, 292547841, 292736443, 292924802, 293113702, 293302359, 293491162, 264041629, 293868825, 294057684, 294246688, 294435453, 294624753, 294813814, 295003019, 265823670, 295381486, 295570748, 295760153, 295949322, 296139021, 296328484, 296518090, 267576733, 296897358, 297087021, 297276827, 297466398, 297656495, 297846358, 298036363, 269346330, 298416430, 298606493, 298796697, 298986669, 299177163, 299367425, 299557828, 0, 299938691, 300129152, 300319754, 300510125, 300701014, 300891674, 301082473, 272889996, 301464130, 301654988, 301845985, 302036755, 302228038, 302419093, 302610288, 274664219, 302992735, 303183989, 303375380, 303566547, 303758222, 303949672, 304141261, 276454510, 304524496, 304716143, 304907928, 305099490, 305291556, 305483399, 305675380, 278217833, 306059400, 306251440, 306443616, 306635573, 306828028, 307020264, 307212635, 280039710, 307597437, 307789868, 307982435, 308174784, 308367627, 308560253, 308753014, 281834777, 309138596, 309331417, 309524372, 309717112, 309910342, 310103357, 310296506, 283645432, 310682864, 310876073, 311069416, 311262546, 311456161, 311649564, 311843099, 0, 312230231, 312423827, 312617555, 312811074, 313005073, 313198862, 313392782, 287271512, 313780685, 313974666, 314168779, 314362685, 314557066, 314751240, 314945544, 289087102, 315334214, 315528580, 315723076, 315917368, 316112129, 316306686, 316501373, 290917784, 316890808, 317085556, 317280434, 317475110, 317670250, 317865189, 318060257, 292723794, 318450454, 318645584, 318840841, 319035900, 319231418, 319426738, 319622185, 294584079, 320013141, 320208651, 320404287, 320599727, 320795622, 320991320, 321187145, 296419861, 321578858, 321774746, 321970759, 322166579, 322362849, 322558925, 322755126, 298270228, 323147592, 323343857, 323540246, 323736444, 323933088, 324129540, 324326117, 0, 324719333, 324915973, 325112736, 325309312, 325506327, 325703154, 325900104, 301976060, 326294068, 326491082, 326688218, 326885169, 327082554, 327279755, 327477077, 303831699, 327871785, 328069172, 328266679, 328464004, 328661759, 328859331, 329057024, 305701402, 329452474, 329650231, 329848109, 330045806, 330243928, 330441871, 330639933, 307548861, 331036121, 331234248, 331432494, 331630563, 331829051, 332027362, 332225792, 309446079, 332622716, 332821211, 333019824, 333218262, 333417115, 333615793, 333814589, 311321232, 334212246, 334411108, 334610086, 334808892, 335008108, 335207152, 335406313, 313209915, 335804699, 336003926, 336203268, 336402441, 336602019, 336801427, 337000951, 0, 337400064, 337599654, 337799359, 337998897, 338198835, 338398606, 338598491, 316992732, 338998329, 339198281, 339398346, 339598248, 339798544, 339998677, 340198922, 318887049, 340599480, 340799793, 341000218, 341200482, 341401135, 341601628, 341802232, 320794350, 342203507, 342404179, 342604962, 342805587, 343006595, 343207446, 343408407, 322681962, 343810398, 344011427, 344212566, 344413550, 344614913, 344816120, 345017437, 324614588, 345420139, 345621524, 345823018, 346024360, 346226075, 346427638, 346629309, 326527713, 347032719, 347234459, 347436306, 347638004, 347840070, 348041987, 348244010, 328453263, 348648126, 348850219, 349052418, 349254471, 349456886, 349659154, 349861529, 0, 350266347, 350468791, 350671341, 350873747, 351076509, 351279129, 351481853, 332310191, 351887370, 352090165, 352293063, 352495820, 352698929, 352901897, 353104969, 334241760, 353511183, 353714326, 353917571, 354120678, 354324132, 354527448, 354730866, 336185184, 355137773, 355341263, 355544853, 355748309, 355952106, 356155768, 356359531, 338111595, 356767128, 356970963, 357174898, 357378700, 357582839, 357786845, 357990951, 340078054, 358399236, 358603414, 358807691, 359011839, 359216317, 359420667, 359625115, 342027697, 360034083, 360238603, 360443221, 360647713, 360852530, 361057220, 361262008, 343988611, 361671657, 361876518, 362081475, 362286309, 362491463, 362696493, 362901620, 0, 363311946, 363517145, 363722441, 363927616, 364133104, 364338473, 364543936, 347916662, 364954936, 365160474, 365366105, 365571619, 365777441, 365983146, 366188945, 349884001, 366600616, 366806489, 367012455, 367218307, 367424461, 367630501, 367836634, 351862015, 368248973, 368455180, 368661479, 368867666, 369074151, 369280524, 369486989, 353825808, 369899993, 370106533, 370313163, 370519684, 370726498, 370933203, 371139998, 355824472, 371553664, 371760534, 371967494, 372174349, 372381489, 372588525, 372795649, 357809119, 373209972, 373417172, 373624460, 373831646, 374039112, 374246476, 374453927, 359803836, 374868906, 375076434, 375284048, 375491563, 375699353, 375907044, 376114821, 0, 376530452,

Окончание таблицы Б.3

376738305, 376946244, 377154088, 377362200, 377570216, 377778317, 363799907, 378194596, 378402774, 378611037, 378819206, 379027639, 379235978, 379444402, 365801470, 379861326, 380069827, 380278411, 380486905, 380695657, 380904318, 381113063, 367812483, 381530629, 381739451, 381948355, 382157172, 382366241, 382575223, 382784286, 369812175, 383202491, 383411632, 383620855, 383829993, 384039377, 384248678, 384458059, 371841360, 384876899, 385086359, 385295897, 385505355, 385715054, 385924672, 386134368, 373859436, 386553840, 386763616, 386973469, 387183245, 387393256, 387603189, 387813200, 375886332, 388233300, 388443391, 388653557, 388863650, 389073971, 389284218, 389494541, 0, 389915267, 390125670, 390336148, 390546555, 390757185, 390967744, 391178378, 379947190, 391599725, 391810440, 392021228, 392231948, 392442885, 392653755, 392864697, 381981368, 393286663, 393497687, 393708784, 393919815, 394131058, 394342236, 394553485, 384023724, 394976067, 395187398, 395398801, 395610142, 395821688, 396033173, 396244729, 386057764, 396667922, 396879559, 397091267, 397302916, 397514764, 397726554, 397938413, 388115722, 398362215, 398574157, 398786167, 398998122, 399210271, 399422364, 399634526, 390165581, 400058932, 400271176, 400483488, 400695748, 400908195, 401120590, 401333052, 392222967, 401758059, 401970605, 402183216, 402395779, 402608523, 402821218, 403033979, 0, 403459584, 403672428, 403885337, 404098201, 404311240, 404524234, 404737291, 396345243, 405163490, 405376632, 405589838, 405803000, 406016333, 406229623, 406442976, 398410355, 406869766, 407083203, 407296703, 407510163, 407723787, 407937372, 408151018, 400482330, 408578395, 408792127, 409005919, 409219675, 409433589, 409647467, 409861404, 402549087, 410289366, 410503389, 410717472, 410931522, 411145724, 411359893, 411574121, 404634005, 412002662, 412216976, 412431348, 412645690, 412860178, 413074636, 413289152, 406713930, 413718270, 413932873, 414147532, 414362164, 414576937, 414791683, 415006485, 408800043, 415436176, 415651065, 415866010, 416080931, 416295986, 416511018, 416726104, 0, 417156365, 417371539, 417586767, 417801975, 418017311, 418232627, 418447996, 412980217, 418878822, 419094280, 419309789, 419525282, 419740897, 419956495, 420172145, 415074506, 420603534, 420819273, 421035062, 421250839, 421466730, 421682609, 421898538, 417174297, 422330485, 422546503, 422762571, 422978629, 423194796, 423410953, 423627159, 419272060, 424059661, 424275957, 424492301, 424708638, 424925079, 425141513, 425357995, 421382050, 425791047, 426007619, 426224238, 426440852, 426657565, 426874274, 427091029, 423490242, 427524629, 427741475, 427958366, 428175256, 428392238, 428609220, 428826247, 425603240, 429260391, 429477509, 429694670, 429911835, 430129085, 430346338, 430563634, 0, 430998318, 431215706, 431433137, 431650574, 431868089, 432085612, 432303176, 429837630, 432738396, 432956052, 433173750, 433391457, 433609237, 433827027, 434044857, 431959253, 434480609, 434698532, 434916495, 435134470, 435352512, 435570567, 435788662, 434084977, 436224943, 436443130, 436661356, 436879598, 437097900, 437316219, 437534575, 436211943, 437971381, 438189831, 438408318, 438626824, 438845385, 439063965, 439282582, 438345038, 439719909, 439938619, 440157366, 440376135, 440594952, 440813792, 441032667, 440479609, 441470511, 441689480, 441908484, 442127514, 442346585, 442565683, 442784814, 442617565, 443223171, 443442398, 443661656, 443880945, 444100269, 444319623, 444539009
};

Библиография

[1] ИСО/МЭК 14496–3:2009 Информационные технологии. Кодирование аудиовизуальных объектов. Часть 3. Аудио (ИСО/МЭК 14496–3:2009 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 *Н. А. Аргунова*
Технический редактор *В. Н. Прусакова*
Корректор *Л. Я. Митрофанова*
Компьютерная верстка *З. И. Мартыновой*

Сдано в набор 17.06.2014. Подписано в печать 03.09.2014. Формат 60×84 1/8. Бумага офсетная. Гарнитура Ариал. Печать офсетная. Усл. печ. л. 7,44. Уч.-изд. л. 7,00. Тираж 41 экз. Зак. 1038.

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